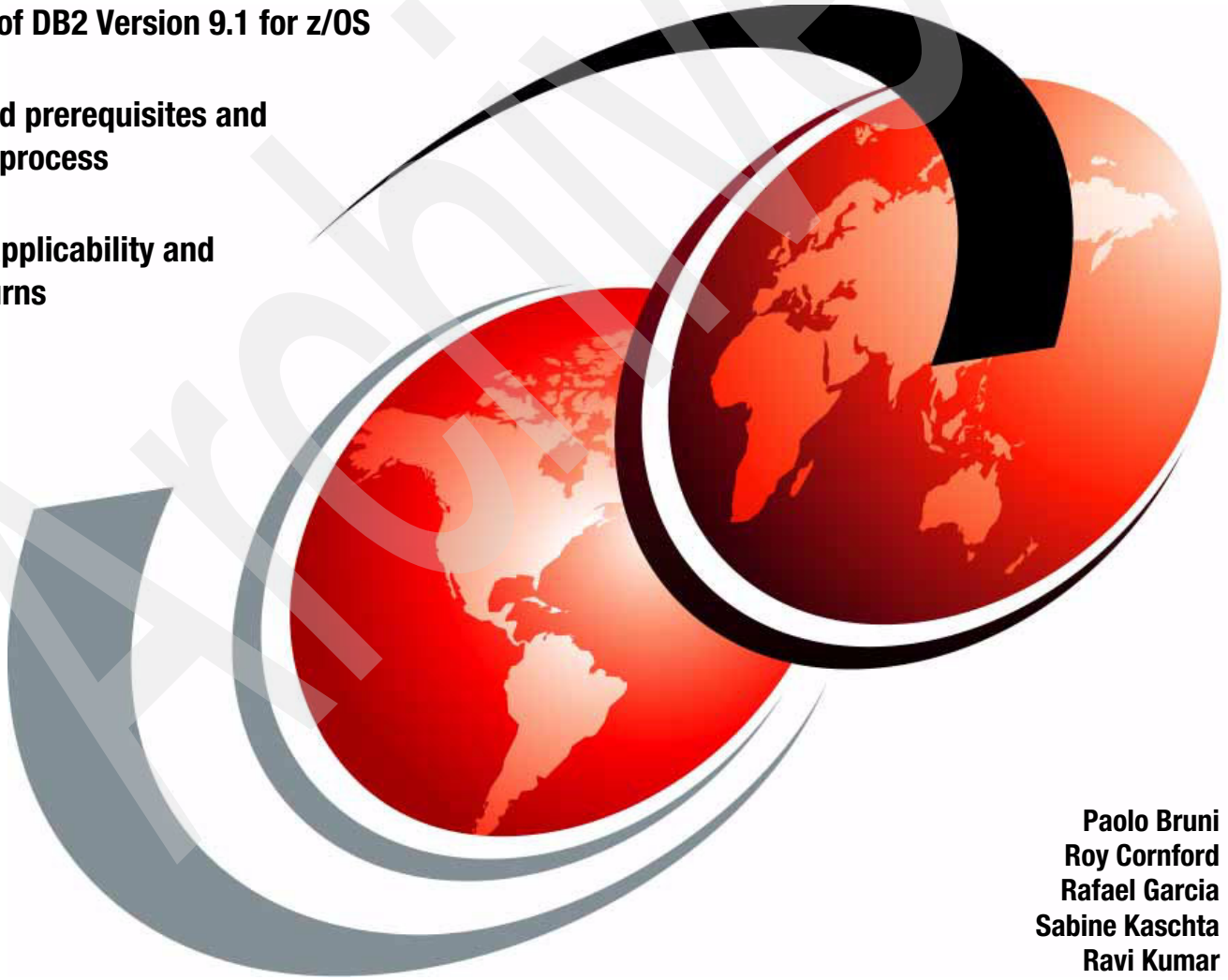
Information Management software

IBM

DB2 9 for z/OS Technical Overview

- Read about the new features and functions of DB2 Version 9.1 for z/OS
- Understand prerequisites and migration process
- Evaluate applicability and major returns



Archived

Paolo Bruni
Roy Cornford
Rafael Garcia
Sabine Kaschta
Ravi Kumar

Redbooks

ibm.com/redbooks

Archived

International Technical Support Organization

DB2 9 for z/OS Technical Overview

June 2007

Archived

SG24-7330-00

Note: Before using this information and the product it supports, read the information in “Notices” on page xxv.

Archived

First Edition (June 2007)

This edition applies to IBM DB2 Version 9.1 for z/OS (program number 5635-DB2).

© Copyright International Business Machines Corporation 2007. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures xiii

Examples xxi

Tables xxiii

Notices xxv
Trademarks xxvi

Preface xxvii
The team that wrote this IBM Redbooks publication xxvii
Become a published author xxxi
Comments welcome. xxxi

Summary of changes xxxiii
June 2007, First Edition xxxiii
July 2007, First Update xxxiii
November 2007, Second Update xxxiii
July 2008, Third Update xxxiv
March 2009, Fourth Update xxxiv
April 2009, Fifth Update xxxiv
March 2010, Sixth Update xxxiv

Chapter 1. DB2 9 for z/OS at a glance 1
1.1 The main goals of DB2 9 for z/OS 2
1.2 The innovations. 2
1.2.1 Enable high-volume transaction processing for next wave of Web applications. . . 3
1.2.2 Extend the lead in transaction processing availability, scalability, and performance 9
1.2.3 Reduce cost of ownership and System z specific skill needs 13
1.2.4 Improve data warehousing and optimizer 18

Part 1. Subsystem functions 1

Chapter 2. Synergy with System z 3
2.1 zSynergy in general 4
2.2 Synergy with new I/O and new CPU hardware 4
2.2.1 Sequential I/O 4
2.2.2 CPU 8
2.3 System z9 Integrated Information Processor and specialty engines. 11
2.3.1 The special purpose processors 11
2.3.2 z/OS support for zIIPs 14
2.3.3 DB2 support for zIIPs 14
2.3.4 DB2 monitoring of zIIPs 16
2.3.5 The zIIPed workloads 16
2.3.6 Index maintenance for DB2 Utilities 22
2.3.7 Utility processing eligible for IBM zIIP redirect 22
2.4 Activating zIIP exploitation 23
2.4.1 IEAOPTxx parmlib member 23
2.4.2 zIIP processors 24
2.4.3 SDSF zIIP and zAAP support 25

© Copyright IBM Corp. 2007. All rights reserved.

2.4.4 WLM services for zIIP execution . . . 25
2.5 Capacity planning for zIIPs . . . 26
2.5.1 Estimating zIIP usage prior to DB2 V8 . . . 26
2.6 RMF support for zIIPs . . . 28
2.6.1 RMF zIIP implementation . . . 28
2.6.2 RMF CPU activity for zIIPs . . . 29
2.6.3 RMF Partition Data Report for zIIPs . . . 30
2.6.4 RMF Service Class Periods Report for zIIPs . . . 30
2.6.5 RMF Distributed Data Server for zIIPs . . . 32
2.7 SMF records for zIIP support . . . 33
2.7.1 SMF type 70 records for zIIPs . . . 33
2.7.2 SMF type 72 record for zIIPs . . . 34
2.7.3 SMF type 79 record for zIIPs . . . 36
2.8 RMF Monitor III support for zIIPs . . . 36
2.9 Advanced data serving on System z . . . 39

Chapter 3. Scalability . . . 43
3.1 Virtual storage management . . . 44
3.1.1 64-bit Distributed Data Facility . . . 44
3.1.2 Runtime structures above the bar . . . 46
3.2 WLM assisted buffer pool management . . . 47
3.3 NOT LOGGED table spaces . . . 48
3.3.1 The logging attribute . . . 51
3.3.2 Indexes . . . 51
3.3.3 XML table spaces . . . 51
3.3.4 LOB table spaces . . . 52
3.3.5 SYSLGRNX and SYSCOPY . . . 53
3.3.6 Recovery implications for table spaces that are not logged . . . 54
3.3.7 Recovering NOT LOGGED LOB or XML table spaces . . . 56
3.3.8 Deadlock and time-out considerations . . . 56
3.3.9 LOAD and REORG interaction . . . 56
3.3.10 Further REORG considerations . . . 57
3.3.11 Online LOAD considerations . . . 57
3.3.12 REBUILD INDEX considerations . . . 57
3.3.13 CHECK DATA considerations . . . 57
3.3.14 QUIESCE considerations . . . 58
3.3.15 Application programming considerations . . . 58
3.3.16 Operational considerations . . . 58
3.3.17 Performance objectives . . . 59
3.4 Other logging enhancements . . . 59
3.5 Exploit WLM to resolve latch contention . . . 60
3.6 Universal table space . . . 62
3.6.1 Partition by growth . . . 63
3.6.2 Partition by range . . . 63
3.7 Reordered row format . . . 63
3.8 Index compression . . . 66
3.9 Native SQL procedures . . . 68
3.10 Other enhancements . . . 69
3.10.1 Support for Internet Protocol Version 6 (IPv6) . . . 69
3.10.2 Increase the number of user-defined indexes on the catalog . . . 70
3.10.3 Trace filtering . . . 70
3.10.4 Allow EXCLUDE qualification in DSNDWQAL for READS . . . 71
3.10.5 Improve the operations of Audit Trace function . . . 71

3.10.6 Greater than 4 KB page size for indexes . . . 71
3.10.7 Asymmetrical split of index pages . . . 72

Chapter 4. Availability . . . 73
4.1 Universal table space . . . 74
4.1.1 Partition-by-growth table spaces . . . 74
4.1.2 Range-partitioned universal table spaces . . . 80
4.2 Clone table support . . . 81
4.2.1 Creation of a clone table . . . 81
4.2.2 Commands and clone tables . . . 85
4.2.3 Exchanging clone and base table . . . 88
4.2.4 Dropping a clone table . . . 91
4.2.5 Restrictions . . . 92
4.2.6 Utilities and clone tables . . . 92
4.3 Recovery to point in time with consistency . . . 100
4.4 APPEND . . . 102
4.5 Conditional Restart Log Truncation by Timestamp . . . 103
4.5.1 ENDTIME usage . . . 104
4.5.2 SYSPITRT usage . . . 106
4.6 REFRESH for EARLY code . . . 107
4.7 RENAME INDEX . . . 108
4.8 RENAME COLUMN . . . 109
4.8.1 Effects of ALTER TABLE RENAME COLUMN . . . 109
4.8.2 Usage restrictions . . . 110
4.9 UPDATE SCHEMA . . . 111
4.9.1 Considerations . . . 113
4.10 Toleration of DBET errors during DB2 restart . . . 113
4.10.1 Steps taken when DBET errors occur . . . 114
4.11 Miscellaneous availability and serviceability enhancements . . . 114
4.11.1 Add keyword RECOVERBEFORE to RECOVER utility . . . 114
4.11.2 Timeout or deadlock condition documented in utility output . . . 115
4.11.3 Julian date and time stamp in utility output . . . 116
4.11.4 DSN1PRNT output for broken pages formatted . . . 117
4.11.5 More messages during LOGAPPLY phase of recover utility . . . 117
4.11.6 Avoid copypending restrictive state when broken page detected . . . 118
4.11.7 New ability to cancel DB commands . . . 119

Chapter 5. Data sharing . . . 121
5.1 Data sharing logging improvement . . . 122
5.2 Index improvements . . . 122
5.3 Reduction in LOB locks . . . 122
5.4 Locking constraint relief . . . 123
5.5 Improved group buffer pool write performance . . . 123
5.6 Improved WLM routing based on DB2 health . . . 123
5.7 Improved workload balancing within the same LPAR . . . 123
5.8 Group buffer pool dependency removal by command . . . 124
5.9 Open dataset ahead of use via command . . . 124
5.10 Enhanced messages when unable to get p-locks . . . 124
5.11 Initiating Auto GRECP recovery at the end of restart . . . 125
5.12 Deferring the updates of SYSLGRNX till after end of restart . . . 126
5.13 Opening data sets earlier in restart processing . . . 126
5.14 Allowing table level retained locks to support postponed abort URs . . . 127
5.15 Simplification of the special open processing . . . 127

5.16 Coupling Facility Control Code (CFCC) Level 15 128

Part 2. Application functions 129

Chapter 6. SQL 131
6.1 New data types: BIGINT, BINARY, VARBINARY, DECFLOAT 132
6.1.1 BIGINT 132
6.1.2 BINARY 132
6.1.3 DECFLOAT 133
6.2 INSTEAD OF triggers 136
6.2.1 Considerations 138
6.2.2 Authorization considerations 139
6.2.3 Catalog changes 139
6.3 MERGE 139
6.3.1 EXPLAIN for MERGE 141
6.3.2 Authorization 142
6.4 SELECT FROM MERGE/UPDATE/DELETE 142
6.4.1 Selecting values as you merge: SELECT FROM MERGE 143
6.4.2 Selecting values as you update: SELECT FROM UPDATE 145
6.4.3 Selecting values as you delete: SELECT FROM DELETE 146
6.5 ORDER BY and FETCH FIRST in subselect 147
6.5.1 Returning fullselect results in the order of the subquery result table 149
6.5.2 Other considerations 150
6.6 TRUNCATE 150
6.6.1 TRUNCATE statement 151
6.6.2 Using the TRUNCATE statement with multilevel security 152
6.6.3 Other considerations 152
6.7 INTERSECT and EXCEPT 152
6.8 Built-in functions 155
6.8.1 ASCII_CHR 156
6.8.2 DIFFERENCE 157
6.8.3 EXTRACT date values 157
6.8.4 EXTRACT time values 158
6.8.5 MONTHS_BETWEEN 159
6.8.6 SOUNDEX 160
6.8.7 TIMESTAMPDIFF 160
6.8.8 TIMESTAMP_FORMAT 162
6.8.9 TIMESTAMP_ISO 163
6.8.10 VARCHAR_FORMAT 164

Chapter 7. Application enablement 167
7.1 Automatic creation of objects 168
7.1.1 Implicit database 168
7.1.2 Implicit table space 170
7.1.3 Implicit system required objects 173
7.1.4 Authorization considerations 174
7.2 Index on expression 175
7.2.1 Examples 176
7.2.2 Considerations on index on expression 177
7.2.3 Restrictions 178
7.3 Optimistic concurrency control and update detection 178
7.3.1 Optimistic concurrency control and cursor stability isolation 179
7.3.2 Determining when a row was changed 180
7.3.3 Finding the rows that were changed within a specified period of time 181

7.3.4 Considerations for tables containing a ROW CHANGE TIMESTAMP column . . 182
7.4 Cultural sort and case insensitive comparison 183
7.4.1 z/OS support for Unicode 183
7.4.2 Collation 184
7.4.3 The Unicode Collation Algorithm. 185
7.4.4 DB2 implementation 185
7.4.5 COLLATION_KEY built-in function 186
7.4.6 Extended UPPER and LOWER built-in functions 189
7.5 CURRENT SCHEMA 189
7.5.1 Unqualified alias, index, JAR, sequence, table, trigger, and view names. 190
7.5.2 Summary. 191
7.6 Skip locked rows 191
7.7 LOB file reference 193
7.7.1 LOB locators 195
7.7.2 Purpose of LOB locators. 196
7.7.3 Different types of LOB locators 197
7.7.4 LOB file reference variables 197
7.7.5 DB2-generated file reference variable constructs. 198
7.7.6 Language support for LOB file reference variables. 199
7.8 FETCH CONTINUE 202
7.8.1 Example using dynamic SQL 203
7.8.2 Example (using static SQL). 205
7.8.3 Scrollable and allocated cursors 207
7.8.4 Restrictions 207
7.9 Spatial support 207
7.9.1 IBM Spatial support for DB2 for z/OS 208
7.10 Allow ALTER TABLE ... LONG VARCHAR TO VARCHAR 214
7.11 RENAME COLUMN 215
7.12 RENAME INDEX. 215
7.13 SMS constructs 215
7.13.1 CREATE STOGROUP 216
7.13.2 ALTER STOGROUP. 217
7.13.3 Catalog changes 218
7.14 Native SQL procedures. 218
7.14.1 CREATE PROCEDURE syntax for native SQL procedures 219
7.14.2 ALTER PROCEDURE syntax for native SQL procedures 226
7.14.3 Sample versioning and deployment scenarios 231
7.14.4 GET DIAGNOSTICS. 234
7.14.5 FOR SQL control statement 235
7.14.6 New catalog table SYSIBM.SYSENVIRONMENT 236
7.15 Nested compound statements in SQL stored procedures 238
7.15.1 Using a compound statement within a condition handler declaration 239
7.15.2 Using statement labels with nested compound statements 240
7.15.3 Scoping SQL variable declarations. 242
7.15.4 Scoping cursor definitions. 243
7.15.5 Scoping condition names 244
7.15.6 Scoping condition handler declarations 244
7.15.7 Summary of name scoping in compound statements. 246
7.15.8 Name resolution differences between native and external SPs 248
7.16 New stored procedure related special registers 249
7.17 DB2 and DSN commands for native SQL procedures 251
7.17.1 -DISPLAY PROCEDURE 251
7.17.2 -START/STOP PROCEDURE 251

7.17.3 REBIND PACKAGE 251
7.18 Changes to DB2-supplied dynamic SQL processing applications 252
7.18.1 Change SPUFI plan names on SPUFI defaults panel 255
7.19 Differences from external SQL procedures 255
7.20 Unified Debugger 257
7.20.1 Setting up the Unified Debugger components 257

Chapter 8. XML 263
8.1 Why use XML with DB2 for z/OS 265
8.1.1 DB2 XML Extender 266
8.2 PureXML 267
8.2.1 Platform integration and cost of ownership 267
8.3 XML support 268
8.3.1 The XML data type 270
8.3.2 XML storage infrastructure 271
8.4 Accessing XML data 274
8.4.1 SQL/XML constructors 275
8.5 XPath 283
8.5.1 XML Data Model (XDM) 283
8.5.2 XPath prologs and expressions 285
8.5.3 XMLQUERY 300
8.5.4 XMLEXISTS 305
8.5.5 XPath functions 305
8.6 XML indexing 312
8.6.1 Data types associated with pattern expressions 315
8.6.2 The UNIQUE keyword in an XML index definition 317
8.6.3 Miscellaneous facts and limits on XML indexes 318
8.6.4 XML index exploitation 318
8.6.5 Catalog support for XML indexes 321
8.7 XML schema support 321
8.7.1 XML schema information in the DB2 catalog 321
8.8 XML schema validation 322
8.9 XML decomposition 323
8.10 Utilities support of XML data 323
8.10.1 REPORT 324
8.10.2 CHECK DATA 324
8.10.3 CHECK INDEX 325
8.10.4 COPY 326
8.10.5 REPORT RECOVERY 327
8.10.6 RECOVER table space, RECOVER index, REBUILD index 327
8.10.7 COPYTOCOPY 330
8.10.8 EXEC SQL 330
8.10.9 LISTDEF 330
8.10.10 LOAD 332
8.10.11 QUIESCE 332
8.10.12 REORG INDEX and REORG TABLESPACE 333
8.10.13 REPAIR 333
8.10.14 RUNSTATS 333
8.10.15 UNLOAD 334

Chapter 9. e-business and connectivity 335
9.1 Internet Protocol Version 6 support 336
9.1.1 Textual representation of IPv6 addresses 336

Archived

9.1.2 Dual-mode stack support . . . 338
9.1.3 DB2 commands . . . 339
9.1.4 Utilities . . . 342
9.1.5 Miscellaneous . . . 344
9.2 Positioning DB2 for removal of private protocol . . . 344
9.2.1 DB2 performance trace . . . 346
9.2.2 The private to DRDA protocol REXX tool (DSNTP2DP) . . . 348
9.2.3 DBPROTOCOL Bind option . . . 350
9.3 Run DDF without SNA when using TCP/IP only . . . 351
9.4 IBM DB2 Driver for JDBC and SQLJ . . . 351
9.4.1 Overview . . . 352
9.4.2 Security enhancement . . . 354
9.4.3 Package utility . . . 355
9.4.4 New data types . . . 355
9.4.5 Support for LOB and XML in Java . . . 356
9.4.6 Multi-row SQL operations . . . 356
9.4.7 deferPrepares and sendDataAsIs . . . 357
9.4.8 SQLJ . . . 358
9.4.9 Client reroute support . . . 358

Part 3. Operations and performance . . . 361

Chapter 10. Security . . . 363
10.1 Security functions in general . . . 365
10.2 Enterprise identity mapping . . . 365
10.3 More security options with INSTEAD OF triggers . . . 367
10.4 Improved tracing in DB2 . . . 367
10.5 Audit Management Expert - a new tool for DB2 . . . 370
10.6 The Consul zSecure and InSight Suites . . . 371
10.7 Encryption hardware advances . . . 371
10.8 Encryption in the controllers . . . 372
10.9 Network trusted context . . . 373
10.9.1 Challenges with three-tiered architectures . . . 373
10.9.2 Trusted contexts . . . 374
10.9.3 Secure Socket Layer . . . 376
10.9.4 Roles . . . 377
10.10 Illustrative scenario 1 . . . 377
10.11 Illustrative scenario 2 . . . 379
10.12 Other examples of trusted context and role usage . . . 379
10.12.1 View maintenance on behalf of another user . . . 380
10.12.2 Backing up a DBA, assuming the identity of another user ID . . . 380
10.12.3 Securing DBA activities . . . 381
10.12.4 Reducing risk of a table being dropped by another person . . . 382
10.12.5 Limiting salary updates from a single source . . . 383

Chapter 11. Utilities . . . 385
11.1 REORG enhancements . . . 386
11.1.1 REORG elapsed time reduction . . . 386
11.1.2 Removal of the BUILD2 phase of Online REORG . . . 386
11.1.3 Change to REORG SHRLEVEL REFERENCE by part . . . 388
11.1.4 Online REORG usability and keyword changes . . . 389
11.1.5 Compressed parts handled by Reorg . . . 389
11.1.6 LOB REORG enhancements . . . 390
11.2 CLONE TABLE support . . . 390

11.3 BACKUP and RESTORE SYSTEM 391
11.3.1 Object-level recovery 391
11.3.2 Tape support for BACKUP SYSTEM 393
11.3.3 Incremental FlashCopy 395
11.4 RECOVER enhancements 397
11.4.1 Displaying progress of RECOVER 397
11.4.2 RECOVER using earlier copy 398
11.5 LOAD enhancements 398
11.5.1 Allowing LOAD with SORTKEYS NO 398
11.5.2 Rounding of DECFLOAT by UNLOAD and LOAD 399
11.5.3 Skipping locked rows during UNLOAD 400
11.6 Histogram statistics 400
11.7 COPY improvements 402
11.7.1 COPY with CHECKPAGE 402
11.7.2 SCOPE PENDING for COPY 403
11.8 TEMPLATE switching 403
11.9 MODIFY RECOVERY 404
11.10 DSN1LOGP message enhancement 407
11.11 Online CHECK DATA 407
11.12 Online CHECK LOB 409
11.13 REPAIR LOCATE SHRLEVEL CHANGE 410
11.14 DB2I and DSNU CLIST enhancement 410
11.15 Online REBUILD INDEX 410
11.16 Time stamps in messages 411
11.17 Large format data sets 411
11.18 Support of large block interface 411
11.19 DSN1COPY RESET enhancement 411
11.20 DSNJU004 412
11.21 Enhancements to V8 412
11.21.1 CHECK INDEX SHRLEVEL CHANGE 412
11.21.2 Cross loader support for LOBs 413
11.21.3 LOAD and UNLOAD of LOBs 414
11.21.4 Automatic display of blocking claimers for REORG 415
11.21.5 Online REORG enhancements 415

Chapter 12. Installation and migration 417
12.1 Currency of versions and migration paths 418
12.2 DB2 9 packaging 423
12.2.1 Removed features and functions 423
12.2.2 Deprecated features and functions 424
12.2.3 Base engine and no charge features 425
12.2.4 New optional product 429
12.2.5 Optional chargeable products 429
12.3 Changes to installation and migration 432
12.3.1 Incompatibilities between Version 8 and Version 9 433
12.3.2 Enhanced functions 437
12.3.3 Functions that are deprecated or no longer supported 438
12.4 Removal of support for DB2-managed stored procedures 439
12.5 Installation and migration overview 440
12.5.1 Summary of changes 441
12.5.2 Enhancements to the installation CLIST and panels 441
12.5.3 Installation changes 443
12.5.4 Panels in the MIGRATE flow 450

12.5.5 Panel in the INSTALL flow 470
12.5.6 Panels for ENFM flow 472
12.5.7 Installation CLISTs general troubleshooting tips 476
12.6 Migration and coexistence 476
12.6.1 The migration process 476
12.6.2 High-level migration sequence 479
12.6.3 BSDS conversion required before V9 migration 480
12.6.4 Convergence of temporary space 481
12.6.5 DBMA clean-up 484
12.6.6 DBETE state 485
12.7 Fallback 485
12.8 Other migration considerations 486
12.9 Samples 490
12.9.1 DSNZPARM change summary 490

Chapter 13. Performance 493
13.1 Expectations and best practices 494
13.2 Enhanced sequential key insert 495
13.2.1 Details of enhancement 496
13.3 REOPT AUTO based on parameter marker change 498
13.3.1 REOPT enhancement in V9 499
13.4 RLF enhancements 499
13.4.1 The new Resource Limit Middleware Table and unique index 500
13.4.2 The Resource Limit Middleware Table columns 500
13.4.3 Example of query limits 503
13.4.4 Providing client information to RLF 503
13.4.5 Operational considerations 503
13.4.6 Installation and migration 504
13.5 Histogram statistics 505
13.5.1 Histogram statistics collection 506
13.6 Utilities CPU reduction 508
13.7 CHECK INDEX SHRLEVEL REFERENCE parallelism 509
13.8 Global query optimization 509
13.8.1 Background information 509
13.8.2 Version 9 optimizer enhancements 512
13.8.3 EXPLAIN output 514
13.8.4 Other considerations 514
13.8.5 Subquery tuning 516
13.9 Generalized sparse index and in-memory data cache 517
13.9.1 Pre-V9 scenario 517
13.9.2 DB2 V9 in memory enhancement 518
13.9.3 New DSNZPARM parameter MXDTCACH 519
13.10 Dynamic index ANDing for star join queries 519
13.10.1 Pair-Wise Join 519
13.10.2 Pair-Wise example 521
13.10.3 Runtime optimization 523
13.11 Large Object (LOB/XML) flow optimization 523
13.11.1 IBM DB2 Driver for JDBC and SQLJ 525
13.11.2 DRDA LOB flow optimization performance 527
13.12 LOBs performance improvements 528
13.12.1 LOB locking enhancements 529
13.12.2 Database design and implementation 530
13.12.3 Performance monitoring and tuning 530

13.13 WLM-assisted buffer pool management 530
13.14 Performance scalability 532
13.15 Other performance improvements 534

Appendix A. DB2 Tools for z/OS 537
Overview of tools Announcement 207-040 538
DB2 Optimization Expert for z/OS. 538
DataQuant for z/OS 539
DB2 Utilities Suite for z/OS 540
DB2 Accessories Suite for z/OS 542

Abbreviations and acronyms 543

Related publications 547
IBM Redbooks 547
Other publications 547
Online resources 548
How to get IBM Redbooks 549
Help from IBM 549

Index 551

Archived

Figures

1-1 XML processing . . . 3
1-2 pureXML . . . 4
1-3 Major SQL enhancements . . . 5
1-4 Schema evolution . . . 9
1-5 Virtual storage constraint relief . . . 11
1-6 Performance, scalability, availability . . . 12
1-7 Security in DB2 9 for z/OS . . . 14
1-8 Reducing cost of ownership . . . 15
1-9 zSynergy at a glance . . . 17
1-10 Data warehousing improvements . . . 19
1-11 Index compression . . . 20
2-1 zSynergy at a glance . . . 4
2-2 Synergy with I/O technology advances . . . 5
2-3 Sequential processing with MIDAW . . . 6
2-4 Impact of Adaptive Multi-stream Prefetching . . . 8
2-5 Synergy with new CPU hardware . . . 9
2-6 CPU multiplier across the evolution . . . 9
2-7 Uniprocessor relative power . . . 10
2-8 IBM special processors . . . 11
2-9 Special purpose processors . . . 13
2-10 Summary of current maintenance . . . 16
2-11 DRDA zIIP usage . . . 17
2-12 Flow of work in DRDA example . . . 18
2-13 Complex parallel queries via DRDA connection — BI application . . . 19
2-14 Complex star schema parallel queries via local connection — BI application . . . 20
2-15 SAP workload and zIIPs . . . 21
2-16 DB2 for z/OS V8 utilities used to maintain index structures . . . 22
2-17 D M=CPU command displays zIIPs . . . 24
2-18 SDSF ENC panel showing new columns for zAAP and zIIP times . . . 25
2-19 CPU Activity report showing zIIP activity . . . 29
2-20 RMF Partition Data Report showing zIIPs . . . 30
2-21 RMF Service Class Periods Report showing zIIPs . . . 32
2-22 SMF type 70 record for zIIPs . . . 33
2-23 RMF Overview Report . . . 33
2-24 New overview conditions for zIIPs . . . 34
2-25 SMF type 72 record for zIIPs . . . 34
2-26 New overview conditions based on SMF record 72-3 Workload activity . . . 35
2-27 SMF type 79 record for zIIPs . . . 36
2-28 Sample CPC capacity report indicating usage of zIIPs . . . 36
2-29 Monitor III enclave report indicating zIIPs usage . . . 37
2-30 Monitor III enclave detail report . . . 37
2-31 Sample RMF System Information Report indicating zIIPs usage . . . 38
2-32 Service class detail information using zIIPs . . . 39
2-33 Advanced data serving . . . 40
3-1 z/OS DISPLAY VIRTSTOR,HVSHARE sample output . . . 45
3-2 z/OS V1R5 and later address space . . . 46
3-3 Additional runtime structures above the 2 GB bar . . . 47
3-4 NOT LOGGED scenario . . . 49

© Copyright IBM Corp. 2007. All rights reserved.

3-5 Use NOT LOGGED option with caution 50
3-6 DISPLAY THREAD(*) TYPE(SYSTEM) output 61
3-7 Sample DSNV508I, DSNV510I, and DSNV512I messages 61
3-8 DISPLAY THREAD(*) SERVICE(STORAGE) output 62
3-9 CREATE TABLE RRF and INSERT statements 64
3-10 DSN1PRNT of rows in reordered row format 65
3-11 SELECT * output showing unchanged column sequence 65
3-12 Sample DSN1COMP output for estimating index compression 67
3-13 Sample BPXPRMxx member implementing TCP/IP dual-mode stack 70
4-1 PARTITION BY SIZE syntax 76
4-2 Allocation of new partitions 77
4-3 Condensed data in partition-by-growth table space after REORG 78
4-4 Point-in-time recovery after partition has been added 79
4-5 Creation of a clone table without any dependent objects on base table 82
4-6 Allocated VSAM CLUSTER before ALTER TABLE ADD CLONE 83
4-7 Allocated VSAM CLUSTER after ALTER TABLE ADD CLONE 85
4-8 -DISPLAY DATABASE output with clone objects 86
4-9 -START DATABASE SYNTAX 88
4-10 EXCHANGE DATA SQL statement 88
4-11 EXCHANGE DATA 89
4-12 -DIS DB output with exception states prior to EXCHANGE 90
4-13 -DIS DB output with exception states after EXCHANGE 91
4-14 ALTER TABLE DROP CLONE syntax 91
4-15 REPAIR UTILITY syntax 94
4-16 Base and clone table in SYSTABLES 97
4-17 CREATE base and clone table statements 97
4-18 DSN1LOGP summary of UR that created the base and clone table 98
4-19 Usage of high order bit for clone tables 99
4-20 RECOVER TORBA 102
4-21 ENDTIME in GMT 104
4-22 DSNJU003 CRESTART with ENDTIME 105
4-23 Active CRCR with ENDTIME usage 105
4-24 ENDTIME usage DB2 start message in MSTR address space 106
4-25 DSNJU003 CRESTART with SYSPITRT 106
4-26 CRCR with SYSPITRT 107
4-27 REFRESH command 107
4-28 REFRESH - Error example 107
4-29 REFRESH - Log information 108
4-30 RENAME INDEX syntax 108
4-31 Update schema syntax diagram 111
4-32 RESTOREBEFORE usage scenarios 115
4-33 Timeout condition information in utility output 116
4-34 Julian date and time stamp in utility output 116
4-35 Broken page output in DSN1PRNT 117
4-36 -DIS UTILITY sample output for LOGAPPLY phase of Recover 118
4-37 Sample error message, issued by COPY when broken page is detected 118
4-38 SYSIBM.SYSCOPY - new TTYPE column 118
4-39 STOPP after -STOP DB command 119
4-40 -DIS THREAD(*) TYPE(SYSTEM) command output 119
6-1 MERGE example 140
6-2 Sample EXPLAIN for MERGE 141
6-3 SELECT from MERGE example 143
6-4 Syntax of TRUNCATE statement 151

6-5 ASCII_CHR syntax 156
6-6 expression-1,expression-2 159
6-7 Time stamp syntax 162
7-1 Reuse of implicitly created databases 169
7-2 CREATE INDEX statement for index on expression. 175
7-3 Positioned updates and deletes without optimistic concurrency control 179
7-4 Positioned updates and deletes with optimistic concurrency control 180
7-5 Comparison levels 185
7-6 COLLATION_KEY 186
7-7 Collation name 187
7-8 Collation name examples 187
7-9 CURRENT SCHEMA in DB2 V9. 191
7-10 Association between base table, ROWID, auxiliary table, and LOB table space . . . 194
7-11 Assigning a LOB locator for a LOB value 196
7-12 DSNTIPE installation panel. 202
7-13 DB2 Spatial Support (1 of 2) 209
7-14 Data representing geographic features. 209
7-15 Tables with spatial columns added 210
7-16 Table that includes new spatial data derived from existing spatial data 211
7-17 DB2 Spatial Support (2 of 2) 212
7-18 Examples of spatial applications. 213
7-19 Steps to follow to use spatial support 214
7-20 CREATE STOGROUP syntax 216
7-21 ALTER STOGROUP statement 217
7-22 Start of CREATE PROCEDURE syntax 219
7-23 CREATE PROCEDURE built-in-type 220
7-24 CREATE PROCEDURE option list 221
7-25 Start of ALTER PROCEDURE syntax 227
7-26 ALTER PROCEDURE built-in type 228
7-27 ALTER PROCEDURE option-list 229
7-28 CREATE PROCEDURE UPDATE_BALANCE V1 231
7-29 ALTER PROCEDURE ADD VERSION example 232
7-30 CREATE PROCEDURE SAMPLE for deployment on server SANJOSE 233
7-31 Bind Package statement with DEPLOY option 234
7-32 ALTER PROCEDURE add V2 234
7-33 Bind package statement with deploy option for V2 234
7-34 FOR statement syntax 235
7-35 FOR statement - very simple usage sample. 235
7-36 FOR statement - utilizing more syntax parts. 236
7-37 Nested compound statement 239
7-38 Compound statement in condition handler 240
7-39 Scoping label names. 241
7-40 Correct and incorrect label usage 242
7-41 Scope of variable definitions 243
7-42 Scoping cursor declarations 244
7-43 Scoping condition handler declaration 245
7-44 Scoping of condition handler declarations II 246
7-45 Name resolution in external and native SQL stored procedures. 248
7-46 Better practice with name resolution in external and native SQL stored procedures 249
7-47 SET CURRENT DEBUG MODE syntax 249
7-48 SET CURRENT ROUTINE VERSION syntax. 250
7-49 DISPLAY PROCEDURE output 251
7-50 SPUFI defaults panel - DSNESP02 253

7-51 DSNESP07 255
7-52 RACF definitions for Session Manager 259
7-53 Job to create a file in HFS to hold the environment settings 260
7-54 Sample started task JCL for the Unified Debugger Session Manager on z/OS 261
8-1 Hybrid data server 268
8-2 Types of XML nodes 269
8-3 CREATE TABLE with XML column 272
8-4 Implicitly and explicitly created objects for XML column 273
8-5 -DIS DB command output for table holding XML column 274
8-6 XML standards 275
8-7 XMLDOCUMENT syntax 277
8-8 XMLPI syntax diagram 278
8-9 XMLTEXT example 279
8-10 XMLTEXT example result 279
8-11 Serialization and parsing 280
8-12 XMLSERIALIZE syntax diagram 280
8-13 XMLSERIALIZE example 1 281
8-14 XMLSERIALIZE with hexadecimal column 281
8-15 SQLCODE -433 282
8-16 XMLSERIALIZE to select whole XML document from XML column 282
8-17 XMLPARSE syntax diagram 282
8-18 Document order 285
8-19 XPath expression syntax 286
8-20 Prolog syntax diagram 286
8-21 XPath expression with numeric literals example 287
8-22 XPath expression with string literal containing double quotation mark 287
8-23 XPath expression with variable reference to a string literal 288
8-24 XPath expression with context item 288
8-25 XPath expression with function call 288
8-26 XPath expression with variable reference to a string literal 288
8-27 Path expression syntax 289
8-28 Supported Axes in DB2 9 289
8-29 Sample XML document 291
8-30 Sample XML Document II 298
8-31 XMLQUERY syntax 300
8-32 Sample XML documents III 301
8-33 XML documents containing namespace definitions 304
8-34 XMLEXISTS syntax diagram 305
8-35 XML Document sample with multiple numeric values 312
8-36 XML related parts of CREATE INDEX statement - I 313
8-37 XML-related parts of CREATE INDEX statement - I I 314
8-38 XML index example 315
8-39 DSNXSR schema repository database 322
9-1 Introduction to DB2 V9 support for IPv6 336
9-2 Textual representation of IPv6 addresses 337
9-3 Dual-mode stack support 338
9-4 Changes to DB2 commands 339
9-5 DB2 command reports changed 340
9-6 Display Thread Active/Inactive (Pre-V9) 341
9-7 Display Thread Active/Inactive (V9) 342
9-8 CHANGE LOG INVENTORY (DSNJU003) 343
9-9 Why Remove Private Protocol? 345
9-10 The Tools to Prepare for Private Protocol Removal 345

9-11 The Trace Tool . . . 347
9-12 DB2 Performance Expert Record Trace Report before Alias Resolution . . . 347
9-13 DB2 Performance Expert Record Trace Report after Alias Resolution. . . . 348
9-14 The Catalog Analysis Tool . . . 348
9-15 Options for tool DSNTP2DP . . . 349
9-16 BIND considerations . . . 350
9-17 DB2 Connect 0r Direct . . . 353
9-18 Security enhancement . . . 354
9-19 Decimal Floating Point (1 of 2) . . . 355
9-20 Decimal Floating Point (2 of 2) . . . 356
9-21 deferPrepares and sendDataAsIs . . . 357
9-22 SQLJ . . . 358
10-1 IFCID 269 . . . 368
10-2 IFCID 270 . . . 369
10-3 IFCID 141 and 142 . . . 369
10-4 IPv6 IFCIDs. . . . 370
10-5 Three-tier application architecture . . . 373
10-6 New option for switching IDs. . . . 376
10-7 SQL to establish a trusted context . . . 378
10-8 SQL to create a role . . . 378
10-9 SQL for trusted context and role . . . 379
10-10 Job to define a view maintenance role . . . 380
10-11 Setting AS USER ID . . . 381
11-1 Online REORG phases in V8 for a partition and NPIs . . . 387
11-2 Online REORG phases in V9 for a partition and NPIs . . . 387
11-3 REORG by partition range - Output . . . 388
11-4 Panel: DSNTIP6 . . . 392
11-5 BACKUP SYSTEM syntax . . . 393
11-6 RESTORE SYSTEM syntax . . . 395
11-7 Incremental FlashCopy with single version. . . . 396
11-8 Incremental FlashCopy with two versions. . . . 397
11-9 SORTKEYS syntax for LOAD . . . 399
11-10 DECFLOAT ROUNDMODE syntax for LOAD and UNLOAD . . . 399
11-11 Histogram statistics . . . 401
11-12 Histogram syntax for RUNSTATS. . . . 402
11-13 SCOPE syntax . . . 403
11-14 Modify recovery syntax . . . 405
11-15 SYSCOPY entries ahead of MODIFY . . . 406
11-16 MODIFY RECOVERY with RETAIN LIMIT and GDG . . . 406
11-17 Job output from MODIFY . . . 406
11-18 SYSCOPY entries after of MODIFY . . . 407
11-19 Online CHECK DATA . . . 408
11-20 Online CHECK LOB . . . 409
11-21 DB2I utilities panel . . . 410
11-22 CHECK INDEX with parallel index checking. . . . 413
11-23 LOAD from reference variables . . . 414
11-24 Unload using reference variables . . . 414
11-25 Claimers shown by REORG . . . 415
12-1 DB2 Versions . . . 418
12-2 z/OS versions . . . 419
12-3 DB2 catalog evolution: V9 counts do not include objects for XML repository . . . 422
12-4 Base engine and optional no charge features . . . 426
12-5 The Management Clients Package . . . 428

12-6 The Accessories Suite 429
12-7 Optional chargeable function 431
12-8 DSNX930 message when trying to start DSNSPAS address space. 439
12-9 Main panel DSNTIPA1 442
12-10 SMS-managed install data sets 443
12-11 WORKFILE database panel 445
12-12 DSNTWFG and description of its parameters. 446
12-13 Main panel DSNTIPA1 450
12-14 Panel DSNTIPA2 450
12-15 If you specity NO for field 3 USE SMS of panel DSNTIAP2 451
12-16 If you specified YES for field 3 USE SMS of panel DSNTIAP2. 452
12-17 DSNTIPH 453
12-18 DSNTIPT. 453
12-19 DSNTIPU 454
12-20 DSNTIPW 454
12-21 DSNTIPD 455
12-22 DNTIP7 455
12-23 DSNTIP9. 456
12-24 DSNTIP9 with 32 KB table spaces 456
12-25 DSNTIP 9 with Max Temp 457
12-26 DSNTIPE 457
12-27 DSNTIP1. 458
12-28 DSNTIP2. 458
12-29 DSNTIPN 459
12-30 DSNTIPO 459
12-31 DSNTIPF. 460
12-32 DSNTIP4. 460
12-33 DSNTIP8. 461
12-34 DSNTIPI 461
12-35 DSNTIPJ. 462
12-36 DSNTIP6. 462
12-37 DSNTIPP 463
12-38 DSNTIPM 463
12-39 DSNTIPL. 464
12-40 DSNTIPA 464
12-41 DSNTIPS 465
12-42 DSNTIPR 465
12-43 DSNTIP5. 466
12-44 DSNTIPX 466
12-45 DSNTIPZ. 467
12-46 DSNTIPY 467
12-47 DSNTIPC 468
12-48 Output of CLIST calculations 468
12-49 DSNTIPA1 469
12-50 Main panel for INSTALL 470
12-51 DSNTIPC1 470
12-52 Typical output of the install CLIST in INSTALL mode. 471
12-53 Main panel for ENFM 472
12-54 The only field you can change here is field 2 472
12-55 Enable New Function Mode 473
12-56 CLIST calculations 473
12-57 Typical output of install CLIST in ENFM 474
12-58 DB2 V9 migration flow 477

12-59 The allowed flows from one mode to another 478
13-1 V8 best practice performance plan example. 494
13-2 DB2 9 z9 and z990 performance plan example 495
13-3 Index with multiple increasing sequential insert patterns 496
13-4 Index after pages 1 and 2 have split using the 50/50 split algorithm 497
13-5 Index after pages 1 and 2 have split asymmetrically 498
13-6 Resource Limit Middleware Table DSNRLMTnn and Unique Index DSNMRLnn . . . 500
13-7 Example of histogram statistics benefits when gaps exist in ranges 507
13-8 RUNSTATS syntax changes 1 507
13-9 RUNSTATS syntax changes 2 508
13-10 A sample of lab measurements 508
13-11 EXPLAIN - Use of virtual table in noncorrelated subquery 514
13-12 How does sparse index work? 518
13-13 Star Join Parallel Filtering of RID Lists 520
13-14 Star Join - Intersect RID List, Access Fact Table, and Join Back 521
13-15 Star Join with Dynamic Index ANDing Scheme 522
13-16 Increasing LOB usage 523
13-17 Progressive Reference Return of LOB Data 524
13-18 Progressive Streaming (1 of 3) 525
13-19 Progressive Streaming (2 of 3) 526
13-20 Progressive Streaming (3 of 3) 526
13-21 Performance of LOB Progressive Streaming 527
13-22 LOB overview 528
13-23 LOB operations and LOB Locks 529
13-24 Buffer Pool Management by WLM 531
13-25 Performance scalability enhancements 532
13-26 Other performance improvements 534

Archive

Archived

Examples

4-1 Output of REPORT TABLESPACESET DB1.TS1 SHOWDSNS 95
4-2 Source column referenced and equally named column added afterwards 110
6-1 BIGINT examples 132
6-2 Example of INSTEAD OF trigger 137
6-3 MERGE example 141
6-4 SELECT FROM MERGE 144
6-5 Output of SELECT FROM MERGE statement 145
7-1 Host variable definitions for LOB locators in COBOL 197
7-2 What the DB2 precompiler makes of LOB locators. 197
7-3 Host variable definition for BLOB file reference variable in COBOL 199
7-4 What the DB2 precompiler makes of BLOB file reference variable 200
7-5 Empty compound statement 247
7-6 SQLFORMAT parameter 254
8-1 List all addresses where element CITY contains text TORONTO 296
8-2 List all ASSISTANT phone numbers for TYPE HOME 296
8-3 List all ASSISTANT information for those who have phone TYPE HOME 297
8-4 List ADDR information 297
8-5 Arithmetic expression example 298
8-6 Comparison expression example - true/false 299
8-7 Example of comparison expression 299
8-8 Logical comparison 299
8-9 Sample XMLQUERY statement I 301
8-10 XMLQUERY statement sample with XMLSERIALIZE 302
8-11 Non-empty sequences returned by XMLQUERY 302
8-12 Empty sequence returned by XMLQUERY 303
8-13 XMLQUERY statement for XML docs with namespace 304
8-14 XMLEXISTS sample 305
8-15 vn:abs example 306
8-16 fn:compare values of attribute country to literal 'Germany' 307
8-17 fn:concat function example 307
8-18 fn:contains function 307
8-19 fn:count function 308
8-20 fn:data example for an atomic value 309
8-21 fn:data function example for a node 309
8-22 XPath on a node as comparison to fn:data 309
8-23 fn:round function 310
8-24 fn:string function 310
8-25 vn:substring function with cardinality problem 311
8-26 fn:substring function with solved cardinality problem 311
8-27 fn:sum function 312
8-28 Numeric index 315
8-29 XML index example 316
8-30 XML index not allowing insert 316
8-31 Insert character that is larger than Unicode codepoint U+007F 317
8-32 CREATE INDEX on column that already contains XML documents. 317
8-33 XPath query example for indexing 1 319
8-34 XML index usage example 1. 319
8-35 XPath query example for indexing 2 320

© Copyright IBM Corp. 2007. All rights reserved.

8-36 XML index usage example 2. 320
8-37 XPath query example for indexing 3 320
8-38 XML index usage example 3. 320
8-39 XPath query example for indexing 4 320
8-40 XML index usage example 4. 320
8-41 Schema validation. 323
8-42 REPORT TABLESPACESET sample output 324
8-43 CHECK INDEX sample utility output. 325
8-44 -DIS DB(DSN00003) output 326
8-45 COPY base and XML table. 326
8-46 Page sets after RECOVER base table 327
8-47 SELECT from XML column after RECOVER base table space 327
8-48 ISPF 3.4 after recovering base and XML table space 328
8-49 Attempt to select from SABIXML5 column INFO 328
8-50 -DIS DB(..) after recovering base and XML table space. 328
8-51 ISPF 3.4 after REBUILD INDEX(ALL) TABLESPACE DSN00003.SABIXML5. 329
8-52 Complete list of page sets associated to SABIXML5 329
8-53 Copy all table spaces related to SABIXML5 330
8-54 Recover all objects related to SABIXML5 330
8-55 Job output for LISTDEF RECOVER 331
8-56 QUIESCE TABLESPACESET job output 332
8-57 UNLOAD DATA without specifying UNICODE - SYSREC 334
8-58 UNLOAD DATA with specifying Unicode - SYSREC 334
11-1 DSNU116I message 398
11-2 Use of keyword LIMIT in TEMPLATE 404
11-3 New message from DSN1LOGP 407
11-4 Utility message with day and time stamp highlighted 411
11-5 Checkpoint queue output 412

Archive

Tables

2-1 PTFs needed to support zIIPs in RMF 28
3-1 LOG column values 53
3-2 LOAD and REORG parameters 56
4-1 Maximum value for MAXPARTITIONS given the page size and DSSIZE 75
4-2 SEGSIZE and MAXPARTITIONS keyword combinations. 76
4-3 SYSIBM.SYSTABLES 83
4-4 SYSIBM.SYSTABLESPACE. 83
4-5 SYSIBM.SYSCOPY 83
4-6 SYSIBM.SYSINDEXES 83
4-7 SYSIBM.SYSTABLES 84
4-8 SYSIBM.SYSTABLESPACE. 84
4-9 SYSIBM.SYSCOPY 84
4-10 SYSIBM.SYSINDEXES 84
6-1 STOCK table before SELECT FROM MERGE statement 144
6-2 Host variable arrays of stock activity. 144
6-3 STOCK table after SELECT FROM MERGE statement. 145
6-4 Example of UNION, EXCEPT, and INTERSECT on result tables R1 and R2 154
6-5 Specifying the type of time interval 161
6-6 Valid input value ranges for string-expression and equivalent Datetime formats 161
6-7 Format string values 164
7-1 Characteristics of an implicitly created database 168
7-2 Maximum record size calculation for different buffer pools. 172
7-3 SYSIBM.SYSKEY for index upper_empname 176
7-4 SYSIBM.SYSKEYTARGETS for index upper_name 176
7-5 SYSIBM.SYSKEYS for index total_compensation 177
7-6 SYSIBM.SYSKEYTARGETS for index total_compensation 177
7-7 Determining the length attribute 188
7-8 Maximum number of LOB columns by partitions 195
7-9 DB2-generated construct 198
7-10 File option constants 200
7-11 SYSIBM.SYSENVIRONMENT catalog table 236
7-12 Summary of name scoping 246
8-1 Performance involving XML stores 270
8-2 Well-formed XML document examples 271
8-3 XML - Base table and auxiliary table spaces 272
8-4 Sample table SHOWXMLTEXT 279
8-5 Supported axes in XPath 290
8-6 Abbreviated syntax for path expressions 291
8-7 Path expression examples 292
8-8 Comparison of unabbreviated and abbreviated path expression syntax 293
8-9 Supported kind tests in DB2 XPath. 294
8-10 New access methods 319
8-11 CHECK DATA error keywords 325
12-1 Differences in installation jobs 475
13-1 Reasons for COST_CATEGORY B 502
13-2 Qualifying rows for DSNRLMTxx 503
13-3 Frequency statistics for YRS_OF_EXPERIENCE column in EMPLOYEE table. . . . 505
13-4 Histogram statistics for YRS_OF_EXPERIENCE column in EMPLOYEE table 506

© Copyright IBM Corp. 2007. All rights reserved.

Archived

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

© Copyright IBM Corp. 2007. All rights reserved.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Redbooks (logo) ®
eServer™
iSeries®
i5/OS®
pureXML™
z/Architecture®
z/OS®
zSeries®
z9™
AIX®
Collation®
CICS®
Database 2™
Distributed Relational Database Architecture™
DB2 Connect™
DB2 Universal Database™
DB2®
DFSMS™
DFSMSdss™
DFSMShsm™
DRDA®
DS8000™
Enterprise Storage Server®
ESCON®
FlashCopy®
FICON®
Geographically Dispersed Parallel Sysplex™
GDPS®
Informix®
IBM®
IMS™
Language Environment®
MQSeries®
MVS™
Net.Data®
OMEGAMON®
OS/390®
Parallel Sysplex®
POWER™
PR/SM™
QMF™
Redbooks®
RACF®
RETAIN®
REXX™
RMF™
System z™
System z9™
System Storage™
System/390®
Tivoli®
VTAM®
WebSphere®

The following terms are trademarks of other companies:

SAP NetWeaver, SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

EJB, Java, Java Naming and Directory Interface, JDBC, JVM, J2EE, Solaris, Streamline, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Expression, Visual Studio, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Archived

Preface

IBM® DATABASE 2 Version 9.1 for z/OS® (DB2® 9 for z/OS throughout this publication) is the thirteenth release of DB2 for MVS™. It brings improved synergy with the System z™ hardware and more opportunities to drive business value in the following areas:

- Business insight innovations
  - Rich hybrid support for relational and pureXML™ data
  - Enhanced support for key business partners that allow you to get more from your data in critical business disciplines like ERP
  - A brand new user interface for IBM DB2 Query Management Facility (QMF™) to let you view and analyze data from either a workstation client or an ordinary Web browser
  - Also announced is IBM DataQuant V1.1, a new business analytics tool for z/OS and for Linux®, UNIX®, and Windows® designed to assist analysts in leveraging new business intelligence from their data
- Cost savings through optimized innovations
  - Streamline™ security and regulatory compliance through the implementation of roles, network-trusted contexts, and enhanced auditing
  - Performance-boosting innovations such as pervasive CPU reduction in index management for most utilities, improved performance for varying length data, and improved logging and insert performance
  - Query management enhancements to make accessing data even faster and more accurate with indexing improvements
- Business resiliency innovations
  - Database on demand capabilities to ensure that information design can be changed dynamically, often without database outages
  - DB2 recovery improvements enhancing performance usability and availability in a data sharing environment

The DB2 9 environment is available either for brand new installations of DB2, or for migrations exclusively from DB2 UDB for z/OS Version 8 subsystems.

This IBM Redbooks® publication introduces the enhancements made available with DB2 9 for z/OS. It is meant to help DB2 for z/OS users understand the functions offered by DB2 9, justify their investment in installing or migrating, and start planning for exploiting the key new capabilities.

The team that wrote this IBM Redbooks publication

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, San Jose Center.

Paolo Bruni is a DB2 Information Management Project Leader at the International Technical Support Organization, San Jose Center. He has authored several Redbooks about DB2 for z/OS and related tools, and has conducted workshops and seminars worldwide. During Paolo's many years with IBM, in development, and in the field, his work has been mostly related to database systems.

© Copyright IBM Corp. 2007. All rights reserved.

Roy Cornford is a DB2 Specialist based in Hursley, UK, working within Europe for the IBM Silicon Valley Laboratory. He worked in California and Denmark as a developer of DB2 Utilities and the DB2 Administration Tool between 1998 and 2004. Prior to that he managed the Introduction Program for DB2 z/OS in Europe and was heavily involved in the early data sharing implementations. His first contact with DB2 was 16 years ago as a Systems Programmer for the IBM IT department in the UK. He holds a Bachelor of Science degree in Chemistry.

Rafael Garcia has been in the IT business for 24 years and has held various positions. He was a COBOL and CICS® Developer, an Application Development Manager, and a DB2 Applications DBA for one of the top 10 banks in the US. For the last 10 years he has been a field DB2 Technical Specialist working for the IBM Silicon Valley Laboratory supporting DB2 for z/OS customers across various industries, including migrations to data sharing. He has an Associate's Degree in Arts and an Associate's Degree in Science in Business Data Processing from Miami-Dade Community College.

Sabine Kaschta is a DB2 Specialist working for IBM Global Learning Services in Germany as an education consultant. She has 14 years of experience working with DB2. Before joining IBM in 1998, she worked for a third-party vendor providing second-level support for DB2 utilities. She is experienced in DB2 system programming and client/server implementations within the insurance industry in Germany. She is also a co-author of the IBM Redbooks *DB2 UDB for OS/390 and Continuous Availability*, SG24-54866; *Cross-Platform DB2 Distributed Stored Procedures: Building and Debugging*, SG24-5485-01; and *DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ... and More*, SG24-6079.

Ravi Kumar is a Senior Instructor and Specialist for DB2 with IBM Software Group, Australia. He has over 20 years of experience in DB2. He was on assignment at the International Technical Support Organization, San Jose Center, as a Data Management Specialist from 1994 to 1997. He is currently on virtual assignment as a Course Developer in the Education Planning and Development team, Information Management, IBM Software Group, USA.

A photo of the team is in Figure 1.

Figure 1 Left to right: Sabine, Ravi, Roy, Rafael, and Paolo

Thanks to the following people for their contributions to this project:

Rich Conway
Bob Haimowitz
Emma Jacobs
Sangam Racherla
International Technical Support Organization

Adrian Lobo
Akiko Hoshikawa
Akira Shibamiya
Allen Yang
Andrei Lurie
Basuki Soetarman
Bill Bireley
Bill Franklin
Bilung Lee
Binghua Zhen
Bob Lyle
Bruce McAlister
Bryan Smith
Cathy Win
Chan-Hua Liu
Ching Lee
Chris Crone
Chris S Leung
Chung Wu
Claire McFeely
Craig Friske
Dan Weis
Dave Levish
Dongmei Ren
Don Smith
Dorothy Lin
Eric Bateman
Frances Villafuerte
Frank Butt
Fung Lee
Gayathiri Chandran
Gene Fuh
Gopal Krishnan
Grace Tzeng
Guogen Zhang
Haakon Roberts
Hsiuying Cheng
Hugh Smith
Irene Liu
Jack Shedden
James Guo
James Pickel
James Teng
Jay Bruce
Jay Yothers
Jeff Josten
Jeffrey Berger
Jerry Kenyon

Archived

Jerry Mukai
Jim Reed
Jim Ruddy
John B Tobler
John Campbell
John Garth
John Lyle
John Tobler
Julie Watts
Keith Howell
Kevin S Liu
Koshy John
Laura Kunioka-Weis
Lee Lin
Li-Mey Lee
Marichu Scanlon
Marko Dimitrijevic
Maryela Weihrauch
Maureen Zoric
Meg Bernal
Mengchu Cai
Michael Shadduck
Michelle Zheng
Michelle Guo
Moira Casey
Patrick D Bossman
Paul Ostler
Paul Webb
Peggy Zagelow
Qing-ying Li
Roger Miller
Ron Parrish
Ruiping Li
Shahneela Rahman
Sherry Guo
Sherry Yang
Shirley L Zhou
Shu-Ling Win
Steve Turnbaugh
Tammie Dang
Terence P Purcell
Terrie Jacopi
Terry Allen
Thomas Beavin
Thomas Majithia
Todd Munk
Tom Miller
Wendy Koontz
Xun Li
Yoichi Tsuji
Ying-lin Chen
Yumi Tsuji
IBM Silicon Valley Lab

Archived

Satheesh Bandaram
Oakland, IBM

Rick Butler
Bank of Montreal (BMO) Financial Group, Toronto

Bart Steegmans
Belgium, IBM

Mark Wilson
Dallas, IBM

Ian Cook
Gareth Jones
IBM UK

Oliver Harm
Christian Michel
Dirk Nakott
Timm Zimmermann
IBM Boeblingen Lab

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbooks publication dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Obtain more information about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review IBM Redbooks publication form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099

Archived

2455 South Road
Poughkeepsie, NY 12601-5400

Archived

Summary of changes

This section describes the technical changes made in this edition of the book and in previous editions. This edition may also include minor corrections and editorial changes that are not identified.

Summary of Changes
for SG24-7330-00
for DB2 9 for z/OS Technical Overview
as updated on March 30, 2010.

June 2007, First Edition

The revision of this First Edition, first published on June 28, 2007, reflects the changes and additions described below.

July 2007, First Update

Several typographical errors have been corrected. The other changes and additions are marked with change bars.

Changed information

- Changed some text in section 2.2.1, “Sequential I/O” on page 4 to help the added information described below.
- Corrected Table 4-8 on page 84.
- Minor corrections in section 6.3.1, “EXPLAIN for MERGE” on page 141 and 6.4.1, “Selecting values as you merge: SELECT FROM MERGE” on page 143.
- Corrected Figure 8-4 on page 273.

Added information

- Added List of Tables and List of Examples.
- Added DS8000 Adaptive Multi-stream Prefetching (AMP) information to section 2.2.1, “Sequential I/O” on page 4.
- Added Figure 2-4 on page 8 to show AMP measurements.

November 2007, Second Update

Changes and additions are marked with change bars.

Changed information

- Removed the mention of XML in “Deallocation of partitions” on page 78.
- Deleted the reference to index controlled partitioning in 4.1.2, “Range-partitioned universal table spaces” on page 80.
- Changed text in 4.2.4, “Dropping a clone table” on page 91.
- Corrected Figure 4-19 on page 99 on clone tables and changed some text in the description.

© Copyright IBM Corp. 2007. All rights reserved.

Added information

- Added a description of Option 9, Statistics Clustering, of panel DSNTIP6 shown in Figure 11-4 on page 392.

July 2008, Third Update

Changed information

- Corrected text at “COPY utility” on page 79 to reflect that when you run a SHRLEVEL(CHANGE) copy and the table space is expanded to one more partition while the copy runs, this new partition will also be copied.

Added information

- Added text under **13.13, “WLM-assisted buffer pool management” on page 530** to include a reference to z/OS APAR OA18461.
- Updated or added information at **“Other publications” on page 547**.

March 2009, Fourth Update

Changed information

- Corrected module names at 7.20.1, “Setting up the Unified Debugger components” on page 257.
- STUSED column is used also for static queries in 5.2, “Index improvements” page 122.
- Added an Note box at 12.5.1, “Summary of changes” on page 441 to state that the term *conversion mode* has recently replaced *compatibility mode*.

April 2009, Fifth Update

Changed information

- Corrected log archive data sets to log archive volumes in 12.6.3, “BSDS conversion required before V9 migration” on page 480.
- Removed the previous section 13.8 “Optimization of complex query”

March 2010, Sixth Update

Changes and additions are marked with change bars.

Changed information

- Removed the reference to access path when discussing option 9 of DSNTIP6 under Figure 11-4 on page 392.

Archived

1

DB2 9 for z/OS at a glance

In this introductory chapter we briefly discuss the main new topics of DB2 9 for z/OS.

First of all, the official name of the thirteenth version of DB2 for z/OS is DB2 Version 9.1 for z/OS, program number 5635-DB2. Notice that UDB is no longer in the name, and 9.1 indicates version and release, as for the other platforms. The abbreviated approved name is *DB2 9 for z/OS*. However, you will find the old fashioned V9 often in this book, and V9.1 in the new DB2 manuals. The introduction program announcement was on May 2, 2006, with the program beginning June 9, 2006. The General Availability date was March 16, 2007, and details on contents and prerequisites are included in the USA Announcement letter 270-41, dated March 6, 2007.

Archived

© Copyright IBM Corp. 2007. All rights reserved.

1.1 The main goals of DB2 9 for z/OS

The vision for DB2 9 for z/OS is to enhance DB2's ability to handle new and enterprise applications.

DB2 9 for z/OS (V9) enhances DB2's ability to handle new and enterprise applications. V9 improves with XML, large objects, and many SQL and security improvements. V9 also builds upon and extends DB2 traditional strengths and the ground-breaking Version 8 in many areas — online schema evolution, Unicode, XML, DB2 family SQL, utilities, security, and 64-bit virtual storage.

V9 enhances DB2's ability to handle new and enterprise applications. V9 improves with XML, large objects, and many SQL and security improvements. V9 builds upon and extends DB2 traditional strengths and the ground-breaking V8 in many areas: online schema evolution, Unicode, XML, DB2 family SQL, utilities, security, and 64-bit virtual storage.

These are primary areas for V9, carrying on some of the key deliveries from Version 8. Migration to V9 is only from DB2 for z/OS Version 8 and requires z/OS 1.7 Some items require z/OS 1.8.

Data definition on demand extends the theme of online schema revolution from V8. Additional Unicode enhancements continue the work from V7 and V8. XML work across the DB2 family is a much larger step than in V7 or V8.

While V7 and V8 removed many differences in SQL functions from DB2 for Linux, UNIX, and Windows, V9 takes the next big step to improved SQL consistency. SQL Procedures also become more consistent across the family. Utility enhancements help with new function, more LOB and XML support, better performance, and improved availability, removing the BUILD2 step from online REORG.

Additional Unicode and text enhancements continue the work from V7 and V8.

For general pointers to current information on DB2 9, go to:

http://www.ibm.com/software/data/db2/zos/db2zosv91.html

1.2 The innovations

SQL family consistency, pureXML, and synergy with the platform are key innovations. Data definition on demand extends the theme of online schema revolution from V8. Utility enhancements help with new function, new XML and more LOB support, better performance, and improved availability, removing the BUILD2 step from online REORG.

The innovations are grouped into the following four sets:

- Enable high-volume transaction processing for the next wave of Web applications.
- Extend the lead in transaction processing availability, scalability, and performance.
- Reduce cost of ownership and System z specific skill needs.
- Improve warehousing functions.

Archived

1.2.1 Enable high-volume transaction processing for next wave of Web applications

Insight for the business is provided by:

- Rich new hybrid data server support for both relational and pureXML storage, with the necessary services to support both data structures.
- Rich new SQL capabilities, including additional data types and built-in functions, expanded SQL commonality with DB2 family, and enriched text handling with functions like caseless comparisons, cultural sort, and index on expression.
- Enhanced support to allow key business partners, especially with ERP offerings like SAP®, to continue to push the envelope with capabilities that drive business value to their customers. Specific items include new autonomic features, optimization and availability enhancements, and support for new data types.

XML capabilities inside the engine

The XML support includes native storage and the usage of schema, indexes, functions, and utilities. It also offers powerful querying and transformation capabilities. See Figure 1-1.

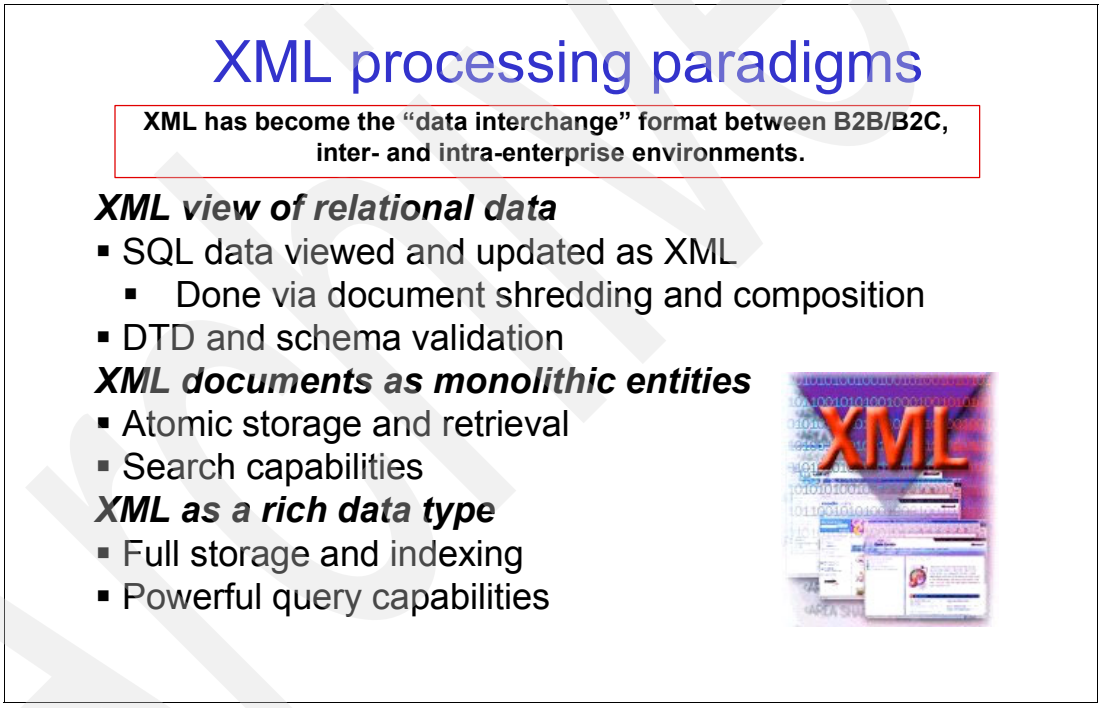


Figure 1-1 XML processing

XML is an important innovation for several reasons: It enables business-to-business communication, making it easy to send semi-structured data across the Web so that nothing gets lost in translation, masking differences in end-point infrastructures. It provides independence from the presentation layer — information can be rendered appropriately for a variety of devices, because XML separates the document content from the document presentation. In this regard, it differs significantly from HTML.

It enables the use of smart agents, because XML includes a description of the data and how it relates to other pieces of data the agent may already know, enabling it to take smarter actions. It enables the use of smart searches, because XML provides a context for search

arguments, for example, the name *chip* rather than the food chip. From the standpoint of data management, there are three distinct paradigms for XML usage and support:

- XML view of relational data: This is for users who merely want to XML-ize their relational data. They want to receive XML documents and shred them into distinct relational tables. Also, they want to send XML documents generated out of their relational tables. There is really no trace of XML inside the data management storage. XML is used as a transient on-the-wire format.
- XML documents as monolithic entities: This is for users who want to store XML documents as-is and retrieve them as a whole. They may also want some form of rudimentary querying capabilities on XML.
- XML as a rich data type: This is for users who want complete support from XML in a data management system. This implies efficient storage and indexing.

The XML extender was the first one in the industry to support XML. DB2 for z/OS extensibility has been implemented via extenders. Extenders for text, image, audio, video, and XML are delivered with DB2 and are optional installation components. They provide the tooling, user defined data types (UDT), user defined functions (UDF), and stored procedures to manage non-relational data. The XML extender provides for the storage and indexing of an XML document as a character large object (CLOB), or for the shredding of the XML document into relational columns for storage and query.

DB2 Version 8 expanded on XML support by implementing several XML publishing operations as built-in DB2 functions. This allows you to perform XML document composition from relational data with improved performance without the XML Extender.

There is expanded support of XML in DB2 by integrating more features into the engine. A wide range of new function has been provided inside the DB2 for z/OS engine. This includes an XML data type, native storage of XML documents, integration of the XPath language, and catalog extensions to support definitions of XML schemas. Utilities will support creation and maintenance of XML data. This work is being done in parallel with similar changes in DB2 for Linux, UNIX, and Windows. DB2 for Linux, UNIX, and Windows delivers this picture and adds an XQuery interface to the data. This is called pureXML support (see Figure 1-2).

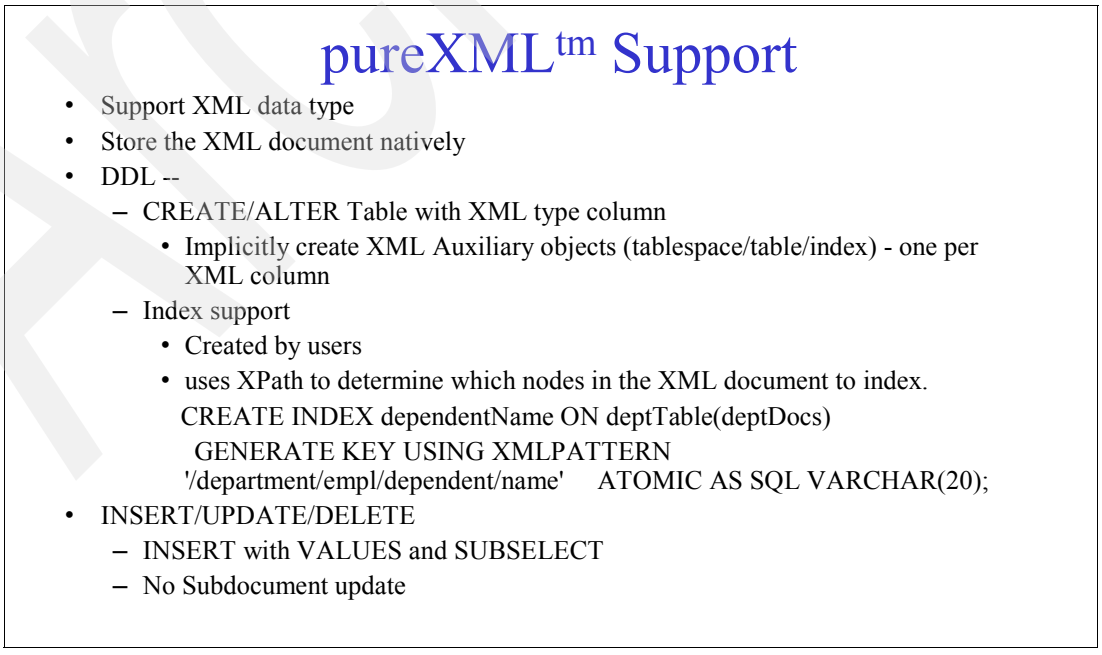


Figure 1-2 pureXML

The pureXML implementation on DB2 for z/OS allows handling XML documents by storing them in binary encoded format. New indexing technology allows for efficient search of XML. New query syntax with XPath expressions can be used to perform the searches. Schema validation of the XML document can be performed as you insert the data. Such support information allows the DB2 server to participate in XML-oriented applications and be a key component of the SOA solutions.

XQuery is not currently supported. This is not a major limitation since XPath is available, and SQL is likely to see higher use in most XML applications since XQuery does not support things like parameter markers.

SQL: productivity, DB2 family, and porting

In V8, there are many improvements for SQL. These improvements in the SQL have made migrating from other platforms, such as UNIX, LINUX, and Windows, much easier.

DB2 9 for z/OS continues the progress in SQL, with many new functions, statements, and clauses. We have seen the biggest change in pureXML. There are new SQL data manipulation statements in MERGE and TRUNCATE. There are new data types with DECIMAL FLOAT, BIGINT, and VARBINARY types. Improvements in LOBs provide more consistent handling and improved performance. Security is improved with ROLEs and trusted context. Data definition consistency and usability are improved. DB2 9 for z/OS is another leap in DB2 family consistency and in the ability to port applications to DB2 for z/OS.

Figure 1-3 highlights the major additions.

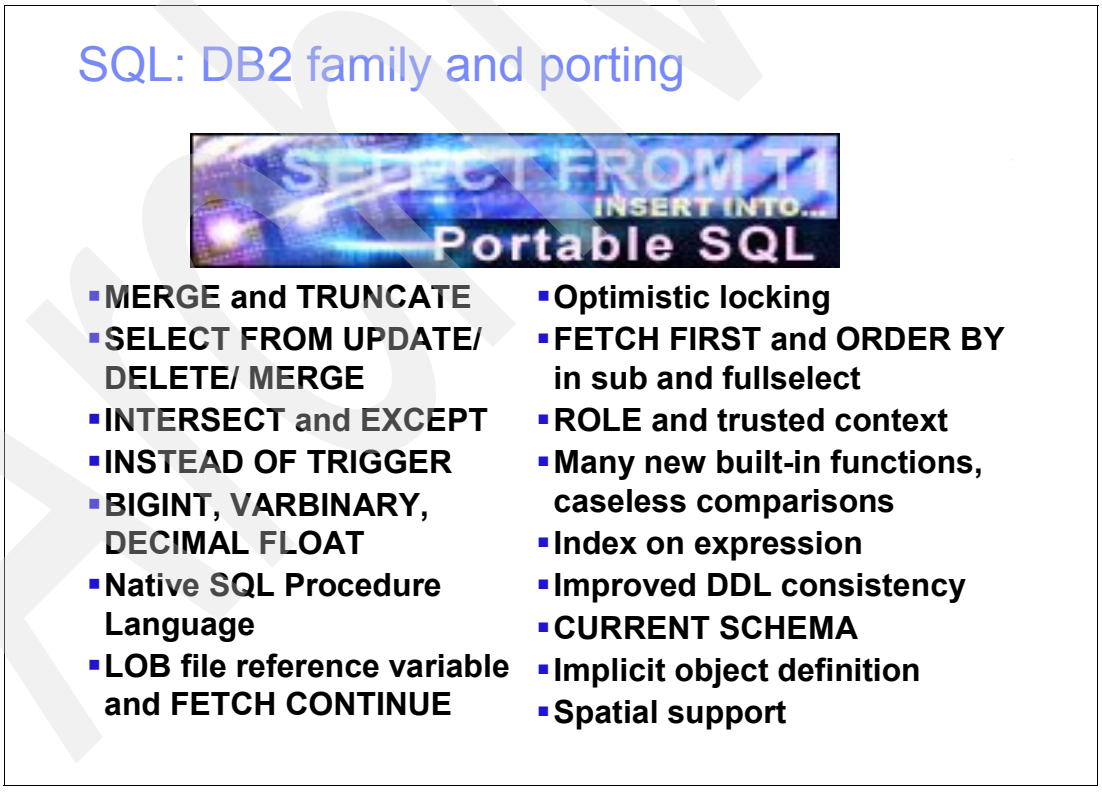


Figure 1-3 Major SQL enhancements

New SQL statements

There are new SQL data manipulation statements in MERGE and TRUNCATE.

Online transaction processing (OLTP) workloads that need to add or replace data for several rows into a table will benefit from MERGE SQL. This capability inserts a record to a table in a database if the record does not exist. If the record already exists, an update operation is performed. MERGE is an SQL standard. The same function is sometimes referred to as UPSERT.

Multi-row insert-type capability is extended, via the MERGE statement, to take an array of column values from a program and perform insert and update operations against a table. DB2 will use the matching criteria in the merge SQL to update existing rows and perform inserts for rows that do not exist, through a single SQL statement.

TRUNCATE provides a fast way to delete rows with SQL, with better application portability. Truncate Table provides for the rapid removal of rows from a table. You can use this function to delete the contents of a table before applying data from a DSN1COPY or INSERT.

The TRUNCATE TABLE statement deletes all data rows for either base tables or declared global temporary tables without activating delete triggers defined on the table. DB2 transforms the new TRUNCATE TABLE operation into a mass delete operation. Moreover, it provides a *statement commit* option (IMMEDIATE option) for the statement that allows the truncate operation to become permanent (that is, cannot undo) and immediately makes deallocated space available for new, subsequent inserts in the same unit of work. When the IMMEDIATE option is specified, the table must not contain any uncommitted updates.

Select from DELETE, UPDATE, and MERGE. The object-relational capabilities of DB2 allow for the incorporation of business logic into the database. This extends the power of SQL. Sometimes the application needs to know the results of this logic when applied to the SQL issued. A subsequent SELECT for the data adds complexity and execution time to the application.

The insert within select feature of DB2 for z/OS Version 8 has been expanded to include the retrieval of columns from rows that are modified via DELETE, UPDATE, and MERGE SQL. One SQL call to DB2 modifies the table contents and returns the resultant changes to the application program.

When used with DELETE, the application now has the option to code a destructive read from a table. This is particularly useful when a table is used as a data queue, as with many vendor packages.

INTERSECT and EXCEPT set operations make some SQL operations simpler to specify.

The INSTEAD OF trigger is an SQL technique that allows a trigger to be used in place of a view, consistent with DB2 for LUW. Since we have no way to influence read access on the base table, our customers are forced to build awareness into their applications that read access goes against one object (for example, the view) and write access goes against another (for example, the base table). This greatly increases the amount of work to introduce new logic into the system and also leaves the integrity of the system dependent on the logic within the applications. The primary reason for looking at INSTEAD OF triggers is that they provide a mechanism to unify the target for all read/write access by an application while permitting separate and distinct actions to be taken for the individual read and write actions.

New data types

These are:

- There are new data types with DECIMAL FLOAT, BIGINT, BINARY, and VARBINARY types. Decimal floating point numbers, similar to calculator mathematics and supporting the IEEE standard.
- BIGINT support of double word (8 byte) integer values.
- VARBINARY, providing better comparison support for binary strings.

LOBs

Improvements in LOBs provide more consistent handling and improved performance. LOB objects can be implicitly defined.

Security

Security is improved with ROLEs and network trusted context. Data definition consistency and usability are improved. V9 is another big step in DB2 family consistency and in the ability to port applications to DB2 for z/OS.

Native SQL stored procedures

Stored procedures written in SQL procedure language enhance portability and ease of use when using DB2 for z/OS as your enterprise information source. This language is an ANSI standard language. It is similar to the proprietary stored procedure languages of several competitive databases, which assists in migrating and porting to DB2 for z/OS.

SQL stored procedures are supported by the DB2 Developer Workbench tooling, providing an environment to code, test, and debug modules from your connected workstation. This language is currently converted to C when the CREATE PROCEDURE statement is executed. The C program is then automatically prepared, compiled, linked, and bound. The developer does not need to work with the C code.

SQL stored procedures code is natively integrated into the DB2 engine, eliminating the conversion to C and the need for C libraries. Additionally, extensions to the BIND command allow for the promotion of the program and access paths between environments without needing to recreate the stored procedure.

When native stored procedure requests are invoked from DRDA® TCP/IP connections, the processing within the native stored procedure is eligible for zIIP engine processing.

Subquery improvements

Correlated and non-correlated subqueries will benefit from improved optimization. They will provide added flexibility with the support of ORDER BY and FETCH FIRST clauses.

Optimistic locking

This change helps with migration of applications that use application consistency techniques.

This function includes the following parts:

- A row change timestamp column: a new GENERATED column that is defined as GENERATED ALWAYS FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP or GENERATED BY DEFAULT FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP. It is used to store the time stamp value that reflects the time when the row was last changed.
- RID: a new built-in function that returns the record identifier (RID) of a row.

Archived

- ROW CHANGE TOKEN FOR table designator: a new expression that returns a token in BIGINT that represents when the row in table-designator was last changed.
- ROW CHANGE TIMESTAMP FOR table designator: a new expression that returns a time stamp value that represents the time when the row in table-designator was last changed.

This provides an easier and more efficient approach for detecting a change of a row. An application does not need to know all of the old values that are marked for update. The new expression, ROW CHANGE TOKEN, is introduced to return a token that represents a relative point in the modification sequence of a row. An application can compare the current ROW CHANGE TOKEN value of a row with the ROW CHANGE TOKEN value that was stored when the row was last fetched to determine whether the row has been updated.

Default databases and table spaces

Automatic unique indexes to support defined primary keys and the default for a table space will change to a segmented structure partition-by-growth table space, improving performance and manageability for many customers.

Spatial support

IBM Spatial Support for DB2 for z/OS contains a set of spatial data types, user-defined functions, and stored procedures for spatial-related queries. These spatial queries can be invoked for local and remote clients to answer questions based on geographic relationships. Relationships such as contains, crosses, equals, intersects, overlaps, touches, within, distance, and envelopeintersect can be used in a WHERE clause of the SELECT statement by using predicate functions. Spatial support provides this capability to create a spatial index on spatial columns.

The spatial index can improve query performance when using spatial predicate functions. The stored procedure interfaces allow the user to manage geographic coordinate systems, spatial indexes, and spatial column usages. Spatial support includes an ODBC program to enable and disable the spatial feature. This program can invoke each of the stored procedures through a set of command line arguments.

Spatial is a function included in the DB2 for z/OS Accessories Suite, together with the Optimization Service Center and the International Components for Unicode.

Archived

1.2.2 Extend the lead in transaction processing availability, scalability, and performance

One of the key initiatives of V8 was online schema evolution, and that theme is expanding and changing to be data definition on demand. The improvements continue with V9. See Figure 1-4.

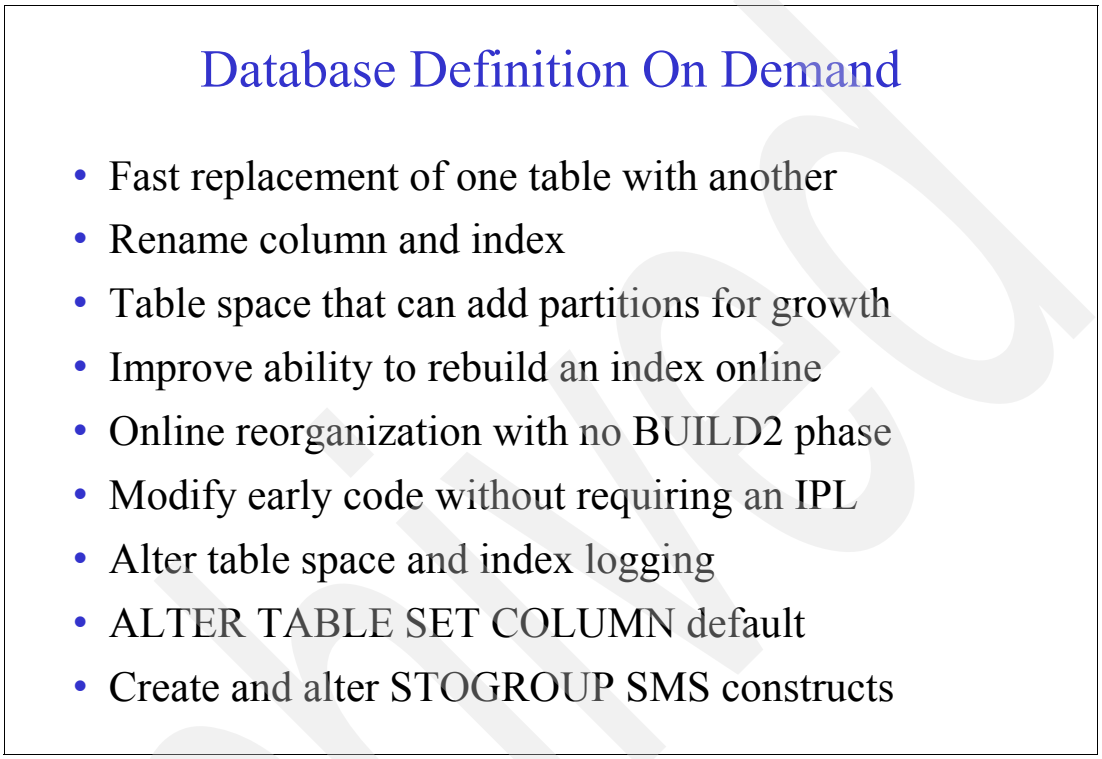


Figure 1-4 Schema evolution

One important change with V9 is to be able to replace one table quickly with another (CLONE table). This function allows you to generate a copy of a current table with the same attributes and same data, in the same table space. It appears to an application or user as a very fast replacement of data within a table. Web-based applications striving for maximum availability will benefit from the option of implementing a pair of tables that are clones of each others' structure. Copies for application testing and auditing can be easily created. These clone tables will have the unique ability to change names quickly. Applications can quickly and almost transparently switch between dual mirror tables.

The tables can be partitioned or non-partitioned, and are created with the CREATE TABLE syntax. The primary table's structure, including indexes, large objects (LOBs), and before triggers, will be copied. Information can then be inserted or loaded into the copy table, and the copy can have its own image copies. When the data in the copy table needs to become active to an application, an ALTER statement will switch the name, providing fast replacement of the original data.

Another change is to be able to RENAME a column or an index. Index page size can now be altered to larger values. A new type of table space combines the attributes of segmented and partitioned, without a partitioning key. The rebuild index can be run with much less disruption. Online table space reorganization for a few partitions is improved a lot, removing the BUILD2 phase for all types of secondary indexes. Table space and index logging can be altered. A new ability to change the DB2 early code does not require an IPL.

Partitioned tables have required key ranges to determine the target partition for row placement. When a table is partitioned, you gain the benefits of scaling objects to hold more data. You also benefit from more granular locking and parallel operations by spreading the data over more data sets. The key functions of segmented and partitioned table spaces have been combined in the new universal table space (UTS). UTS can be defined with two options:

- The option to partition by growth (PBG) allows segmented tables to be partitioned as they grow, without needing key ranges. These segmented tables will gain increased table space limits and the SQL and utility parallelism afforded for partitioned tables.

```
CREATE TABLESPACE …  (explicit specification)
    MAXPARTITIONS integer
CREATE TABLE …             (implicit specification)
   PARTITIONED BY SIZE EVERY integer G
```

- The option partition by range (PBR) defines the UTS via:

```
CREATE TABLESPACE … SEGSIZE integer NUMPARTS integer
```

Several changes improve the scalability and the performance of massive INSERT applications.

With V8, DB2 acquires a lock on the LOB value while performing the INSERT, DELETE, UPDATE, and SELECT operations. The purpose of the LOB lock is to serialize the LOB table space access and to determine whether the previously de-allocated LOB space can be reallocated. The V9 enhancement eliminates the acquisition of the LOB lock on every LOB operation including the current LOB locks used in the space allocation. The V8 LOB lock is removed for Insert, Delete, Update or Select operations. Furthermore, for a UR reader, the current LOB lock that is required to serialize the consistency between the LOB value and the base row column's value is also removed. DB2 9 for z/OS will not see lock escalation for the LOB locks to the table space level.

The processing of large objects (LOBs), particularly from a distributed perspective, has been optimized for the retrieval of larger amounts of data. Many applications effectively use locators to retrieve LOB data regardless of the size of the data being retrieved. This mechanism incurs a separate network flow to get the length of the data to be returned, so that the requester can determine the proper offset and length for SUBSTR operations on the data to avoid any unnecessary blank padding of the value. For small LOB data, returning the LOB value directly instead of using a locator would be more efficient.

APARs on V8 deliver the ability to use utilities for loading and unloading large LOB data. File reference variables are used to let the large objects be accessed from data sets instead of from storage. The abilities to reorganize and to recover space are provided. In V9 you can use file reference variables in SQL.

Virtual storage constraint relief

With DB2 V8, several DB2 data related memory structures have been moved above the 2 GB bar. The two address spaces that saw these changes were the DBM1 and the IRLM address space.

With V9, additional changes help with virtual storage constraint. See Figure 1-5.

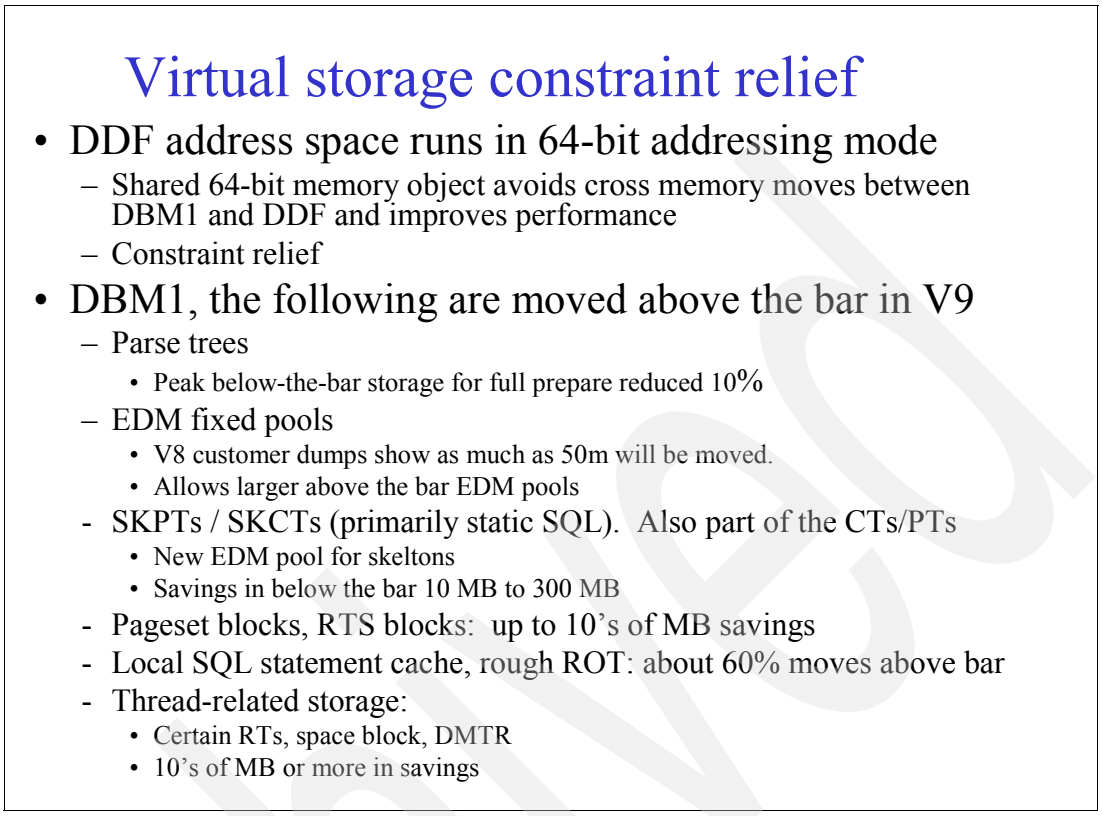


Figure 1-5 Virtual storage constraint relief

The Distributed Data Facility (DDF) is moved above the bar (DB2 DIST address space). This removes restrictions in storage for communications buffers and DRDA intensive usage by vendor applications.

With 64 bit, DB2 DDF uses the z/OS Shared Memory Facility to reduce data moves between DBM1 and DDF. Shared memory is a relatively new type of virtual storage allowing multiple address spaces to easily address storage introduced in z/OS 1.5. Shared memory resides above the 2 GB bar. 64-bit DDF is a performance enhancement for distributed server processing, but it also provides virtual storage constraint relief: no cross memory move is necessary for the shared blocks, and this storage no longer needs to be allocated in the DBM1 address space below the 2 GB bar.

Skeletons above the bar: For customers that experience heavy package and plan activity such as banks, this is the most significant DBM1 below the bar storage relief in V9. For customers that use very few or small packages, such as SAP environments, the savings are smaller.

Performance, scalability, and availability

There are a large numbers of scalability improvements implemented in DB2 9 for z/OS to eliminate or reduce inhibitors to the full exploitation of faster and more powerful hardware. In particular there are several improvements in the area of INSERT, UPDATE, or DELETE intensive workloads, more recovery functions, and CPU reduction for utilities. See Figure 1-6.

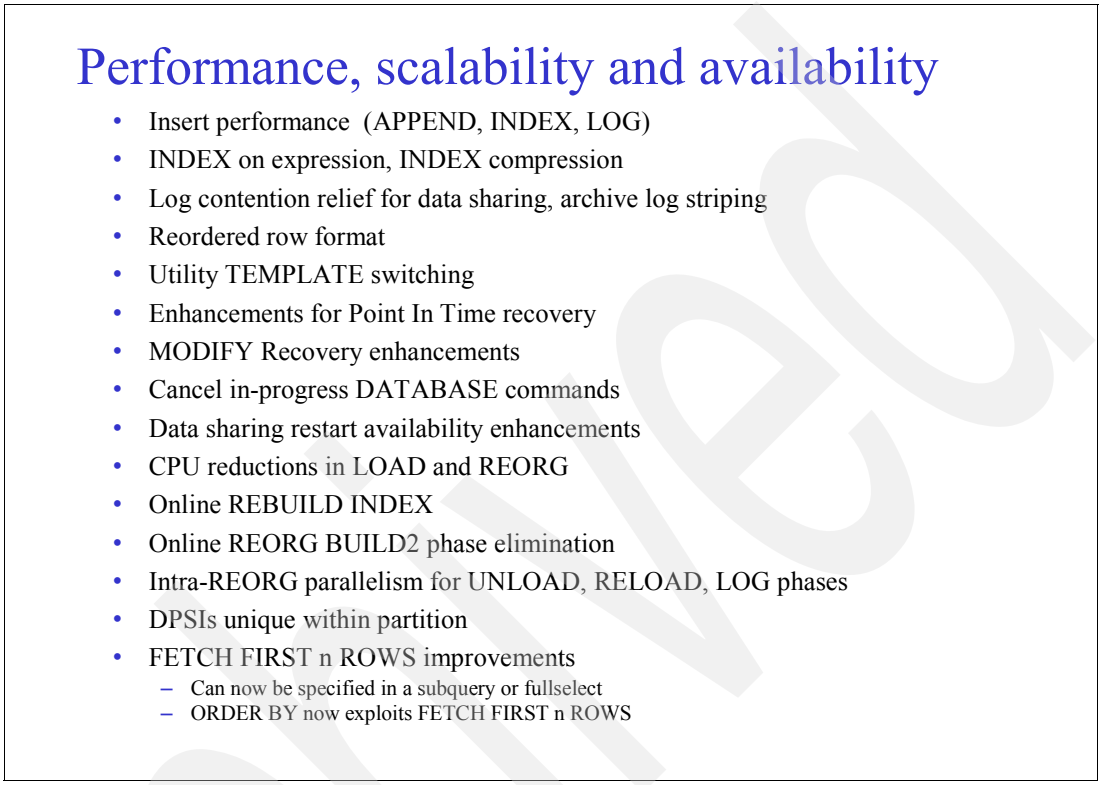


Figure 1-6 Performance, scalability, availability

INSERT processing improvements

The table APPEND option offers increased performance for inserting data into the end of a table. It reduces the instructions used to target the locations for new rows. Index maintenance techniques are also made more efficient when DB2 detects that entries are added to the beginning or ending of the index.

Sequential insert performance is improved by avoiding page splits with larger index page sizes and the ability to split a page more effectively.

Other changes improve logging rates and allow striping for archive logs. The NOT LOGGED option for table space may reduce I/O contention and latch contention.

Data sharing will benefit the most from these changes.

MODIFY Recovery

You can specify a certain number of image copies to retain. A new keyword RETAIN®(n) is added to the MODIFY RECOVERY utility control statement. If RETAIN(n) is specified together with AGE or DATE it specifies the minimum number of full copies to be retained. The delete point specified by AGE or DATE will be adjusted backwards, if necessary, to retain the specified number of full copies. If RETAIN(n) is specified without AGE or DATE, an equivalent DATE value is generated that retains the specified number of full copies.

Archived

Reordered row format

Varying length data can improve performance substantially if there are large numbers of varying length columns. The new format allow for direct addressing of length columns without scanning the length of preceding variable length rows.

Template switching

Provides a mechanism to allocate utility output data sets either to DASD or to TAPE depending on the size of the data set to be allocated.

CPU reduction with index processing in utilities

Significant reductions in CPU usage are provided with the utilities.

FETCH FIRST clause

Improved CPU and elapsed times can be achieved with the FETCH FIRST clause specified on a subquery. Fetch First N Rows with Order By by avoiding tournament tree sort for small N (up to -50% CPU). (With the small n, you do not need to sort the entire answer set. Only the top n needs to be sorted.) ORDER BY now exploits FETCH FIRST n ROWS so that work files are not created (less I/O).

Index-related enhancements

Index on an expression can be combined with caseless comparisons to improve text search.

Index compression will save disk space, especially in data warehousing environments.

1.2.3 Reduce cost of ownership and System z specific skill needs

Most of the key items in this version help key enterprise application partners, such as SAP, but also improve many other applications and customers. Customers working on the Web and Service Oriented Architecture (SOA) see most of these benefits, too. The SQL flexibility improvements allow DB2 to be efficient in performance and in productivity for our partners. The many items improving the performance, scalability, continuous availability. SQL and portability contribute to a net improvement in the total cost of operation.

We now examine the enhancements resulting from synergy with the z hardware and software platform. The trends in cost of ownership for various platforms, as noted in customers' surveys, show how hardware costs continue to decline on all platforms. People costs are climbing the most, and software increases its percentage, so people cost is the first obstacle for reducing total cost of ownership.

Regulatory compliance

Today's computing environment is subject to increasing regulatory pressures and potentially malicious attacks. The security of the information to which you have been entrusted has never been more critical. DB2 for z/OS already resides on one of the most secure platforms in the industry. It provides secure access to your data through internal or external authorities and privileges, encryption, and multilevel security with row-level granularity enforcement.

Security can be difficult to manage. The plan for DB2 is to provide you with additional capability while assisting you with security management.

While DB2 for z/OS V8 provides many enhancements for security, there are still many more needs and much more work to do. V9 helps in this area with some key functions (see Figure 1-7).

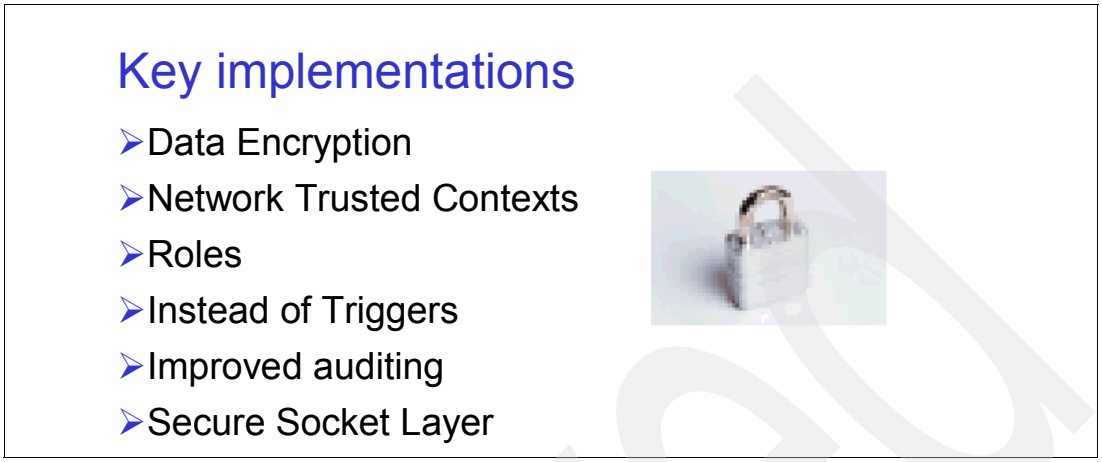


Figure 1-7 Security in DB2 9 for z/OS

A network-trusted context provides a technique to work with other environments more easily, improving flexibility. Roles are used to provide a more flexible technique than groups or users in assigning and controlling authorization, while improving consistency with the industry. The instead of trigger is an SQL technique that allows a trigger to be used in place of a view, consistent with DB2 for LUW. Improved audit selectivity is needed for being able to see that security is functioning. Secure Socket Layer (SSL) implementation provides encryption of data on the wire. Some additional techniques for data encryption will help protect data at rest and in backups.

Database role

A database role is a virtual authorization ID that is assigned to the user. DB2 privileges are assigned to the defined role. You only get the privileges associated with the role when you come through the trusted context.

The role exists as an object independent of its creator, so creation of the role does not produce a dependency on its creator.

This capability can allow a DBA to have privileges to create objects and manage them for a time, even though ownership is to be another ID.

The role can be assigned and removed from individuals via the trusted authorization context as needed. This allows a DBA to perform object maintenance during a change control window on a Saturday night, for example. But when Monday arrives, they do not have the authority to do this same work.

Auditing trails of the work completed during the maintenance window are available for verification by a security administrator or auditor.

Trusted security context

With DB2 9 for z/OS, you have the option to set a system parameter that indicates to DB2 that all connections are to be trusted. It is unlikely that all connection types (such as DRDA, RRS, TSO, and batch) from all sources will fit into this category. It is likely that only a subset of connection requests for any type and source may be trusted or that you want to restrict trusted connections to a specific server. More granular flexibility allows for the definition of trusted connection objects.

Once defined, connections from specific users via defined attachments and source servers will allow trusted connections to DB2. The users defined in this context can also be defined to obtain a database role. One typical environment for trusted security context applicability is WebSphere® connection.

Trace filtering

Improved trace filtering makes the job of auditing and of performance management easier.

Other TCO reductions

We now examine several other functions, related to the z/OS environment, which help in reducing people cost by reducing complexity of operations. See Figure 1-8.

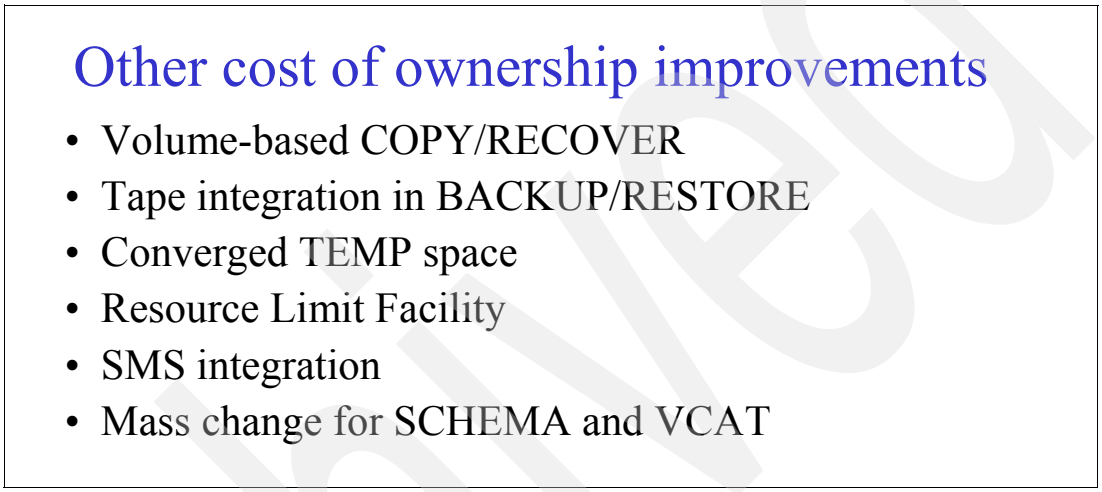


Figure 1-8 Reducing cost of ownership

Volume-based object recovery

FlashCopy® technology can be used to capture the entire content of disk volumes, thereby enormously simplifying the data back up. V8 provided a new BACKUP utility using FlashCopy technology to take very fast backups of the entire subsystem without any disruption. RECOVER is only for the whole subsystem.

The RECOVER utility has been modified to enable object-level recovery from a volume FlashCopy. This eliminates labor associated with setting up COPY jobs for each database and table space.

Tape integration

DB2 9 for z/OS provides full integration of tape into BACKUP/RESTORE SYSTEM utilities. Tapes can encrypt the data for safe remote recovery storage.

Temporary storage architecture and use

DB2 9 for z/OS introduces a single source for all temporary space in DB2, replacing DSNDB07, temporary databases, and workfile database.

Access is virtualized for small amounts of data, eliminating the cost of work file creation (reduced CPU and I/O) for most simple cases.

The workfile now supports 4 KB and 32 KB page sizes, with an automatic, optimized selection of the appropriate page size. A new ZPARM is meant to prevent workfile monopolization by single users.

Archived

Many other recent improvements have enhanced the use of temporary storage when it is needed for interim materialized result sets.

Resource limit facility

RLF is extended to support the Set Client Information APIs that are exposed in CLI and JDBC™. This allows the customer to control CPU cost of dynamic queries for packaged applications and user-written applications that exploit these APIs.

RLF allows CPU cost to be controlled based on:

- Client workstation name
- Client application name
- Client user ID
- IP address

This helps applications that exploit the Set Client Information APIs (SAP, PeopleSoft®, Siebel®, WebSphere).

SMS integration

The existing VOLUME clause is now optional. It can be omitted if any of the DFSMS™ classes is specified. If explicitly specified to DB2, the new attributes are recorded in SYSIBM.SYSSTOGROUP.

CREATE | ALTER STOGROUP

DATACLAS	dcname
MGMTCLAS	mcname
STORCLAS	scname

SCHEMA and VCAT changes

This enhancement supports changing schema, creator, owner, and VCAT name of objects using the CATMAINT utility. This is a system-wide function, meaning that all objects are altered from one schema name to another. Creator, owner, and VCAT names are also updated in a similar manner.

Three new options (SCHEMA, OWNER, and VCAT) are added to the CATMAINT utility.

Archived

Synergy with System z

Several enhancements take advantage of the synergy with IBM System z and z/OS in areas that include Long Displacement Instructions support, XML parsing, zIIP, MIDAW channel improvements, encryption, IPv6, and Secure Socket Layer (SSL). Figure 1-9 shows the major enhancements.

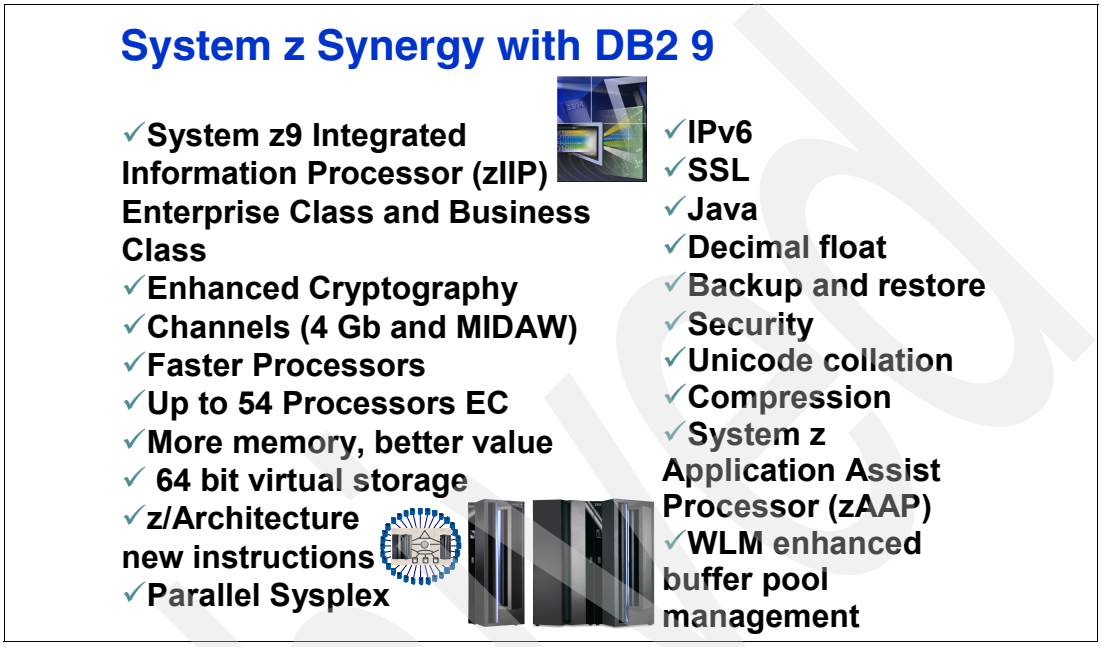


Figure 1-9 zSynergy at a glance

New with DB2 9, remote native SQL procedures are enabled for processing on the IBM System z9™ Integrated Information Processor (zIIP). DB2 synergy with System z9 continues with fiber connection (FICON®) and I/O improvements. Tests with Version 8 that involve parallel access on one channel have shown faster response times for sequential access. Similar improvements are available for DB2 9 for z/OS. V9 also takes advantage of new System z9 hardware support with the new data type, DECFLOAT, that lets you use decimal floating-point numbers with greater precision. FlashCopy can be used for DB2 database backup and restore operations. Other improvements for V9 on System z9 include added security and encryption and more Unicode collation.

The latest System z processor improvements for DB2 are the zIIP and the new Business Class and Enterprise Class processors. DB2 9 for z/OS remote native SQL procedures are enabled for zIIP processing. V9 adds IPv6, SSL, and decimal float and BIGINT data types, with enhancements for Parallel Sysplex®, backup and restore, added security and encryption, and more Unicode with collation, and uses the WLM in new ways. Channel enhancements (MIDAW) and improved DS8000™ performance were included with the System z9 announcements. DB2 uses the latest improvements in hardware and operating system to provide better performance, improved value, more resilience, and better function.

DB2 benefits from large real memory support, faster processors, specialty engines, and better hardware compression. DB2 uses Parallel Access Volume and Multiple Allegiance features of the IBM DS8000 and Enterprise Storage Server®. FlashCopy can be used for DB2 backup and restore.

DB2 makes unique use of the z/Architecture® instruction set, and recent instructions provide improvements in reliability, performance, and availability. DB2 continues to deliver synergy

with hardware data compression, FICON (fiber connector) channels, disk storage, advanced networking function, and Workload Manager (WLM).

DB2 for z/OS Version 8 news

Version 8 continues to benefit functionally from the maintenance stream and hardware innovation:

- Cross loader with LOBs
- Built-in functions ASCII, TIMESTAMPDIFF
- DSN1COPY with 1000 OBIDs
- QMF with multirow fetch
- Online Check Index
- z/OS 1.7 up to 7257 extents
- LOAD, UNLOAD with LOBs
- IBM System z9 Integrated Information Processor (zIIP)
- New and updated books: general library refresh in Feb. 2007

1.2.4 Improve data warehousing and optimizer

Today's complex applications include both transactions and reporting, so performing both well is imperative. In warehousing environments, indexes can be larger than data, and compression can reduce storage cost. Several tools can help reduce warehousing query complexity.

Warehousing and optimizer

The key improvements for reporting are optimization enhancements to improve query and reporting performance and ease of use. Improved data is provided for the optimizer, with improved algorithms and a rewritten approach to handling performance exceptions.

Archived

More queries can be expressed in SQL with the provided enhancements. The set operators INTERSECT and EXCEPT clauses make SQL easier to write. OLAP extensions for RANK, DENSE_RANK, and ROW_NUMBER add new capabilities. Other SQL statements improve consistency with the DBMS industry. Query management enhancements to make accessing your data even faster and more accurate with indexing improvements include index on expression, randomization, larger index page sizes and optimization improvements provide better data for the optimizer, improved optimization techniques, and better management with optimization services. See Figure 1-10.

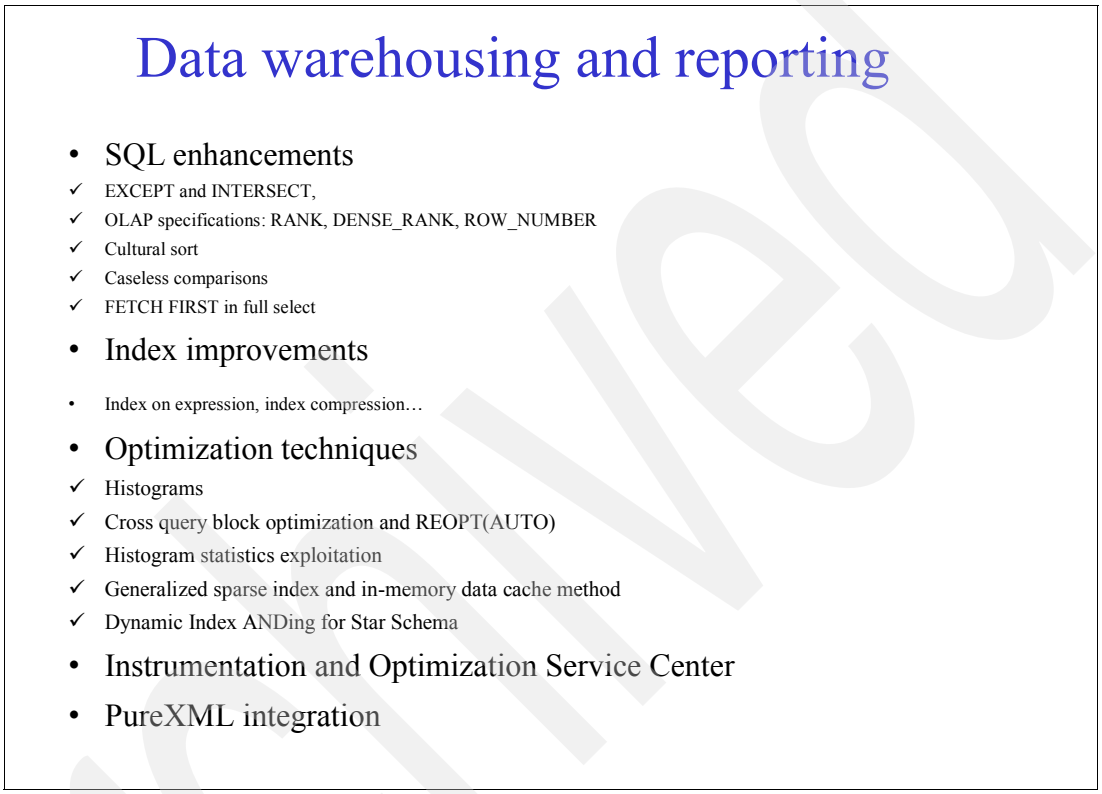


Figure 1-10 Data warehousing improvements

Global query optimization allows DB2 to optimize a query as a whole rather than as independent parts. This is accomplished by allowing DB2 to:

- Consider the effect of one queryblock on another.
- Consider reordering queryblocks.

Histogram statistics are a new type of data distribution statistics. The goal is to enhance predicate selectivity estimation to enhance the DB2 access path selection in general. Histogram statistics is a way of summarizing data distribution on an interval scale (either discrete or continuous). It divides up the range of possible values in a data set into quantiles, for which a set of statistics parameters are collected.

Improved data is provided for the optimizer, with improved algorithms.

Dynamic Index ANDing avoids the need for multi-column indexes with star schema, since it can AND together multiple single column indexes instead.

Indexes on expression are supported.

Generalized sparse index and in-memory data caching are extended from the star join schema to all queries.

Archived

Index compression

DB2 9 introduces a dictionaryless, software-managed index compression at the page level. Indexes are compressed at write time, decompressed at read time. They are uncompressed in the buffer pools. See Figure 1-11.

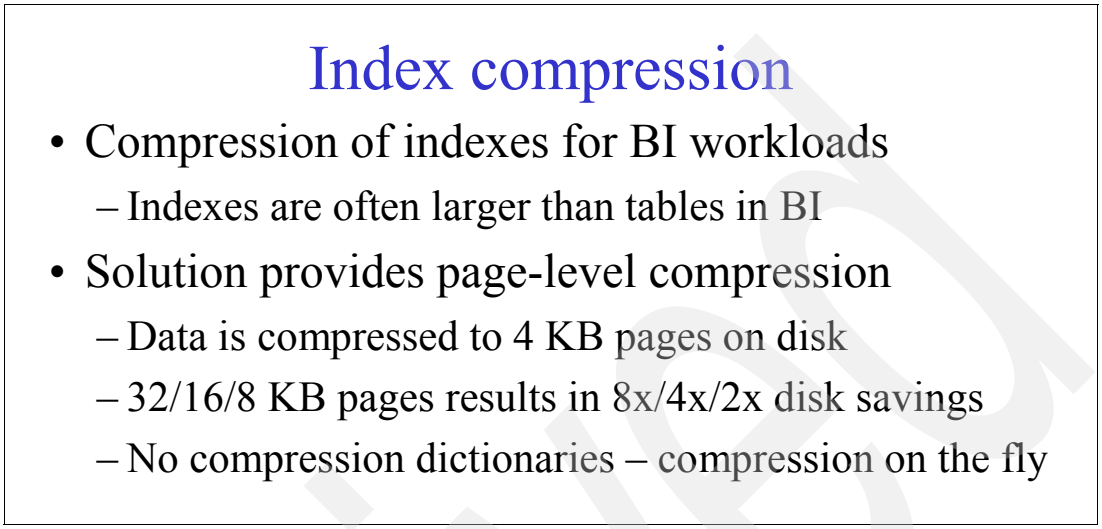


Figure 1-11 Index compression

The index compression relies upon page level compression instead of row-level compression (technique used for tables). Indexes with 32 KB page sizes can save up to eight times the disk space with the compression feature turned on. The solution has to guarantee that a given page will compress to 4 KB on disk, so DB2 will sometimes force pages in memory to be *partial* so that the compression will indeed result in no more than a 4 KB byte page on disk.

Query tools

QMF has been enhanced to include several new warehousing functions. A new tool, DataQuant, adds functionality in this area.

QMF

QMF Version 9 Enterprise Edition delivers:

- Support for DB2 9 for z/OS, including XML and enhancements to SQL.
- Drag-and-drop development of OLAP analytics, SQL queries, tabular reports, graphical reports, pivot tables, and data analysis views. Drag and drop development of interactive visual solutions such as executive dashboards, capable of graphically rendering data simultaneously drawn from multiple data sources Rich security model, supports the personalization and dissemination of data and reports on a per-user/group basis.
- New object repository simplifies the storage, access control, and distribution of QMF objects across the enterprise.
- Re-engineered Eclipse-based desktop application extends QMF to numerous workstation operating systems including Windows, Linux, Solaris™, and AIX®.
- Former QMF Visionary solution fully assimilated into core QMF product, extending the Visionary graphical objects to QMF reports and unifying the development of visual solutions within a single application.
- Redesigned QMF for WebSphere extends key QMF functionality to browser-based users across multiple platforms.

Archived

- With this release, DB2 QMF offers a completely redesigned, cross-platform workstation and Web-based solution, providing on demand access to data, reports, and interactive visual solutions using a rich desktop application or ordinary Web browser.

DataQuant

DataQuant V1.1 is a business analytics tool built on Eclipse, which delivers a comprehensive query, reporting, and data visualization platform for both Web and workstation-based environments. It is available for z/OS, IBM DataQuant for z/OS (program number 5697-N64), as well as Linux, UNIX, and Windows, IBM DataQuant for Multiplatforms (program number 5724-R90).

DataQuant introduces a variety of powerful business intelligence capabilities, including:

- Self-service reporting

  Empower your end users with the ability to answer their own business questions while also decreasing the demands on your IT group.

- Turnkey solution

  Rapidly create and distribute business intelligence queries, reports, and interactive dashboards across your enterprise.

- Perform analytics

  Create graphical reports and interactive visual solutions on your distributed or z/OS-based data warehouses, without moving your data, with 100 built-in mathematical and analytical functions.

- Sharing information

  Service-oriented architecture (SOA) infrastructure, enabling enterprises to share specific business intelligence content and data with partners over secure Internet or intranet connections.

- Distributed access

  Concurrently draw data from multiple relational data stores, such as multiple instances of DB2 for z/OS, DB2 for Linux, UNIX and Windows, and Informix®.

Tools for complex query management

Two new tools, DB2 for z/OS Optimization Service Center (a component of the no-charge DB2 Accessories Suite) and Optimization Expert (a separate tool to buy), propose to lower TCO, reduce specialized skills, and simplify workload management by providing a rich set of reactive, proactive, and autonomic functions to optimize query performance and physical database design. OSC and OE allow DB2 users to transition from manual query tuning and problem identification using performance monitoring tools, through automated analysis and recommendations, to policy-driven autonomic optimization based on best practices and business requirements.

Optimization Service Center

IBM Optimization Service Center for DB2 for z/OS is a Windows workstation tool that helps you tune and get expert recommendations for statistics to collect to improve the performance of your queries and query workloads. Optimization Service Center (OSC) is offered as part of the DB2 Accessories Suite for z/OS, a separate no-extra charge product with number 5655-R14. The Accessories Suite also includes IBM Spatial Support for DB2 for z/OS and the International Components for Unicode for DB/2 for z/OS.

OSC is built on Eclipse, an award-winning, open-source platform for the construction of powerful software development tools.

Archived

OSC represents an innovative set of capabilities to improve the performance of DB2 SQL queries at both the individual query and full SQL workload level. For individual queries, OSC can annotate the query, draw an access plan graph, generate query reports, facilitate generation of optimization hints, and suggest statistics needed by SQL Optimizer. At your request, OSC can send query-related information back to IBM service. For SQL workloads, OSC can capture workloads from various kinds of sources such as dynamic statement cache, the catalog tables, files and directories, and so on.

Profile monitoring provides a lightweight exception monitoring mechanism because only those exception statements are pushed out by DB2, and OSC fully supports the use and management of profile monitoring.

The main features of OSC are:

- Reactive tuning — optimization tools for problem SQL queries

  You can use OSC to identify and analyze problem SQL statements and to receive expert advice about statistics that you might gather to improve the performance of problematic and poorly performing SQL statements on a DB2 for z/OS subsystem.

- Proactive tuning — optimization tools for monitoring and tuning SQL workloads

  You can use Optimization Service Center to identify and analyze groups of statements and receive expert advice about statistics that you can gather to improve the performance of entire SQL workloads.

- Advanced tuning — optimization tools for experienced DBAs

  Powerful OSC optimization tools enable the experienced DBA to understand, analyze, format, and optimize the SQL statements that run on a DB2 for z/OS subsystem.

DB2 Optimization Expert

DB2 Optimization Expert for z/OS V1.1 (program number 5655-S19) DB2 Optimization Expert offers a range of advanced tools to help experienced DBAs understand, analyze, format, and optimize SQL statements. Through these tools, DBAs can gather valuable information, graph access plans through an easy-to-use interface, and even set optimizer hints to improve the access path of a query.

- Optimize DB2 query performance to help lower total cost of ownership.
- Proactively solve problems before they occur to help increase quality of service.
- Decrease reliance on specialized DBA skills through autonomics and open tools.

DB2 Optimization Expert builds on the basic capabilities of the DB2 Optimization Service Center to offer advanced proactive monitoring technologies, enabling organizations to transition from reactive, manual query tuning to proactive optimization based on best practices by building on an open standards platform.

DB2 Optimization Expert enables you to quickly snap the cache of your DB2 subsystem to obtain what dynamic queries have been running and which of those queries might be causing performance problems — from application outages to migration obstacles — on your DB2 for z/OS subsystem. You can also view query activity from a number of other sources, such as the DB2 catalog, or even import a query from a file. In addition, DB2 Optimization Expert lets you record information about the normal execution of static and dynamic SQL statements, as well as exception processing information, such as when the execution of SQL statements exceeds specific thresholds.

DB2 Optimization Expert makes it easy for you to get and implement expert advice to improve query performance. After you have identified a problem query, you can run any or all of the expert advisors to get tuning recommendations, including:

- Statistics advisor

  Recommends statistics to update or collect to improve the performance of a query

- Query advisor

  Recommends ways to rewrite an SQL query to improve performance

- Access path advisor

  Alerts you to problematic access paths in the access plan for the query that might cause poor performance

- Index advisor

  Recommends new indexes to enhance the performance of an SQL query

The tool includes the ability to create monitor profiles that monitor the health of query processing on the subsystem and alert you when problems develop and when more tuning activities might be advised. DB2 Optimization Expert also goes beyond single query tuning advice to offer expert tuning advisor functions to help optimize entire workloads.

Archived

Archived

Part 1

Subsystem functions

This part contains the chapters that deal with functions generally related to the DB2 subsystem and the z/OS platform.

This part contains the following chapters:

- Chapter 2, “Synergy with System z” on page 3
- Chapter 3, “Scalability” on page 43
- Chapter 4, “Availability” on page 73
- Chapter 5, “Data sharing” on page 121

© Copyright IBM Corp. 2007. All rights reserved.

Archived

2

Synergy with System z

As with previous versions, DB2 9 for z/OS takes advantage of the latest improvements in the platform. DB2 9 increases the synergy with System z hardware and software to provide better performance, improved value, more resilience, and better function.

DB2 benefits from large real memory support, faster processors, and better hardware compression. DB2 uses Parallel Access Volume and Multiple Allegiance features of the IBM DS8000 and Enterprise Storage Server. FlashCopy can be used for DB2 backup and restore. DB2 makes unique use of the z/Architecture instruction set, and recent instructions provide improvements in reliability, performance, and availability. DB2 continues to deliver synergy with hardware data compression, FICON (fiber connector) channels, disk storage, advanced networking function, and Workload Manager (WLM).

The latest System z9 processor improvements for DB2 are the zIIP specialty engine and the new Business Class and Enterprise Class processors. IBM announced cryptography improvements in late 2005. Channel enhancements (MIDAW) and improved DS8000 performance were included with the System z9 announcement. DB2 uses the latest improvements in hardware and operating system to provide better performance, improved value, more resilience, and better function.

This chapter discusses the following:

- zSynergy in general
- Synergy with new I/O and new CPU hardware
- System z9 Integrated Information Processor and specialty engines
- Activating zIIP exploitation
- Capacity planning for zIIPs
- RMF support for zIIPs
- SMF records for zIIP support
- RMF Monitor III support for zIIPs

© Copyright IBM Corp. 2007. All rights reserved.

2.1 zSynergy in general

New with DB2 9, remote native SQL procedures are now enabled for processing on the IBM System z9 Integrated Information Processor (zIIP). DB2 synergy with System z9 continues with fiber connection (FICON) and I/O improvements. Tests with Version 8 that involve parallel access on one channel have shown faster response times for sequential access. Similar improvements are anticipated for V9. V9 also takes advantage of new System z9 millicode support with the new data type, DECFLOAT, that lets you use decimal floating-point numbers with greater precision. FlashCopy can be used for DB2 database backup and restore operations. Other improvements for V9 on System z9 include added security and more Unicode collation. Figure 2-1 summarizes the major enhancements.

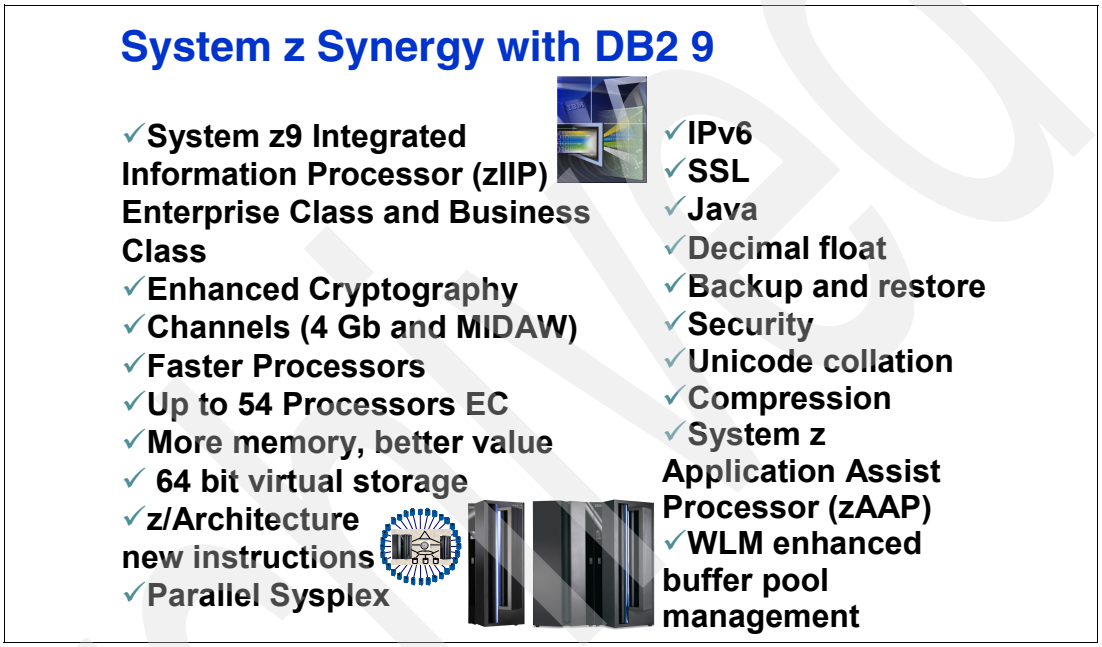


Figure 2-1 zSynergy at a glance

2.2 Synergy with new I/O and new CPU hardware

DB2 9 for z/OS works in synergy with new I/O and CPU hardware. We look briefly at both.

2.2.1 Sequential I/O

The performance of DB2 queries, utilities, and mass inserts are all very sensitive to the performance of the data paths between the processor and the disks.
In particular, the performance of large LOBs and large rows, and the performance of the Copy and Recover utilities, is much affected by the data paths. Thus, I/O hardware performance can affect data availability as well as performance.

The data path between a host and the storage server is made up of several components that together act like a conveyor belt with baggage handlers at each end.

The host has a *channel adapter* and the storage server has a *host adapter*, putting and pulling data on and off a FICON link.

In between the channel and the host adapter, the FICON link operates as the conveyor belt, which moves data bidirectionally at up to 4 Gbit/sec today.

Collectively we refer to the *data transfer speed* of this portion of the data path.

Figure 2-2 summarizes the synergy with new I/O hardware.

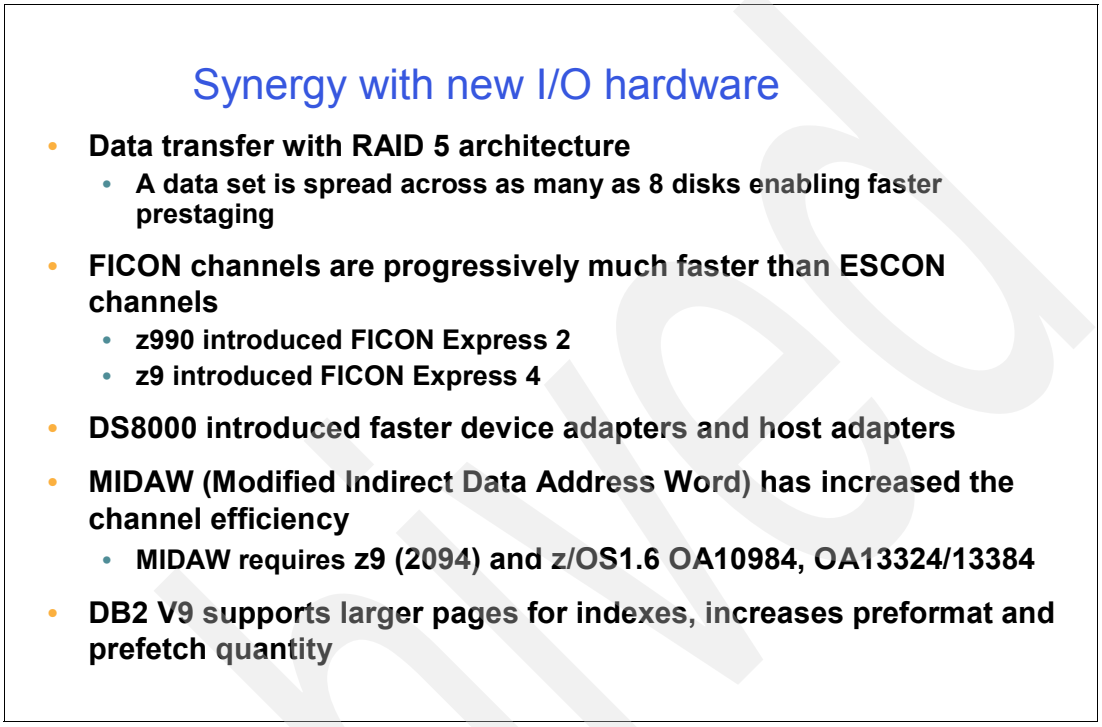


Figure 2-2 Synergy with I/O technology advances

The MIDAW facility increases the efficiency of the FICON link by improving the way that the host does scatter reads and writes to and from discontiguous memory. Channel efficiency is defined as the relationship between throughput and channel utilization. A more efficient channel is one that achieves higher throughput for a given utilization.

MIDAW is supported by all IBM storage servers and by the z9 processor. The MIDAW is exploited by the Media Manager component of z/OS, which DB2 for z/OS uses. MIDAW is especially important for Extended Format (EF) data sets that use a small page size (especially 4 KB).

DB2 uses Extended Format data sets to support large data sets (>4 GB) and striped data sets. MIDAW also improves the performance of non-EF DB2 data sets under conditions where the channel utilization is stressed.

One of the unique hardware synergy features of DB2 pertains to the DSNZPARM *SEQCACH* parameter. It assumes the values BYPASS (default) and SEQ, sequential. If SEQCACH is set to SEQ, then DB2 requests *prestaging* for all sequential and dynamic prefetch I/Os.

Note: Generally, in DB2 terminology, *prefetching* is the I/O that DB2 requests asynchronously with respect to the application thread while *prestaging* is the disk access that the storage server does asynchronously with respect to the host I/O.
However, they are conceptually the same thing, and in most data storage documents you will find that the term sequential prefetching is used instead of prestaging.

While the FICON link is used to read data from the storage server to the host, the IBM storage server may also asynchronously prestage data from up to eight disks that generally form a Raid 5 rank.

Most storage servers today have the ability to detect sequentiality under some conditions. When the storage server detects sequentiality, it uses prestaging no matter how SEQCACH is set.

However, the sequential detection algorithms in the servers are often conservative to avoid consuming too much disk bandwidth or cache space. A scattering of DB2 buffer hits may cause a table scan to read only a subset of the pages from the disk. When this happens, the storage server might not be able to detect sequentiality. So, if there is a chance for a table or index scan to experience a non-zero buffer hit ratio, IBM advises customers to use SEQCACH=SEQ.

To track the progress of the I/O hardware technology IBM focuses in particular on sequential prefetch using 4 KB page sizes. Figure 2-3 shows the progression of changes to the hardware technology which have improved DB2 sequential prefetch.

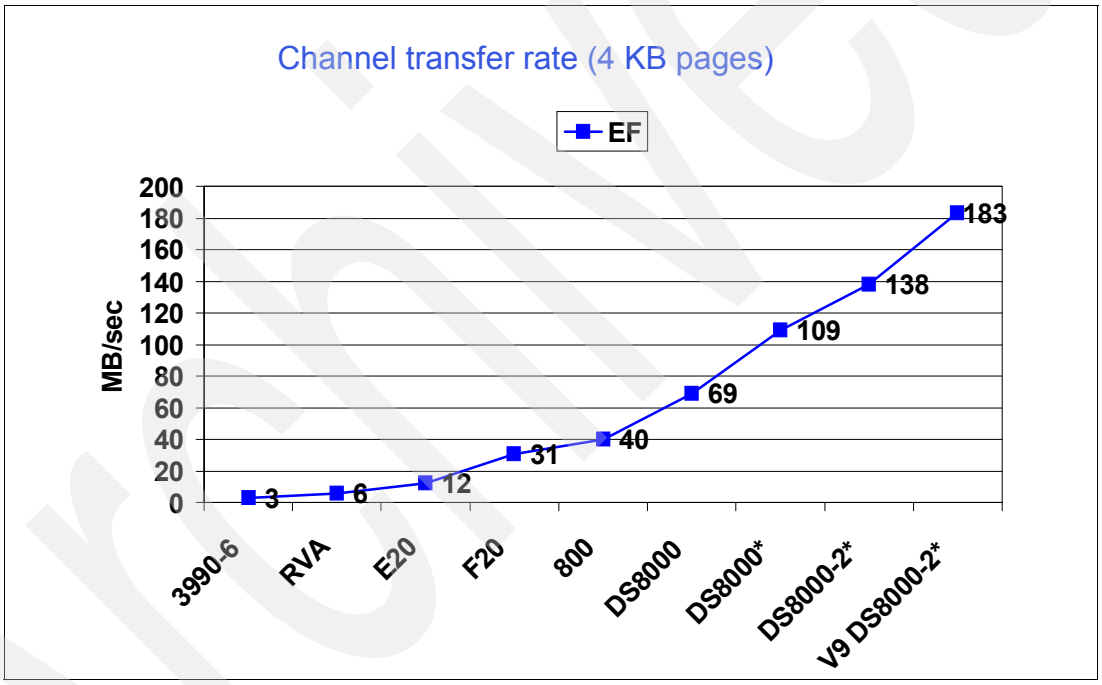


Figure 2-3 Sequential processing with MIDAW

In Y2000 IBM processors and storage servers began supporting FICON channels to replace the older ESCON® channel technology. The G6 and z900 processors were the first to support FICON.

The z900 introduced FICON Express channels.

In 2005 IBM introduced FICON Express 2 channels for the z990 processor, and the following year IBM introduced FICON Express 4 channels for the z9 processor.

FICON Express supports 1 Gbit/sec FICON links, FICON Express 2 supports 2 Gbit/sec FICON links, and FICON Express 4 supports 4 Gbit/sec FICON links.

The ESS F20 was the first IBM storage server to support FICON.

For DB2 table scans of 4 KB pages, the F20 host adapter was capable of transferring up to 28 MB/sec.

The ESS 800 increased this speed to about 40 MB/sec., which was the maximum bandwidth of one Raid rank.

Since the ESS prestaging rates far exceed the bandwidth of a host adapter, applications were never much aware that prestaging was occurring.

In 2005, IBM introduced FICON Express 2, the first DS8000 model, and the MIDAW facility.

The DS8000 storage server has much faster host adapters and device adapters than the ESS.

While the faster host adapters enable higher data transfer rates, the faster host adapters enable faster prestaging rates.

In 2006, IBM introduced FICON Express 4 and the DS8000 Turbo, which both support 4 Gbit/sec FICON links.

Yet since prestaging in the Turbo performed the same as the non-Turbo, an individual table scan would perform very similarly on the two models. The primary advantage of the faster host adapters in the Turbo was that fewer host adapters were needed to handle a heavily sequential I/O workload. The Turbo also increased the throughput of sequential writes performed by DB2 utilities and the DB2 logs.

In 2007, IBM recently made yet another big breakthrough in sequential I/O performance when it introduced AMP (Adaptive Multi-Stream Prefetching), available with the 2.4G LIC release of the DS8000 (DS8000 R2.4.2 Maintenance Release Letter: R11p.9b070621 (Bundle 62.42.77.0)). AMP optimizes cache efficiency by incorporating an autonomic, workload-responsive, self-optimizing prestaging technology. The new algorithm dynamically decides when and what to prestage, delivering up to a two-fold increase in the sequential read capacity of RAID-5 arrays. AMP improves the prestaging performance of the DS8000, thereby enabling a DB2 prefetch operation to more fully take advantage of FICON Express 4 channels and the fast host adapters in the DS8000 Turbo. See Figure 2-4 on page 8 for a graph including AMP measurements.

Using DB2 V8, prior to AMP, the DS8000 was able to prestage 4 KB pages at 92 MB/sec. The DS8000 host adapter was able to transfer this data across the FICON link at about 97 MB/sec. and the DS8000 Turbo host adapter was able to transfer this data at 132 MB/sec. Nevertheless, prior to AMP, the prestaging rates of the DS8000 Turbo for DB2 V8 were limited to 92 MB/sec. AMP increases the prestaging rate, enabling DB2 Version 8 to prefetch data from the disk at 130 MB/sec.

DB2 9 doubles the prefetch quantity. Doing so with AMP increases the prefetch rates using 4 KB pages from 130 MB/sec to 160 MB/sec.

Doubling the prefetch quantity also causes the prefetch rates for a 32 KB page size to exceed 200 MB/sec. These rates also require a z9 processor with the MIDAW facility.

Striping a data set across multiple Raid 5 disk arrays using DFSMS enables yet higher data transfer and prestaging speeds.

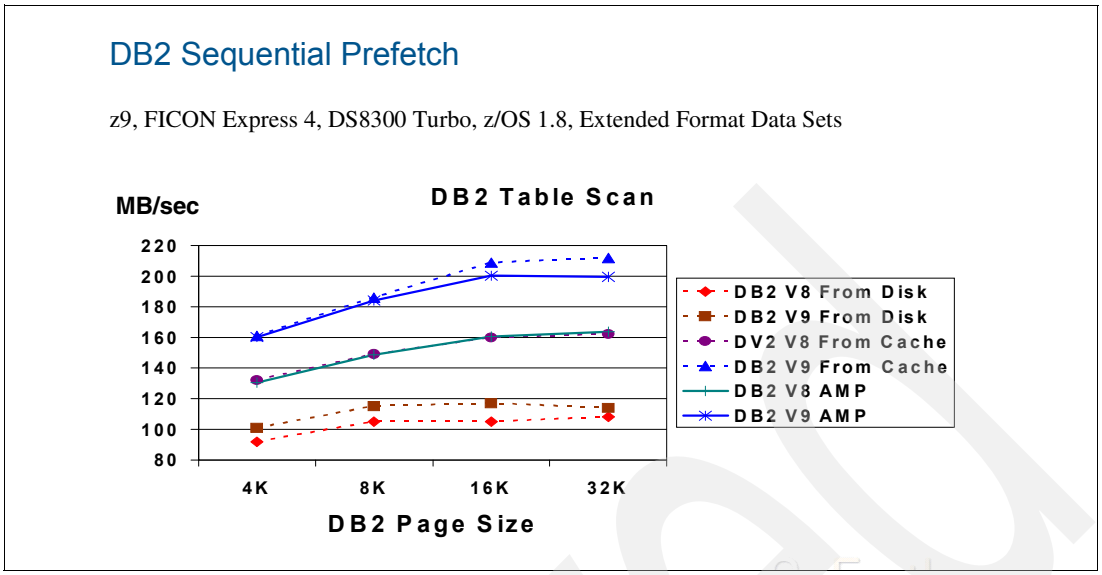


Figure 2-4 Impact of Adaptive Multi-stream Prefetching

When mass inserts cause an individual DB2 object to grow, DB2 has to preformat the storage before a Getpage can use a new page. After a chunk of storage is preformatted, the high used Relative Byte Address (RBA) is updated in the ICF catalog. As the I/O time to preformat a chunk of storage has gone down, the time to update the ICF catalog has not gone down by as much. To reduce ICF catalog overhead, DB2 9 has increased the preformat quantity from 2 cylinders to 16 cylinders. Using a DS8000 Turbo with FICON Express 4, this change has the effect of increasing the preformat throughput for 4 KB pages from 44 MB/sec. to 66 MB/sec.

2.2.2 CPU

The System z9 processors can have as many as 64 engines, 54-way. In addition, there are specific performance improvements for some instructions as we move to later models.

The z990, z890, and z9 have a new instruction for encryption and decryption, providing more than two times faster row-level encryption.

The z990, z890, and z9 processors have faster processing for long displacement instructions. While DB2 V8 used long displacement instructions, those instructions are used more in V9. The largest impact of this difference is measured in input and output column processing. If you run on these processors, then you expect to find the same or reduced CPU times, ranging from 0% to -10%. On the z900 or z800, you may expect to find increased CPU time of +5% to +10%, and the CPU could be more for a large number of columns.

The z9 processors have zIIP processors to reduce the cost of ownership and channel improvements to make I/O faster.

Figure 2-5 shows the synergy with new CPU hardware.

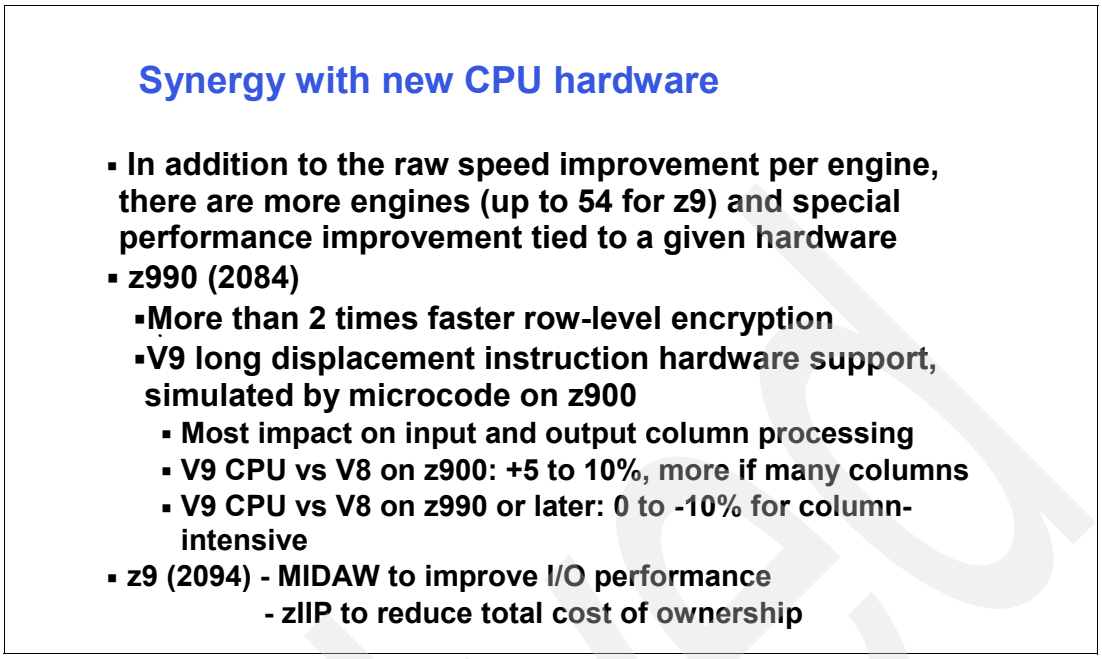


Figure 2-5 Synergy with new CPU hardware

Figure 2-6 shows the improvements in performance from a single uniprocessor, expressed as a CPU multiplier. So if a process takes one second of CPU time on the z900, it runs .37 seconds on the z9.

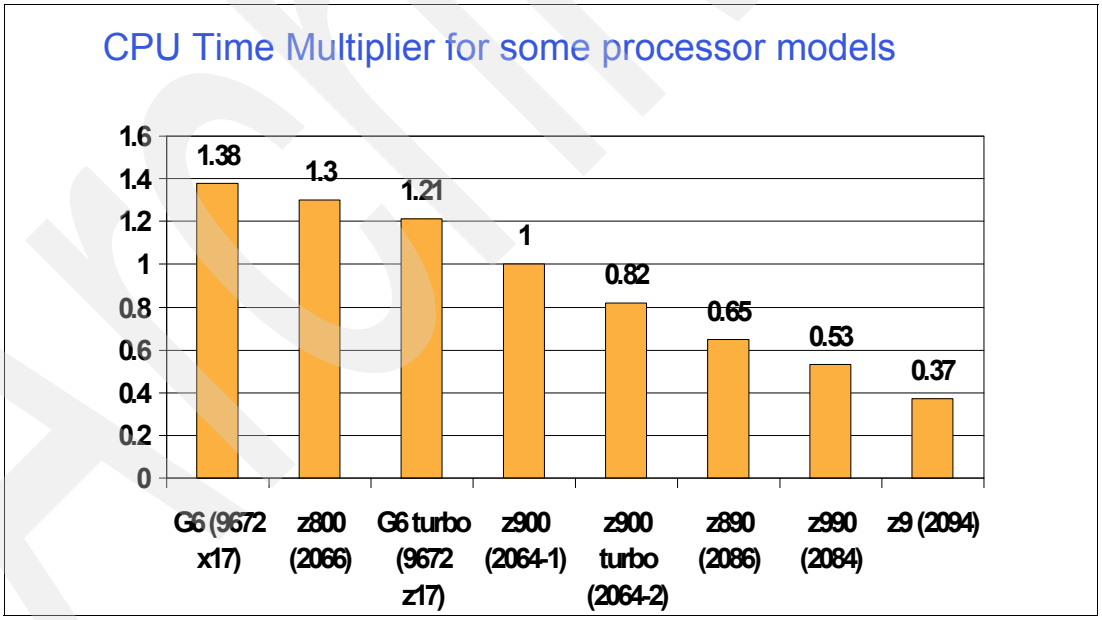


Figure 2-6 CPU multiplier across the evolution

Figure 2-7 shows the improvements in performance expressed as the power of the uniprocessors. Of course, we also gain improvements from the larger numbers of engines that come with each processor and hardware support of long displacement instructions.

Archive

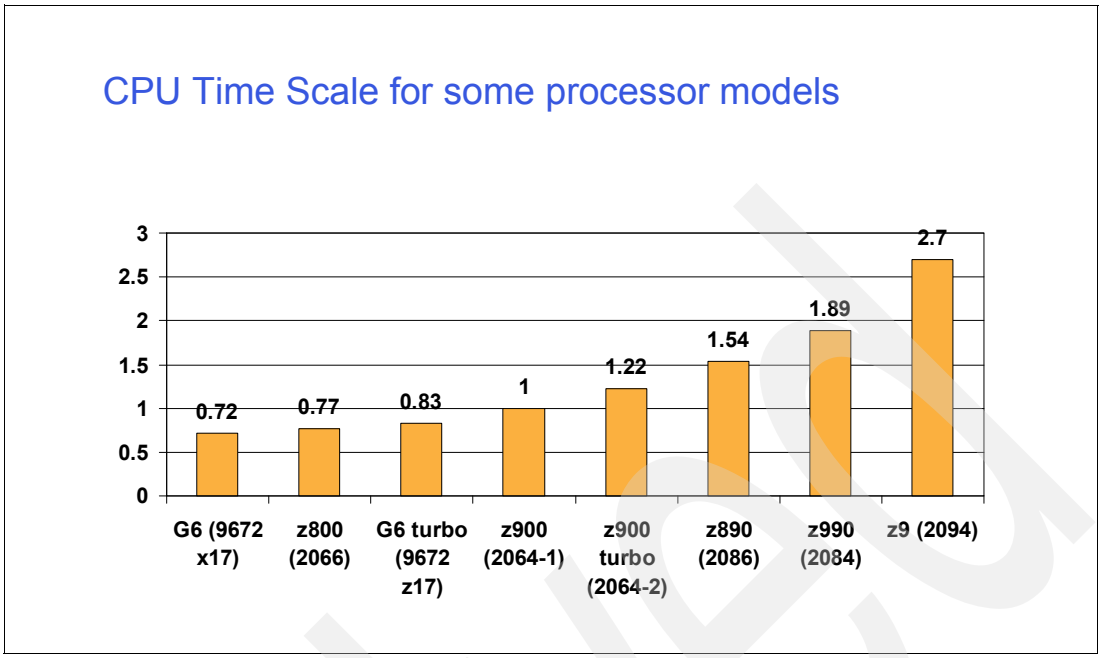


Figure 2-7 Uniprocessor relative power

The System z9 processors can have as many as 54 engines. In addition, there are specific performance improvements for some instructions as we move to later models.

The z990, z890, and z9 have a new instruction for encryption and decryption, providing more than two times faster row-level encryption.

The z990, z890, and z9 processors have faster processing for long displacement instructions. While DB2 V8 used long displacement instructions, those instructions are used more in V9. The largest impact of this difference is measured in input and output column processing. If you run on these processors, then we expect to find the same or reduced CPU times, ranging from 0% to -10%. On the z900 or z800, we expect to find increased CPU time of +5% to +10%, and the CPU could be more for a large number of columns because the long displacement instructions are software simulated on z900 and z800.

The z990, z890, and z9 processors have faster processing for long displacement instructions.

IBM has announced the implementation of hardware decimal floating point facilities in System z9. The facilities include 4-byte, 8-byte, and 16-byte data formats, an encoded decimal (base-10) representation for data, instructions for performing decimal floating point computations, and an instruction that performs data conversions to and from the decimal floating point representation. Initial software support for hardware decimal floating point is limited to High Level Assembler support running in z/OS and z/OS.e on System z9. z/OS V1.9 will provide support for hardware decimal floating point instructions and decimal floating point data types in the C and C++ compilers as a programmer-specified option. Support is also provided in the C Run Time Library and the dbx debugger.

Coupling Facility Control Code (CFCC) Level 15 is being made available on System z9 EC and BC. The enhancement includes an increase of the allowable tasks in the Coupling Facility (CF) from 48 to 112.

The z9 processors have zIIP processors to reduce the cost of ownership and channel improvements to make I/O faster.

2.3 System z9 Integrated Information Processor and specialty engines

The IBM family of specialty engines have been delivered over a number of years for the diverse workloads, ranging from a Coupling Facility in 1997 and Linux in 2000 to the Java™ workloads in 2004 and some database work in 2006.

The latest change to deliver is the ability to use the zIIP for IPSec encryption in 2007. A statement of direction was provided for z/OS XML to be able to use zAAP or zIIP for XML parsing.

In this section we discuss the special purpose processors called System z9 Integrated Information Processors (zIIPs). The z9 BC and z9 EC servers support zIIPs. The main objective of zIIP is to lower total cost of operations and improve resource optimization, strengthening your decision of making the z9 your data server of choice. The zIIP moves eligible DRDA, selected DB2 utility, and part of Business Intelligence (BI) workloads to a zIIP, reducing software cost and improving available capacity of existing general purpose engines. IBM does not impose software charges on zIIP capacity. The amount of general purpose processor savings varies based on the amount of workload executed by the zIIPs.

2.3.1 The special purpose processors

IBM has provided special processors for some time (see Figure 2-8).

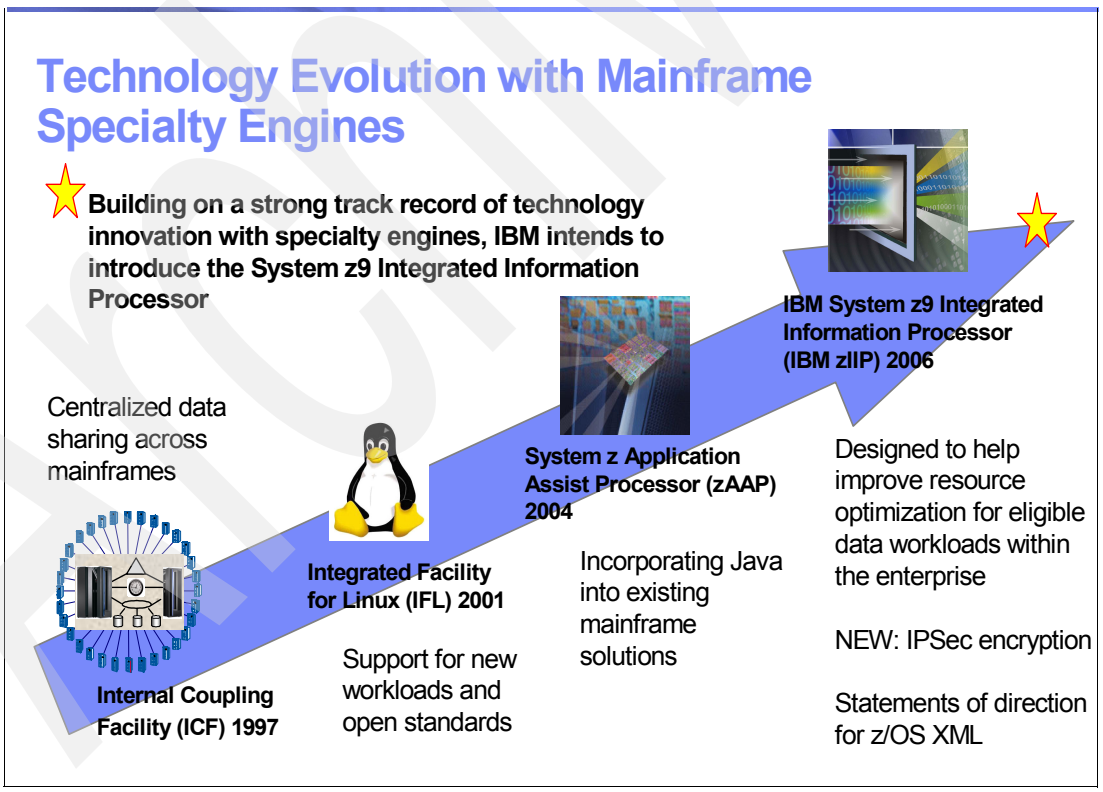


Figure 2-8 IBM special processors

The special purpose processors are:

- System Assistant Processor

The first special purpose processor was the System Assistant Processor (SAP) implemented to perform I/O processing. It was introduced in 1994 with the 9672 processors (not in the chart).

- Internal Coupling Facility

  The second special purpose processor was the ICF. Since the Coupling Facility (CF) was implemented using 9672 CP chips, it was possible to implement *internal CFs*. The introduction of ICFs in 1997 allowed the backup CF to be an LPAR on a production processor. Also with the introduction of CF duplexing, internal CFs started to be used on both production processors.

- zSeries® Integrated Facility for Linux

  The third special purpose processor was the zSeries Integrated Facility for Linux (IFL). IFLs were introduced in 2001. IFLs are processors dedicated to Linux-only workloads and cannot be mixed with general-purpose processors in an LPAR.

- z9 Application Assist Processor

  The fourth special purpose processor was the System z9 Application Assist Processor (zAAP). This system was introduced in 2004. The first *applications* exploiting the zAAP are Java workloads under z/OS. They can be used with the z990, z890, z9 EC, and z9 BC. This is currently used for the IBM Java Virtual Machine on z/OS.

- System z9 Integrated Information Processor

  In 2006, with the new z9 EC and BC servers, a new special purpose processor was introduced: the System z9 Integrated Information Processor (zIIP). The first *information processor* to exploit the zIIP is DB2 for z/OS. Prerequisites for zIIP usage are:

  – z/OS V1R6 or later
  – DB2 for z/OS V8 or V9 (first subsystem exploiting zIIPs)
  – z9 EC or z9 BC with zIIP processors

IBM United States Hardware Announcement 107-190 on April 18, 2007 states that the zIIP could be used as a high-speed encryption engine that is designed to provide better price performance for eligible IPSec workload.

The same announcement also mentions the intent to enable the z/OS XML component to take advantage of zAAPs for z/OS XML System Services processing. z/OS XML System Services parsing executing in TCB mode will be redirected to the zAAP.

In addition, IBM announced the intent to enable the z/OS XML component to take advantage of zIIPs. z/OS XML processing may be partially directed to zIIPs when utilized as part of a distributed request (like DB2 DRDA).

Figure 2-9 shows a combination of special processors on the same System z9.

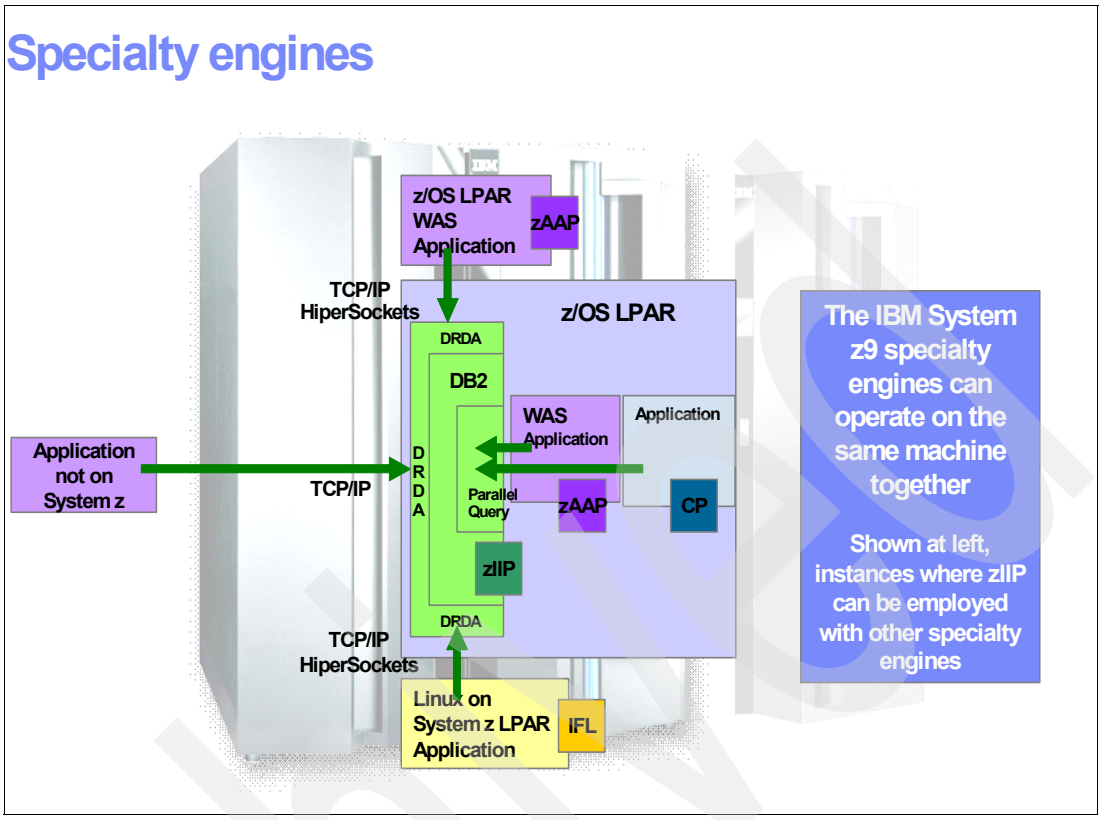


Figure 2-9 Special purpose processors

The specialty engines can be used to improve the cost of ownership, providing a low price for the hardware and not incurring software charges, since they are not general purpose processors. Customers can use all of the engines together with DB2. The ICF provides the Coupling Facility for DB2 data sharing with Parallel Sysplex for availability and scalability. The IFL can run Linux applications using DB2 Connect™ over a communication link or hipersockets to DB2 for z/OS. The zAAP can run Java applications, while the zIIP runs part of the DB2 work.

The z9 Business Class and Enterprise Class extend zIIP capabilities to many more customers. Only the largest customers needed the z9-109 processors, and the upgrade steps were very large ones. The new z9 Business Class and Enterprise Class processors have a much greater range of processing power with more granular upgrade options.

Query work is broadened beyond just the star schema parallel queries to all large parallel queries. If you have a warehouse that uses parallel processing and significant CPU time, then the zIIP may provide a benefit.

The zIIP is for customers who are concerned about costs for growth. The big cost reduction is in hardware cost, which is much less than a standard processor. The biggest cost reductions are in software, as IBM does not charge for software running on the specialty processors. A zIIP specialty engine is priced considerably less than a general purpose engine for a z9. The zIIP will fit some customers very well, but will not apply for all. As a specialty processor, not all work can use the zIIP, which will only process work running under an enclave SRB. Most applications cannot run in SRB mode. The specifics of the software charging need to be considered.

For details and recent information see:

http://www.ibm.com/systems/z/ziip/

2.3.2 z/OS support for zIIPs

The zIIPs execution environment accepts eligible work from z/OS. z/OS is designed to manage and direct the work between the general purpose processor and the zIIP engine. DB2 for z/OS V8 exploits the zIIP capability for eligible workloads. Similar to a zAAP, the z/OS dispatcher can direct certain work to a zIIP. In more specific terms, z/OS uses a zIIP only to handle SRB routines that are part of a defined enclave.

However, other than these operating system-related functions, the full instruction set is available to zAAP and zIIP processors. zIIP support is incorporated in z/OS V1R8 and is available as a Web-installable FMID for z/OS V1R6 and z/OS V1R7. zIIP support is based on the changes implemented for dispatcher affinity (off) in z/OS V1R8. z/OS V1R8 is the first z/OS operating system that has zIIP support included in the base product. For the previous releases, the support is via FMIDs as follows:

- z/OS V1R6 - JBB77S9
- z/OS V1R7 - JBB772S

CVT bit

A new CVTZIIP bit can be used by software to determine whether the z/OS release supports the new zIIP execution environment.

TIMEUSED macro

The TIMEUSED macro enables you to record execution times and to measure performance. TIMEUSED returns the amount of processor a task has used since being created (attached). The TIMEUSED macro has been modified to allow zIIP execution time to be requested in addition to the standard CP consumption.

2.3.3 DB2 support for zIIPs

z/OS dispatches work in either Task Control Block (TCB) mode or Service Request Block (SRB) mode. DB2 parallel tasks use SRB mode and are assigned the same importance as the originating address space.

Preemptible enclaves are used to do the work on behalf of the originating TCB or SRB address space. Enclaves are grouped by common characteristics and service requests, and, since they are preemptible, the z/OS dispatcher (and WLM) can interrupt these tasks for higher priority ones. There are two types of preemptible SRBs:

- Client SRBs
- Enclave SRBs

If the DB2 for z/OS V8 request is coming through DRDA over TCP/IP, then most of that work (other than stored procedures and user-defined functions) is executed in enclave SRBs.

If the request is coming over a local or a native connection, then that work is dispatched between TCBs, client SRBs, and enclave SRBs.

Parallel queries, after some CPU usage threshold is exceeded, and some utility index maintenance, also use enclave SRBs.

Archive

As for the zIIP, only the enclave SRB work can be redirected. The client SRB work, non-preemptible SRB work, or TCB work is not eligible to be redirected to the zIIP. DB2 V8 knows how its work is dispatched and directs z/OS to dispatch or redirect a portion of the eligible work to the zIIP. The above types of DB2 V8 work are those executing in enclave SRBs, of which portions can be sent to the zIIP. Not all of this work will be run on zIIP. The zIIP is designed so that a subsystem program like DB2 can work with z/OS to dispatch all or a portion of its enclave SRB work directed to the zIIP with no changes to the application — only changes to DB2 and z/OS.

Customers must be current on hardware (System z9), current on software (z/OS 1.6 or later, DB2 V8 or later), and have a workload peak using the types of work supported by the zIIP:

- Remote SQL processing of DRDA network-connected applications over TCP/IP

  These DRDA applications include ERP (for example, SAP or PeopleSoft), CRM (Siebel), and business intelligence running on other platforms. Remote SQL is expected to provide the primary benefits to customers, as it is commonly part of the peak load. Stored procedures and UDFs run under TCBs, so they are not generally eligible for zIIP, except for the call, commit, and result set processing. V9 remote native SQL Procedure Language is eligible for zIIP processing.

- Parallel queries

  If the work comes in remotely over DRDA using TCP/IP, then the initial work is eligible as remote work. After the initial time, the parallel processing threads are eligible and can process on the zIIP.

- DB2 utility index processing

  Functions of the LOAD, REORG, and REBUILD DB2 utilities that perform index maintenance are eligible for zIIP.

  This is not a common peak capacity constraint, but could be useful in reducing CPU charges.

The zIIP will run some of the work, not all of it. The best way to get an estimate of the work eligible is to put on the needed z/OS and DB2 service, to run your workload, and to take measurements. Use DB2 accounting with any product that can provide DB2 accounting reports, such as IBM Tivoli® OMEGAMON® XE for DB2 Performance Expert on z/OS.

Archived

Table 2-10 summarizes the current recommended maintenance.

```
DB2 V8 zIIP support :
 PK18454  DRDA
 PK19920, PK27712, PK30087  Utilities
 PK19921, PK27578, PK30548 Parallel Queries
 II14219    Info Apar with additional information

Support for zIIP related instrumentation changes in IFCIDs
1,3,124,147,148,231,239,316  for zIIP usage reporting, monitoring and
projection:
IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS V3 : PK25395,
PK32787, OA15898, OA15899, OA15900
DB2 Performance Expert V210 : PK29966, PK32782
DB2 Performance Monitor V810 : PK29967, PK32782

zIIP support maintenance info for z/OS, SDSF,RMF,WLM,BCP
FMIDs : z/OS 1.6 JBB77S9,   z/OS 1.7 JBB772S
Included in the base for z/OS 1.8
http://www.ibm.com/systems/z/ziip/gettingstarted/prereqs.html

RETAIN search keyword zIIP/K for zIIP related APAR/PTF information
```

Figure 2-10 Summary of current maintenance

However, review the DB2 information APAR II14219 related to DB2 V8 zIIP maintenance. Also use RETAIN search keyword zIIP/K for zIIP-related maintenance entries to ensure the correct level needed for your environment and take advantage of the most recent information. These zIIP redirect benefits apply to DB2 V8 and DB2 V9 in compatibility mode.

Once you are in DB2 V9 new-function mode (as described in 3.9, “Native SQL procedures” on page 68), the new remote stored procedures implemented using the native option available with SQL procedures will exploit the zIIP capabilities.

For more information on zIIP see:

http://www.ibm.com/systems/z/ziip/

2.3.4 DB2 monitoring of zIIPs

DB2 accounting trace records can provide information about the zIIP processor. IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS, DB2 Performance Expert, or IBM Tivoli OMEGAMON XE for DB2 Performance Monitor on z/OS can monitor the zIIP information.

2.3.5 The zIIPed workloads

In this section we discuss the zIIPed workloads.

Enterprise applications

Distributed Relational Database Architecture™ (DRDA) is the infrastructure for remote access that DB2 supports between the requesting RDBMS and the serving RDBMS. The other remote access type is DB2 private protocol access, but no enhancements are planned for private protocol.

Note: A database access thread is created when an SQL request is received from the requester. For every database access thread, an independent enclave is created.

By using DRDA work, the need for additional gateway products is reduced. These products may also affect performance and availability. The Open Group adapted DRDA in 1998 as the open standard for database remote access.

An application uses DRDA application requestor or server to access a remote database. DB2 connect is an example of a DRDA application requestor. Regarding the zIIPs, if DB2 for z/OS V8 workload comes over TCP/IP and is DRDA compliant, all or a portion of that DB2 workload is eligible to be redirected to the zIIP. Both TCP/IP and DRDA are needed.

Queries that access DB2 via DRDA over TCP/IP connections are dispatched within z/OS as enclave SRBs. z/OS directs a portion of this work to the zIIP. Only DRDA work that is coming via TCP/IP is zIIP eligible. DRDA work via SNA is not zIIP eligible.

The amount of DRDA via TCP/IP to be redirected might be affected by the DB2 stored procedures and user-defined functions in the application. DB2 stored procedures and user-defined functions run under TCBs, not SRBs, so they are not eligible for zIIPs. The call to the stored procedures, the commit processing, and the result of processing may be redirected, but the SQL processing that is in TCB mode cannot be redirected.

In DB2 9 the native SQL/PL Stored Procedure requests that come in via TCP/IP DRDA run under the DBM1 address space, and it is eligible for zIIP redirect, similar to the other TCP/IP DRDA workloads.

Figure 2-11 shows DRDA zIIP usage. This figure is for illustrative purpose only and shows only a single application. Actual workload redirects may vary.

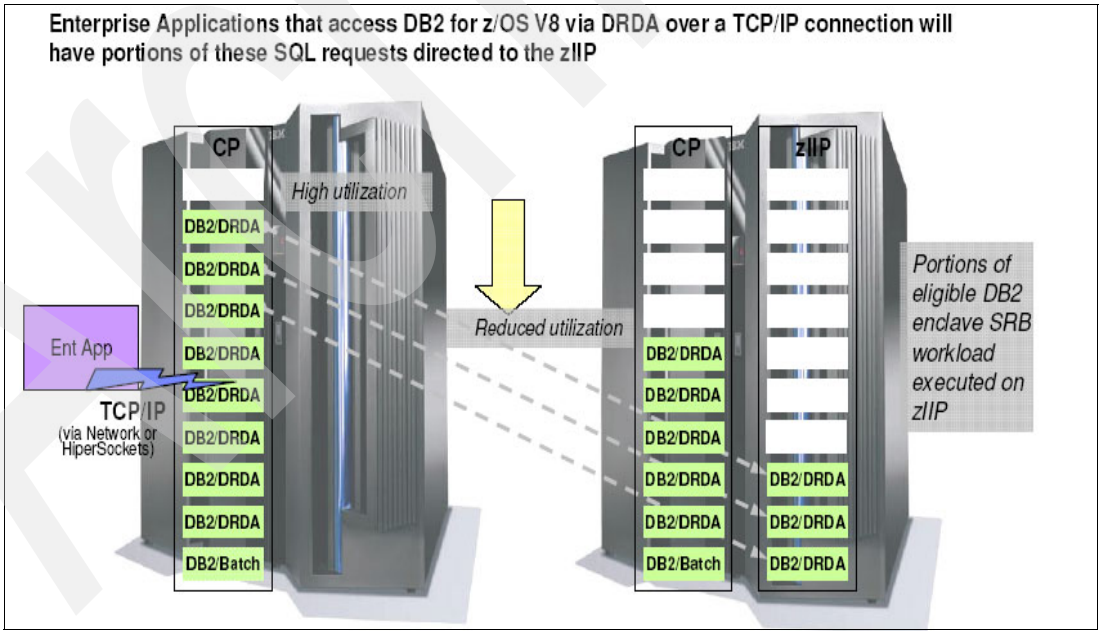


Figure 2-11 DRDA zIIP usage

One objective with the zIIP is to help bring the costs of network access to DB2 more closely in line with the costs of running similar workloads under CICS, IMS™, or Batch on the mainframe.

Database workloads such as CICS, IMS, stored procedures, and batch have become increasingly efficient and cost effective on the mainframe. Today, customers are looking to further leverage their data on the mainframe, and are turning to the mainframe more and more frequently for new application workloads. These application workloads, such as Enterprise Resource Planning, Customer Relationship Management, and Business Intelligence often use DB2 as a database server.

Added system costs for network access to DB2 results in hardware and software charges that are substantially higher for remote applications, when compared to the same workload running as a local batch application. The zIIP is aimed at helping customers more cost effectively run these database-serving applications on the mainframe.

zIIP processing flow

A flow of zIIP usage in detail using DRDA via TCP/IP as an example is shown in Figure 2-12:

1. A DB2 request comes in DRDA over TCP/IP.
2. DB2 schedules an SRB.
3. DB2 creates an enclave and classifies it. One enclave per transaction is created.
4. DB2 notifies WLM that the enclave is eligible to direct a portion of the work to a zIIP.
5. WLM with z/OS Dispatcher dispatches work to either zIIP or a general purpose processor.

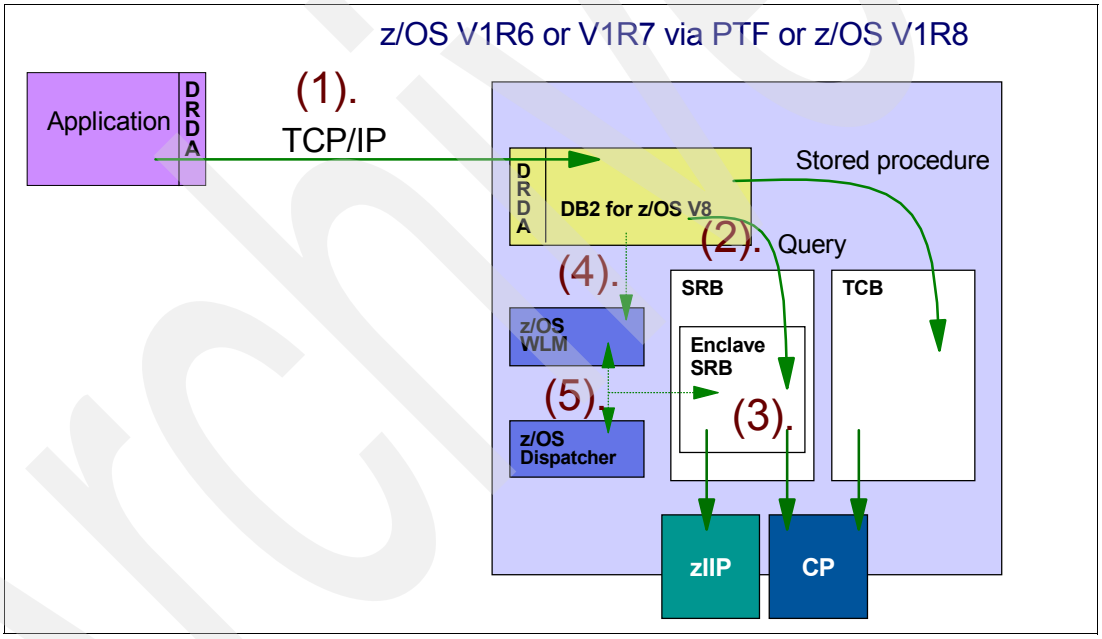


Figure 2-12 Flow of work in DRDA example

Attention: Some DB2 processing like batch and non native SQL stored procedures is not implemented in SRB mode or in enclave SRBs. Only portions of an enclave SRB work are eligible for a zIIP.

Business Intelligence applications

Business Intelligence applications that send requests that utilize DB2 for z/OS complex parallel queries may have portions of these SQL requests directed to the zIIP when DB2 gives z/OS the necessary information.

If you have a warehouse that uses parallel processing and significant CPU time, then the zIIP may provide a benefit. If the work comes in remotely over DRDA using TCP/IP, then the initial

Archived

work is eligible as remote work. If the work comes via local connection and includes parallel queries, after the initial transaction processing time, the parallel processing threads are eligible and can process on the zIIP.

Query work has been broadened beyond just the initial star schema parallel queries to all large parallel queries with APAR PQ27578. Any long-running parallel query can now benefit from zIIP.

Star join is a special join technique that DB2 uses to efficiently join tables that form a star schema. A star schema is a logical database design that is included in decision support applications.

Complex parallel queries now use enclave SRBs. z/OS directs a portion of this work to the zIIP.

Parallel query workloads may benefit from two redirected tasks — the main task, which is a DRDA request, and a child task, which is the parallel query. The child and main tasks are additive, which means that a combination of them is expected to yield a larger amount of redirect than that of just DRDA via TCP/IP alone. Longer running queries see higher benefit. Benefits to a data warehousing application may vary significantly depending on the details of that application.

Figure 2-13 is an example of a business intelligence application type in which complex parallel queries via DRDA are being used.

Important: Only parallel queries can be zIIP eligible. Any parallel star join can exploit zIIPs in any environment (BATCH, IMS, CICS, JDBC, and so on).

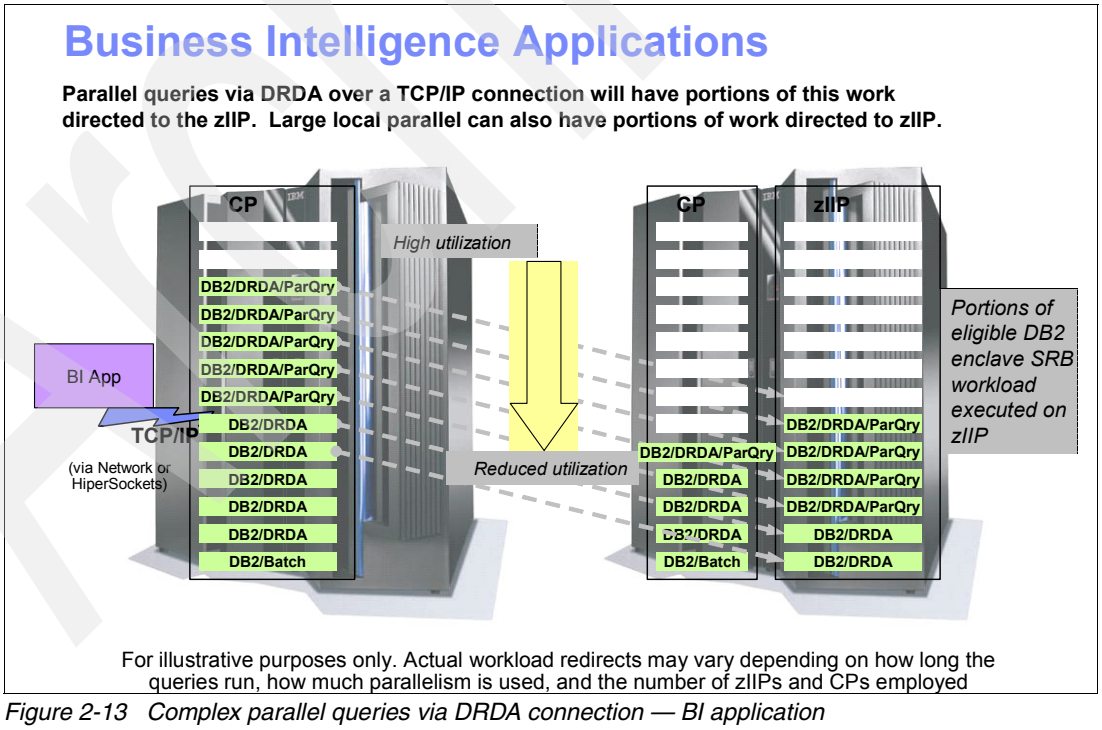


Figure 2-13 Complex parallel queries via DRDA connection — BI application

Figure 2-14 is an example of a business intelligence application type in which complex parallel queries via LOCAL connection are being used. No DRDA is being used.

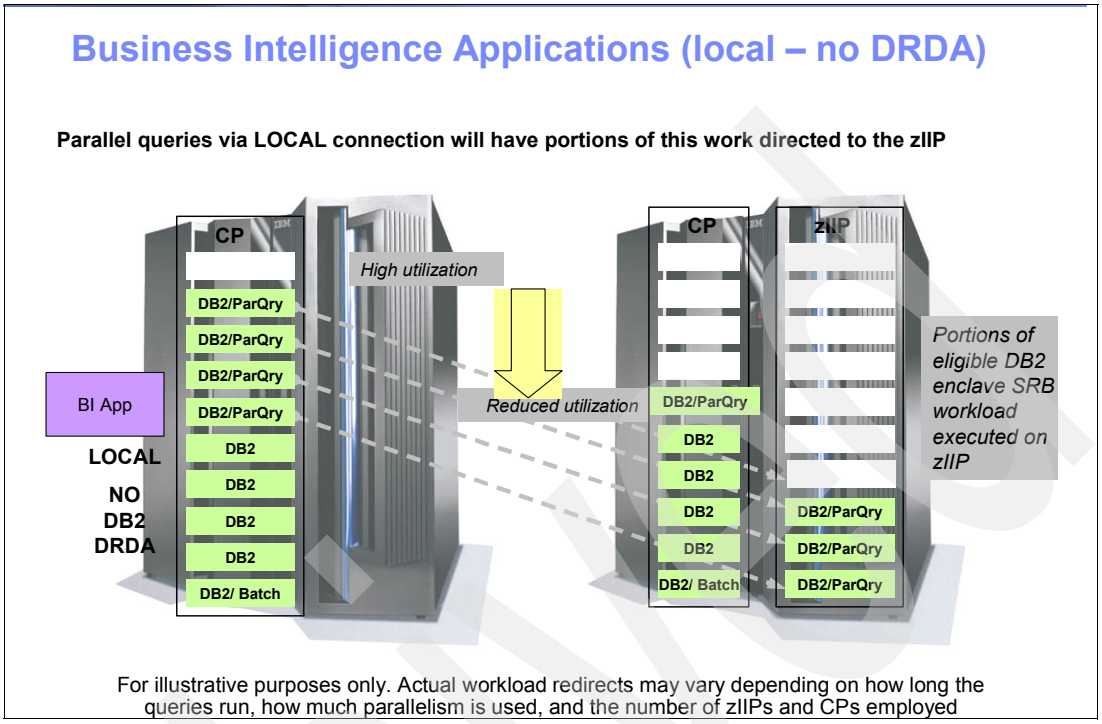


Figure 2-14 Complex star schema parallel queries via local connection — BI application

The actual work of complex parallel queries redirects may vary depending on the following:

- How long the queries run
- How much parallelism is used
- The table design being used
- The number of zIIPs and CPs employed

The following factors are taken into consideration by DB2 for the exploitation of query parallelism:

- DB2 attempts to exploit sysplex parallelism if the underlying DB2 instance is a DS member (not supported for star join queries).
- DB2 attempts to exploit CP parallelism if there is more than one CP available.
- DB2 attempts to exploit I/O parallelism if neither sysplex parallelism nor CP parallelism is eligible (no zIIP benefit).
- The estimated cost of each query block needs to exceed a certain value.
- DB2 9 has enabled more query parallelism with Star Schema Index ANDing capability.
- Number of CPs if CPU-bound (in general, parallelism is disabled if only 1 CP).
- Number of partitions of the leading table if I/O bound (in general, parallelism is disabled if table is not partitioned).

SAP workload on zIIPs

IBM System z9 advantage for SAP Applications is available with both the IBM System z9 Enterprise Class (z9 EC) and IBM System z9 Business Class (z9 BC) servers, which are ideal for SAP environments in both medium and large enterprises. These new offerings can provide benefits to companies wishing to deploy their first SAP application functions on

System z, as well as to customers running existing SAP application landscapes on System z, especially those planning to install or upgrade to the latest application based on SAP NetWeaver®.

As an enterprise data hub, the IBM System z9 can provide customers with a platform that can help manage costs and improve resource utilization. By leveraging DB2 Version 8, the new zIIP specialty processor for data-centric applications, and the Integrated Facility for Linux (IFL), this new offering can enable consolidation of SAP application landscapes onto a single system.

SAP workloads can benefit from zIIP exploitation. Workload types that are eligible are the same as mentioned above. See Figure 2-15. For estimations, SAP customers should use special SAP data collection procedures.

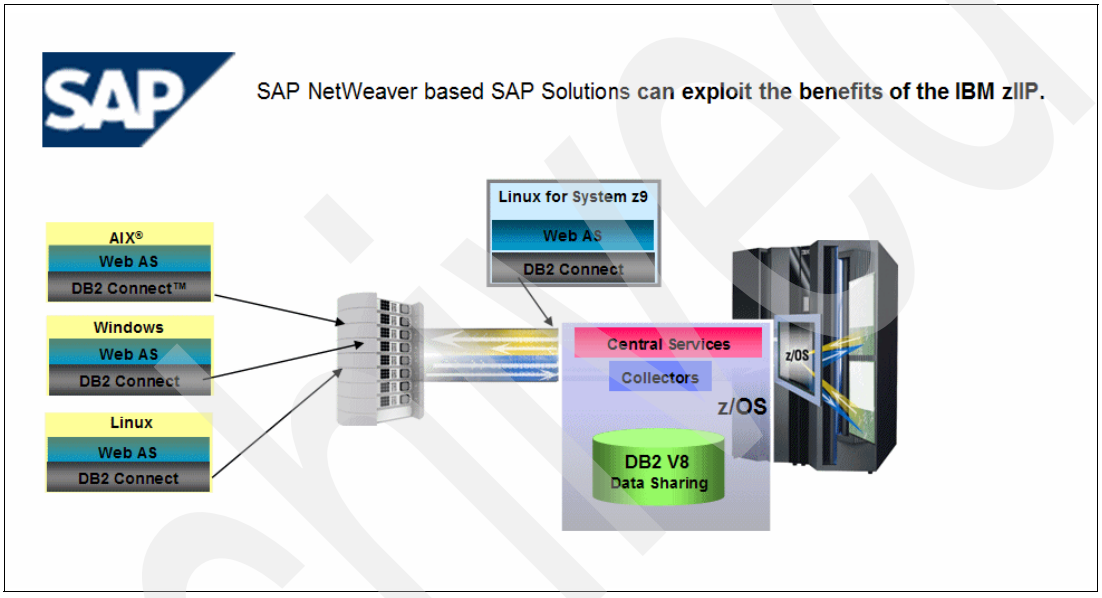


Figure 2-15 SAP workload and zIIPs

Archive

2.3.6 Index maintenance for DB2 Utilities

An index allows quick access to the rows in a table. Indexes are created using one or more columns of a table. Over time, as data in a large database is manipulated indexes can become less efficient. They need to be updated and maintained. This can be a very big task. See Figure 2-16.

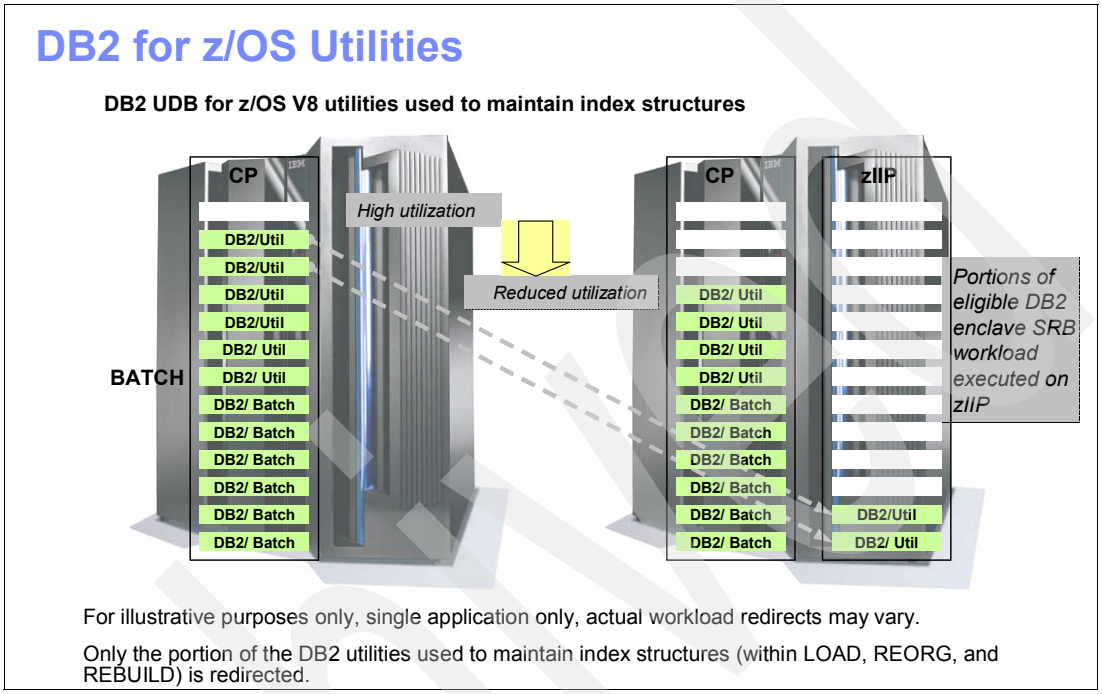


Figure 2-16 DB2 for z/OS V8 utilities used to maintain index structures

The DB2 utilities LOAD, REORG, and REBUILD INDEX now use enclave SRBs for the portion of the process that is eligible to execute in a zIIP. The LOAD utility loads tables, and the REORG utility improves index performance, while the REBUILD INDEX utility creates or rebuilds your indexes. Only the build portion of the LOAD, REORG, and REBUILD DB2 utility processing is related to index maintenance and can be redirected to zIIPs. The amount of workload eligible for zIIPs depends on the workload as follows:

- More redirect for LOAD and REORG when there are multiple indexes
- If partitioned and how many partitions are in the table

2.3.7 Utility processing eligible for IBM zIIP redirect

Estimating the amount of processing that is eligible for redirect to a zIIP without DB2 z/OS V8 zIIP maintenance or prior to DB2 for z/OS V8 is as follows:

- The following rule-of-thumb estimates identify the percentage of Load and Reorg CPU time that can be expected to be offloaded.
- The percentages are derived from a set of measurements from DB2 laboratory testing. They should be used for capacity planning purposes only.
- The true offload can vary depending on the customer environment.
- The range of percentage is dependent on the number of partitions in the table.

Test results with DB2 for z/OS V8

The low end of the range is for tables that have few partitions (for example, 1 to 49 partitions), and the high end of the range is for tables that have many partitions (for example, more than 49 partitions), as follows:

- Load or Reorg of one partition with only one index or entire table space:
  - 10 to 20%
- Load or Reorg of one partition with two or more indexes:
  - 60 to 75% if two indexes on a table
  - 70 to 85% if four indexes on a table

The rebuild Index offload behavior is opposite of what is expected. Instead of having higher percentage offload with more indexes, there is a lower percentage offload with more indexes. The reason is that the sort cost goes up non linearly as the number of indexes increases such that the cost of sort starts to dominate as the number of indexes increases, reducing the percentage impact of offloading an index build. The rebuild index is 20 to 30% higher if fewer indexes and lower if more indexes.

Note: We should note again that these values should be used for capacity planning purposes only. The true offload can vary depending on the customer environment.

2.4 Activating zIIP exploitation

The hardware zIIP engines are available with the z9 BC and z9 EC models. Once the appropriate hardware and software is installed, no further action is required to implement zIIP exploitation. As mentioned, DB2 V8 support for zIIP is described in informational APAR II14219, and the RETAIN keyword is zIIP/K. We look at the jobs and parameters for the zIIP set up.

2.4.1 IEAOPTxx parmlib member

The OPT parameters allow the installation to change the special assist processor (like zAAPs and zIIPs) options, as follows:

- [PROJECTCPU=YES/NO]

  The PROJECTCPU=YES option (also available on z/OS V1R6 and z/OS V1R7 as part of the zIIP FMIDs) now also allows zAAP projection to occur, without requiring any per JVM™ configuration changes. Previously, each impacted JVM had to be individually configured to cause zAAP statistics to be collected in RMF™ and SMF.

  To aid in determining the number of zIIP engines required to satisfy a specific customer's usage, this new parmlib option is available once all of the software updates have been applied. The PROJECTCPU=YES parameter enables z/OS to collect zIIP usage as though there was one configured, when the target workload is being run. This projection capability can be run at any time, on a production environment if desired. RMF and SMF now show this calculated zIIP time so that an accurate zIIP projection can be made.

  PROJECTCPU=YES can be used for zIIP capacity planning purposes prior to installing the z9 hardware.

Archived

- [IFAHONORPRIORITY=YES/NO]

  IFAHONORPRIORITY=YES allows zAAP eligible work to run on standard CPs if zAAP work is not completed in a reasonable time period. This is the default and recommended value.

- [IIPHONORPRIORITY=YES/NO] (added by APAR OA20045)

  IIPHONORPRIORITY allows zAAP eligible work to run on standard CPs, if zIIP work is not completed in a reasonable time period. This is the default and recommended value.

Attention: The other IEAOPTxx parmlib parameter that is also related to zAAPs is IFACROSSOVER=YES. This previous zAAP configuration option has become obsolete due to lack of customer demand and functional complexity. It still exists in IEAOPTxx but is ignored.

For details refer to *z/OSMVS V1R8.0 Initialization and Tuning Reference*, SA22-7592-14.

2.4.2 zIIP processors

The D M=CPU command output has been modified to use an *I* to represent a zIIP, as shown in Figure 2-17.

```
IEE174I 15.39.06 DISPLAY M 135
PROCESSOR STATUS
ID  CPU                SERIAL
00  +                   01991E2094
01  +                   01991E2094
02  +                   01991E2094
03  +                   01991E2094
04  -
05  -
06  +I                  01991E2094
07  +I                  01991E2094

CPC ND = 002094.S18.IBM.02.00000002991E
CPC SI = 2094.710.IBM.02.0000000000002991E
CPC ID = 00
CPC NAME = SCZP101
LP NAME = A01       LP ID =  1
CSS ID  = 0
MIF ID  = 1

+ ONLINE    - OFFLINE    . DOES NOT EXIST    W WLM-MANAGED
N NOT AVAILABLE

I       INTEGRATED INFORMATION PROCESSOR (zIIP)
```

Figure 2-17 D M=CPU command displays zIIPs

2.4.3 SDSF zIIP and zAAP support

SDSF adds information by scrolling to the right of the DA panel and the ENC panel to show usage of the zAAP (zSeries Application Assist Processor or System z9 Application Assist Processor) and the zIIP (System z9 Integrated Information Processor). See Figure 2-18. Most of the changes for the zAAP were introduced in APAR PK06616, which has been incorporated into z/OS V1R8 SDSF.

```
  Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
SDSF ENCLAVE DISPLAY  SC74     ALL                      LINE 0-0 (0)
COMMAND INPUT ===>                                        SCROLL ===> HALF
NP   TOKEN            SysName  Subsys   zAAP-Time zACP-Time zIIP-Time zICP-Time
```

Figure 2-18 SDSF ENC panel showing new columns for zAAP and zIIP times

Note: SDSF support for zIIPs is similar to zAAP.

2.4.4 WLM services for zIIP execution

The following WLM services have been enhanced to support zIIPs with the specified new parameters:

IWMEQTME The IWMEQTME service has been enhanced to include zIIP usage, as follows:

ZIIPONCPTIME=ziiponcptime An optional output parameter, which will contain the total accumulated time spent on a standard processor for zIIP eligible work for the enclave that is associated with the current dispatchable work unit. The time will be in TOD clock format, normalized to standard processor speed. To code, specify the RS-type address, or address in register (2)-(12), of an 8-character field.

ZIIPQUALTIME=ziipqualtime An optional output parameter, which will contain the total time the enclave that is associated with the current dispatchable work unit was qualified to run on an integrated information processor (zIIP). The time will be in TOD clock format. To code, specify the RS-type address, or address in register (2)-(12), of an 8-character field.

ZIIPTIME=ziiptime An optional output parameter, which will contain the total accumulated time spent on an integrated information processor (zIIP) for the enclave that is associated with the current dispatchable work unit. The zIIP time will be in TOD clock format, normalized to standard processor speed. To code, specify the RS-type address, or address in register (2)-(12), of an 8-character field.

IWM4EDEL The IWM4EDEL service has been enhanced to include zIIP usage, as follows:

ZIIPSERVICE=ziipservice An optional output parameter, which contains the integrated information processor (zIIP) service

Archived

accumulated by the enclave on the local system. The service is normalized to standard processor speed. To code, specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

ZIIPTIME=ziiptime An optional output parameter, which contains the total integrated information processor (zIIP) time accumulated by the enclave on the local system. The time is normalized to standard processor speed. To code, specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

Mixing zIIP-enabled sysplex members with non-zIIP-enabled members is fully supported. IBM does not charge any software licensing fees for offloaded zIIP work, but customers should contact their ISVs to learn their zIIP licensing policies.

2.5 Capacity planning for zIIPs

In order to help customers in determining the number of zIIP engines required to satisfy a specific customer usage, a new parmlib member option is available once all the software updates have been applied. Software requirements are mentioned in 2.4, “Activating zIIP exploitation” on page 23. The PROJECTCPU=YES parameter enables z/OS to collect zIIP usage as though there was one configured, when the target workload is being run. By specifying the PROJECTCPU option in the IEAOPTxx parmlib member, zIIP consumption can be projected without the z9 hardware. The zIIP software support is needed for the zIIP usage projection. The PROJECTCPU=YES option enables RMF to monitor DB2 for how zIIP consumption would be. This projection capability can be run at any time, on a production environment if desired. RMF and SMF will show the calculated zIIP time so that an accurate zIIP projection can be made.

This single PROJECTCPU=YES option, which is also available on z/OS V1R6 and z/OS V1R7 as part of the zIIP FMID, also allows zAAP projection to occur, without requiring any per-JVM configuration changes. Previously, each impacted JVM had to be individually configured to cause zAAP statistics to be collected in SMF and RMF. In summary, the PROJECTCPU parmlib member in IEAOPTxx projects zIIP or zAAP consumption without the hardware installed.

2.5.1 Estimating zIIP usage prior to DB2 V8

This section summarizes the process of how to estimate the zIIP offload for DRDA, query parallelism, and utilities for customers who have not migrated to DB2 for z/OS V8.

Estimating DRDA zIIP redirect

This list summarizes the process of how to estimate the DRDA zIIP offload:

- Ensure that the WLM policy is set up with service classes for SUBSYSTEM TYPE=DDF.
- Run the RMF Workload activity report (SYSRPTS) for the peak periods of interest showing DB2 DDF work-related service and reporting classes.

- The APPL% CP under the DDF work service or reporting classes for the DB2 Subsystem can be used to determine the DB2 enclave SRB CPU eligible to be redirected.
  - A portion of the eligible work will be redirected to zIIP.
  - If the eligible work is significant, contact IBM Tech Line/FTSS to help with the estimation for DRDA zIIP redirect.
  - Indicate whether stored procedures or user-defined functions are used extensively.

Rules of thumb for parallel query zIIP redirect

This list summarizes a process by which parallel query zIIP redirect can be estimated.

1. Gather DB2 Accounting trace class (1, 2, 3) for the period of interest.
2. Run the Accounting report using a tool. With IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS you can specify EXCLUDE(CONNTYPE(UTILITY)).
3. If the average Class 2 'PAR.TASKS' field in the Grand Total section has a significant value (> 50 ms) then there may be a potential for zIIP redirect.
   a. Run Accounting Trace.
   b. Using IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS performance database or another tool, filter the queries that use more than 100 ms of PAR.TASKS CPU.
   c. Subtract 0.1 sec multiplied by the number of parallel groups executed for each query.
   d. Sum the result for all such parallel queries.
   e. A significant portion of the result will be eligible for zIIP redirect.
4. More V8 parallelism potential with parallel sort and parallel multi column merge join.
5. Increased zIIP redirect potential with Star Join dynamic index ANDing enhancements in DB2 9.

Rules of thumb for Utilities zIIP redirect

This list summarizes the rules of thumb for estimating zIIP redirect for DB2 Utilities for customers who have not migrated to DB2 for z/OS V8.

The rules of thumb are based on measurements done with a tables pace with 10 partitions and a varying number of indices up to 6.

We have measured LOAD, REBUILD INDEX, and REORG Utilities.

The percentage of zIIP redirect depends on the percentage of CPU consumed by the Build Index phase of the Utility.

We have observed the following Class 1 CPU reduction for a configuration with 4 CPs and 2 zIIPs with fixed length Index key:

- 5 to 20% for Rebuild Index
- 10 to 20% for Load or Reorg of a partition with one Index only, or Load of entire table, or Reorg of entire table space
- 40% for Rebuild Index of logical partition of non partitioning index
- 40 to 50% for Reorg index
- 30 to 60% for Load or Reorg of a partition with more than one index

The CPU overhead incurred during execution unit switch from TCB to enclave SRB during Index Rebuild phase is typically less than 10%, and it is eligible for offload.

Archived

2.6 RMF support for zIIPs

Once a zIIP is installed (with z/OS V1R6 or z/OS V1R7 with a PTF) and DB2 V8 (with PTFs), the monitoring of zIIP activity is similar to monitoring zAAP activity. The monitoring of the zIIP activity is done as follows:

- Set up a WLM policy with service classes for SUBSYSTEM TYPE=DDF.
- RMF Monitor 1 Type 70 records will monitor overall zIIP activity, as follows:
  - Logical processor busy as seen by z/OS is reported.
  - Physical processor busy as seen by the LPAR is reported.
- RMF Monitor 1 Type 72 records will show more detail, as follows:
  - The amount of time spent executing on zIIP processors is reported.
  - Usage and delay sample counts for zIIP eligible work is reported.

RMF support for zIIP is introduced with z/OS V1R8. It is also rolled-back as an SPE with APAR OA13499. Table 2-1 shows the necessary PTFs for RMF to support zIIPs.

Table 2-1 PTFs needed to support zIIPs in RMF

PTF number	FMID	Release
UA90521	HRM7708	z/OS V1R5 RMF Base *
UA90252	JRM77J8	z/OS V1R5 RMF Kanji *
UA90253	HRM7720	z/OS V1R7 RMF Base
UA90254	JRM772J	z/OS V1R7 RMF Kanji
		* Only in combination with z/OS V1R6

Attention: RMF uses the term *IIP* to denote zIIP processors in the affected reports, messages, and help panels. To use consistent terms for zAAPs and zIIPs, any IFA fields in affected RMF reports have been renamed to AAP.

2.6.1 RMF zIIP implementation

RMF support for the zIIP is like support for the zAAP. RMF distinguishes between general purpose CPs and special purpose processors (zAAP and zIIP) where necessary, collects and reports about zIIP consumption, and collects and reports about using delay states for the zIIP. RMF provides measurements about zIIP activity in the following reports:

- Postprocessor:
  - CPU Activity report and its Partition Data Report section
  - Workload Activity report
- Monitor III:
  - CPC Capacity report
  - Enclave Report
  - System Information report
- New Postprocessor overview conditions for zIIP are introduced.
- New fields with zIIP measurements in SMF 70.1, 72.3, 79.1, and 79.2 are introduced.
- New DDS and RMF PM metrics about zIIP activity are introduced.
- Monitor II data services now return the system's zIIP utilization.

Archive

2.6.2 RMF CPU activity for zIIPs

Besides general purpose CPs and zAAPs (IFA), the CPU Activity section formats additional lines for each zIIP configured with the following columns: online, LPAR busy, and MVS busy time percentages, as shown in Figure 2-19. The following changes are:

- A summary line is printed with the average percentage values for the zIIPs.
- The term IFA is replaced by AAP (zAAP)

```
                                                  C P U   A C T I V I T Y

            z/OS V1R8              SYSTEM ID S5C              DATE 05/13/2006            INTERVAL 15.0
                                   RPT VERSION V1R8 RMF       TIME 15.00.00              CYCLE 1.000 S
CPU  2094   MODEL  743   H/W MODELS54
---CPU---  ONLINE TIME   LPAR BUSY MVS BUSY      CPU SERIAL  I/O TOTAL          % I/O INTERRUPTS
NUM  TYPE  PERCENTAGE    TIME PERC TIME PERC     NUMBER      INTERRUPT RATE     HANDLED VIA TPI
 0   CP    100.00        78.44      78.44         030CFE       5.63               0.00
 1   CP    100.00        78.03      78.04         030CFE       6.17               0.02
 2   CP    100.00        76.38      76.38         030CFE       5.90               0.00
 3   CP    100.00        75.64      75.64         030CFE       5.88               0.00
 4   CP    100.00        74.50      74.51         030CFE       6.01               0.00
 5   CP    100.00        73.62      73.63         030CFE       5.79               0.02
CP   TOTAL/AVERAGE       74.70      74.71                     83.28               0.00
 F   AAP   100.00        99.87      99.88         030CFE
10   AAP   100.00        99.84      99.86         030CFE
11   AAP   100.00        99.88      99.89         030CFE
12   AAP   100.00        99.86      99.87         030CFE
AAP  AVERAGE             99.87      99.88
13   IIP   100.00        99.99      100.0         030CFE
14   IIP   100.00        99.99      100.0         030CFE
15   IIP   100.00        99.99      100.0         030CFE
16   IIP   100.00        99.99      100.0         030CFE
17   IIP   100.00        99.99      100.0         030CFE
IIP  AVERAGE             99.99      100.0
```

Figure 2-19 CPU Activity report showing zIIP activity

Archived

2.6.3 RMF Partition Data Report for zIIPs

The Partition Data section of the CPU Activity Report shows the number of physical zIIPs, as shown in Figure 2-20. If zIIPs are configured, an additional data block is formatted with one line per LPAR exploiting zIIPs followed by the PHYSICAL and TOTAL line for the zIIP resource pool. For zIIPs, the same report columns as for other special purpose processors (zAAPs, IFLs, ICFs) are printed.

```
                                        P A R T I T I O N   D A T A   R E P O R T

            z/OS V1R8                SYSTEM ID S5C              DATE 05/13/2006            INTERVAL 15.00.000
                                     RPT VERSION V1R8 RMF       TIME 15.00.00              CYCLE 1.000 SECONDS

MVS PARTITION NAME                       S5C        NUMBER OF PHYSICAL PROCESSORS          54                  GR
IMAGE CAPACITY                            721                  CP                            43
NUMBER OF CONFIGURED PARTITIONS            25                  AAP                            4
WAIT COMPLETION                            NO                  IFL                            0
DISPATCH INTERVAL                     DYNAMIC                  ICF                            2
                                                               IIP                            5

--------- PARTITION DATA -----------------  -- LOGICAL PARTITION PROCESSOR DATA --   -- AVERAGE PROCESSOR UTIL
                   ----MSU----  -CAPPING--  PROCESSOR-  ----DISPATCH TIME DATA----   LOGICAL PROCESSORS  --- P
NAME       S   WGT  DEF    ACT  DEF   WLM%  NUM   TYPE   EFFECTIVE       TOTAL       EFFECTIVE    TOTAL  LPAR
S5C        A   900    0    539  NO     0.0 15.0   CP    02.48.03.847  02.48.04.742       74.70    74.70      0
S51        A   100    0      0  NO     0.0 10.0   CP    00.00.00.000  00.00.00.000        0.00     0.00      0
S5D        A   100    0      0  NO     0.0   15   CP    00.00.00.000  00.00.00.000        0.00     0.00      0
S5G        A   100    0      0  NO     0.0 15.0   CP    00.00.00.000  00.00.00.000        0.00     0.00      0
S5H        A   100    0      0  NO     0.0 15.0   CP    00.00.00.000  00.00.00.000        0.00     0.00      0
*PHYSICAL*                                                        00.00.02.598                          0
                                                        ------------  ------------                          ---
  TOTAL                                                 02.48.03.847  02.48.07.341                            0

S5C        A   900                            4   AAP   00.59.55.019  00.59.55.147       99.86    99.87      0
S51        A   100                            2   AAP   00.00.00.000  00.00.00.000        0.00     0.00      0
S5D        A   100                            4   AAP   00.00.00.000  00.00.00.000        0.00     0.00      0
S5G        A   100                            4   AAP   00.00.00.000  00.00.00.000        0.00     0.00      0
S5H        A   100                            4   AAP   00.00.00.000  00.00.00.000        0.00     0.00      0
*PHYSICAL*                                                        00.00.00.345                          0
                                                        ------------  ------------                          ---
  TOTAL                                                 00.59.55.019  00.59.55.492                            0

S5C        A   900                            5   IIP   01.14.59.421  01.14.59.558       99.99    99.99      0
S51        A   100                            2   IIP   00.00.00.000  00.00.00.000        0.00     0.00      0
S5D        A   100                            5   IIP   00.00.00.000  00.00.00.000        0.00     0.00      0
S5G        A   100                            5   IIP   00.00.00.000  00.00.00.000        0.00     0.00      0
S5H        A   100                            5   IIP   00.00.00.000  00.00.00.000        0.00     0.00      0
*PHYSICAL*                                                           00.00.00.428
                                                        ------------  ------------                          ---
  TOTAL                                                 01.14.59.421 01.14.59.987 -
```

Figure 2-20 RMF Partition Data Report showing zIIPs

2.6.4 RMF Service Class Periods Report for zIIPs

The Resource Consumption and Goal vs. Actual section of the WLMGL report is changed to format the following for zIIPs, as shown in Figure 2-21 on page 32:

- zIIP service times and zIIP using and delay state samples.
- TCB time reflects CPU time spent on regular CPs as well as on zIIPs and zAAPs.
- The SERVICE TIMES block is extended by a new field called IIP, reporting the zIIP service time in seconds.
- All APPL% values are moved to a new block, which is inserted between the SERVICE TIMES and PAGE-IN RATES blocks.

Archived

- To gain space for this new block, the STORAGE and TRANS-TIME blocks are squeezed. This is achieved by shortening a couple of field names:
  - R/S AFFINITY is changed into R/S AFFIN in the TRANS-TIME block.
  - In the STORAGE block, TOTAL is changed into TOT, CENTRAL into CEN, EXPAND into EXP, and SHARED into SHR.
- The percentage of CPU time on zIIPs is added. This field is called APPL% IIP.
- To assess the portion of IIP work executed on a standard CP, a new field called APPL% IPPCP is added, which is a subset of APPL% CP. The calculation is based on the IIP service time spent on standard CPs. Thus, this value is only reported as a percentage of CPU time, but not as a value in time of seconds. However, IIP time on CPs can be formatted by means of overview reporting.
- APPL% AAPCP and APPL% IIPCP might be reported with values greater than zero even without hardware to project zIIP and zAAP consumption (see PROJECTCPU option).
- The USING% block of the Goals versus Actuals section is extended by IIP using samples.
- The EXECUTION DELAY% block is extended. IIP delay samples can appear as new contributor to the total delay samples.

APPL % column

In Figure 2-21 on page 32, there are two zIIP percentages in this column, as follows:

- The IIP for zIIPs is the percentage of CPU time used by transactions executed on zIIPs in the service or report class period. The calculation is as follows:

$$\text{APPL\% IIP} = \frac{\text{IIP} * 100}{\text{Interval Length}}$$

  APPL% shows the CPU utilization based on uniprocessor capacity. This means that the values can exceed 100% in systems with more than one processor. In a sysplex, the values for seconds and CPU time percentage are meaningful only if all processors have the same speed.
- The IIPCP for zIIPs is the percentage of CPU time used by zIIP eligible transactions running on standard CPs. This is a subset of APPL% CP.

Archived

EXECUTION DELAYS % column

In Figure 2-21, for the EXECUTION DELAYS %, the IIP column of the zIIP-eligible work is delayed because it is waiting for a processor that can run zIIP work.

USING % column

Figure 2-21 shows the WLM service class information for the DDF enclave SRB processing. Most of the zIIP redirect will be under the DDF enclaves under DDF service class (DDFWORK in this example). The IIP column is the samples of zIIP work executing on standard CPs.

```
REPORT BY: POLICY=DRDAIC1    WORKLOAD=DB2        SERVICE CLASS=DDFWORK    RESOURCE GROUP=*NONE      PERIOD=1 IMPORTANCE=2
                                                 CRITICAL     =NONE

-TRANSACTIONS-  TRANS-TIME HHH.MM.SS.TTT  --DASD I/O--  ---SERVICE----  --SERVICE TIMES--  ---APPL %---  -----STORAGE-----
AVG       7.74  ACTUAL                16  SSCHRT 893.4  IOC          0  CPU       445.404  CP     74.25  AVG          0.00
MPL       7.74  EXECUTION             16  RESP     0.2  CPU      12636K  SRB         0.000  AAPCP   0.00  TOTAL        0.00
ENDED   140419  QUEUED                 0  CONN     0.0  MSO          0  RCT         0.000  IIPCP   5.49  SHARED       0.00
END/S   475.50  R/S AFFIN              0  DISC     0.0  SRB          0  IIT         0.000
#SWAPS       0  INELIGIBLE             0  Q+PEND   0.1  TOT      12636K  HST         0.000  AAP     0.00  --PAGE-IN RATES--
EXCTD        0  CONVERSION             0  IOSQ     0.0  /SEC     42788  AAP         0.000  IIP    76.58  SINGLE         0.0
AVG ENC   7.74  STD DEV               19                                 IIP       226.147                BLOCK          0.0
REM ENC   0.00                                          ABSRPTN  5529                                    SHARED         0.0
MS ENC    0.00                                          TRX SERV 5529   PROMOTED    0.000               HSP            0.0

GOAL: EXECUTION VELOCITY 80.0%     VELOCITY MIGRATION:   I/O MGMT  48.1%     INIT MGMT 48.2%

         RESPONSE TIME EX   PERF  AVG   ------ USING% ----- ------------ EXECUTION DELAYS % ------------ ---DLY%-- -CRYPTO%-    %
SYSTEM                 VEL% INDX ADRSP   CPU  AAP  IIP  I/O  TOT  IIP  CPU                            UNKN IDLE  USG  DLY QUIE

H2          --N/A--    48.2  1.7   7.7   8.2  0.0  8.8  0.0 18.4 15.4  3.0                            64.6  0.0  0.0  0.0  0.0
```

Figure 2-21 RMF Service Class Periods Report showing zIIPs

2.6.5 RMF Distributed Data Server for zIIPs

The RMF Distributed Data Server (DDS) is extended to provide the same variety of metrics for the zIIP as for the zAAP. These metrics can be displayed either via an RMF PM client or via a RMF Monitor III Data Portal. Here are the changes related to metrics to support zIIPs:

- New metrics for resource sysplex
  - % total utilization (zIIP) by partition
  - % total physical utilization (zIIP) by CPC
  - % total LPAR management time (zIIP) for PHYSICAL by CPC
  - % zIIP by job
- New metrics for resource processor
  - % effective physical utilization (zIIP)
  - % total physical utilization (zIIP)
  - % total LPAR management time (zIIP) for PHYSICAL
  - % effective physical utilization (zIIP) by partition
  - % total physical utilization (zIIP) by partition
- New metrics for resource processor
  - % zIIP
  - % zIIP on CP
  - # zIIPs online
  - % zIIP by enclave
  - % zIIP delay by enclave
  - total / delta zIIP seconds by enclave
  - % zIIP by job/service or report class
  - % zIIP by service or report class period

Archived

2.7 SMF records for zIIP support

The SMF type 30 record (IFASMFR3) has been updated to include new zIIP consumption fields. The SMF type 30 record may also be used to enhance existing charge back applications to include explicit zIIP charge back.

The SMF records for types 70, 72, and 79 have been updated to include information about zIIPs.

2.7.1 SMF type 70 records for zIIPs

The SMF type 70 record type 1 reports on the availability of zIIPs in the processor flags section of the RMF product section and the CPU data section, as shown in Figure 2-22.

Offsets	Name	Len	Format	Description
SMF record types 70 to 79 - RMF product section				
49 x31	PRF	1	binary	Processor flags BIT MEANING WHEN SET 0 System has expanded storage 1 Processor enabled for ESCON 2 ESCON Director in configuration 3 zAAPs available 4 zIIPs available
SMF 70.1 - CPU control section				
64 x40	SMF70SUP	4	Binary	Number of zIIPs online at the end of interval
SMF 70.1 - CPU data section				
15 x0F	SMF70TYP	1	Binary	CPU type (0=CP, 1=zAAP, 2=zIIP)
SMF 70.1 - CPU identification section				
0 x00	SMF70CIN	16	EBCDIC	CPU-identification name
16 x10	SMF70CTN	2	Binary	Number of physical CPUs of this type
18 x12		2		reserved

Figure 2-22 SMF type 70 record for zIIPs

New overview conditions based on SMF type 70 subtype 1 records (CPU activity) are available in z/OS V1R8, as shown in Figure 2-23.

```
                R M F   O V E R V I E W   R E P O R T

NUMBER OF INTERVALS 5
DATE     TIME     INT      NUMIIP    IIPBSY   IIPMBSY
MM/DD HH.MM.SS MM.SS
03/14 12.33.59 06.00           1       0.4       0.2
03/14 12.40.00 19.59           1       0.4       0.2
03/14 13.00.00 20.00           1       0.3       0.2
03/14 13.20.00 20.00           1       0.3       0.2
03/14 13.40.00 19.59           1       0.3       0.2
```

Figure 2-23 RMF Overview Report

As shown in Figure 2-23 on page 33 and in Figure 2-24, IIPBSY and IIPMBSY can be specified with a processor identifier as qualifier. If omitted, the value represents the average for all zIIPs.

Condition	Name	Algorithm
Number of zIIPs available at the end of the reporting interval	NUMIIP	SMF70SUP
CPU processor busy for zIIPs	IIPBSY	Like CPUBSY but for zIIPs instead of standard CPs
MVS processor busy for zIIPs	IIPMBSY	Like MVSBSY but for zIIPs instead of standard CPs

Figure 2-24 New overview conditions for zIIPs

2.7.2 SMF type 72 record for zIIPs

The SMF type 72 record subtype 3 (Workload Activity), as shown in Figure 2-25, is extended as follows:

- The workload manager control section is extended by the zIIP normalization factor. In addition, the description of the zAAP normalization factor (R723NFFI) is updated.
- The service report class period data section is extended starting at offset x214.

While zAAP service time (see fields R723IFAT and R723IFCT) was provided in units of microseconds by the WLM RCOLL interface, zIIP service is provided in terms of service units.

Note: For consistency purposes, zAAP service is now provided in terms of service units, also. zAAP service in terms of microsecond units (R723IFAT and R723IFCT) is kept for compatibility.

Offsets	Name	Len	Format	Description
SMF 72.3 - Workload manager control section				
240 xF0	R723NFFI	4	Binary	Normalization factor for zAAP (IFA). Multiply zAAP times or service units with this value and divide by 256 to calculate the CP equivalent value.
244 xF4	R723NFFS	4	Binary	Normalization factor for zIIP. Multiply zIIP service units with this value and divide by 256 to calculate the CP equivalent value.
SMF 72.3 - Service / Report class period data section				
532 x214	R723SUPU	4	Binary	zIIP using samples
536 x218	R723SUCU	4	Binary	zIIP on CP using samples (included in R723CCUS)
540 x21C	R723SUPD	4	Binary	zIIP delay samples
544 x220	R723CSUP	8	Float	zIIP service units. Multiply with R723NFFS and divide by 256 to calculate the CP equivalent value
552 x228	R723CSUC	8	Float	zIIP service units spent on CPs
560 x230	R723CIFA	8	Float	zAAP service units. Multiply with R723NFFI and divide by 256 to calculate the CP equivalent value
568 x238	R723CIFC	8	Float	zAAP service units spent on CPs

Figure 2-25 SMF type 72 record for zIIPs

Overview fields for type 72 records

The new overview condition fields for SMF type 72 are shown in Figure 2-26. OVW conditions IIPDLYP, IIPUSGP, and IPCUSGP are specified for a service or report class period. All other new OVW conditions can be specified for service and report classes, service and report class periods, workload groups, or the entire service policy.

To use consistent terms for zIIP-based and zAAP-based OVW conditions, an alternate set of condition names for IFASEC, IFANSEC, IFACPSEC, APPLIFA, APPLIFCP, IFAUSGP, IFCUSGP, and IFADLYP is provided. The new names are AAPSEC, AAPNSEC, AAPCPSEC, APPLAAP, APPLAPCP, AAPUSGP, APCUSGP, and AAPDLYP.

zAAP consumption is now also available in terms of service units. Previously, it was only available in terms of service time (IFASEC and IFACPSEC):

- zAAP service units per second: AAPSRV … Sum(R723CIFA) / SMF72INT
- zAAP service units on standard CPs per second: AAPCPSRV … Sum(R723CIFC) / SMF72INT

Condition	Name	Algorithm
zIIP service units per second	IIPSRV	Sum(R723CSUP) / SMF72INT
zIIP service units on standard CPs per second	IIPCPSRV	Sum(R723CSUC) / SMF72INT
zIIP service time in seconds	IIPSEC	Sum((R723CSUP x R723MADJ) / (1600 x R723MCPU))
zIIP service time in seconds (normalized)	IIPNSEC	Sum((R723CSUP x R723MADJ) / (1600 x R723MCPU)) x R723NFFS / 256
zIIP service time in seconds spent on standard CPs	IIPCPSEC	Sum((R723CSUC x R723MADJ) / (1600 x R723MCPU))
zIIP application execution time %	APPLIIP	Like IIPSEC… / SMF72INT x 100
zIIP on CP application execution time %	APPLIPCP	Like IIPCPSEC… / SMF72INT x 100
zIIP delay %	IIPDLYP	R723SUPD / R723CTSA x 100
zIIP using %	IIPUSGP	R723SUPU / R723CTSA x 100
zIIP on CP using %	IPCUSGP	R723SUCU / R723CTSA x 100

Figure 2-26 New overview conditions based on SMF record 72-3 Workload activity

Archived

2.7.3 SMF type 79 record for zIIPs

The SMF type 79 record for subtypes 1 and 2, as shown in Figure 2-27, has zIIP information.

Offsets	Name	Len	Format	Description
SMF 79.1 – Address space state data – ASD and ASDJ data section				
208 xD0	R791TSUP	4	Binary	CPU time consumed on zIIPs (ASSB_TIME_ON_zIIP)
212 xD4	R791TSUC	4	Binary	CPU time consumed on standard CPs by zIIP eligible work (ASSB_TIME_zIIP_ON_CP)
216 xD8	R791NFFS	4	Binray	Normalization factor for zIIP time. Used to convert between real and normalized zIIP times, i.e. the equivalent time on a standard CP. Multiply R791TSUP by this value and divide by 256.
SMF 79.2 – Address space resource data – ARD and ARDJ data section				
196 xC4	R792TSUP	4	Binary	CPU time consumed on zIIPs (ASSB_TIME_ON_zIIP)
200 xC8	R792TSUC	4	Binary	CPU time consumed on standard CPs by zIIP eligible work (ASSB_TIME_zIIP_ON_CP)
204 xCC	R792NFFS	4	Binray	Normalization factor for zIIP time. Used to convert between real and normalized zIIP times, i.e. the equivalent time on a standard CP. Multiply R792TSUP by this value and divide by 256.

Figure 2-27 SMF type 79 record for zIIPs

2.8 RMF Monitor III support for zIIPs

RMF Monitor III supports the following reports for zIIPs:

- CPC Capacity report in Figure 2-28
- Enclave Report in Figure 2-29 on page 37 and Figure 2-30 on page 37
- System Information report in Figure 2-31 on page 38

RMF CPC Capacity report

Figure 2-28 shows a sample CPC Capacity report showing zIIP activity.

```
                        RMF V1R8   CPC Capacity                  Line 18 of 26
Command ===>                                                  Scroll ===> CSR

Samples: 120     System: S5C   Date: 05/13/06  Time: 15.07.30  Range: 120   Sec

Partition:   S5C         2094 Model 743
CPC Capacity:   2067     Weight % of Max: 90.1    4h Avg:  160   Group:   N/A
Image Capacity:  721     WLM Capping %:   ****    4h Max:  538   Limit:   N/A

Partition  --- MSU ---  Cap  Proc    Logical Util %   - Physical Util % -
             Def   Act  Def   Num    Effect   Total   LPAR  Effect  Total

PHYSICAL                                               0.0             0.0

*IIP                                                   0.0     100     100
S5C                      NO    5.0       100     100   0.0     100     100
S5D                      NO    5.0       0.0     0.0   0.0     0.0     0.0
S5G                      NO    5.0       0.0     0.0   0.0     0.0     0.0
S5H                      NO    5.0       0.0     0.0   0.0     0.0     0.0
S51                      NO    2.0       0.0     0.0   0.0     0.0     0.0
PHYSICAL                                               0.0             0.0
```

Figure 2-28 Sample CPC capacity report indicating usage of zIIPs

RMF Enclave Report

Enclave reports provide detailed information about the activities of enclaves. An enclave is a transaction that can span multiple dispatchable units (SRBs and tasks) in one or more address spaces and is reported on and managed as units. Figure 2-29 is an example of zIIP usage information on the Monitor III Enclave Reports.

```
                         RMF V1R8   Enclave Report                    Line 1 of 14
Command ===> _                                                    Scroll ===> CSR

Samples: 120     System: S5C   Date: 05/13/06  Time: 15.41.30  Range: 120   Sec

Current options:    Subsystem Type: ALL                          -- CPU Util --
                    Enclave Owner:                                Appl%   EAppl%
                    Class/Group:                                   14.8    135.6

Enclave   Attribute  CLS/GRP  P Goal    % D X   EAppl%   TCPU    USG  DLY  IDL

*SUMMARY                                        58.88
ENC00001             SYSOTHER 1 N/A             12.82  24215     21   73  0.0
ENC00003             SYSOTHER 1 N/A             12.68  23914     34   61  0.0
ENC00007             ZIPHIGH  1        70 Y     4.124   5409     99  0.8  0.0
ENC00005             ZIPHIGH  1        70 Y     4.122   5601     99  0.8  0.0
ENC00004             ZIPHIGH  1        70 Y     4.121   5735     98  1.7  0.0
ENC00008             ZIPHIGH  1        70 Y     4.121   5395     99  0.8  0.0
ENC00009             ZIPHIGH  1        70 Y     4.120   5224     98  1.7  0.0
ENC00006             ZIPHIGH  1        70 Y     4.120   5418     99  0.8  0.0
ENC00010             ZIPHIGH  1        70 Y     4.119   4087     98  1.7  0.0
ENC00013             SYSOTHER 1 N/A             2.088  61.60    1.3   97  0.0
ENC00012             SYSOTHER 1 N/A             0.854  79.78    0.0  0.0  0.0
ENC00002             SYSOTHER 1 N/A             0.798   1356    0.0  0.0  0.0
ENC00011             SYSOTHER 1 N/A             0.787  230.5    0.0  0.0  0.0
```

Dependent Enclave

Figure 2-29 Monitor III enclave report indicating zIIPs usage

RMF Enclave Classification Data report

If you place the cursor on any data field of an enclave, shown in Figure 2-29, a pop-up panel appears, shown in Figure 2-30, showing performance statistics and classification attributes for the selected enclave.

The using percentage of all types of processors can be seen in this panel also. All percentages are based on indicated numbers of state samples, which is 120 in this example.

```
                         RMF V1R8   Enclave Report                    Line 1 of 14
Command ===>                                                      Scroll ===> CSR

Sampl                System: S5C   Date: 05/13/06  Time: 15.39.30  Range: 120   Sec

Cur
                         RMF Enclave Classification Data          Top of data

                 enclave ENC00006 with token 00000038 00000008
Enc              to return to the Report panel.

*SU    - CPU Time -       -zAAP Time--       -zIIP Time--
ENC    Total    5300      Total   0.000      Total    4038
ENC    Delta   118.9      Delta   0.000      Delta   90.43
ENC
ENC    State    ---- Using ----  ---------- Delay ----------  IDL  UNK
ENC   Samples   CPU AAP IIP I/O  CPU AAP IIP I/O STO CAP QUE
ENC      120     21 0.0  79 0.0  0.0 0.0 0.0 0.0 0.0 0.0 0.0  0.0  0.0
ENC
ENC   Classification Attributes:
ENC                                                        More:      +
ENC    Subsystem Type: JES    Owner: SKZPH026   System: S5C
ENC    Accounting Information . . :
ENC     ?,?
ENC

      F1=Help       F2=SplitScr   F3=End        F6=RMFHelp    F7=Backward
      F8=Forward    F9=SwapScr   F12=Return
```

Total CPU time in seconds consumed by enclave on general purpose CPs + zIIPs

Total CPU time in seconds consumed by enclave on zIIPs

Figure 2-30 Monitor III enclave detail report

Archived

RMF System Information report

Figure 2-31 shows the sample monitor III system information report showing the usage of zIIP processors. The Appl% IIP field shows the percentage of CPU time on zIIPs used by all address spaces and enclaves during the report interval. This value is divided by the number of zIIPs that have been active during this interval.

```
                        RMF V1R8   System Information              Line 1 of 27
Command ===> _                                                  Scroll ===> CSR

Samples: 120     System: S5C   Date: 05/13/06  Time: 15.57.30  Range: 120   Sec

Partition:    S5C     2094 Model 743          Appl%:      15  Policy: BASEPOL
CPs Online:  15.0     Avg CPU Util%:   77     EAppl%:     76  Date:   05/11/06
AAPs Online:  4.0     Avg MVS Util%:   77     Appl% AAP: 100  Time:   14.14.03
IIPs Online:  5.0                             Appl% IIP:  99
Group      T WFL --Users--  RESP TRANS -AVG USG-   -Average Number Delayed For -
               %   TOT  ACT  Time  /SEC PROC  DEV   PROC  DEV STOR SUBS OPER  ENQ

*SYSTEM       89  359    6        0.80 13.8  0.0    1.7  0.0  0.0  0.0  0.0  0.0
*TSO         100    5    0        0.80  0.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
*BATCH       100   19    1        0.00  1.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
*STC         100  309    0        0.00  0.2  0.0    0.0  0.0  0.0  0.0  0.0  0.0
*ASCH              0    0        0.00  0.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
*OMVS         97   14    5        0.00  4.9  0.0    0.2  0.0  0.0  0.0  0.0  0.0
*ENCLAVE      84   12  N/A         N/A  7.6  N/A    1.5  N/A  0.0  N/A  N/A  N/A
BATCH      W  98   20    1  .000  0.00  7.9  0.0    0.2  0.0  0.0  0.0  0.0  0.0
CPUHIGH    S 100    1    1  .000  0.00  1.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
IFAHIGH    S        5    0  .000  0.00  0.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
ZIPHIGH    S  98   14    0  .000  0.00  6.9  0.0    0.2  0.0  0.0  0.0  0.0  0.0
IWEB_WLD   W        0    0  .000  0.00  0.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
IWEBLOW    S        0    0  .000  0.00  0.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
JES_WLD    W 100    6    0  .000  0.00  0.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
JESLOW     S 100    6    0  .000  0.00  0.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
OMVS_WLD   W  97   14    5  42.8  0.06  4.9  0.0    0.2  0.0  0.0  0.0  0.0  0.0
OMVSDEMN   S        4    0   300  0.01  0.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
```

Figure 2-31 Sample RMF System Information Report indicating zIIPs usage

Archive

RMF Delay Report

From the system information panel, when you press any key under the service class of enclaves (which is ZIPHIGH in this example) you will reach to panel where you can get detailed information about enclaves service class ZIPHIGH. See Figure 2-32, “Service class detail information using zIIPs” on page 39.

```
                        RMF V1R8   Delay Report                      Line 1 of 15
Command ===> _                                                  Scroll ===> CSR

Samples: 120     System: S5C   Date: 05/13/06  Time: 14.37.30  Range: 120   Sec

            Service     WFL USG DLY IDL UKN ---- % Delayed for ---- Primary
Name     CX Class    Cr  %   %   %   %   %  PRC DEV STR SUB OPR ENQ Reason

*ZIPHIGH                 99  50   0   0  50   0   0   0   0   0   0
ENC00008 E  ZIPHIGH      98  98   2   0   0   2 N/A   0 N/A N/A N/A N/A
ENC00004 E  ZIPHIGH      99  99   1   0   0   1 N/A   0 N/A N/A N/A N/A
ENC00006 E  ZIPHIGH      99  99   1   0   0   1 N/A   0 N/A N/A N/A N/A
ENC00007 E  ZIPHIGH      99 100   1   0   0   1 N/A   0 N/A N/A N/A N/A
ENC00009 E  ZIPHIGH      99  98   1   0   1   1 N/A   0 N/A N/A N/A N/A
ENC00010 E  ZIPHIGH      99  99   1   0   0   1 N/A   0 N/A N/A N/A N/A
ENC00005 E  ZIPHIGH     100 100   0   0   0   0 N/A   0 N/A N/A N/A N/A
SKZPH024 B  ZIPHIGH           0   0   0 100   0   0   0   0   0   0
SKZPH025 B  ZIPHIGH           0   0   0 100   0   0   0   0   0   0
SKZPH026 B  ZIPHIGH           0   0   0 100   0   0   0   0   0   0
SKZPH027 B  ZIPHIGH           0   0   0 100   0   0   0   0   0   0
SKZPH028 B  ZIPHIGH           0   0   0 100   0   0   0   0   0   0
SKZPH030 B  ZIPHIGH           0   0   0 100   0   0   0   0   0   0
SKZPH014 B  ZIPHIGH           0   0   0 100   0   0   0   0   0   0



                  Report is for service class ZIPHIGH only.
 F1=HELP      F                                                  F6=TOGGLE
```

Figure 2-32 Service class detail information using zIIPs

2.9 Advanced data serving on System z

IBM has been working on specialty processors for a long time, and has just delivered new capabilities with the zIIP. The intent is to improve the cost of ownership. There have been several large, recent changes. We have seen that V8 of DB2 has broadened the applicability to zIIP of all large parallel queries, allowing significant savings to business analytics applications. DB2 9 for z/OS has added native remote SQL stored procedures as further candidates for zIIP, and increased the applicability of parallelism for start join queries as well as others.

IBM announced the following additional zIIP and zAAP enablement in April 2007, which are summarized in Figure 2-33 on page 40.

The Data Serving Roadmap provides more information about this slide at:

http://www.ibm.com/systems/z/ziip/data.html

Here we highlight two functions that are related to specialty engines exploitation:

- The z/OS 1.8 Communication Server will take advantage the zIIP processor for the IPSec end-to-end encryption security protocol processing. IPSec Authentication Header (AH) and Encapsulation Security Payload (ESP) protocol routine CPU processing will be eligible for zIIP redirect. The planned availability is August 2007.
- z/OS XML Services for Parsing (Statement of Direction): z/OS 1.8 will provide an Assembler interface. DB2 for z/OS 9 will take advantage of this interface. z/OS 1.9 will

provide a C/C++ interface. It will be available on z/OS 1.7 via SPE. When DB2 for z/OS 9 applications do inserts or update XML data, DB2 invokes the z/OS XML System Services to parse the data. DB2 for z/OS 9 XML data is loaded by utilities. The data needs to be parsed, and it invokes z/OS XML System Services to parse the data. zIIP redirect will be used for DRDA remote requests. zAAP redirect will be used for local z/OS requests.

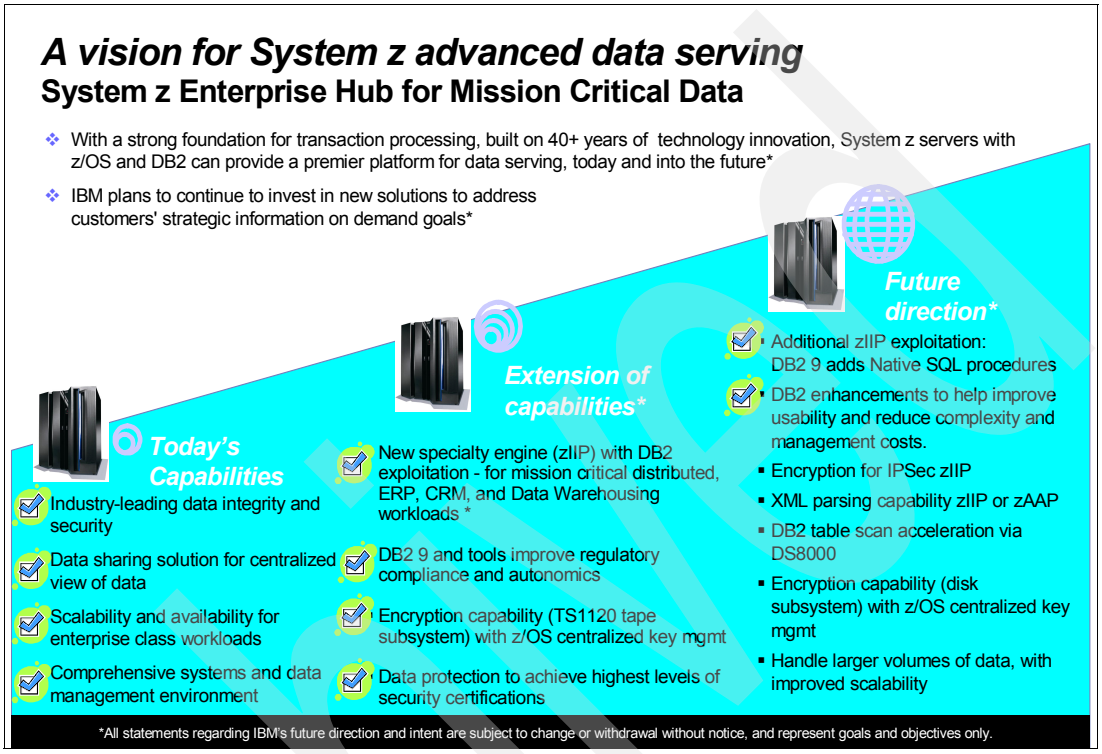


Figure 2-33 Advanced data serving

The zAAP is the System z Application Assist Processor designed to help implement new application technologies on System z. Java was the first exploiter.

The zIIP is the System z9 Integrated Information Processor designed to help integrate data and transaction processing across the enterprise. DB2 V8 was the first exploiter.

The self-describing nature of XML lends itself to both application and information processing environments.

z/OS XML System Services enabled for zAAP and zIIP means that you can have the advantages of XML processing on z/OS with TCO benefits of either the zIIP or the zAAP processor, regardless of the invocation environment. Specifically, all z/OS XML System Services parsing executing in TCB mode will be eligible for the zAAP. Note that not all DB2 9 XML processing is done using z/OS XML System Services. XML Validation is not eligible.

Examples of DB2 9 SQL/XML processing via local connection, executing in TCB mode:

- Applications (queries) running locally on z/OS

  When DB2 9 inserts or updates XML data, the data has to be parsed, and therefore DB2 invokes z/OS XML System Services (and zAAP, when present).

- Utilities

  When XML data is loaded into tables, then the XML data needs to be parsed, and therefore DB2 9 invokes z/OS XML System Services (and zAAP, when present).

How much DB2 9 work is eligible for the zAAP will depend on the amount of XML data being processed. Middleware and applications requesting z/OS XML System Services will have the z/OS XML System Services parsing eligible to execute on the zAAP. Only the XML parsing being performed by z/OS XML System Services (a base element of z/OS) is eligible for zAAP.

Note that there is a Java-based XML parser in the IBM SDK 1.3 (and later). The technology is called XML4J. This Java-based XML parser is already fully eligible for zAAP because it is Java workload. Other C++, COBOL, PL/I, and roll-your own parsers are not eligible for zAAP.

Please note that not all DB2 9 XML processing is done using z/OS XML System Services. XML Validation is not eligible.

Archived

Archived

3

Scalability

In this chapter we discuss the scalability improvements implemented in DB2 9 for z/OS to eliminate or reduce inhibitors to the full exploitation of faster and more powerful hardware. In particular, there are several improvements in the area of INSERT, UPDATE, and DELETE intensive workloads.

This chapter discusses the following:

- Virtual storage management
- WLM assisted buffer pool management
- NOT LOGGED table spaces
- Other Other logging enhancements
- Exploit WLM to resolve latch contention
- Universal table space
- Reordered row format
- Index compression
- Native SQL procedures
- Other enhancements

Archived

© Copyright IBM Corp. 2007. All rights reserved.

3.1 Virtual storage management

Virtual storage constraint is still an important issue for key high-end DB2 accounts. Extensions (long names, UTF-8 names, and so on) introduced in DB2 V8 significantly increased the size of control blocks that remained in virtual storage below the 2 GB bar. Due to this, some customers had much smaller virtual storage constraint relief (VSCR) in DB2 V8 than they expected. DB2 9 for z/OS provides some additional relief. In this section we discuss the major improvements that provide additional virtual storage constraint relief.

3.1.1 64-bit Distributed Data Facility

Beginning with DB2 V8, there has been an increase in demand for storage in DDF (DB2 DIST address space). This in part has been due to the fact that the communications buffers, which reside in the DIST address space, increased in size in DB2 V8. Additionally, ERP applications like SAP have switched to using DRDA instead of using their private ICLI and putting additional demands on DDF. Also, other vendor applications have been putting additional demand with heavy use of packages. In order to alleviate the pressure put on the DDF address space and to improve the interface with the DBM1 address space and TCP/IP, DDF now supports 64-bit addressability.

With 64-bit Distributed Data Facility (DDF), DB2 uses the z/OS Shared Memory Facility to reduce data moves between DBM1 and DDF. Shared Memory is a relatively new type of virtual storage allowing multiple address spaces to easily address storage. The Shared Memory Facility was introduced in z/OS 1.5. It is similar to ECSA since it is always addressable and no address register mode (AR) or cross memory (XM) moves are needed. It is different from ECSA since it is not available to all address spaces on the system. Only those registered with z/OS as being able to share this storage have visibility to this storage. Shared memory resides above the 2 GB bar.

Although 64 bit DDF is a performance enhancement for distributed server processing, it also provides virtual storage constraint relief. Before DB2 9 for z/OS and 64 bit DDF, when DDF invoked DB2 to process a request, data was copied from the DDF address space to the DBM1 address space at the beginning of the request, and copied back from DBM1 to DDF at the completion of the request. In DB2 9 for z/OS many of these interface areas are now in shared memory, so no cross memory move will be necessary for those blocks, and this storage may no longer be allocated in the DBM1 address space below the 2 GB bar. Additionally, some control blocks that used to reside in ECSA have now been moved to shared memory. DDF is changed to run in 64-bit mode (AMODE(64)) to be able to access Shared Memory areas.

A new shared memory object, Virtual Shared Object (VSO), is created at DB2 initialization by the DB2 MSTR address space. As MSTR, DBM1 and DIST go through their local storage initialization, they are registered to use this VSO. As each address space is terminated during shutdown, they request that their interest in the VSO be deleted. The shared memory object is freed at DB2 termination. Some DB2 utilities also register to use the DB2 VSO.

Important: This enhancement requires z/Architecture, z/OS V1R7 or later, enablement of 64-bit Virtual Shared Storage, and your system must be configured with sufficient shared private to allow *all* shared storage exploiters on this LPAR to allocate their shared objects. Each DB2 9 for z/OS requires a minimum of 128 GB of 64-bit shared virtual storage at startup time. Messages DSNY011I and DSNY012I are changed to reflect these requirements on DB2 startup.

All distributed SQL statements processed by DB2 9 for z/OS will see some improvement in performance and storage usage. SQL statements that use host variables, like INSERT, UPDATE WHERE, DELETE WHERE, CALL with parameters, and any statements that use LOBs (particularly cursors with LOB columns) will see the largest improvements.

You can monitor how much of this storage is being utilized by creating reports from IFCID225 and IFCID217 records. IFCID225 and IFCID217 have been modified to add new fields to report on shared storage.

- IFCID217 is changed to add QW0217SG Total getmained shared storage above the bar for ASID.
- IFCID225 is changed to add:
  - QW0225SF Total fixed shared storage above the bar for ASID.
  - QW0225SG Total getmained shared storage above the bar for ASID.
  - QW0225SV Total variable shared storage above the bar for ASID.

Be aware that the additional shared storage is charged against the DB2 registered address spaces. When looking at RMF and other storage reports you might see an increase in the total amount of virtual and real storage frames attributed to the DBM1 address space. However, if you take into account the storage allocated to the DIST address space, you may in fact see a decrease in overall storage allocation. Shared memory is not charged against MEMLIMIT of an address space. You can issue the z/OS command DISPLAY VIRTSTOR,HVSHARE to see the current defined storage and how much of it is currently allocated, as shown in Figure 3-1.

```
D VIRTSTOR,HVSHARE
IAR019I  14.21.42 DISPLAY VIRTSTOR 731
 SOURCE = DEFAULT
 TOTAL SHARED = 522240G
 SHARED RANGE = 2048G-524288G
 SHARED ALLOCATED = 131073M
```

Figure 3-1 z/OS DISPLAY VIRTSTOR,HVSHARE sample output

Informational APAR II14203 describes TCP/IP and z/OS pre-req APARs required to enable this functionality. Make sure that TCP/IP and z/OS have the required maintenance prior to migrating to DB2 9 for z/OS. Shared virtual storage is defined via the IEASYSxx parmlib member by specifying parameter HVSHARE or at IPL time through an operator-supplied reply to message IEA101A. You may specify the HVSHARE value in *nnnnnnnnnnG* (Gigabytes), or *nnnnnnnnT* (Terabytes), or *nnnnnP* (Petabytes).

As you can see in Figure 3-2, the default shared area starts at 2 TB and ends at 512 TB. The minimum size is zero and the maximum size is 1 exabyte (which is 1048576 terabytes or 1024 petabytes). However, your system must be configured with sufficient shared private to allow DB2 9 for z/OS to obtain a minimum of 128 GB at DB2 startup time. You must configure enough shared virtual storage to allow for at least 128 GB of shared storage for *each* DB2 subsystem or data sharing member that may be concurrently started on that LPAR, plus storage for any other exploiter of VSO. Shared storage is not backed at startup time. This only sets up the *dividing* lines.

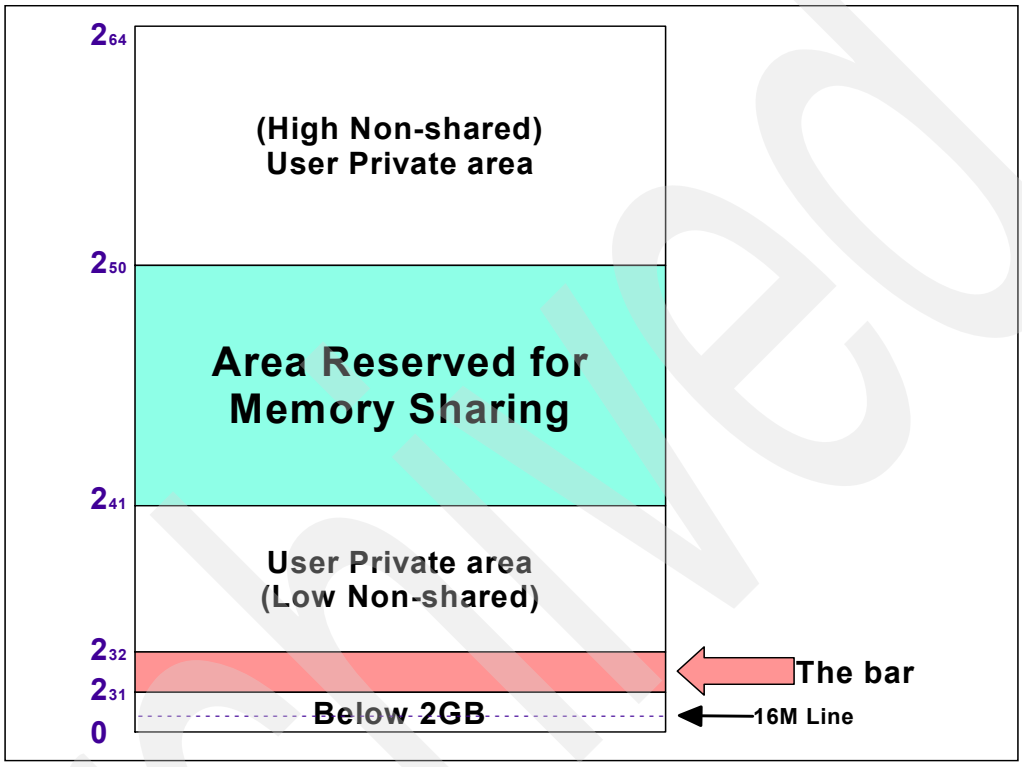


Figure 3-2 z/OS V1R5 and later address space

3.1.2 Runtime structures above the bar

DB2 9 for z/OS implements critical infrastructure enhancements for moving runtime structures above the 2 GB bar. The changes are large and are being staged, with many already available in V8, more added with V9, and probably some later. DB2 will maintain both an above-the-bar storage area and a below-the-bar area. DB2 9 for z/OS now places portions of the bound/prepared DML statements (SQL statement text, SQLDA for DESCRIBE output, part of the native SQL PL package, and some other areas) into storage above the 2 GB bar.

This should provide further virtual storage constraint relief below the bar. The infrastructure is now in place to move other runtime structures above the bar in the future.

Archived

In Figure 3-3, you can see additional structures that have been moved above the 2 GB bar. In DB2 9 for z/OS the EDMPOOL plan and package skeletons (SKCT/SKPT) are now above the 2 GB bar. Also, static SQL sections (CT/PT) are split between above and below the 2 GB bar. Some storage acquired for distributed applications and some storage acquired for dynamic SQL statement execution also moved above the 2 GB bar (the Parse Trees and portion of Run Time blocks moved above the 2 GB bar). The goal is to reduce virtual storage requirements below the 2 GB bar by 10 to 15% for both static SQL and dynamic SQL. Your savings will vary depending on the current storage utilization of your workload mix.

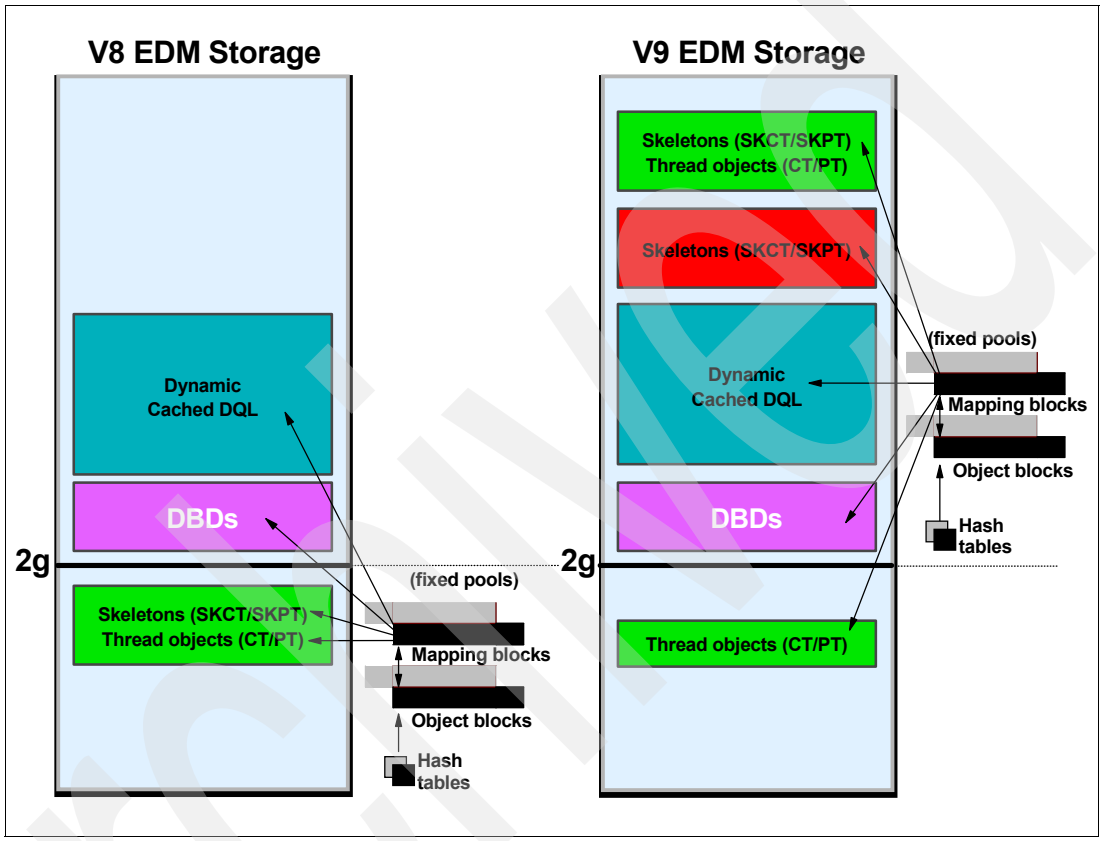


Figure 3-3 Additional runtime structures above the 2 GB bar

3.2 WLM assisted buffer pool management

z/OS 1.8 delivers new WLM services that can assist DB2 in making dynamic buffer pool size adjustments based on real-time workload monitoring. DB2 9 for z/OS in compatibility mode exploits these new services to allow for dynamic buffer pool size adjustments so that the system's memory resources can be more effectively utilized to achieve workload performance goals. This functionality should lead to better usage of existing memory resources for important work and improve throughput of that work. For example, a buffer pool on a non-critical DB2 subsystem can be shrunk to reassign its storage to a buffer pool on a mission-critical DB2 subsystem on the same LPAR if important transactions are not meeting their performance goals.

You can enable or disable this functionality via a new AUTOSIZE(YES/NO) option of the ALTER BUFFERPOOL command at the individual buffer pool level. By default, automatic buffer pool adjustment is turned off. Only the size attribute of the buffer pool is changed.

Automatic management of buffer pool storage entails the following:

- DB2 registers the BPOOL with WLM.
- DB2 provides sizing information to WLM.
- DB2 communicates to WLM each time allied agents encounter delays due to read I/O.
- DB2 periodically reports BPOOL size and random read hit ratios to WLM.

DB2 notifies WLM each time an allied agent encounters a delay caused by a random Get Page having to wait for read I/O. Periodically, DB2 reports to WLM the buffer pool sizes and hit ratio for random reads. WLM maintains a histogram and plots the size and hit ratio over time and projects the effects of changing the size of the buffer pool. It determines the best course of action to take if work is not achieving its goals. If WLM determines that buffer pool I/O is the predominant delay, it will determine whether increasing the buffer pool size could help achieve the performance goal. Depending on the amount of storage available, WLM may instruct DB2 to increase the buffer pool or first decrease another buffer pool and then increase the buffer pool in question. If a buffer pool is adjusted, the results will be just as though an ALTER BUFFERPOOL VPSIZE command had been issued. DB2 9 for z/OS restricts the total adjustment to +/- 25% the size of the buffer pool at DB2 startup. However, be aware that, for example, if a buffer pool size is changed and later DB2 is shut down and subsequently brought up, the last used buffer pool sizes will be remembered across DB2 restarts and you can potentially change that size by another +/- 25% of the new value.

Keep in mind that there are implications to utilizing this feature. Since DB2 V8, buffer pools have been allocated above the 2 GB bar. For each DB2 system, you can define up to a total of 1 TB of virtual space for your buffer pools. If you subscribe to the DB2 V8 recommendation to page fix your I/O-bound buffer pools with low buffer pool hit ratios in order to save the CPU overhead of continually page fixing and freeing those pages, it is now possible that you may add up to 25% more demand on real storage to back those buffer pools. For example, if you have 800 MB of buffer pools defined and they are page fixed, if they grow 25%, you would need an additional 200 MB of real storage to back them. If you do not have the extra capacity or are already paging to auxiliary storage, you could severely impact the operation of your system. We recommend that you closely monitor your real storage consumption when turning on WLM assisted buffer pool management for buffer pools defined with PGFIX(YES).

You can find additional information about WLM assisted buffer pool management in **13.13, “WLM-assisted buffer pool management” on page 530**.

Note: DB2 offers real storage protection for other workloads on the same LPAR from being *squeezed* by page fixing of buffer pools. DB2 will page fix buffer pools up to 80% of the real storage of the z/OS LPAR. However, this is by DB2 subsystems, and if you have more than one DB2 on that LPAR, you could potentially page fix 100% of the real storage.

3.3 NOT LOGGED table spaces

The purpose of this enhancement is two fold. The first purpose allows for the reduction of the amount of information written to the log in situations where absolute recoverability of the data is not required. This will provide for the ability to scale/increase this type of workload without impacting the amount of logging to be performed, and other work that depends on logging for recoverability of the data. A typical example would be for large MQTs.

The second purpose for this enhancement is to relieve scalability issues for those customers who run many concurrent/parallel heavy insert or update batch processes. They will now have the option to turn off logging during those specific batch processes for the objects involved in the heavy insert or update activity. After the batch processes are complete, the

customer can again turn on logging and image copy the objects involved so that they are recoverable. This allows for additional horizontal scalability (ability to run more concurrent/parallel streams) of these types of workload.

Applying a NOT LOGGED logging attribute to a particular table space suppresses only the logging of Undo and Redo information. Control records with respect to that table space, such as open and close page set log records, continue to be logged, regardless of the logging attribute. As a result, there will be a reduction in the amount of recovery log data generated, and there may be an increased speed of log reads for RECOVER and DSN1LOGP.

The suppression of writing log records can be useful in a number of different situations to reduce I/O and latch contention on LOGs, as long as recoverability is preserved by data, in effect being duplicated. If the data is lost, it can be recreated or regenerated from its original source, rather than from an image copy and applying log records.

Figure 3-4 shows a scenario when there may be no need for logging.

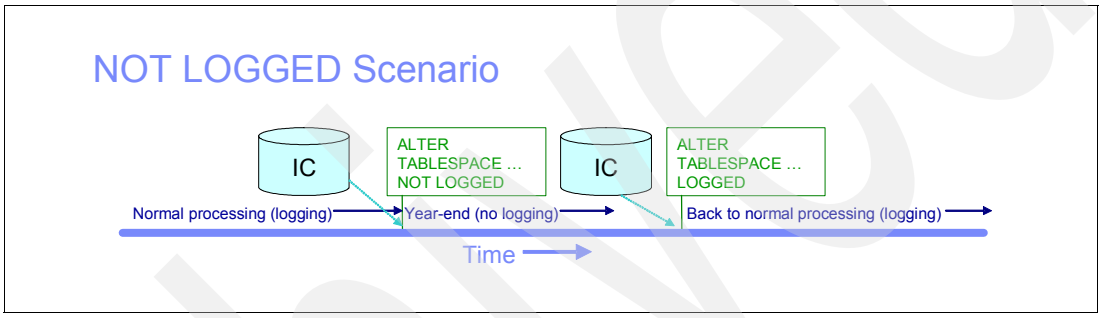


Figure 3-4 NOT LOGGED scenario

For example:

- There are situations where modifications to the data are very infrequent and do not result in very many rows being changed. During this time, the changes to the table would be logged, as usual. However, from time to time (for example, year end processing) changes to the data are extensive and frequent, but the duration of this large amount of change to the data is relatively short. In these cases, it would be advantageous to take an image copy of the data, turn off logging, make the massive changes, turn on logging, and take another image copy. Should anything go awry during the period logging is not done, the original data can be recovered from the prior image copy and the process that makes the massive changes can be rerun.

  You can also use this approach with Online LOAD RESUME processing and also to make modifications to the data after the data is loaded.

  Important: You should take care to avoid other concurrent changes to the data, since those changes would be lost if anything were to go wrong.

- Materialized Query Tables are a summarization of data that exists elsewhere.

  Since an MQT can be rebuilt from data that exists elsewhere, there is no need for the changes to the MQT to be logged.

- Non-MQTs that contain summarized information in other tables.

  These tables would then be directly referenced in SQL statements. Since the information is a summary of data that exists elsewhere, there is no need for the changes to it to be logged.

- It may also be advantageous to avoid writing log records for change to tables to which data is propagated.

  Since the data exists elsewhere, there is no need to log the changes, as they are applied to the copied table. If necessary, the entire table can be refreshed from its source.

- In a Business Intelligence/Query/Reporting environment *temporary* tables are populated with the result set from a query as an intermediate step by the application, and then used by a subsequent query step.

  The results are temporary and are deleted by the application. Data can be recovered by rerunning the application.

This enhancement provides for a new logging attribute that can be specified at a table space level. You can either specify a LOGGED attribute to indicate that modifications should be recorded on the log, or specify a NOT LOGGED attribute to indicate that modifications should not be recorded on the log. The default logging attribute is LOGGED, except for table spaces in the work file database, and LOB table spaces containing LOBs with a maximum length greater than 2 GB. For these types of table spaces, the logging attribute is always NOT LOGGED. You cannot specify a logging attribute of LOGGED.

Figure 3-5 has words of caution about the use of the NOT LOGGED option.

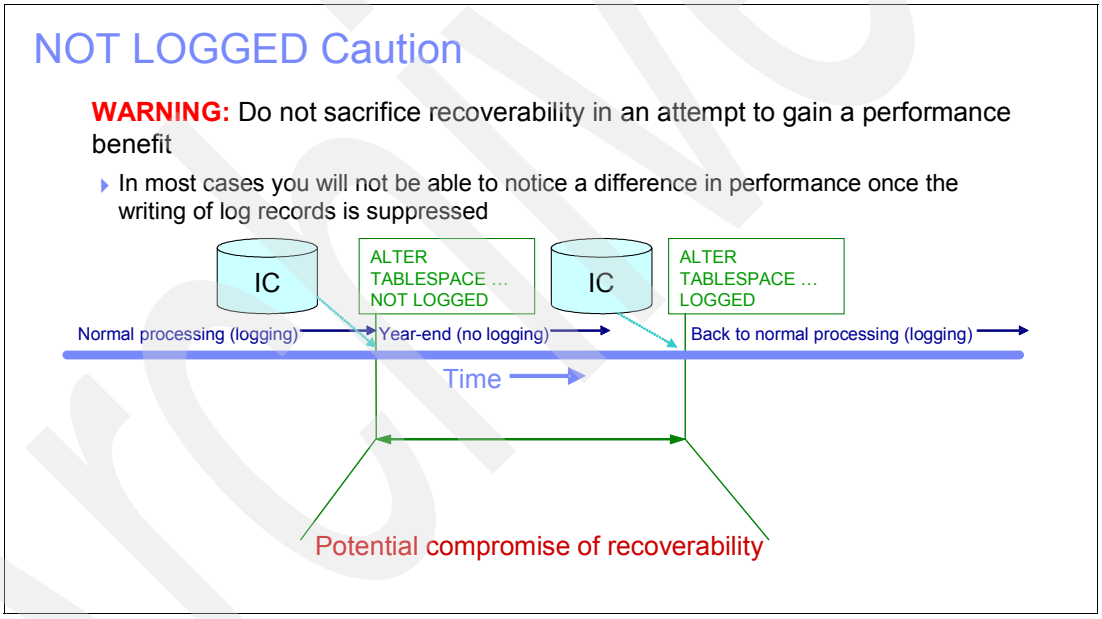


Figure 3-5 Use NOT LOGGED option with caution

Attention: Suppressing the writing of log records is, in general, not going to improve the performance of your system. The logging facilities in DB2 are specifically tailored to the way DB2 writes log records and has been finely tuned. Do not sacrifice the recoverability of your data in an attempt to gain performance because, in the vast majority of situations, you will not be able to notice a difference in performance when the writing of log records is suppressed. There are a number of other ways to avoid logging, such as by using LOAD, work files, declared and global temporary tables, LOBs, and REORG LOG NO.

Archived

3.3.1 The logging attribute

The logging attribute is a table space attribute. For a partitioned table space, the logging attribute applies at partition level. You can have NOT LOGGED for some partitions and LOGGED for other partitions. The default logging attribute is LOGGED.

The logging attribute can be altered. When the logging attribute is altered, it affects all of the tables contained in the table space.

When specifying the logging attributes for a table space, you can specify LOG YES as a synonym for LOGGED, and LOG NO as a synonym for NOT LOGGED. Although these keywords are supported as alternatives, they are not the preferred syntax.

The LOG column in SYSIBM.SYSTABLESPACE is now used for non-LOB table spaces as well as for LOB table spaces.

NOT LOGGED is mutually exclusive with DATA CAPTURE CHANGES. Therefore, any table in the table space with the DATA CAPTURE CHANGE attribute prevents NOT LOGGED from being applied to the table space.

NOT LOGGED cannot be specified for table spaces in the following databases:

- DSNDB06 (the DB2 catalog)
- A work file database
- The real-time statistics database

3.3.2 Indexes

The NOT LOGGED parameter specified either with the CREATE TABLESPACE or ALTER TABLESPACE statement applies to all tables that are created in the specified table space and to all indexes of those tables.

Altering the logging attribute of a table space from LOGGED to NOT LOGGED establishes a point of recovery for the table space. The logging attribute of a COPY YES index automatically changes from LOGGED to NOT LOGGED (due to inheritance from its table space) and establishes a point of recovery for the index if a valid full copy exists for the index. After that, taking each image copy establishes another point of recovery for the table space and its associated indexes.

Altering the logging attribute of a table space from NOT LOGGED to LOGGED marks the table space as Copy Pending (a point of recovery must be established before logging resumes). The indexes (on the tables in the table space) that have the COPY YES attribute are marked as Informational Copy Pending.

You can take full and incremental SHRLEVEL REFERENCE image copies even though the table space has the NOT LOGGED attribute. You cannot take SHRLEVEL CHANGE copies because the NOT LOGGED attribute suppresses the logging of changes necessary for recovery.

XML table spaces and their indexes inherit the logging attribute from the associated base table space. Auxiliary indexes inherit the logging attribute from the associated base table space.

3.3.3 XML table spaces

The LOGGED or NOT LOGGED option cannot be specified for XML table spaces.

An XML table space has the same log attributes as its base table space. If the base table space is logged, the XML table space is logged. If the base table space is not logged, the XML table space is not logged.

When the logging attribute of a base table space is altered from LOGGED to NOT LOGGED, all associated XML table spaces and indexes are also implicitly altered to force their logging attribute to NOT LOGGED to match the base table space. When the logging attribute of an XML table space is implicitly altered in this way, its logging attribute is said to be *linked* to the logging attribute of the base table. This is indicated by a value X stored in column LOG in SYSIBM.SYSTABLES. Should the logging attribute of the base table space be subsequently altered back to LOGGED, all linked XML table spaces are also implicitly altered to return their logging attribute to LOGGED. At this point, the table spaces are no longer linked.

3.3.4 LOB table spaces

Prior to V9, LOB table spaces could have a LOG YES (equivalent of LOGGED) or a LOG NO (equivalent of NOT LOGGED) logging attribute. This logging attribute was independent of the required and implied LOGGED logging attribute of their respective base table spaces.

In V9, it is a requirement that if the base table space has the NOT LOGGED logging attribute, all associated LOB table spaces must also have the NOT LOGGED logging attribute.

If the base table space has a LOGGED logging attribute, the logging attribute of the LOB table space continues to be independent of the base table space. In this case, the LOB table space may have either a LOGGED or a NOT LOGGED logging attribute. Further, the logging attribute of the LOB table space may be altered without restriction. This is as in previous releases.

A LOB table space with the LOGGED logging attribute has its logging attribute altered to NOT LOGGED when the logging attribute of the associated base table space is altered from LOGGED to NOT LOGGED. When this happens, the logging attribute of the LOB table space is said to be linked to the logging attribute of the base table space. This is indicated by a value X stored in column LOG in SYSIBM.SYSTABLES. When the logging attribute of the base table space is altered back to LOGGED, all logging attributes that are linked for the associated LOB table spaces are altered back to LOGGED, and all of these links are dissolved.

You can also dissolve the link between the logging attributes of the base table space and a LOB table space by altering the LOB table space to specify NOT LOGGED, even if it has already been implicitly given this attribute from the base table space. After the link between the logging attributes of the base table space and the LOB table space has been dissolved, the logging attribute of the LOB table space is unaffected when the logging attribute of the base table space is altered.

Table 3-1 shows a progression of SYSIBM.SYSTABLESPACE LOG column values through a set of ALTER TABLESPACE statements that alter the logging attribute of a base table space, and one XML table space and two LOB table spaces associated with a table in the base table space. The values are:

- Y - Logged
- N - Not logged
- X - Not logged, linked to base

Table 3-1 LOG column values

Alter	Base	LOB1	LOB2	XML1	Notes
Initially all LOGGED	Y	Y	Y	Y	
LOB2 to NOT LOGGED	Y	Y	N	Y	
Base to NOT LOGGED	N	X	N	X	LOB1 and XML1 linked to Base.
LOB2 to LOGGED	N	X	N	X	Rejected, SQLCODE -763. It is a LOB table space and LOGGED was specified, but the associated base table space is defined as NOT LOGGED.
Base to LOGGED	Y	Y	N	Y	LOB1-XML1 links dissolved.
LOB2 to LOGGED	Y	Y	Y	Y	
Base to NOT LOGGED	N	X	X	X	LOB1, LOB2, and XML1 linked to Base.
Base to LOGGED	Y	Y	Y	Y	LOB1, LOB2, and XML1 links dissolved.
Base to NOT LOGGED	N	X	X	X	LOB1, LOB2, and XML1 linked to Base.
LOB2 to NOT LOGGED	N	X	N	X	LOB2 link dissolved.
Base to LOGGED	Y	Y	N	Y	LOB1-XML1 links dissolved.

3.3.5 SYSLGRNX and SYSCOPY

SYSLGRNX records are not maintained for NOT LOGGED table spaces.

When a table space is created with the LOGGED option, a SYSCOPY record is written with ICTYPE='C' and STYPE='L'.

When a table space is created with the NOT LOGGED option, a SYSCOPY record is written with ICTYPE='C' and STYPE='O'.

When a table space is altered to LOGGED, a SYSCOPY record is written with ICTYPE='A' and STYPE='L'.

When a table space is altered to NOT LOGGED, a SYSCOPY record is written with ICTYPE='A' and STYPE='O'.

The LRSN in these SYSCOPY records reflects the point in the log at which the logging attribute was altered.

Archive

A new column, LOGGED, is added to SYSCOPY. The values can be:

- Y - to indicate that the logging attribute is LOGGED.
- N - to indicate that the logging attribute is NOT LOGGED.
- blank - to indicate that the row was inserted prior to V9. For a non-LOB table space or an index space, this is an indication that the logging attribute is LOGGED. For LOB table spaces, this does not indicate the logging attribute.

3.3.6 Recovery implications for table spaces that are not logged

You can use the RECOVER utility on table spaces that have the NOT LOGGED logging attribute. The NOT LOGGED attribute does not mean that the contents of a table space are non-recoverable. However, the modifications to a table space that is not logged are non-recoverable.

Recovery can be to any recoverable point. A *recoverable point* is established when a table space is altered from logged to not logged or when an image copy is taken against a table space that is not logged. The TORBA or TOLOGPOINT keywords can also be used for a point-in-time recovery on an object that is not logged, but the RBA or LRSN must correspond to a recoverable point or message DSNU1504I is issued.

If a base table space is altered so that it is not logged, and its associated LOB table spaces already have the NOT LOGGED attribute, then the point where the table space is altered is not a recoverable point.

Informational Copy Pending status

If you update a table space while it has the NOT LOGGED logging attribute, the table space is marked Informational Copy Pending (ICOPY). You can use the DISPLAY DATABASE ADVISORY command to display the ICOPY status for table spaces. For example:

```
DSNT36I1 - * DISPLAY DATABASE SUMMARY

               *     ADVISORY

   DSNT360I - ************************************
   DSNT362I - DATABASE = DBIQUA01 STATUS = RW
              DBD LENGTH = 8066

   DSNT397I -
   NAME      TYPE PART STATUS      HYERRLO PHYERRHI CATALOG PIECE
   --------  ---- ---- ----------- ------- -------- ------- -----
   TPIQUQ01  TS   001  RW,AUXW
   TPIQUQ01  TS   002  RW,AUXW
   TPIQUQ01  TS   003  RW,AUXW
   TPIQUQ01  TS   004  RW,ICOPY
```

To clear the ICOPY status, you must take a full image copy of the table space.

RECOVER-pending status

If it becomes necessary for DB2 to undo work that has not been logged (as when a rollback occurs), the table space has lost its data integrity and is marked RECOVER-pending. To prevent access to corrupt data, the pages are placed in the LPL.

If DB2 restart recovery determines that a not-logged table space may have been in the process of being updated at the time of the failure, then the table space or partition is placed

in the LPL and is marked RECOVER-pending. You have several options for removing a table space from the LPL and resetting the RECOVER-pending status:

- Dropping and re-creating the table space, and repopulating the table
- Using a REFRESH TABLE statement

  Use the REFRESH TABLE statement to repopulate a materialized query table, but only if the materialized query table is alone in its table space. If the table is not alone in its table space, a utility must be used to reset the table space and remove it from RECOVER-pending status.
- Using the RECOVER utility

  Use the RECOVER utility to recover to a recoverable point.

  You can run the RECOVER utility against a table space with the NOT LOGGED logging attribute. To do so, the current logging attribute of the table space must match the logging attribute of the recovery base (that is, the logging attribute of the table space when the image copy was taken). If no changes have been made to the table space since the last point of recovery, the utility completes successfully. If changes have been made, the utility completes with message DSNU1504I.

  You can use RECOVER with the TOCOPY, TOLASTFULLCOPY, or TOLASTCOPY keyword to identify which image copy to use. You can also use TORBA or TOLOGPOINT, but the RBA or LRSN must correspond to a recoverable point.

  You cannot use RECOVER with the LOGONLY keyword.
- Using the LOAD REPLACE utility

  Use the LOAD REPLACE utility or the LOAD REPLACE PART utility in the following situations:

  – With an input data set to empty the table space and repopulate the table
  – Without an input data set to empty the table space to prepare for one or more INSERT statements to repopulate the table
- Using a DELETE statement without a WHERE clause

  Use the DELETE statement without a WHERE clause to empty the table when the table space is segmented or universal. The table is alone in its table space and the table does not have:

  – A VALIDPROC
  – Referential constraints
  – Delete triggers
  – A SECURITY LABEL column (Or it does have such a column, but multilevel security with row level granularity is not in effect.)
- Using a TRUNCATE TABLE statement

  Use the TRUNCATE TABLE statement to empty the table when the table space is segmented and the table is alone in its table space and the table does not have:

  – A VALIDPROC
  – Referential constraints
  – A SECURITY LABEL column (Or it does have such a column, but multilevel security with row level granularity is not in effect.)

Archived

3.3.7 Recovering NOT LOGGED LOB or XML table spaces

Use image copies that are taken for NOT LOGGED table spaces that have associated LOB or XML table spaces as a recovery set, so that the base table space and all the associated LOB or XML table spaces are copied at the same point in time. This way a RECOVER TO LASTCOPY of the entire set results in consistent data across the base table space and all of the associated LOB or XML table spaces.

3.3.8 Deadlock and time-out considerations

Rolling back a unit of recovery that encompasses modifications to a not logged table space has profound effects on that table space. Such a rollback makes the entire table space inaccessible. Since this is the case, when a transaction that has made modifications to a not logged table space (a) is engaged in a deadlock with a transaction that has not made changes to a not logged table space (b), the latter transaction (b) is chosen as the victim of the deadlock regardless of how many log records have been written on its behalf. If all transactions engaged in a deadlock have made modifications to not logged table spaces, the criteria for choosing a victim remains the number of log records written. The one with the least number of log records written is chosen as the victim.

For time outs, a transaction that has made modifications to a not logged table space is assumed to have a time-out multiplier of the maximum of three or the thread's current time-out multiplier.

3.3.9 LOAD and REORG interaction

Table 3-2 shows the effect of LOAD and REORG on NOT LOGGED table spaces.

Table 3-2 LOAD and REORG parameters

LOAD REORG LOG keyword	Table space logging attribute	Table space type	What is logged	Table space status after utility completes
LOG YES	NOT LOGGED	Non-LOB	LOG YES changed to LOG NO	No pending status or ICOPY-pending[1]
LOG YES	NOT LOGGED	LOB	Control information	No pending status
LOG NO	NOT LOGGED	Non-LOB	Nothing	No pending status or ICOPY-pending[1]
LOG NO	NOT LOGGED	LOB	Nothing	No pending status

1. The table space is set to ICOPY-pending status if the records are discarded and no pending status if the records are not discarded.

The LOAD utility with the LOG NO option may be run on a table space with the NOT LOGGED logging attribute. If the LOG YES option is specified, it is changed to LOG NO and the processing continues (message DSNU1153I is issued). Since an inline image copy is taken before discard processing, when a delete is needed due to discard processing, the table space is marked ICOPY pending.

Should an inline image copy be taken as part of such a load, a recover to currency using that image copy results in a condition code of 4 and message DSNU1505I. This message means that a full recovery was requested, but because the object is not a logged object, it was recovered to its last recoverable point (that is, any changes made to the object after the

recoverable point are lost). This message warns that the object has been modified since the last recoverable point. If no inline copy is requested the table space is marked ICOPY.

3.3.10 Further REORG considerations

REORG SHRLEVEL(REFERENCE) or SHRLEVEL (NONE) may be run against a NOT LOGGED table space. For SHRLEVEL NONE, if the LOG YES option is specified, it is changed to LOG NO and message DSNU1153I is issued. If no inline copy is requested the table space is marked ICOPY pending. Note that, for SHRLEVEL REFERENCE, the LOG parameter is not allowed and an inline copy is always produced.

Running REORG TABLESPACE PART SHRLEVEL (REFERENCE) with nonparticipant indexes defined on the partitioned table space is not allowed on a NOT LOGGED table space, due to the log phase that occurs (DSNU1152I).

REORG SHRLEVEL (CHANGE) is not allowed on a NOT LOGGED table space or index space (DSNU1152I).

3.3.11 Online LOAD considerations

LOAD LOG NO has an optional NOCOPYPEND option that indicates that the LOAD should not leave the table space in COPY pending. Since a LOAD run against a table in a NOT LOGGED table space never leaves the table space in COPY pending, the NOCOPYPEND option is ignored. This means that if discard processing is necessary, the table space is marked ICOPY pending even if NOCOPYPEND is specified.

You cannot specify LOG NO for Online Load (LOAD RESUME YES SHRLEVEL CHANGE). Online Load without logging can be accomplished by altering the table space to NOT LOGGED, running the Online Load, altering the table space back to LOGGED, and taking an image copy. However, unlike Offline LOAD RESUME, Online Load against a NOT LOGGED table space is not restartable if the LOAD fails. If an Online Load abends, and rollback is necessary for the NOT LOGGED table space, then the table space is placed in LPL/RECP status. An attempt to restart the LOAD results in a return code of 8 and message DSNU1154I. To recover from such a failure, the failed LOAD job must be terminated, the data must be recovered from a prior image copy, and the Online Load job rerun.

3.3.12 REBUILD INDEX considerations

When the NOT LOGGED attribute is specified at a table space level, the attribute property also is applied to also is applied to indexes defined on XML and LOB tables. This indicates that modifications are not recorded on the log.

Specifying REBUILD INDEX SHRLEVEL(CHANGE) requires log records to be applied. Therefore, REBUILD INDEX SHRLEVEL(CHANGE) is not allowed on a NOT LOGGED table space (DSNU1152I).

3.3.13 CHECK DATA considerations

CHECK DATA supports a list of table spaces. If DELETE YES LOG YES is specified or defaulted, deleted rows are logged. If LOG NO is specified and rows are deleted, the changed table space is marked COPY pending.

For NOT LOGGED table spaces in the list, if LOG YES is in effect, CHECK DATA issues the message DSNU1153I to inform the user that the LOG YES option is ignored. If rows are

deleted in a NOT LOGGED table space, CHECK DATA sets ICOPY pending regardless of the LOG option.

3.3.14 QUIESCE considerations

QUIESCE on a table space with the NOT LOGGED attribute does not create a recoverable point, because there are no log records that a subsequent RECOVER utility can apply to recover the data to the quiesced point.

If Quiesce with the WRITE(NO) option is requested on a NOT LOGGED table space, then informational message DSNU485I is issued and the object is skipped. If QUIESCE with the WRITE(YES) option is requested on a NOT LOGGED table space, the table space and its indexes are drained of writers, and the corresponding pages are written from the buffer pool to DASD.

3.3.15 Application programming considerations

Application programmers should commit frequently and try to avoid duplicate key or referential integrity violations when modifying a table in a NOT LOGGED table space. They should also take care to avoid issuing a ROLLBACK after modifying a table in a NOT LOGGED table space. This includes ROLLBACK TO SAVEPOINT.

3.3.16 Operational considerations

Since the data in a NOT LOGGED table space is not protected by information on the log, it is desirable to externalize modifications to this data relatively quickly, but without elongating the commit processing or elapsed time. To accomplish this, RO SWITCH CHKPTS and RO SWITCH TIME are both considered to be set to 1 for not logged table spaces.

RO SWITCH CHKPTS and RO SWITCH TIME together establish the amount of time to elapse between the last update of a table space and when DB2 converts it from read-write to read-only. When a table space is converted from read-write to read-only, among other things, its modified pages are written from its buffer pool to external media.

RO SWITCH CHKPTS on installation panel DSNTIPL (ACTIVE LOG DATA SET PARAMETERS) corresponding to DSNZPARM PCLOSEN indicates the number of consecutive DB2 checkpoints since a table space was last updated after which DB2 converts the table space from read-write to read-only. The number specified for this system parameter applies only to logged table spaces. For not logged table spaces, RO SWITCH CHKPTS is considered to be 1. This means that when a DB2 checkpoint occurs, all read-write not logged table spaces that are not in current use are converted to read-only.

RO SWITCH TIME on installation panel DSNTIPL (ACTIVE LOG DATA SET PARAMETERS) corresponding to DSNZPARM PCLOSET indicates the number of minutes since a table space was last updated after which DB2 converts the table space from read-write to read-only. The number specified for this system parameter applies only to logged table spaces. For not logged table spaces, RO SWITCH TIME is considered to be 1. This means that approximately one minute after the commit of the last update to a not logged table space, it is converted from read-write to read-only, if a DB2 checkpoint has not already done that conversion.

DB2 writes the table space from the buffer pool to external media when it converts the table space from read-write to read-only, externalizing any unprotected modifications to the data.

The DSN1LOGP SUMMARY(YES) report is enhanced to show whether an object is logged or not logged, when that is possible. Depending on the log range specified for DSN1LOGP, it may not be possible to say correctly whether the object is logged or not logged. That accuracy depends on whether the range of log records that is specified contains the Begin UR log record. Also, DSN1LOGP formatting of the page set control records (PSCRs) is enhanced to give this indication.

3.3.17 Performance objectives

The main performance objective for this enhancement is to maintain current performance regardless of whether INSERTS are done against a logged or not logged table space. In some cases, parallel INSERTS to a not logged table space may show significant (greater than 5%) improvement over a logged table space.

For those cases where parallel INSERTS are done against a not logged table space, performance is expected to show a greater than 10% improvement.

3.4 Other logging enhancements

DB2 9 for z/OS implements various logging enhancements to remove previous restrictions that caused bottlenecks in logging activity. In this section, we discuss these enhancements.

Remove data sharing LRSN increment wait

In data sharing prior to DB2 9 for z/OS, the LRSN value of each log record on a given member had to be unique. If two successive log records for the same member had the same LRSN value, DB2 would re-drive the store clock (STCK) for the LRSN until a different one was generated. The LRSN value increments every 16 microseconds. On today's processors, 16 microseconds is a long time to wait, and this can cause significant delays for workloads that have a high logging rate. In DB2 9 for z/OS NFM and ENFM modes, LRSN values only have to be unique within a given page. That is, successive updates, inserts, and deletes into a given page must have unique LRSNs in order to guarantee the order the changes arrived. So, DB2 9 for z/OS (NFM and ENFM) only drives the LRSN increment wait when successive updates, inserts, and deletes into a given page have the same LRSN value. With this enhancement DB2 also removes the need to keep holding the log latch (LC19) while the LRSN is incremented. This should provide for less wasted resources and increased log and transaction throughput. Initial testing has shown up to nearly a two times improvement in logging rate through the reduction of how often DB2 performs the LRSN spin and the corresponding reduction of the log latch. Log latch contention may be significantly reduced or eliminated for data sharing workloads that were experiencing contention.

Note: DSN1LOGP output may now include multiple log records with the same LRSN value on a single DB2 data sharing member. DSNJU004 may list consecutive active or archive log data sets where the end LRSN value of the first is the same as the beginning LRSN value of the second. These conditions can occur in DB2 9 for z/OS NFM, ENFM, or in CM after falling back to that mode.

Convert disk archive log reads from BDAM to BSAM

Prior to DB2 9 for z/OS, all archive log reads were done using BDAM. While BDAM allows for rapid access to the target CI, it does not provide support for striped data sets or DFSMS DASD compression (Extended Format data sets). Since active logs support Extended Format data sets and can be striped, it takes longer to write an archive log copy than the active log that the archive data is coming from. If this situation continues on a busy system for a

sufficient period of time, the possibility exists that logging may be suspended while waiting for the archive log processing to catch up. Beginning with CM, DB2 9 for z/OS uses BSAM to read DASD archive logs (QSAM is still used to write archive logs), and in DB2 9 for z/OS NFM these data sets can be defined as Extended Format data sets. This allows DASD archive logs to be striped and compressed. With striped archive logs, DB2 can now match the performance of striped active logs, thus ensuring that as logging volumes increase, log archiving can scale at the same rate.

Note: After converting to NFM, if you decide to have archive logs striped or compressed, any recovery actions involving those archive logs must be performed on a DB2 V9 system. You cannot create archive logs as extended format data sets while in CM, but reading them in CM is tolerated. Extended Format data sets must be SMS managed.

Convert archive log processing from AMODE(24) to AMODE(31)

Prior to DB2 9 for z/OS, some customers had reported running short of storage below the 16 MB line in the MSTR address space. Part of the problem was the fact that DB2 used 24-bit mode to process log buffers used to read and write archive logs, and this required to define them below the 16 MB line (this also significantly limited the size of these buffers). In order to alleviate the storage constraint and to provide for increased archive log processing, archive log processing (reads and writes) in the MSTR address space is converted from 24-bit mode to 31-bit mode. By converting to using 31-bit mode, DB2 can now move these buffers above the 16 MB line, allowing them to be much larger. In addition, DB2 now uses dual buffering (fill the next buffer while processing the current one) for archive log reads when possible and the size of the log buffers has been greatly increased. This is expected to result in improvements of archive log read and write performance and throughput. This enhancement is in effect starting with V9 in compatibility mode.

Exploit z/OS DSNTYPE=LARGE disk data set support

Since the inception of the *x*SAM access methods, there has been a sequential and partitioned data set size limit of 64 KB tracks per DASD volume. z/OS 1.7 lifts this limit with the introduction of DSNTYPE=LARGE support. The new limit for these types of data sets is 16 million tracks on a single DASD volume (or across 59 DASD volumes). DB2 currently supports active logs as large as 4 GB. You can define them larger but DB2 only uses 4 GB. For 3390 geometry DASD, this requires 87,382 tracks, and for 3380 geometry the requirement is 104,858 tracks. Both of these are above the 64 KB track limit, forcing installations that use 4 GB logs to put archive logs on tape or to use multi-volume archive logs. By exploiting the new z/OS support for large data sets, installations will be allowed to have 4 GB DASD archive logs. This may simplify management of DASD archive logs since 4 GB active logs will no longer require multi-volume allocations. You can create these large DASD archive logs only in NFM or ENFM modes. However, DB2 tolerates reading them in CM.

3.5 Exploit WLM to resolve latch contention

Two issues that have sometimes impacted DB2 in the past were CPU stalls causing latch contention and DBM1 below the bar storage constraints. While CPU stalls and storage constraints can be identified, DB2 did not provide automated means to identify such issues. DB2 V7 introduced a serviceability command, DISPLAY THREAD(*) SERVICE(WAIT), to help identify and rectify some CPU stalls. But, this was a manual process. Customers first had to identify that they were in this type of situation and then manually issue the command. Typically, most customers identified DBM1 storage constraint issues analyzing the IFCID 225 records with a DB2 monitor program or home-grown solution. Because the command and the

IFCID 225 record analysis were not automated, many customers avoided such monitoring altogether. This often led to unhealthy systems and outages.

With DB2 9 for z/OS in compatibility mode, you get the benefit of a built-in monitor that will help automate this type of problem identification and correction. The built-in monitor will run from restart to shutdown and will check the health of the system in 1-minute intervals. The built-in monitor will identify CPU stalls (for system, DBAT, and Allied agents) that result in latch contention. The monitor will attempt to clear the latch contention via a temporary priority boost via WLM services to the latch holder. This should allow customers to run closer to 100% CPU utilization by reducing the chances that less important work can hold a latch for an extended period of time, causing important work to stall. In addition, DBM1 storage below the 2 GB bar will be monitored for critical storage increases. You can view the health of your system by issuing the DISPLAY THREAD(*) TYPE(SYSTEM) command, as shown in Figure 3-6.

```
-DISPLAY THREAD(*) TYPE(SYSTEM)
DSNV401I  -DB9B DISPLAY THREAD REPORT FOLLOWS -
DSNV497I  -DB9B SYSTEM THREADS -  778
DB2 ACTIVE
NAME     ST A   REQ ID           AUTHID   PLAN     ASID TOKEN
DB9B     N  *     0 002.VMON  01 SYSOPR            0069     0
V507-ACTIVE MONITOR, INTERVALS=429522, STG=7%, BOOSTS=0, HEALTH=100%
. . .
```

Figure 3-6 DISPLAY THREAD(*) TYPE(SYSTEM) output

When DBM1 storage below the 2 GB bar reaches a thresholds of 88, 92, 96, or 98% of the available storage, messages DSNV508I, DSNV510I, DSNV511I, and DSNV512I will be issued, reporting current DBM1 storage consumption and the agents consuming the most storage. See Figure 3-7 for a sample DSNV508I, DSNV510I, and DSNV512I messages.

```
DSNV508I -SE20 DSNVMON - DB2 DBM1 BELOW-THE-BAR 09
STORAGE NOTIFICATION
     91% CONSUMED
     87% CONSUMED BY DB2

DSNV510I -SE20 DSNVMON - BEGINING DISPLAY OF LARGEST
STORAGE CONSUMERS IN DBM1
DSNV512I -SE20 DSNVMON - AGENT 1: 094
NAME    ST A  REQ ID           AUTHID   PLAN
----    -- -  --- --           ------   -----
SERVER  RA * 18461 SE2DIA004     R3USER   DISTSERV
LONG 1720K VLONG 388K 64BIT 2056K
DSNV512I -SE20 DSNVMON - AGENT 2: 095
NAME    ST A  REQ ID           AUTHID   PLAN
----    -- -  --- --           ------   -----
SERVER  RA * 9270  SE2DIA001     R3USER   DISTSERV
LONG 1672K VLONG 388K 64BIT 2056K
```

Figure 3-7 Sample DSNV508I, DSNV510I, and DSNV512I messages

Customers can easily write automation based on these messages and take proactive actions to prevent the problem from becoming serious. Furthermore, the DISPLAY THREAD command is extended to include STORAGE as an option. You can now issue the DISPLAY THREAD(*) SERVICE(STORAGE) command, as shown in Figure 3-8.

```
DSNV401I  -DB9A DISPLAY THREAD REPORT FOLLOWS -
DSNV402I  -DB9A ACTIVE THREADS -
NAME     ST A   REQ ID           AUTHID   PLAN     ASID TOKEN
RRSAF    T        8 DB9AADMT0066 STC      ?RRSAF   0081     2
V492-LONG 140 K VLONG 28 K 64 1028 K
RRSAF    T     4780 DB9AADMT0001 STC      ?RRSAF   0081     3
V492-LONG 140 K VLONG 28 K 64 1028 K
BATCH    T  *    41 PAOLOR7C     PAOLOR7  DSNTEP91 002D    63
V492-LONG 716 K VLONG 404 K 64 1028 K
TSO      T  *     3 PAOLOR7      PAOLOR7           0068    64
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I  -DB9A DSNVDT '-DIS THREAD' NORMAL COMPLETION
***
```

Figure 3-8 DISPLAY THREAD(*) SERVICE(STORAGE) output

This feature should correct some problems, help provide early warning of others, and at least provide additional diagnostic information. This automated monitor should help lower the cost of ownership for DB2 and help ensure customers maintain healthy DB2 systems.

Note: In order to properly detect these CPU stalls, we recommend that you run the started task for DB2 MSTR in the SYSSTC dispatching priority.

3.6 Universal table space

In DB2 V4 and prior releases, the maximum table space size was 64 GB. This limit was for partitioned as well are non-partitioned table spaces. DB2 V5 introduced the concept of LARGE partitioned table spaces. This allowed for 254 partitions of 4 GB each for a maximum partitioned table space size of 1 TB. DB2 V6 introduced extended format data sets that allowed us to have partitions (a single data set) up to 64 GB in size and a maximum partitioned table space size of 16 TB. In V8 the maximum number of partitions was raised from 254 to 4096 partitions. This allowed a 32 KB page size partitioned table space to grow to a maximum of 128 TB. As you can see, these have been tremendous strides in the scalability of the size of partitioned table spaces. Traditionally, as non-partitioned table spaces neared the 64 GB limit, customers converted them to partitioned table spaces in order to grow beyond 64 GB.

When converting to a partitioned table spaces, an adequate partitioning key had to be defined. This is not always simple, and is sometimes even impossible (sometimes an arbitrary column was chosen for the partitioning key). This sometimes caused growth management issues (re-partition to balance the partition sizes, add partitions, remove partitions, and so on). Although relief was provided in DB2 V8 with table controlled partitioning and the ability to add, rotate, and rebalance partitions, a simpler method was needed to be able to scale the size of non-partitioned table spaces.

DB2 9 for z/OS introduces the universal table space. Prior to DB2 9 for z/OS, you could not define a table space using both the SEGSIZE and NUMPARTS parameters. These parameters were mutually exclusive. In DB2 9 for z/OS this restriction is removed and the

table spaces that are both segmented and partitioned are called universal table spaces (UTSs). DB2 9 for z/OS introduces two flavors of universal table spaces — partition-by-growth (PBG) and partition-by-range (PBR). Below we discuss these at a high level. You can find more details on UTS in 4.1, “Universal table space” on page 74, and in the *DB2 Version 9.1 for z/OS SQL Reference,* SC18-9854.

3.6.1 Partition by growth

Partitioned-by-growth UTS have better space management and improved delete performance than traditional partitioned table spaces due to their segmented space organization. There is no need to define a partitioning key to bound the data within a table space. Partitioning is managed by DB2 depending on the DSSIZE and MAXPARTITIONS chosen. Space must be STOGROUP defined, and only non-partitioning indexes (NPIs) are allowed to be defined on PBG UTS. Free space, caching, define, logging, and TRACKMOD attributes are the same for each partition, and each partition will have the same compression dictionary if compressed.

When you define a PBG UTS, DB2 starts out with one partition. When a partition is filled, DB2 automatically defines the next partition and starts using it (just like extending a no-partitioned table space). So this table space acts like a non-partitioned table space but with the added benefit of being able to grow to a maximum of 128 TB in size (for 32 KB page size). You also get the benefit of running utilities in parallel (at the partition level), except LOAD, which must be run at the table space level. Running a REORG will eliminate holes (deleted rows) and condense the entire table space, so it is possible to have empty partitions at the end of the table space. There is no need to ALTER ADD PART, ALTER ROTATE PART, or DROP PART to manage the partitions (as a matter of fact, you will not be allowed to issue these statements against a PBG UTS). You can find more details on PBG UTS in 4.1.1, “Partition-by-growth table spaces” on page 74, and in the *DB2 Version 9.1 for z/OS SQL Reference,* SC18-9854-00.

3.6.2 Partition by range

Partitioned-by-range UTS are just like existing partitioned table spaces but are segmented. Even though PBR UTS are segmented, in DB2 9 for z/OS, you can only define *one table in them*. PBR UTS have better space management and improved delete performance due to their segmented space organization. You can now convert existing segmented table spaces that are reaching the 64 GB limit to PBR UTS and maintain most of the benefits of segmented table spaces (except you can only have one table per table space). PBR UTS can be up to 128 TB in size and you can choose your DSSIZE. You can find more details on PBR UTS in 4.1.2, “Range-partitioned universal table spaces” on page 80, and in the *DB2 Version 9.1 for z/OS SQL Reference,* SC18-9854.

3.7 Reordered row format

Prior to DB2 9 for z/OS, each variable length column in a data row was stored with its length preceding it. If you needed to access a column that was preceded by a variable length column, DB2 had to traverse through all the columns that preceded it, starting with the first variable length column until it reached the column you wanted to process. If you updated a variable length column, DB2 logged the changed column and everything to the end of the row. For these reasons, we used to recommend that all variable length columns be placed at the end of the row and the most frequently changed ones at the very end of the row. In today’s world of ERP, CRM and many other canned applications, it is impossible for customers to control the placement of columns. DB2 9 for z/OS introduces a new format for data rows

called *reordered row format* that helps reduce the impact and overhead of processing variable length columns.

In DB2 9 for z/OS NFM, you do not have to worry about the order of columns within the row. You can specify the order however you wish, and DB2 automatically reorders the columns within the row and places all of the variable length columns at the end of the physical row within the data page. Instead of storing the columns with their length, DB2 stores all of the offsets for the variable length columns in an area following the last fixed length column and prior to the first variable length column. DB2 can now directly access each variable length column and know its length by doing simple calculations with the offsets. This is a much more efficient way to process individual varying length columns. This enhancement may also help reduce the amount of logging perform for canned applications since all the variable length columns will physically be located at the end of the row no matter how the vendor defined the order of the columns.

Let us take a look at an example of how this is done. We used the statement shown in Figure 3-9 to create a table with four columns, two variable length character columns, and two fixed length character columns. In our example, the variable length columns are in the middle of the row. We also inserted two rows.

```
CREATE TABLE RRF
    (C1 CHAR(10),
     C2 VARCHAR(20),
     C3 VARCHAR(15),
     C4 CHAR(10));

INSERT INTO RRF
VALUES('A11111111A','A2222222222222A','A33333333A','A44444444A');

INSERT INTO RRF
VALUES('B11111111B',NULL,'B3333333333333B',NULL);
```

Figure 3-9 CREATE TABLE RRF and INSERT statements

As you can see from the DSN1PRNT output in Figure 3-10 on page 65, in the first row (record at offset ‘00CF’ within the page), the data for the fixed length columns (A11111111A for column C1 and A44444444A for column C4) is at the front of the row and the data for the variable length columns (A2222222222222A for column C2 and A33333333A for column C3)

Archive

is at the end of the row. Immediately after the last fixed length column (at location x'16' into the row) we find the two 2-byte offsets to the two varying length columns (x'001A' for column C2 and x'002A for column C3).

```
PAGE: # 00000002 ------------------------------------------------------------------------------------
DATA PAGE:  PGCOMB='10'X  PGLOGRBA='00002C3C3D8A'X  PGNUM='00000002'X  PGFLAGS='00'X  PGFREE=3952
            PGFREE='0F70'X  PGFREEP=2D7  PGFREEP='00CF'X  PGHOLE1='0014'X  PGMAXID='02'X  PGNANCH=1
PGTAIL:  PGIDFREE='00'X  PGEND='N'
ID-MAP FOLLOWS:
01  009E 0059

RECORD:  XOFFSET='00CF'X  PGSFLAGS='02'X  PGSLTH=59  PGSLTH='003B'X  PGSOBD='0000'X  PGSBID='01'X
00C1F1F1 F1F1F1F1 F1F1C100 C1F4F4F4 F4F4F4F4 F4C10001A 002A00C1 F2F2F2F2   .A11111111A.A44444444A.....A2222
F2F2F2F2 F2F2F2F2 F2C100C1 F3F3F3F3 F3F3F3F3 C1                           222222222A.A33333333A

RECORD:  XOFFSET='010A'X  PGSFLAGS='02'X  PGSLTH=49  PGSLTH='0031'X  PGSOBD='0000'X  PGSBID='02'X
00C2F1F1 F1F1F1F1 F1F1C2FF 00000000 00000000 0000001A 001BFF00 C2F3F3F3   .B11111111B.................B333
F3F3F3F3 F3F3F3F3 F3F3C2                                                 3333333333B

DSN1994I DSN1PRNT COMPLETED SUCCESSFULLY,  00000003 PAGES PROCESSED
```

Figure 3-10 DSN1PRNT of rows in reordered row format

In the second row (record at offset '010A' within the page), the data for the fixed length columns (B11111111B for column C1 and nulls for column C4) is at the front of the row and the data for the variable length columns (nulls for column C2 and B3333333333333B for column C3) is at the end of the row. Immediately after the last fixed length column (at location x'16' into the row) we find the two 2-byte offsets to the two varying length columns (x'001A' for column C2 and this time x'001B for column C3).

Using the example in Figure 3-9 on page 64, if C3 and C4 were updated, V8 would log C3 and C4, but V9 would log C4, C2, and C3. On the other hand, if C3 was updated, V8 would log C3 and C4, but V9 would log C3 only. So reordered row format can either increase or decrease the amount of log data. As a general rule, if you have both fixed and varying length columns in a row and varying length columns are spread throughout the row (not at the end), the more columns you have in the row, the higher the possibility of reducing the amount of logging being done in DB2 9 for z/OS.

As you can see in Figure 3-11, even though the physical order of the columns within the row has changed, if you do a SELECT * to retrieve all of the columns, DB2 still returns them in the order specified in your define statement.

```
---------+---------+---------+---------+---------+---------+---------+
 SELECT * FROM RRF;
---------+---------+---------+---------+---------+---------+---------+
C1         C2                  C3              C4
---------+---------+---------+---------+---------+---------+---------+
A11111111A  A2222222222222A       A33333333A       A44444444A
B11111111B  --------------------  B3333333333333B  ----------
DSNE610I NUMBER OF ROWS DISPLAYED IS 2
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
```

*Figure 3-11 SELECT * output showing unchanged column sequence*

The first time you run a reorg or load replace on DB2 9 for z/OS NFM, DB2 automatically converts the table space from the old basic row format to the new reordered row format. With reordered row format, DB2 no longer stores the length of varying length columns, and the data can look significantly different within the row/page. The first time you REORG a compressed table space on DB2 9 for z/OS NFM, DB2 will automatically build a new

compression dictionary even if you specified the KEEPDICTIONARY keyword in your REORG/LOAD control statement.

Important: When using a DSN1COPY of a table space to populate another table space, you need to be sure that the row formats of the DSN1COPY and the table space to be populated match. The results of using a DSN1COPY to populate a table space with a different row format are unpredictable.

If all of the columns in a row are VARCHAR, prior to V9 DB2 needed to do offset calculations to access each column, and this could consume a lot of CPU. With the new reordered row format DB2 9 for z/OS no longer needs to do this processing and all columns are directly accessible, thus reducing the cost of processing columns. There is no negative impact in performance if you do not have any varying length columns.

3.8 Index compression

In data warehouse type applications, it is common to have many very large indexes defined on large tables. It is also possible to have index spaces that are much larger than the tables they are based upon. In order to lower the cost of ownership and to improve scalability (more index entries in a single physical index data set), in DB2 9 for z/OS NFM you can choose to compress your indexes and you can define indexes with page sizes larger than 4 KB.

Unlike table compression, the compression mechanisms implemented for index compression do not require a compression dictionary. Because DB2 performs dictionary-less compression, newly created indexes may begin compressing their contents immediately. The implementation of index compression only compresses the data in the leaf pages. The goal is to reduce the cost of ownership, hence index pages are stored on disk in their compressed format (physical 4 KB index page on disk) and will be expanded when read from disk into 8 KB, 16 KB, or 32 KB pages. In order to turn on compression for an index, it must be defined in an 8 KB, 16 KB, or 32 KB buffer pool.

The CREATE INDEX and ALTER INDEX now have a new keyword COMPRESS YES/NO. If you ALTER INDEX COMPRESS YES/NO, the index is placed in Rebuild Pending. Compression takes place for the whole index. It cannot be activated at the partition level.

Note: IMAGE COPY of a compressed index creates an uncompressed format output file. This will likely cause the size of the space required to store the image copy data set to exceed the size of the source index.

As with table compression, you can use DSN1COMP to estimate the effectiveness of compressing an index. This helps you decide which indexes will benefit from compression. No parameters are required and only one parameter is allowed for index estimates, PARM='LEAFLIM(x)', where *x* is the number of leaf pages to be scanned. If you wish to scan all index pages, do not specify the LEAFLIM parameter. The estimate will provide you with an evaluation of compression using different index page sizes. It is very important that you do not choose a larger index page size than what is recommended. You can end up wasting valuable buffer pool space if you do. In Figure 3-12 on page 67 you can see that, as the page size increases, there is more unused space. In this example, it would be best to specify an 8 KB page size, specifying a larger page size would waste too much space. A good rule of thumb is to stop below 50% unused space when choosing the compressed index page size.

```
DSN1999I START OF DSN1COMP FOR JOB PAOLOR8C S0010
DSN1998I INPUT DSNAME = DB9BU.DSNDBC.DSN8D91A.XEMP1.I0001.A001      , VSAM

DSN1944I DSN1COMP INPUT PARAMETERS
                 PROCESSING PARMS FOR INDEX DATASET:
              10  LEAF LEAFLIM REQUESTED

DSN1940I DSN1COMP COMPRESSION REPORT

              10  REQUESTED LEAF LIMIT NOT REACHED
               1  Index Leaf Pages Processed
              32  Keys Processed
              32  Rids Processed
               0  KB of Key Data Processed
               0  KB of Compressed Keys Produced


                 EVALUATION OF COMPRESSION WITH DIFFERENT INDEX PAGE SIZES:

                 ------------------------------------------
               8  K Page Buffer Size yields a
              37  % Reduction in Index Leaf Page Space
                 The Resulting Index would have approximately
              63  % of the original index's Leaf Page Space
              20  % of Bufferpool Space would be unused to
                 ensure keys fit into compressed buffers
                 ------------------------------------------

                 ------------------------------------------
              16  K Page Buffer Size yields a
              38  % Reduction in Index Leaf Page Space
                 The Resulting Index would have approximately
              62  % of the original index's Leaf Page Space
              59  % of Bufferpool Space would be unused to
                 ensure keys fit into compressed buffers
                 ------------------------------------------

                 ------------------------------------------
              32  K Page Buffer Size yields a
              38  % Reduction in Index Leaf Page Space
                 The Resulting Index would have approximately
              62  % of the original index's Leaf Page Space
              79  % of Bufferpool Space would be unused to
                 ensure keys fit into compressed buffers
                 ------------------------------------------


DSN1994I DSN1COMP COMPLETED SUCCESSFULLY,             1  PAGES PROCESSED
```

Figure 3-12 Sample DSN1COMP output for estimating index compression

Archived

Note: Index compression always compresses down, whatever index page size you choose, to a 4 KB page on disk. As inserts and deletes are processed, DB2 keeps track of the available space remaining for compressed data (keys/RIDs) as it would fit on disk (4 KB). Once DB2 determines that there is no more available space in the compressed version of the page, DB2 does not allow additional inserts against the non-compressed index page. This can cause you to waste space in the uncompressed index page and your buffer pool pages. It is best to choose a buffer/page size that minimizes unused buffer space. One size does not fit all.

Utility DSN1PRNT is changed to print the contents of compressed VSAM index data sets. If you specify PARM='PRINT', unformatted compressed 4 KB pages will be printed. If you specify PARM='PRINT,EXPAND', unformatted uncompressed 8 KB, 16 KB, or 32 KB pages will be printed. If you specify PARM='PRINT,FORMAT,EXPAND', formatted uncompressed 8 KB, 16 KB, or 32 KB pages will be printed. If you specify PARM='PRINT,FORMAT', formatted header portions of pages will be printed. DSN1COPY automatically manages an 8 KB, 16 KB, or 32 KB IMAGE copy datasets for a compressed 4 KB index space physical VSAM file

Compressed indexes are only available in NFM, and compression applies to the whole index, not at the index partition level.

3.9 Native SQL procedures

Starting with DB2 V8, all new stored procedures were required to run under a WLM-managed stored procedure address space. Running stored procedures under a WLM address space incurred the overhead of communications between the WLM address space and DBM1 address space for each SQL call. For systems running heavy stored procedure workloads composed of many short-running stored procedures, at or near 100% CPU utilization, this added overhead could potentially inhibit the amount of work being processed. DB2 9 for z/OS provides support for native stored procedures that run entirely within DB2 and do not require a WLM stored procedure address space. DB2 9 eliminates the cross address space overhead between the DBM1 and stored procedure address space as well as the round-trip between Workload Manager and DBM1. This can produce a significant increase in throughput for short-running stored procedures. There will be very little difference for long-running stored procedures. An added benefit of native procedures is that they run under an SRB and are now zIIP eligible.

Native SQL Procedural Language is fully integrated into DB2. It provides improved compatibility with DB2 for Linux, UNIX, and Windows and eliminates the SQL Procedural Language's prior requirement for a C compiler. SQL Procedure Language now has extensive support for versioning, including a VERSION keyword on CREATE PROCEDURE, a CURRENT ROUTINE VERSION special register, ALTER ADD VERSION, ALTER REPLACE VERSION, and ALTER ACTIVATE VERSION. You can BIND a package with the new DEPLOY keyword.

When you create a native SQL procedure, its procedural statements are converted to a native representation that is stored in the database directory, as is done with other SQL statements. The parameter list and procedure options are stored in the database catalog tables as in the prior releases. When a native SQL procedure is called, the native representation is loaded from the directory and the DB2 engine executes the procedure.

The intent is to simplify the definition and use of SQL procedures, which currently involves multiple setup steps and a level of complexity in the build process. It also provides additional

capability that makes SQL procedures more usable and a more viable alternative to having to use external procedures. See also 7.14, “Native SQL procedures” on page 218.

In terms of performance, implementing support for SQL procedures in the engine rather than generating a C routine means that general performance improvements can be expected for applications for which the API trips were a significant contributing factor to the total execution time. This is because the native SQL procedure execution is now done with a single SQL API call for the entire routine.

3.10 Other enhancements

We list other miscellaneous functions related to subsystem functions that help with the scalability issues.

3.10.1 Support for Internet Protocol Version 6 (IPv6)

DB2 9 for z/OS supports Internet Protocol Version 6. IPv6 is the next generation of the Internet protocol designed to replace the current version, Internet Protocol Version 4 (IPv4). Most of today's Internets use IPv4, which is approximately 20 years old and is approaching the end of its physical limits. The most significant issue surrounding IPv4 is the growing shortage of IPv4 addresses. In theory, the 32-bit IPv4 address allows over 4 billion nodes, each with a globally unique address. In practice, the interaction between routing and addressing makes it impossible to exploit more than a small fraction of that number of nodes. Consequently, there is a growing concern that the continued growth of the Internet will lead to the exhaustion of IPv4 addresses early in the 21st century. IPv6 fixes a number of problems in IPv4, such as the limited number of IPv4 addresses. IPv6 uses 128-bit addresses, which theoretically provides 6.6×10^{23} addresses per square meter of the planet’s surface, an address space large enough to last for the foreseeable future. It also adds many improvements to IPv4 in areas such as routing and network auto-configuration. IPv6 is expected to replace IPv4, with the two coexisting for a number of years during the transition period. DB2 9 for z/OS exploits the functions provided in IPv6, the most obvious benefit being the increased address space, and to allow interaction with both IPv4 and IPv6 partners. To utilize IPv6 addresses, DB2 9 for z/OS requires z/OS Communications Server Version 1 Release 5, meaning that DB2 9 for z/OS requires z/OS V1R5. Although z/OS CS V1R4 was the first release to incorporate IPv6 features, not all IPv6 features were supported. z/OS V1R5 Communications Server made further improvements to its IPv6 function that DB2 9 for z/OS can utilize.

You can transition to IPv6 by implementing TCP/IP dual-mode stack. To enable TCP/IP for IPv4/IPv6 dual-mode stack, modify the BPXPRMxx member to define two NETWORK statements, one for AF_INET and another for AF_INET6. Figure 3-13 shows two NETWORK statements in the BPXPRMxx member.

```
FILESYSTYPE TYPE(INET) ENTRYPOINT(EZBPFINI)
NETWORK DOMAINNAME(AF_INET)
        DOMAINNUMBER(2)
        MAXSOCKETS(12000)
        TYPE(INET)
NETWORK DOMAINNAME(AF_INET6)
        DOMAINNUMBER(19)
        MAXSOCKETS(13000)
        TYPE(INET)
SUBFILESYSTYPE NAME(TCPIP) ENTRYPOINT(EZBPFINI)
        TYPE(INET)
```

Figure 3-13 Sample BPXPRMxx member implementing TCP/IP dual-mode stack

For additional details on IPv6 implementation, refer to 9.1, “Internet Protocol Version 6 support” on page 336. See also *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7339.

3.10.2 Increase the number of user-defined indexes on the catalog

Some customers are already reaching the limit of 100 user-defined indexes on the catalog. The original limit was an arbitrary limit and has become much more of a constraint over time. The number of catalog tables has gradually increased since V4 when you were first allowed to define user indexes on the catalog (see Figure 12-3 on page 422). Some user indexes are now being installed through the IBM SMP/E process. Customers who reach the limit of 100 indexes have to stop and start DB2 to drop a current index that needs to be replaced by a more crucial index. This becomes very disruptive. DB2 9 for z/OS allows 500 user-defined indexes against the catalog, which should allow plenty of room to define additional user indexes on the new catalog tables delivered with DB2 9 for z/OS, as well as for catalog tables existing in prior releases of DB2.

3.10.3 Trace filtering

In today’s large DB2 installations it is very possible to produce very large quantities of trace records in a very short period of time (possibly millions of records) when monitoring your system or collecting data for debugging purposes. As processor power continues to increase and more work can be processed in shorter periods, it has become necessary to be able to more granularly filter trace records and reduce the volume of records collected for specific purposes. Prior to DB2 9 for z/OS, when starting traces you could qualify the trace by PLAN name, AUTHID, LOCATION, and IFCID, but you could not specify an exclude list. In DB2 9 for z/OS you can be more specific on the -START TRACE command by specifying additional include parameters as well as exclude parameters. This will give you the option of dramatically reducing what is being traced and the amount of trace records produced. The new filtering keywords that can be used in the -START TRACE for INCLUDE or EXCLUDE are:

- USERID - Client user ID
- WRKSTN - Client workstation name
- APPNAME - Client application name

- PKGLOC - Package LOCATION name
- PKGCOL - Package COLLECTION name
- PKGPROG - PACKAGE name
- CONNID - Connection ID
- CORRID - Correlation ID
- ROLE - DBA or end user's database ROLE

3.10.4 Allow EXCLUDE qualification in DSNDWQAL for READS

DB2 can experience problems if the system has several hundred active threads. Normally the user does not want to see all of them in a list. Therefore, he has to qualify his request. With the current implementation DB2 allows to specify INCLUDE criteria in the DSNDWQAL mapping macro. Now there are customers who know that 90% of their threads are CICS transactions and that in most cases other threads than CICS threads are responsible for a problem. Therefore, they would like to exclude the CICS transactions from the list. In V4.1 of the OMEGAMON for DB2 Performance Expert provides a more sophisticated filter mechanism on top of the DSNDWQAL.

For example, the user can specify multiple filter criteria for the same field, use a field as include or exclude criteria, and so on. However, this approach requires that all of the DB2 threads are retrieved from DB2 in order to select the desired and expected threads. Moreover, the current implementation of the DB2 IFI READS call uses CSA storage, and therefore we have to be very careful how large we define our READS return area. In a data sharing environment this is even more complex, because the same return area storage will be used by all DSG members. As a result of this, DB2 normally allocates not more than 4 MB for a READS. In large customer environments this 4 MB area is not sufficient to contain all active threads. Enlarging the return area would result in a critical usage of the CSA and has lead to PMRs in the recent past. To solve this situation and in order to satisfy a large customer environment with a large amount of READS data, DB2 requests to support the EXCLUDE capability beside INCLUDE while defining READS qualification within WQAL area.

The capability to have an overview of all data, for example, the number of threads (IFCID 148) and the locked resources (IFCID 150)) using a real-time monitor, is beneficial in a customer environment with high work load.

3.10.5 Improve the operations of Audit Trace function

Many DB2 customers require the function to implement AUDIT tracing capability without modifying application program logic. DB2 9 adds options to reduce the volume and cost of AUDIT trace. See 10.4, “Improved tracing in DB2” on page 367.

3.10.6 Greater than 4 KB page size for indexes

Prior to V9, the size of an index page is limited to 4 KB. The size of an index page limits the number of index keys that the index page can accommodate and can cause contention in indexes that split frequently. V9 lifts these restrictions by offering expanded index page sizes of 8 KB, 16 KB, and 32 KB. An index page size that is greater than 4 KB accommodates more index keys per page and can reduce the frequency of index page splits. You can use the INDEXBP option on both CREATE DATABASE and ALTER DATABASE statements to specify 4 KB, 8 KB, 16 KB, or 32 KB index buffer pools. You can also use the BUFFERPOOL keyword on the CREATE INDEX statement to specify 8 KB, 16 KB, and 32 KB buffer pools.

The larger page size also helps with the compression of indexes.

3.10.7 Asymmetrical split of index pages

This enhancement has the potential to allow index sizes to be reduced and allow for more densely packed indexes. Other *scalability* implications that this may bring are:

- Index page split latch contention relief
- Larger index page
- Fewer index splits
- Latch contention reduction, especially in data sharing environments

When all of the entries of an index leaf page are consumed during inserts, page split processing takes places and DB2 allocates a new page and moves half of the entries from the old page to the new page. This logic works well in general, but for sequential inserts where keys are added in ascending order, the freed up space in the old page will never be reused. In essence, only half of the allocated pages will be used for an index.

Another problem that is associated with this condition is that page split processing takes place more frequently since only half of the space in a leaf page is used. Some new workloads require loading data at extremely high rates (850,000 rows/sec). Some of the indexes (NPI) use keys that are inserted sequentially. As a result, page split processing shows up very frequently. During page split processing, DB2 locks up the entire index tree and blocks every load job, except one, from processing. This may reduce data load rate significantly.

Archive

4

Availability

DB2 9 for z/OS continues to bring changes that improve availability, keeping up with the explosive demands of e-business, transaction processing, and business intelligence. DB2 9 delivers increased application availability with more functions for schema evolution, new ways of defining partitioned table spaces and cloning tables, and additional recovery and restart capabilities.

This chapter discusses the following:

- Universal table space
- Clone table support
- Recovery to point in time with consistency
- APPEND
- REFRESH for EARLY code
- RENAME INDEX
- RENAME COLUMN
- UPDATE SCHEMA
- Toleration of DBET errors during DB2 restart
- Miscellaneous availability and serviceability enhancements
- Conditional Restart Log Truncation by Timestamp

© Copyright IBM Corp. 2007. All rights reserved.

4.1 Universal table space

As a follow-on to all of the changes regarding partitioning in DB2 V8, DB2 9 introduces universal table space (UTS) as a new table space type. Universal table spaces combine the advantages of both partitioned and segmented table spaces. Since UTS helps with scalabilty and growth of data, this topic is also mentioned in 3.6, “Universal table space” on page 62.

There are two different types of universal table spaces:

- Partitioned-by-growth
- Range-partitioned

In this section we describe and discuss both new table space types.

4.1.1 Partition-by-growth table spaces

One of the two types of universal table spaces is the partition-by-growth table space. A partition-by-growth table space is very useful for those of your table spaces whose tables do not have a suitable partitioning key, but are expected to exceed the 64 GB limit that has been in place for simple or segmented table space up to DB2 V8. A partition-by-growth table space can grow up to 128 TB.

A partition-by-growth table space begins its life as a single-partition table space and automatically grows additional partitions as needed to accommodate data growth. When the first partition fills, you will not receive a *resource unavailable* condition. Rather, the next partition is created and so on. This continues up to a maximum number of partitions that you define at creation time of the table space. If your table space finally reaches this maximum value, you will then receive the *resource unavailable* condition.

The data structure of partition-by-growth table spaces is similar to the structure of a segmented table space. This structure is preferable over a simple table space or a regular partitioned table space because segmented table spaces have better space management and mass delete capabilities. Since partition-by-growth table spaces are segmented partitioned, they offer almost all capabilities of partition-level operation and parallelism. However, one exception is the LOAD utility that does not allow LOAD PART for partition-by-growth table spaces.

Explicitly created table spaces

Partition-by-growth table spaces can be created either explicitly or implicitly. The parameters and usage instructions, which are now following, refer to explicitly created table spaces. Refer to “Implicitly created partition-by-growth table spaces” on page 76 for a description of how to create partition-by-growth table spaces implicitly.

Keyword MAXPARTITIONS on CREATE or ALTER TABLESPACE

A new keyword MAXPARTITIONS has been added to the CREATE TABLESPACE and ALTER TABLESPACE statements. With MAXPARTITIONS you specify the maximum number of partitions to which the partition-by-growth table space may grow.

The number of partitions that you can use for MAXPARTITIONS depends upon the page size and segment size that you use in your CREATE TABLESPACE statement. If you omit the DSSIZE keyword on your CREATE TABLESPACE statement and you specify MAXPARTITIONS, the DSSIZE is 4 GB for a 4 KB page, 8 GB for a 8 KB page, 16 GB for a 16 KB page, and finally 32 GB for a 32 KB table space. Refer to Table 4-1 to find the maximum numbers for MAXPARTITION in your CREATE or ALTER TABLESPACE statement.

Table 4-1 Maximum value for MAXPARTITIONS given the page size and DSSIZE

DSSIZE	4 KB page	8 KB page	16 KB page	32 KB page
1 - 4 GB	4096	4096	4096	4096
8 GB	2048	4096	4096	4096
16 GB	1024	2048	4096	4096
32 GB	512	1024	2048	4096
64 GB	256	512	1024	2048

When you start using partition-by-growth table spaces, you might want to start with a number for MAXPARTITIONS that is smaller than the maximum number shown in Table 4-1 in order to be able to protect against run-away applications, for example, an insert in an infinite loop. You can use the ALTER TABLESPACE statement to increase the maximum number of partitions allowed later on.

Involved catalog tables

As with many new functions, the catalog tables have been adjusted to reflect their usage and object characteristics.

- SYSIBM.SYSTABLESPACE

  A new column MAXPARTITIONS has been added to SYSIBM.SYSTABLESPACE. The entry there is the number that you have specified in your CREATE or ALTER TABLESPACE statement. It does not show the actual number of physically existing partitions. The actual number of allocated partitions is shown in column PARTITIONS.

  Column TYPE shows a value of ' G' for partition-by-growth table spaces.

  Note: Even though the number of MAXPARTITIONS might be, for example, 100, and the number shown in PARTITIONS is only 4, the data is only spread over four physical partitions up to now. It is not possible to use ALTER TABLESPACE to reduce MAXPARTITIONS after the table space has been created.

- SYSIBM.SYSTABLEPART

  When you create the table space, only one row is added to SYSIBM.SYSTABLEPART independently from the number that you used for MAXPARTITIONS. Additional rows are added to this catalog table as your table grows and additional partitions are allocated because your amount of data has exceeded the associated DSSIZE.

Archive

SEGSIZE usage

Table 4-2 shows how specifying or omitting the SEGSIZE keyword in combination with MAXPARTITIONS influences which type of table space you create.

Table 4-2 SEGSIZE and MAXPARTITIONS keyword combinations

SEGSIZE clause	MAXPARTITIONS clause	Type of table space
Explicitly specified	Explicitly specified	Partitioned-by-growth table space
Explicitly specified	Not specified	Segmented table space
Not specified	Explicitly specified	Partitioned-by-growth table space with an implicit specification of SEGSIZE 4
Not specified	Not specified	Segmented table space

Implicitly created partition-by-growth table spaces

If you plan to influence the creation of your implicitly created table space so that it will be partition-by-growth table space, you can do so by using the PARTITION BY SIZE clause on your CREATE TABLE statement.

The whole PARTITION BY SIZE syntax part of the CREATE TABLE statement is shown in Figure 4-1.

```
PARTITION BY SIZE EVERY integer G
```

Figure 4-1 PARTITION BY SIZE syntax

If you omit the EVERY integer G part of the syntax, the implicit table space defaults to:

- SEGSIZE 4
- Maximum size for each partition of 4 GB
- Maximum number of partitions set to 256

If the limit of maximum 256 partitions is inadequate, you can alter it according to the numbers shown in Table 4-1 on page 75.

Considerations and restrictions

There are a few other rules concerning the creation of partition-by-growth table spaces:

- Partition-by-growth table spaces must be storage group controlled.
- The MEMBER CLUSTER option of the CREATE TABLESPACE syntax is not valid.
- You cannot use the following options for partition-by-growth table spaces:
  - ALTER TABLE ADD PARTITION
  - ALTER TABLE ROTATE PARTITION
  - ALTER TABLE ALTER PARTITION
- LOB and XML spaces associated with a partition-by-growth table space are always implicitly defined by DB2, independent of whether SQLRULES DB2 or STD is in effect.
- A partition-by-growth table space can only contain one table.
- Only NPIs are allowed on tables residing in partition-by-growth table spaces.
- When a new partition is added, drain and claim are inherited from a prior partition.
- Some DBET states are also inherited from the previous partition when a new partition is added. Those DBET states include RO*, UTUT, UTRO, PSRBD, and ICOPY.

Allocation of partitions

As stated earlier in this section, new partitions of a partition-by-growth table space are added one at a time as the existing ones fill up due to insert or update operations. You will not receive table space full conditions any more as long as the number shown in column PARTITIONS of SYSIBM.SYSTABLESPACE is less than the number of MAXPARTITIONS in the same table.

Once a partition is added and its associated data set is allocated, DB2 will not roll back this allocation even if the entire Unit of Recovery (UR) that caused the addition of the new partition is rolled back. One exception to this rule occurs if the UR that would cause DB2 to add a new partition contains outstanding DDL operation in the same UR. In this case DB2 does not allocate the new partition and rolls back the whole UR instead. To prevent those situations, you should always commit any preceding DDL and process the data changes afterwards. The described behavior is also shown in Figure 4-2.

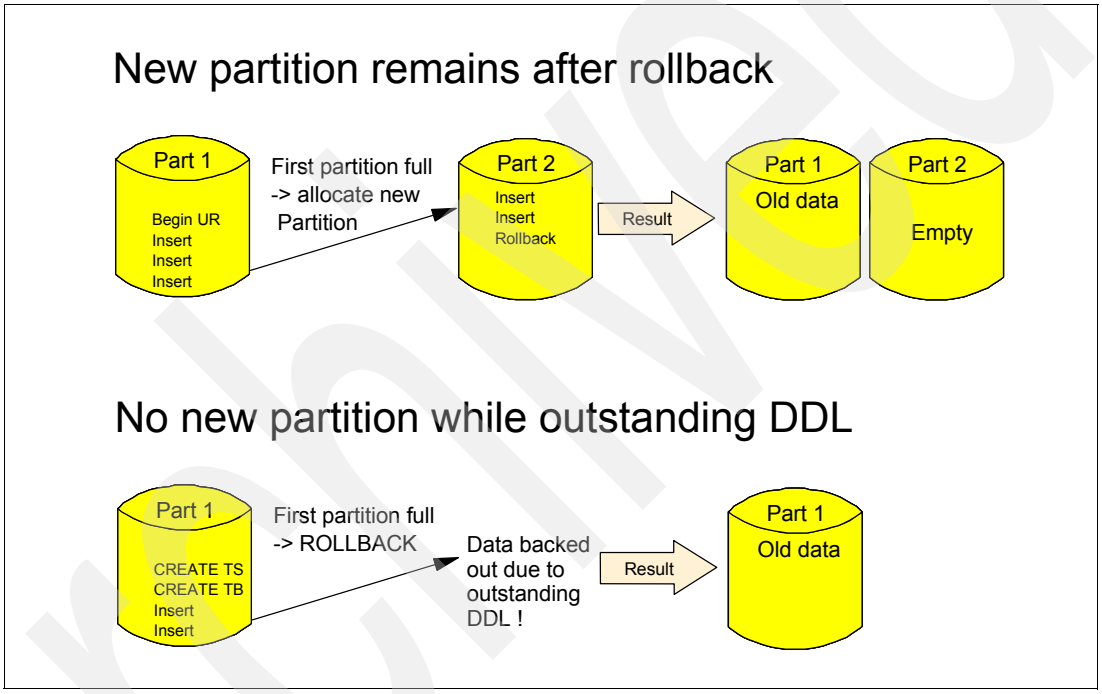


Figure 4-2 Allocation of new partitions

Note: If a partition-by-growth table space contains a table with one or more LOB columns, the behavior regarding the automatic implicit creation of the needed LOB objects, such as the LOB table spaces for the LOB data, depends on the SQLRULES special register setting. If you are using SQLRULES STD, you must create all necessary LOB objects manually for the first partition. With DB2 9 NFM, for partitions two and up, independently from what your SQLRULES setting is, DB2 now creates all needed LOB objects every time a new partitions is added.

When you run the REORG utility, DB2 might have to allocate new partitions, for example, due to a change in free space parameters. As long as your table does not contain XML or LOB columns, DB2 will automatically allocate new partitions as required.

If your table contains XML or LOB columns and REORG requires the data to be moved from one partition to another, the utility fails. This is true for both the reorganization of the entire table space and also for REORG PART. To circumvent this situation you can allocate more

space for the partitions that cause the problem so that REORG is then able to fit the data rows back into the originating partition.

Deallocation of partitions

When you reorganize a partition-by-growth table space, the data may fit in a smaller number of partitions than it currently occupies. If this is the case, and provided that your table does not contain LOB columns, the REORG utility condenses the data into the minimum number or required partitions. As shown in Figure 4-3, even if after the REORG the data would fit into two partitions, the third partition is not deleted.

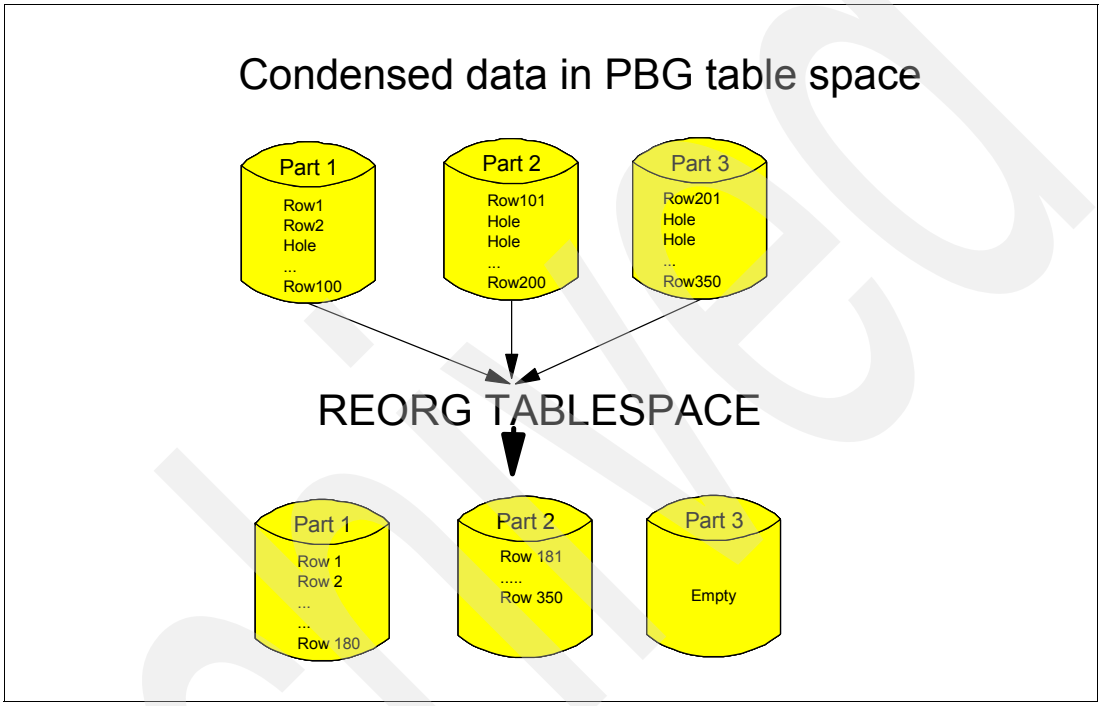


Figure 4-3 Condensed data in partition-by-growth table space after REORG

If your table contains LOB columns, as for the allocation of new partitions, it is not allowed to move rows from one partition to another. Following this, as a result of a REORG of a partition-by-growth table with LOB columns, each partition may shrink by eliminating the existing holes, but no rows are removed from one partition to another and empty partitions are not likely to result.

Utility considerations

Partition-by-growth table spaces are not exactly the same as regular partitioned table spaces. As a result, there are some rules and restrictions when working with utilities that we are going to describe hereafter.

LOAD utility

There are two things that we must mention to you regarding the LOAD utility:

- Although all of the other utilities can operate on partition level even for partition-by-growth table spaces, you cannot specify the PART keyword on your LOAD statement.
- Parallelism does not apply for partition-by-growth table spaces.

REORG utility

Refer to “Allocation of partitions” on page 77 and “Deallocation of partitions” on page 78 for a discussion about implications on the REORG utility.

COPY utility

The COPY utility allows you to copy either the entire table space or perform the copy on a partition basis. If you copy the entire table space, the utility does not just copy those partitions that contain data, but also empty ones that just have a header and space map or system pages.

If you run a SHRLEVEL(CHANGE) copy and the table space is expanded to one more partition while the copy runs, this new partition will also be copied.

RECOVER utility

You can run the RECOVER utility on the entire table space as well as on partition level. Since it might have happened that a new partition has been added after you took the latest image copy, your image copy might contain less partitions than the current number of partitions.

As a consequence of not giving up partitions that have been created once, after a point-in-time recovery to this image copy, excess partitions in the table space will be empty. This behavior is also shown in Figure 4-4.

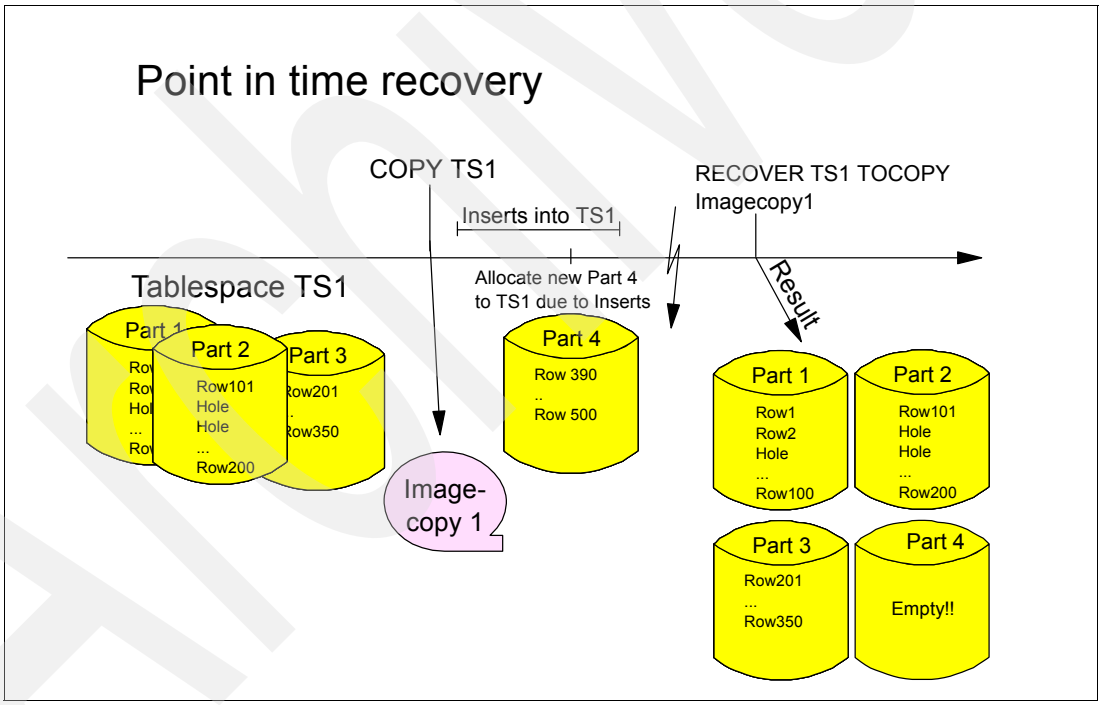


Figure 4-4 Point-in-time recovery after partition has been added

REBUILD INDEX utility

Rebuild Index has a similar situation as REORG and RECOVER. The REBUILD INDEX utility may reset more partitions than it repopulates. Excess index partitions continue to exist as empty partitions and will not be removed.

DSN1COPY utility

You can use DSN1COPY to restore single partitions or the entire table space of a partition-by-growth table space. Since partitions are dynamically created as needed, the total

number of partitions in a DSN1COPY may not be consistent with the number of partitions defined on the current table space. Remember that you should delete data that resides in excess partitions before you use DSN1COPY. Doing so prevents you from having residual data in your table.

Tip: The new SQL function TRUNCATE might be a good means to delete existing data rows quickly. Refer to 6.6, “TRUNCATE” on page 150, to learn more about this functionality.

If you want to use DSN1COPY to copy data from one DB2 subsystem to another, the target system must have an equal or greater number of partitions than the source system.

Note: Let us assume that you have an image copy data set from a production DB2 subsystem. This image copy contains ten partitions. If you try to use this image copy data set as input for DSN1COPY with the goal to copy the data to a test DB2 subsystem, in which the table space only has 8 partitions, the execution would fail.

4.1.2 Range-partitioned universal table spaces

Range-partitioned universal table spaces is the perfect mixture of partitioned and segmented table spaces. You easily create this type of table space when you specify both keywords NUMPARTS and SEGSIZE in one CREATE TABLESPACE statement. You were not allowed to use this combination of keywords in prior releases. Once you have created a range-partitioned table space, you can take any action that used to be allowed on partitioned or segmented table spaces. However, as for regular partitioned table spaces, only one table is allowed.

Some benefits of range-partitioned table spaces are:

- Better space management in conjunction with varying-length rows. This is because a segmented space map page has more information on free space than a partitioned space map page.
- Improved mass delete performance. Mass delete in a segmented table space organization tends to be faster than in other types of table space organizations such as partitioned or simple table spaces.
- DB2 reuses all or most of the segments of a table after you committed the dropping or mass delete of it.

Miscellaneous considerations

There are some things that we want to talk about to give you a complete understanding of what is newly related to range-partitioned universal table spaces:

- All range-partitioned universal table spaces are LARGE table spaces.
- After your successful creation of such a table space, column TYPE in SYSIBM.SYSTABLESPACE contains an ’R’.
- The MEMBER CLUSTER keyword is not allowed for range-partitioned universal table spaces.
- Range partitioned table spaces follow the regular partitioned table space locking scheme, (that is, LOCKSIZE TABLE is not a valid keyword in the CREATE TABLESPACE syntax).
- You need to use the table-controlled partitioning syntax introduced with V8 to create range-partitioned universal table spaces.

- Range-partitioned universal table spaces do *not* replace the regular partitioned table space.
- Mass delete locks are at the table level instead of the table space level.

4.2 Clone table support

Users expressed the need for a functionality that is similar to an online LOAD REPLACE. This is, for instance, the need to reload table data multiple times a day while still being able to access the existing data at least with read-only operations. A new functionality that is being introduced with DB2 V9, clone table support, provides you with the ability to generate a table with the exact same attributes as a table that already exists at the current server. This new table is called the clone table of the base table, and it is created in the same table space as the base table. Once the clone table is created, you can independently work with it (that is, you can, for example, load it or insert rows into it).

The clone table is not only structurally identical to the base table in every way (that is, it has the same number of columns, column names, data types, check constraints, etc.), but it is also created with the same indexes, before triggers, LOB objects, and so on. DB2 creates the clone objects automatically when it creates the clone table. Although the clone table itself has a different name and might have a different schema associated to it, the clone table objects are referred to by the same names as those that are used for the base table objects (that is, the base and clone table share all object descriptions).

4.2.1 Creation of a clone table

You can create a clone table on an existing base table using the new ALTER TABLE ADD CLONE syntax. As mentioned above, with the creation of a clone table, you also initiate the creation of multiple additional clone objects such as indexes, constraints, triggers, and so on. You can independently choose the table name and schema name for the clone table.

One restriction for the usage of those new clone tables is that you can only clone base tables that are created in universal table spaces (that is, you can either use partition by growth or range partitioned table spaces for clone tables). Refer to 4.1, “Universal table space” on page 74, if you want to learn more about this special kind of table spaces. Also, your clone tables only work for single table spaces, which are DB2 managed.

Archive

Figure 4-5 shows what happens when you use ALTER TABLE ADD CLONE for a table that does not have any additional objects defined on it such as indexes, constraint, and so on. Base table paolor7.basetab exists in table space DB1.TS1. The name of the VSAM cluster for this table space is DB2P.DSNDBD.DB1.TS1.I0001.A001. Once you issue the ALTER statement, DB2 creates a second data set whose name is exactly the same as for the cluster holding the base table with the exception of the data set instance number, which is now I0002 rather than I0001 for the base table.

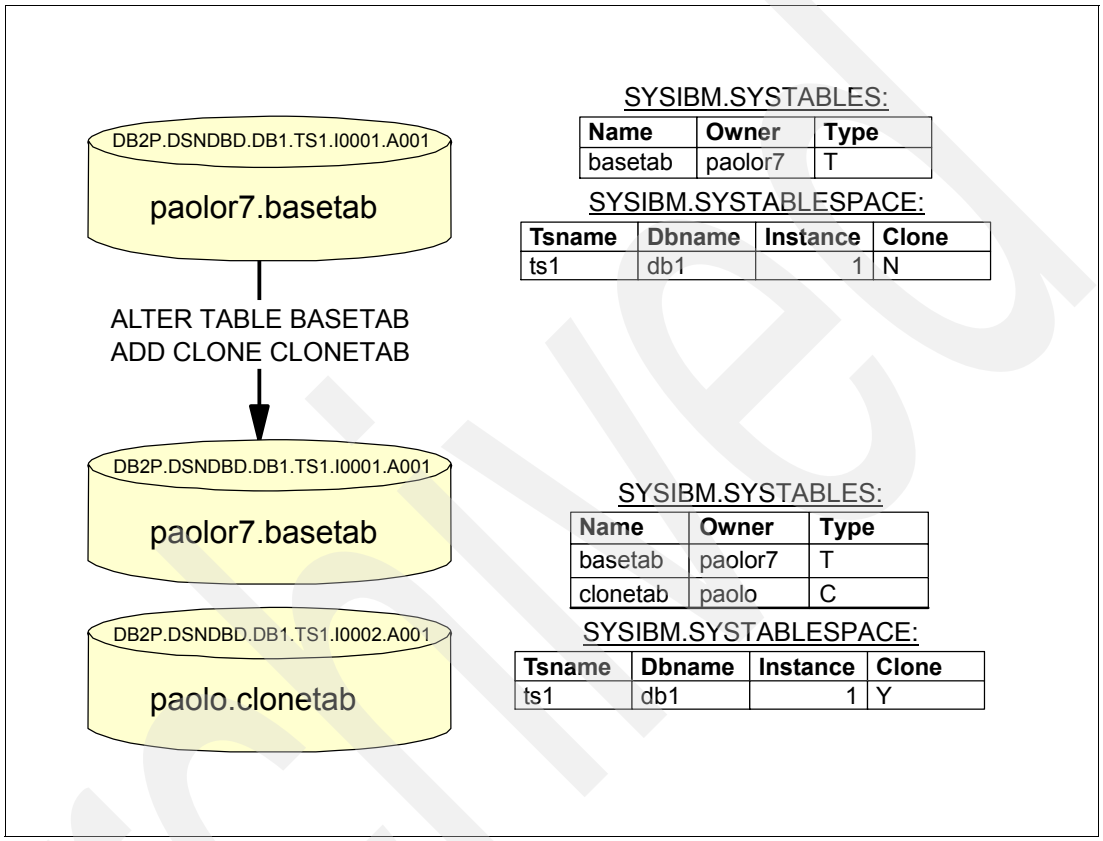


Figure 4-5 Creation of a clone table without any dependent objects on base table

The table space name is the same for both the base and clone table. A new column INSTANCE has been added to a couple of DB2 catalog tables. The instance number can always either be 1 or 2. When you create your first clone table, the instance number of your base table is 1 and 2 for the clone table. This might not be the case any more once you have started exchanging the clone and base tables. We explain the exchange process later in this chapter. Another column CLONE has also been added to SYSIBM.SYSTABLESPACE. This column contains 'Y' if the table space contains any objects that are involved in a clone relationship.

Figure 4-5 also shows that once you create a clone table, DB2 adds one new row to SYSIBM.SYSTABLES. The new ICTYPE 'C' indicates the creation of a new clone table. Column INSTANCE indicates whether the VSAM cluster I0001 or I0002 (or J0001 or J0002) currently stores the base table.

Every CREATE of a page set of a clone leads to an entry in SYSIBM.SYSCOPY with the new ICTYPE = 'C'.

Let us now assume that your base table has two indexes and one LOB column defined on it. Your current catalog entries are as follows.

For tables, see Table 4-3.

Table 4-3 SYSIBM.SYSTABLES

NAME	CREATOR	TSNAME	TYPE	DBNAME
basetab	paolor7	ts1	T	db1
basetablob	paolor7	baselob	X	db1

For table spaces, see Table 4-4.

Table 4-4 SYSIBM.SYSTABLESPACE

NAME	DBNAME	INSTANCE	CLONE
ts1	db1	1	N
baselob	db1	1	N

For entries in SYSCOPY, see Table 4-5.

Table 4-5 SYSIBM.SYSCOPY

TSNAME	DBNAME	ICTYPE	INSTANCE
ts1	db1	C	1
baselob	db1	C	1

For indexes, see Table 4-6.

Table 4-6 SYSIBM.SYSINDEXES

NAME	CREATOR	TBNAME	DBNAME	INDEXSPACE
baseix1	paolor7	basetab	db1	BASERTW1
baseix2	paolor7	basetab	db1	BASEQTEW
baseixlob	paolor7	basetablob	db1	BASETGFD

Additionally to the information in the DB2 catalog, you have the data sets shown in Figure 4-6 allocated.

```
DB2P.DSNDBD.DB1.TS1.I0001.A001
DB2P.DSNDBD.DB1.BASELOB.I0001.A001
DB2P.DSNDBD.DB1.BASERTW1.I0001.A001
DB2P.DSNDBD.DB1.BASEQTEW.I0001.A001
DB2P.DSNDBD.DB1.BASETGFD.I0001.A001
```

Figure 4-6 Allocated VSAM CLUSTER before ALTER TABLE ADD CLONE

Archived

Let us assume that you create a new clone table 'PAOLO.CLONETAB' on table 'BASETAB' now. Table 4-7 shows how this changes the catalog table entries.

Table 4-7 SYSIBM.SYSTABLES

NAME	CREATOR	TSNAME	TYPE	DBNAME
basetab	paolor7	ts1	T	db1
basetablob	paolor7	baselob	X	db1
clonetab	paolo	ts1	C	db1

Table 4-8 shows the changes in SYSTABLESPACE.

Table 4-8 SYSIBM.SYSTABLESPACE

NAME	DBNAME	INSTANCE	CLONE
ts1	db1	1	Y
baselob	db1	1	Y

Table 4-9 shows the entries in SYSCOPY.

Table 4-9 SYSIBM.SYSCOPY

TSNAME	DBNAME	ICTYPE	INSTANCE
ts1	db1	C	1
baselob	db1	C	1

Table 4-10 shows the entries in SYSINDEXES.

Table 4-10 SYSIBM.SYSINDEXES

NAME	CREATOR	TBNAME	DBNAME	INDEXSPACE
baseix1	paolor7	basetab	db1	BASERTW1
baseix2	paolor7	basetab	db1	BASEQTEW
baseixlob	paolor7	basetablob	db1	BASETGFD

Archived

As you can see from the tables above, there is just one change in Table 4-7 on page 84 where DB2 has added the clone table. All other entries remain the same. However, if you refer to Figure 4-7, you can see that every VSAM cluster now has a new extension with the same name but with instance name I0002 instead of I0001.

```
DB2P.DSNDBD.DB1.TS1.I0001.A001
DB2P.DSNDBD.DB1.TS1.I0002.A001
DB2P.DSNDBD.DB1.BASELOB.I0001.A001
DB2P.DSNDBD.DB1.BASELOB.I0002.A001
DB2P.DSNDBD.DB1.BASERTW1.I0001.A001
DB2P.DSNDBD.DB1.BASERTW1.I0002.A001
DB2P.DSNDBD.DB1.BASEQTEW.I0001.A001
DB2P.DSNDBD.DB1.BASEQTEW.I0002.A001
DB2P.DSNDBD.DB1.BASETGFD.I0001.A001
DB2P.DSNDBD.DB1.BASETGFD.I0002.A001
```

Figure 4-7 Allocated VSAM CLUSTER after ALTER TABLE ADD CLONE

Attention: When you create a clone table, no rows are moved or copied to your newly created objects. You must fill the clone table with data. You can, for example, use the LOAD utility to do so.

4.2.2 Commands and clone tables

The following commands have been changed in order to address clone objects explicitly:

- -DISPLAY DATABASE
- -START DATABASE
- -STOP DATABASE

We now look at the details.

-DISPLAY DATABASE command

The -DISPLAY DATABASE command has been extended so that it indicates information for both base table objects and their clones. The output always contains information for both base and clone table objects and not just one or the other. Base table objects with cloning are now noted with an additional TYPE column character of 'B', and cloned objects with a TYPE column character of 'C'. Immediately following the new 'B' and 'C' TYPE characters is the data set instance number that is currently associated with the data sets of these objects. Data set instance numbers are always either a '1' or a '2'. Objects that do not have a cloning TYPE object suffix are not currently involved in cloning.

Archived

Refer to Figure 4-8 for a sample output of the -DISPLAY DATABASE command containing information about clone objects. We have a couple of different base/clone combinations in the displayed database.

```
DSNT360I  -DB9B **********************************
DSNT361I  -DB9B *  DISPLAY DATABASE SUMMARY
                *    GLOBAL
DSNT360I  -DB9B **********************************
DSNT362I  -DB9B     DATABASE = SABINE  STATUS = RW
                  DBD LENGTH = 32294
DSNT397I  -DB9B
NAME     TYPE PART  STATUS            PHYERRLO PHYERRHI CATALOG  PIECE
-------- ---- ----- ----------------- -------- -------- -------- -----
CLONE1   TS         RW
CLONE2   TSB2  0001 RW,COPY
CLONE2   TSC1  0001 RW,COPY
CLONE3   TSB1  0001 RW
CLONE3   TSC2  0001 RW
CLONE4   TSB2  0001 RW
CLONE4   TSC1  0001 RW
CLONE5   TSB2  0001 STOP
CLONE5   TSC1  0001 RW,COPY
CLONE7   TS    0001 RW
CLONE8   TSB2  0001 RW
CLONE1BA IX         RW
CLONE2BA IXB2  L*   RW
CLONE2BA IXC1  L*   RW
CLONE3BA IXB1  L*   RW
CLONE3BA IXC2  L*   RW
CLONE4BA IXB2  L*   RW
CLONE4BA IXC1  L*   RW
CLONE8BA IXB2  L*   RW
CLONE8BA IX    L*   RW
******* DISPLAY OF DATABASE SABINE   ENDED      **********************
```

Figure 4-8 -DISPLAY DATABASE output with clone objects

Let us have a look at the different page sets:

- Table space CLONE1

  This table space does not have a clone. You can tell this, because the type is just shown as 'TS'.

- Table space CLONE2

  The types for this table space are TSB2 and TSC1. TSB means that this is the table space instance that contains the base table. The '2' at the end of TSB2 indicates that the instance number containing the base table is I0002. TSC indicates that the associated instance contains the clone table, which is I0001 in our case. Both instances have restricted database state COPYPENDING associated with them.

  Since the base table is stored in instance number I0002, at least one successful EXCHANGE DATA statement must have been performed in the past.

- Table space CLONE3

  This table space contains clone objects. You can tell this because the type shows up as TSB1 and TSC2, which means that the base table is in I0001 and the clone table is in the

Archived

I0002 instance. This in turn means that the data has either been never exchanged until now or it has been exchange 2, 4, 6, and so on or some even number times already. If you would like to be able to answer how many times the data has been exchange and if at all, you can get a clue looking at the SYSIBM.SYSCOPY column ICTYPE in combination with the STYPE value. Refer to 4.2.3, “Exchanging clone and base table” on page 88, to learn more about the EXCHANGE DATA SQL statement.

- Table space CLONE5

  This table space containing clone objects has got data exchanged between the clone and base table at least once, as you can see from instant number I0002 associated to the base table. In addition to that, this sample table space shows that it is possible to have different restrictive or informational database states assigned to the base and clone page sets. In the example shown in Figure 4-8 on page 86, the clone table is currently in COPYPENDING state, while the base table space has been stopped.

- Table space CLONE8

  Table space CLONE8 has had clone objects in the past. After the data has been exchanged, the clone table was dropped.

-START DATABASE and -STOP DATABASE command

The -START DATABASE and STOP DATABASE commands are extended with the CLONE keyword so that you can specify that clone objects are to be started or stopped. In the absence of the CLONE keyword, base table objects are started or stopped and the clone table objects will not be processed. If you use the CLONE keyword, it indicates to DB2 that you only want to process the CLONE objects.

Note: You cannot start or stop both base table and clone objects with just one command. You must issue two commands — one with and one without the CLONE keyword — if you plan to issue the same command for base and clone objects.

Archive

Refer to Figure 4-9 to see where the new CLONE syntax appears in the -START DATABASE COMMAND.

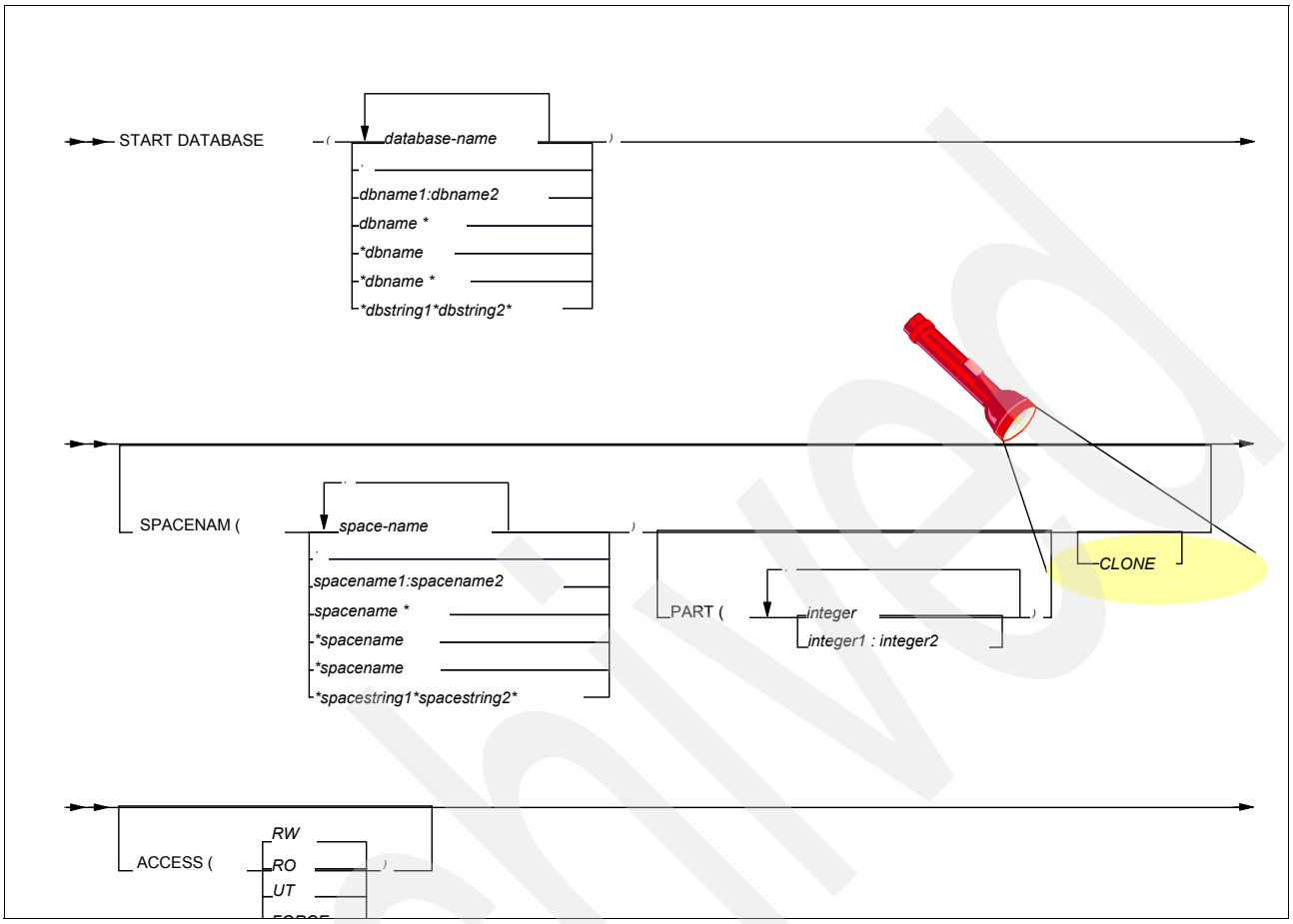


Figure 4-9 -START DATABASE SYNTAX

4.2.3 Exchanging clone and base table

When you create clone tables, the whole purpose is to finally switch from the original base table to your newly created and maybe reloaded clone table. Use the new SQL statement EXCHANGE to initiate the swap. The statement is shown in Figure 4-10.

```
EXCHANGE DATA BETWEEN TABLE table-name1 AND table-name2
```

Figure 4-10 EXCHANGE DATA SQL statement

For this statement, it does not matter which table is the clone table and which table is the base table. No data movement takes place when the tables are exchanged. Only the instance numbers that point to the base or clone table change. What originally was the pre-exchange clone table data instance number becomes the base table data instance number and vice versa after the exchange completes. Figure 4-11 on page 89 illustrates this exchange process.

Prior to issuing the EXCHANGE DATA statement, when you selected data from table PAOLOR7.BASETAB, you were returned all the rows stored on instance I0001 of table space DB1.TS1. After the exchange has successfully been processed, selecting from table

PAOLOR7.BASETAB returns the rows that are stored on instance I0002 of table space DB1.TS1.

You can also see from Figure 4-11 that the EXCHANGE DATA statement leads to an update of column INSTANCE of SYSIBM.SYSTABLESPACE. This now identifies instance 2 as the data set instance number of the current base object.

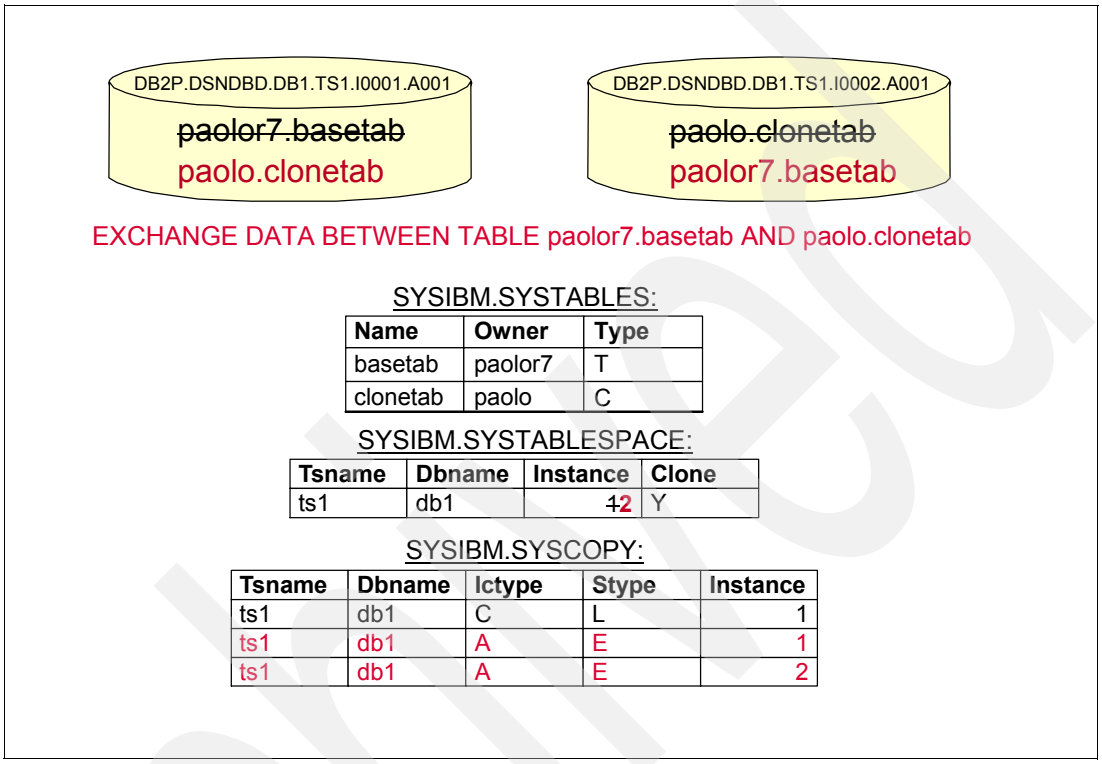


Figure 4-11 EXCHANGE DATA

Additionally, to the change in SYSIBM.SYSTABLESPACE, DB2 adds two new rows to SYSIBM.SYSCOPY. The ICTYPE column contains value 'A' like ALTER, and the STYPE column shows an 'E'. 'E' means that the data set numbers of a base table and its associated clone table are exchanged.

This data change provides an Online Load Replace type of capability. The data exchanges can only be done at the table level. It is not possible to just exchange some partitions of a partitioned table space. Since it is assumed that existing base table statistics do not sufficiently differ from those of the clone table, DB2 does not invalidate statistics during the data exchange process. As a result, the exchange of base and clone table does not impact applications accessing base table data. DB2 also neither invalidates plans or packages nor the dynamic statement cache.

Attention: Since DB2 does not invalidate any plans, packages, or the dynamic statement cache, it is your responsibility to run RUNSTATS and perform binds if you know that the data that you have just exchanged is significantly different from what you had in your base table prior to the exchange process.

Important: In order to prevent the objects that are being exchanged from concurrent access with DML or utilities, the EXCHANGE process drains both base and clone objects. As a result, the exchange process cannot be performed without any impact to the availability of the data. It needs exclusive access to both the base and clone objects.

You can exchange data regardless of the exception states of each object. DB2 tracks the database exception status by the data set instance number, which will be properly enforced after an exchange.

DISPLAY DATABASE command will show information for both the base and clone objects in the database. The -DISPLAY DATABASE output captured in Example 4-12 shows two pairs of base and clone table spaces that have some pending states. For table space CLONE2, the copy pending is the same for base and clone. In contrast to that, CLONE5 shows different states. The table space holding the base table (TSB2) is currently in STOP state, and the table space hosting the clone table is in copy pending.

```
DSNT360I  -DB9B *********************************
DSNT361I  -DB9B *  DISPLAY DATABASE SUMMARY
                *    GLOBAL
DSNT360I  -DB9B *********************************
DSNT362I  -DB9B     DATABASE = SABINE  STATUS = RW
                   DBD LENGTH = 40370
DSNT397I  -DB9B
NAME     TYPE PART  STATUS           PHYERRLO PHYERRHI CATALOG  PIECE
-------- ---- ----- ---------------- -------- -------- -------- -----
CLONE1   TS        RW
CLONE2   TSB2  0001 RW,COPY
CLONE2   TSC1  0001 RW,COPY
CLONE3   TSB1  0001 RW
CLONE3   TSC2  0001 RW
CLONE4   TSB2  0001 RW
CLONE4   TSC1  0001 RW
CLONE5   TSB2  0001 STOP
CLONE5   TSC1  0001 RW,COPY
```

Figure 4-12 -DIS DB output with exception states prior to EXCHANGE

Archive

Regardless of the pending states of CLONE5, we now issue the EXCHANGE command to exchange the clone with the base table for this table space. The result is shown in Figure 4-13. As you can see, the pending states are still the same. One table space of CLONE5 is in STOP state and one has copy pending associated with it.

```
DSNT360I  -DB9B *********************************
DSNT361I  -DB9B *  DISPLAY DATABASE SUMMARY
                *    GLOBAL
DSNT360I  -DB9B *********************************
DSNT362I  -DB9B     DATABASE = SABINE  STATUS = RW
                   DBD LENGTH = 40370
DSNT397I  -DB9B
NAME     TYPE PART  STATUS            PHYERRLO PHYERRHI CATALOG  PIECE
-------- ---- ----- ----------------- -------- -------- -------- -----
CLONE1   TS         RW
CLONE2   TSB2  0001 RW,COPY
CLONE2   TSC1  0001 RW,COPY
CLONE3   TSB1  0001 RW
CLONE3   TSC2  0001 RW
CLONE4   TSB2  0001 RW
CLONE4   TSC1  0001 RW
CLONE5   TSB1  0001 RW,COPY
CLONE5   TSC2  0001 STOP
CLONE7   TS    0001 RW
```

Figure 4-13 -DIS DB output with exception states after EXCHANGE

Attention: If you look at CLONE5 and its associated pending states very carefully, you can now see that after the EXCHANGE statement, it is no longer the clone table that is in copy pending state, but now it is the new base table that cannot be updated due to the restrictive state.

4.2.4 Dropping a clone table

After you have created a clone table, you might want to drop it subsequently. You can always drop a clone table. You can do this after you have exchanged the data with the base table or without having performed an exchange. You cannot use DROP TABLE (base or clone) to drop a clone table. You must use the ALTER TABLE statement with the DROP CLONE clause to drop a clone table. If a base table that is involved in a clone relationship is dropped, the associated clone table is also dropped. You can drop a clone table in the following ways:

- Use the ALTER TABLE statement to drop the CLONE. The full syntax is shown in Figure 4-14.

```
ALTER TABLE base-table-name DROP CLONE;
```

Figure 4-14 ALTER TABLE DROP CLONE syntax

When you use the ALTER TABLE syntax, you must specify the name of the base table rather than the clone table itself to drop the clone table, its indexes, LOB structures, and delete the underlying VSAM data sets from the current server. The base table and its objects remain unchanged.

When you DROP a clone table, DB2 removes all associated SYSCOPY entries. This is different than what happens as result of a DROP of a base table. However, as for base tables, DB2 does not delete SYSLGRNX information until the DBID/PSIDs are reused for a subsequent table space.

- You can use the DROP TABLESPACE statement to drop the base table and all other underlying objects including the clone table.
- DROP DATABASE also removes all underlying objects including the clone table.

4.2.5 Restrictions

When you use the ALTER TABLE table-name ADD CLONE clone-table-name statement to create a clone of the base table, the base table must conform to the following:

- Be a universal table space (that is, either a partition-by-growth or a range-partitioned table space).
- Reside in a DB2 managed table space (that is, must be storage-group-controlled).
- Be the only table in the table space.
- Not already be defined with a clone table.
- Not be involved in any referential constraints.
- Not have any AFTER triggers defined.
- Not be a materialized query table.
- Not have any data sets that have yet to be created. This mean that you cannot create clone tables for base tables that reside in table spaces that have been created using DEFINE NO and that have not yet created the VSAM.
- Not have any pending alterations and not have any active versioning. What is meant here is that the numbers that you find in columns OLDEST_VERSION and CURRENT_VERSION of SYSIBM.SYSTABLESPACE must be identical. If they differ, you receive SQLCODE -148 with REASON 9.
- Not have an incomplete definition.
- Not be a created global temporary table or declared global temporary table

4.2.6 Utilities and clone tables

In order for you to be able to use utilities efficiently not only for base table page sets but also for clone table objects, many DB2 utilities have been extended. Most of these changes consist of adding the CLONE keyword to the utility syntax. We can generally say that when you specify CLONE on a utility statement, the utility is to operate only against a clone objects. If you omit the CLONE statement, the utility operates against a base object as it always used to do. Most utilities have now the CLONE keyword. Exceptions are LOAD, RUNSTATS, and MODIFY STATISTICS.

CHECK DATA utility

If you specify CLONE on the CHECK DATA utility syntax, you indicate that the clone table within the specified table space is to be checked. As listed above, clone tables cannot have referential constraints. As a consequence, if you use CHECK DATA ... CLONE, the utility only checks table check constraints and for consistencies between the clone table data and the corresponding LOB data.

Archived

CHECK INDEX utility

If you specify CLONE on your CHECK INDEX syntax, DB2 checks only the specified indexes that are on clone tables.

CHECK LOB utility

CLONE indicates that you want DB2 to check the LOB table space data for only the table clone, not the LOB data for the base table.

COPY utility

The CLONE keyword on the COPY utility statement indicates that you want DB2 to copy only the table clone data in the specified table spaces or indexes on table clones in the specified index spaces. If you use the LIST keyword to specify a list of objects, COPY processes only those objects in the list that contain table clones or indexes on table clones. DB2 ignores the other objects in the list.

COPYTOCOPY utility

If you specify the CLONE keyword on a COPYTOCOPY statement, you ask DB2 just to process image copy data sets that were taken against table clones or indexes on table clones.

DIAGNOSE utility

If you code CLONE on the DIAGNOSE utility statement, you tell DB2 just to display information for only the specified objects that are table clones, table spaces that contain tables clones, indexes on table clones, or index spaces that contain indexes on table clones.

LISTDEF utility

When you include the CLONE statement on LISTDEF, you indicate that the INCLUDE or EXCLUDE expression will only return clone tables, table spaces that contain clone tables, indexes on table clones, or index spaces that contain indexes on clone tables. CLONE in combination with the ALL keyword also includes LOB objects.

Attention: If you specify a table name on your LISTDEF statement, DB2 ignores the CLONE statement.

Tip: If you omit CLONE and just code the ALL keyword, clone objects are not included.

MERGECOPY utility

When you specify CLONE, only those image copy data sets that were taken against clone objects are processed by the MERGECOPY utility.

MODIFY RECOVERY utility

If you use the CLONE keyword on your MODIFY RECOVERY utility statement, DB2 deletes all corresponding SYSCOPY and SYSLGRNX entries for your clone objects. This is a different behavior that DB2 has when you drop a clone. When you drop a clone, DB2 deletes the SYSCOPY entries, but keeps the SYSLGRNX entries.

QUIESCE utility

If you specify CLONE on the QUIESCE utility, you indicate that you want to create a quiesce point for only the specified table spaces that contain clone tables.

REBUILD INDEX utility

If you specify CLONE, only those indexes are rebuilt that are on table clones. The keywords CLONE and STATISTICS are mutually exclusive. DB2 does not collect statistics for table clones.

RECOVER utility

There is nothing special that you need to care of when running RECOVER on a clone table. If you specify CLONE, DB2 only recovers clone table data in the specified table space or the specified index spaces that contain indexes on table clones.

REORG INDEX utility

You can use the new CLONE keyword to ask DB2 to reorganize only the specified index spaces that contain indexes on table clones.

REORG TABLESPACE utility

If you specify CLONE, DB2 only reorganizes table clones from the specified table spaces. Since you cannot collect statistics on clone tables, specifying both CLONE and STATISTICS is not allowed.

REPAIR utility

If you use CLONE on the REPAIR utility syntax you indicate that you only want REPAIR to work on those objects that are table spaces that contain tables clones, indexes on table clones, or index spaces that contain indexes on table clones. You cannot specify CLONE on REPAIR VERSIONS, because table clones do not have versions.

Refer to the syntax shown in Figure 4-15. If you use multiple SET statements in one REPAIR statement with CLONE specified also, the CLONE keyword applies to all SET statements and LOCATE statements within the same REPAIR utility control statement.

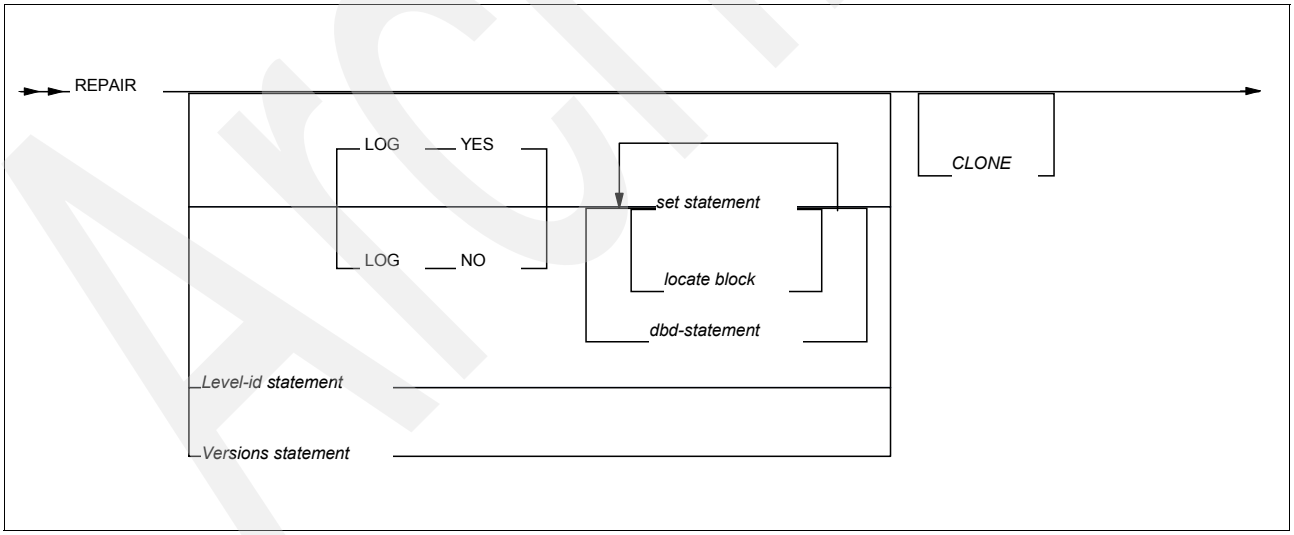


Figure 4-15 REPAIR UTILITY syntax

REPORT utility

If REPORT TABLESPACESET or REPORT RECOVERY is specified and the base objects have been cloned, information for both base and clone objects will be displayed in the output.

If you specify CLONE on the REPORT utility, you ask for information for only the specified objects that are table spaces that contain clone tables, indexes on clone tables, or index spaces that contain indexes on clone tables.

If you specify CLONE together with TABLESPACESET, REPORT also processes related LOBS. Since referential constraints are not allowed for tables with clones, TABLESPACESET does not list depending table objects for your clone tables.

The REPORT TABLESPACESET utility has a new keyword, SHOWDSNS.

SHOWDSNS specifies that the VSAM data set names for each table space or index space are to be included in the TABLESPACESET report. Data set names for base objects are shown in the section titled TABLESPACE SET REPORT. Data set names for CLONE objects are shown in the section titled CLONE TABLESPACE SET REPORT. This report is only prepared if the base objects have been cloned.

The output is shown in Example 4-1.

Example 4-1 Output of REPORT TABLESPACESET DB1.TS1 SHOWDSNS

```
DSNU050I    157 14:32:01.57 DSNUGUTC -  REPORT TABLESPACESET DB1.TS1 SHOWDSNS
DSNU587I  -DB9A 157 14:32:01.57 DSNUPSET - REPORT TABLESPACE SET WITH TABLESPACE

TABLESPACE SET REPORT:

TABLESPACE         : DB1.TS1
  PART: 0001   DSN : DB9AU.DSNDBC.DB1.TS1.I0001.A001
  TABLE            : PAOLOR7.BASETAB
  INDEXSPACE       : DB1.BASEIX1
               DSN : DB9AU.DSNDBC.DB1.BASEIX1.I0001.A001
  INDEX            : PAOLOR7.BASEIX1
  INDEXSPACE       : DB1.BASEIX2
               DSN : DB9AU.DSNDBC.DB1.BASEIX2.I0001.A001
  INDEX            : PAOLOR7.BASEIX2


LOB TABLESPACE SET REPORT:

TABLESPACE                 : DB1.TS1

  BASE TABLE               : PAOLOR7.BASETAB
  PART: 0001         COLUMN : LOBCOL1
    LOB TABLESPACE         : DB1.LXAFSGSF
                       DSN : DB9AU.DSNDBC.DB1.LXAFSGSF.I0001.A001
                       DSN : DB9AU.DSNDBC.DB1.LXAFSGSF.I0001.A001
      AUX TABLE            : PAOLOR7.BASETLOBCOXAFTKXN7
      AUX INDEXSPACE       : DB1.IBASETLO
                       DSN : DB9AU.DSNDBC.DB1.IBASETLO.I0001.A001
      AUX INDEX            : PAOLOR7.IBASETLOBCOXAFT1BD



CLONE TABLESPACE SET REPORT:

TABLESPACE         : DB1.TS1
  PART: 0001   DSN : DB9AU.DSNDBC.DB1.TS1.I0002.A001
  CLONE TABLE      : SABINE.CLONETAB
```

```
  INDEXSPACE       : DB1.BASEIX1
             DSN : DB9AU.DSNDBC.DB1.BASEIX1.I0002.A001
  INDEX            : PAOLOR7.BASEIX1
  INDEXSPACE       : DB1.BASEIX2
             DSN : DB9AU.DSNDBC.DB1.BASEIX2.I0002.A001
  INDEX            : PAOLOR7.BASEIX2


LOB TABLESPACE SET REPORT:

TABLESPACE                : DB1.TS1

  BASE TABLE              : PAOLOR7.BASETAB
  PART: 0001        COLUMN : LOBCOL1
    LOB TABLESPACE        : DB1.LXAFSGSF
                      DSN : DB9AU.DSNDBC.DB1.LXAFSGSF.I0002.A001
      AUX TABLE           : PAOLOR7.BASETLOBC0XAFTKXN7
      AUX INDEXSPACE      : DB1.IBASETLO
                      DSN : DB9AU.DSNDBC.DB1.IBASETLO.I0002.A001
      AUX INDEX           : PAOLOR7.IBASETLOBC0XAFT1BD
```

UNLOAD utility

If you specify the name of the clone table in the FROM TABLE clause, you do not need to specify the CLONE keyword to unload the clone table. If, however, you specify the table space name in your UNLOAD syntax, you must specify CLONE to unload data from the clone table.

DSN1LOGP utility

DSN1LOGP is a so-called stand-alone utility, which in contrast to online utilities can run while DB2 is not up and running. Since this is the case, DB2 cannot determine what data set instance number is associated with a base or clone object. Following this, it would not help to provide you with an additional CLONE keyword. Instead, another mechanism is used to identify which data manipulation belongs to the base table of a table space and which one belongs to the clone table.

Archived

To make it more transparent for you, we must step back a little bit. As described in 4.2.1, “Creation of a clone table” on page 81, when a clone table is created, the OBID and the PSID is the same for both instances of the table space. You can also see this in the same output of SYSIBM.SYSTABLES for base table clone9base and clone table clone9clone and of SYSIBM.SYSTABLESPACE, as shown in Figure 4-16.

```
SELECT NAME, TYPE, DBNAME, DBID,TSNAME
FROM SYSIBM.SYSTABLES WHERE NAME LIKE 'CLONE9%';
---------+---------+---------+---------+---------+-
NAME          TYPE  DBNAME          DBID  TSNAME
---------+---------+---------+---------+---------+-
CLONE9BASE    T     SABINE           267  CLONE9
CLONE9CLONE   C     SABINE           267  CLONE9

SELECT NAME, PSID, DBNAME, DBID,INSTANCE, CLONE
FROM SYSIBM.SYSTABLESPACE WHERE NAME LIKE 'CLONE9%';
---------+---------+---------+---------+---------+---------+--
NAME            PSID  DBNAME          DBID  INSTANCE  CLONE
---------+---------+---------+---------+---------+---------+--
CLONE9            80  SABINE           267         1  Y
```

Figure 4-16 Base and clone table in SYSTABLES

We have used the statements shown in Figure 4-17 to create the base and clone tables and to insert data into them.

```
CREATE TABLESPACE CLONE9 IN SABINE MAXPARTITIONS 10;
CREATE TABLE CLONE9BASE LIKE DSN8910.EMP IN SABINE.CLONE9;
INSERT INTO CLONE9BASE SELECT * FROM DSN8910.EMP;
ALTER TABLE CLONE9BASE ADD CLONE CLONE9CLONE;
INSERT INTO CLONE9CLONE SELECT * FROM DSN8910.EMP;
```

Figure 4-17 CREATE base and clone table statements

As you can see, this is all done in one unit of work (UR). If you now print the summary of this UR, you can get the information shown in Figure 4-18 on page 98. Although the information in the catalog only shows one PSID, the DSN1LOGP output shows that the PSID used is almost the same. The difference is that the page set containing the clone table is identified by a PSID for which the high order bit value is set to ’1’. Due to that, you can identify all log records that

are written for the clone table easily, because their PSIDs, which show up as OBIDs in the DSN1LOGP output, all starting with 8. In our example, table space CLONE9, which has a decimal PSID of 80, is listed with hexadecimal OBID of x'0050' in the DSN1LOGP output for the base table and with x'8050' for the clone table.

```
DSN1LPRT UR  CONNID=TSO       CORRID=PAOLOR7       AUTHID=PAOLOR7   PLAN=DSNESPCS
START DATE=06.255 TIME=20:06:23  DISP=COMMITTED           INFO=COMPLETE
STARTRBA=00001E0D013C  ENDRBA=00001E0F9581  STARTLRSN=BF65AD878B71  ENDLRSN=BF65B1889A13
    NID=* LUWID=USIBMSC.SCPDB9B.BF65AD8779DC.0001
    COORDINATOR=* PARTICIPANTS=*
    DATA MODIFIED:
DATABASE=0006=DSNDB06   PAGE SET=0010=SYSCOPY
DATABASE=0006=DSNDB06   PAGE SET=0072=DSNUCH01
DATABASE=0006=DSNDB06   PAGE SET=0076=DSNUCX01
DATABASE=010B=SABINE    PAGE SET=0050=CLONE9
DATABASE=0006=DSNDB06   PAGE SET=0009=SYSDBASE
DATABASE=0006=DSNDB06   PAGE SET=005C=DSNDSX01
DATABASE=0006=DSNDB06   PAGE SET=00C7=DSNDPX01
DATABASE=0006=DSNDB06   PAGE SET=0163=DSNDPX02
....
....
DATABASE=0006=DSNDB06   PAGE SET=02F5=DSNATX04
DATABASE=010B=SABINE    PAGE SET=8050=CLONE9
DATABASE=0001=DSNDB01   PAGE SET=001F=DBD01
```

Figure 4-18 DSN1LOGP summary of UR that created the base and clone table

You need to know about this if you plan to analyze the DSN1LOGP output, because if you use DBID(xxxx) and OBID(xxxx) in your DSN1LOGP JCL to reduce the number of written log records and you just want to see all changes that have been done for the clone table, you would have to specify DBID(010B) OBID(8050) as SYSIN for DSN1LOGP, while for the base table the input would be DBID(010B) OBID(0050).

Apart from the importance for you to know which table belongs to which log record, it is essential for DB2 itself to be able to identify which log record belongs to which table space (for example, during forward log recoveries). This is described in Figure 4-19 on page 99. The figure shows that when the table space and the base table are created, DB2 adds one row into SYSIBM.SYSTABLESPACE. The PSID shown there is decimal and is the same as what you can see as PAGE SET ID in Figure 4-17 on page 97. When you start inserting rows into the base table subsequently, the associated log records indicate that they are written for a specific OBID, which is the same as the PAGE SET ID. If you add a clone table to this base table and insert rows into it, the log records that are caused by this insertion now refer to OBID x'8050', which is the same OBID as for the base table with the difference that the high order bit is turned on here, which leads to the 8 as first hexadecimal value.

When you now run for example a RECOVER utility to recover the base table data, DB2 refers to SYSTABLESPACE to find out the instance number that applies to the base table. In Figure 4-19 on page 99 it is '1' for the first recovery. DB2 also checks the information in SYSCOPY. Since at the time the first recover is performed, no data exchange has occurred yet, DB2 just has to apply those log records that belong to OBID x'0050'.

Let us assume that we perform EXCHANGE DATA now. The INSTANCE information in SYSTABLESPACE is updated and shows a '2' after the exchange has ended successfully. Also, DB2 adds two new rows to SYSCOPY to document that data exchange has occurred.

If we now insert more rows into the base table, the corresponding log records show x'8050' as the associated OBID, because the instance number for the base table is now 2.

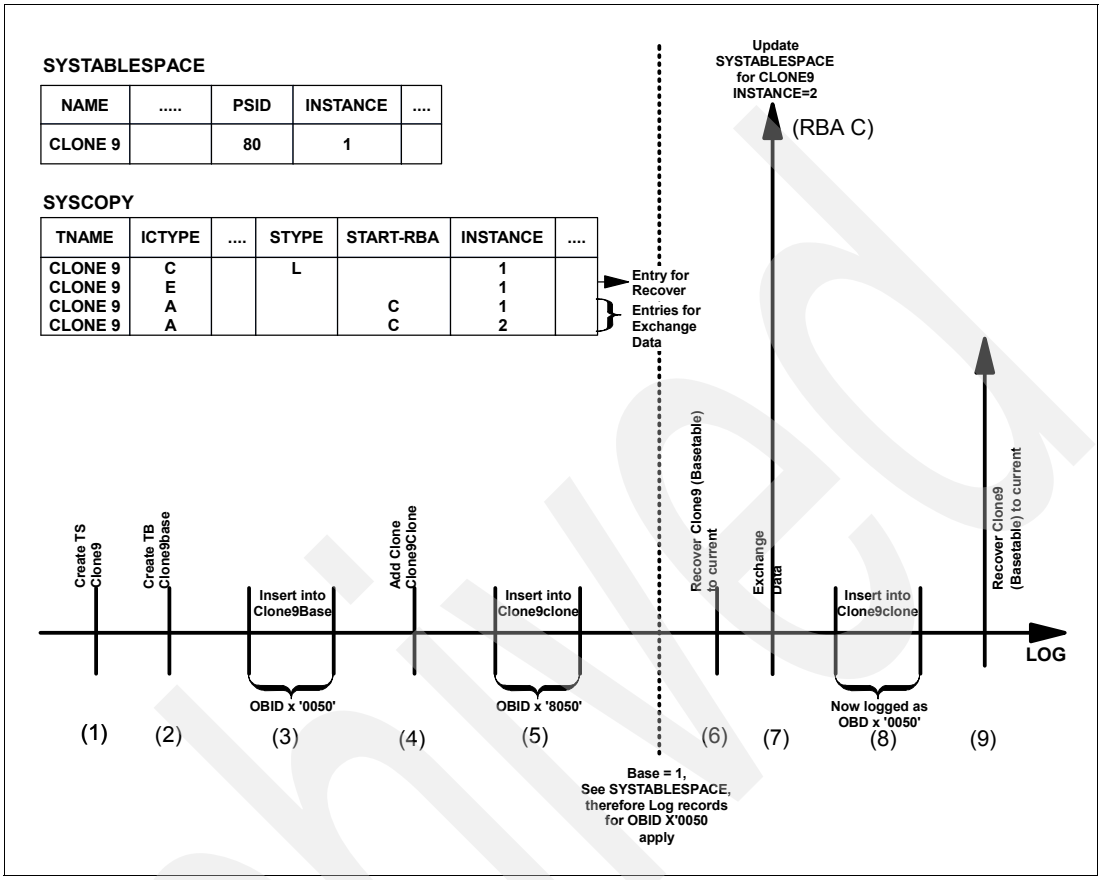


Figure 4-19 Usage of high order bit for clone tables

If we recover the clone table data once again to current, DB2 must now again check SYSTABLESPACE to see which instance number is currently in use for the base table and derive which page set contains the clone table. In our example, SYSIBM.SYSTABLESPACE currently contains 2 as INSTANCE number which means that page set I0001 contains the current clone table data. All changes being made for base or clone objects to page set I0001 are logged as OBD x'0050' in our example.

DB2 can now search for all log records for OBDx'0050' to recover the page set currently holding the clone table.

Note: Information about EXCHANGE DATA is stored in SYSIBM.SYSCOPY. This information is NOT used as information about a LOG NO event.

The sequence of events for RECOVER in Figure 4-19 is as follows:

- (1) - It creates a table space for the clone in page set I0001.
- (2) - It creates the clone table in page set I0001.
- (3) - Inserts go to page set I0001, thus they are logged for PSID x'0050'.
- (4) - It adds page set I0002 (instance 2).
- (5) - Inserts go to page set I0002, thus they are logged for PSID x'8050'.

- (6) - If you omit the CLONE keyword during RECOVER, the page set containing the base table is going to be recovered. In our case, this is page set I0001.

  The reference to the page set is always the instance number in SYSIBM.SYSTABLESPACE. The number stored in column INSTANCE indicates the page set currently containing the base table space.
- (7) - Due to EXCHANGE, the instance number is changed in SYSIBM.SYSTABLESPACE and the EXCHANGE is recorded in SYSIBM.SYSCOPY as A (ALTER).
- (8) - Logged as x'0050'.
- (9) - Recover now recovers everything that belongs to the page set that is currently storing the clone table.

If cloning starts with base table being stored on page set I0001, the records being written to the log for this page set are always those with the high order bit turned off, that is x'0050' in our example.

Once the clone, and therefore page set I0002, is added, log records x'8050' always belong to I0002.

Depending on your Exchange activity, your base table might be stored either on I0001 or on I0002. As a result, when running the utility:

```
RECOVER TABLESPACE .... CLONE
```

the page set containing the clone table is being recovered. The clone table is stored on the page set that is NOT shown under INSTANCE in SYSIBM.SYSTABLESPACE.

Referring to our example, when we recover the clone table after all other actions, the recovery starts at (1), applies (2), the inserts (3) and the inserts that have been made for the clone table after EXCHANGE (7).

4.3 Recovery to point in time with consistency

When an application or part of an application needs to be recovered to a point other than the current time, then the consistency of the data needs to be assured. The QUIESCE utility can be used to establish a point of consistency that can subsequently be used. The -ARCHIVE LOG MODE(QUIESCE) could perform a similar function.

There are, however, two drawbacks to these two options:

- Firstly, the point to recover back to must have been anticipated in advance (that is, there must be a quiesce at the time that the actual recovery is now required).
- Secondly, as workloads have increased, it has become progressively harder to obtain a quiesce or the quiesce has become disruptive. This is especially true at the system level or for large applications. Applications with work that does not commit regularly can significantly disrupt the taking of quiesces.

In V9 NFM the RECOVER utility is enhanced to automatically detect the uncommitted work at the point in time selected for the recovery. DB2 will then roll back the changes on objects being recovered. After the recover, all the objects involved in the recovery will be in a transactionally consistent state. Note that it is the user's responsibility to ensure that *all* related table spaces and index spaces are included in the RECOVER statement. Recovery to a consistent state will occur when RECOVER is run with either the TOLOGPOINT or TORBA keywords.

The existing V8 behavior will apply to RECOVER with TOCOPY, TOLASTCOPY, and TOLASTFULLCOPY using SHRLEVEL CHANGE copy. For these TOCOPY/TOLASTCOPY/TOLASTFULLYCOPY SHRLEVEL CHANGE recoveries, consistency is not ensured. If consistency is desired, the user should locate the respective entry for that image copy in SYSCOPY and use the PIT_RBA column as a recovery point with TORBA or TOLOGPOINT.

RECOVER to current is unaffected.

Two new phases have been added to RECOVER TORBA and TOLOGPOINT — LOGSCR and LOGUNDO. These new phases occur after the LOGAPPLY phase (LOGAPPLY works as normal applying all REDO records).

The first phase after LOGAPPLY, the LOGSCR, is run for each DB2 member for which it is determined there was an active unit of recovery (UR). If there was no UR this phase is skipped. During the LOGCSR phase, for each DB2 member that may have a UR that has changed the objects being recovered and also were not complete at the recovery point, RECOVER processes the log. The log is read forward from the last checkpoint for the specific DB2 member to the recovery point. The URs that were active (INFLIGHT, INABORT, INDOUBT, or POSTPONED ABORT) during the recovery point and also changed the objects being recovered are identified. Message DSNU1553I indicates the URs for each member.

Provided that there are units of recovery requiring objects needing to be backed out, a new phase called LOGUNDO will occur. During the LOGUNDO phase, the RECOVER utility will back out the changes made on recovered objects by active URs. In a data sharing environment, this will be done one member at a time. This backout cannot exploit Fast Log Apply. For each member, RECOVER will read backwards from the recovery point. RECOVER traverses the log just once per member, backing out the updates for the URs identified in LOGCSR phase. During the LOGUNDO phase, the RECOVER utility will periodically issue message DSNU1555I. This can be used to monitor progress.

Note: If there is a long-running UR, then the LOGUNDO phase could take a significant amount of time. By monitoring the DSNU1553I message, the user may determine that it is more appropriate to terminate (-TERM UTIL command) the recovery and select an alternative recovery point.

In V9 an improvement was made to the RECOVER utility where if the -TERM UTIL command is issued in the LOGAPPLY, LOGCSR, or LOGUNDO phases, the command will be recognized at the next commit point and the utility will terminate itself gracefully. In prior releases, the -TERM UTIL command was ignored in the LOGAPPLY phase.

Figure 4-20 shows the output of a recovery job with a TORBA. In this example there was a single transaction active at the recovery point. The transaction affected two tables that were part of an RI set (in two table spaces).

```
1DSNU000I    052 19:58:01.49 DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = COPYTS
 DSNU1044I   052 19:58:01.53 DSNUGTIS - PROCESSING SYSIN AS EBCDIC
0DSNU050I    052 19:58:01.54 DSNUGUTC -  LISTDEF ROY INCLUDE TABLESPACE DSN00218.ROYC INCLUDE TABLESPACE
 DSN00217.ROYP
 DSNU1035I   052 19:58:01.54 DSNUILDR - LISTDEF STATEMENT PROCESSED SUCCESSFULLY
0DSNU050I    052 19:58:01.54 DSNUGUTC -  RECOVER LIST ROY TORBA X'00002C6A42C3'
 DSNU1033I   052 19:58:01.55 DSNUGULM - PROCESSING LIST ITEM: TABLESPACE DSN00218.ROYC
 DSNU1033I   052 19:58:01.55 DSNUGULM - PROCESSING LIST ITEM: TABLESPACE DSN00217.ROYP
 DSNU532I    052 19:58:01.57 DSNUCBMD - RECOVER TABLESPACE DSN00218.ROYC   START
 DSNU515I    052 19:58:01.57 DSNUCBAL - THE IMAGE COPY DATA SET PAOLOR9.COPYA3 WITH DATE=20070221 AND TIME=195306
                       IS PARTICIPATING IN RECOVERY OF TABLESPACE DSN00218.ROYC
 DSNU504I    052 19:58:01.94 DSNUCBMD - MERGE STATISTICS FOR TABLESPACE DSN00218.ROYC   -
                       NUMBER OF COPIES=1
                       NUMBER OF PAGES MERGED=3
                       ELAPSED TIME=00:00:00
 DSNU532I    052 19:58:01.96 DSNUCBMD - RECOVER TABLESPACE DSN00217.ROYP   START
 DSNU515I    052 19:58:01.96 DSNUCBAL - THE IMAGE COPY DATA SET PAOLOR9.COPYP3 WITH DATE=20070221 AND TIME=195235
                       IS PARTICIPATING IN RECOVERY OF TABLESPACE DSN00217.ROYP
 DSNU504I    052 19:58:02.32 DSNUCBMD - MERGE STATISTICS FOR TABLESPACE DSN00217.ROYP   -
                       NUMBER OF COPIES=1
                       NUMBER OF PAGES MERGED=2
                       ELAPSED TIME=00:00:00
DSNU830I  -DB9B 052 19:58:01.59 DSNUCARS - INDEX PAOLOR9.CIN IS IN REBUILD PENDING
 DSNU831I  -DB9B 052 19:58:01.59 DSNUCARS - ALL INDEXES OF DSN00218.ROYC ARE IN REBUILD PENDING
 DSNU830I  -DB9B 052 19:58:01.98 DSNUCARS - INDEX PAOLOR9.ROYPABCD_#_JWY IS IN REBUILD PENDING
 DSNU830I  -DB9B 052 19:58:01.98 DSNUCARS - INDEX PAOLOR9.PIN IS IN REBUILD PENDING
 DSNU831I  -DB9B 052 19:58:01.98 DSNUCARS - ALL INDEXES OF DSN00217.ROYP ARE IN REBUILD PENDING


 DSNU513I  -DB9B 052 19:58:02.34 DSNUCALA - RECOVER UTILITY LOG APPLY RANGE IS RBA 00002C699158 LRSN 00002C699158
TO
                                                      RBA 00002C6A42C3 LRSN 00002C6A42C3
 DSNU1510I   052 19:58:03.42 DSNUCBLA - LOG APPLY PHASE COMPLETE, ELAPSED TIME = 00:00:01
 DSNU1550I -DB9B 052 19:58:03.42 DSNUCALC - LOGCSR IS STARTED FOR MEMBER         , PRIOR CHECKPOINT RBA =
 X'00002A0192C4'
 DSNU1551I -DB9B 052 19:58:04.24 DSNUCALC - LOGCSR IS FINISHED FOR MEMBER         , ELAPSED TIME = 00:00:00
 DSNU1552I -DB9B 052 19:58:04.24 DSNUCALC - LOGCSR PHASE COMPLETE, ELAPSED TIME = 00:00:00
 DSNU1553I -DB9B 052 19:58:04.24 DSNUCALC - RECOVER DETECTS THE FOLLOWING ACTIVE URS:
              INFLIGHT = 1, INABORT = 0, INDOUBT = 0, POSTPONED ABORT = 0

                MEM    T  CONNID     CORRID     AUTHID    PLAN   S     URID       DATE       TIME
                       B TSO      PAOLOR4      PAOLOR4  DSNESPCS F 00002C69A4F2 2007-02-22 00.55.16
                    DBNAME  SPACENAME DBID/PSID PART     RBA
                   DSN00218 ROYC      01E5/0002 0001 00002C69A783
                   DSN00217 ROYP      01E4/0002 0001 00002C69A6B8


DSNU1554I -DB9B 052 19:58:04.38 DSNUCALU - LOGUNDO IS STARTED FOR MEMBER
 DSNU1556I -DB9B 052 19:58:04.38 DSNUCALU - LOGUNDO IS FINISHED FOR MEMBER         , ELAPSED TIME = 00:00:00
 DSNU1557I -DB9B 052 19:58:04.38 DSNUCALU - LOGUNDO PHASE COMPLETE, ELAPSED TIME = 00:00:00
DSNU500I    052 19:58:04.43 DSNUCBDR - RECOVERY COMPLETE, ELAPSED TIME=00:00:02
 DSNU010I    052 19:58:04.44 DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=4
```

Figure 4-20 RECOVER TORBA

The RECOVER is for both table spaces. The transaction is identified as INFLIGHT. Both table spaces are shown as needing to be backed out. See DSNU1553I.

Important: When a table space is being recovered, it is essential that all other table spaces that are being changed by the same transactions at the recovery point are included in the *same* RECOVER statement. In the example above we use a LISTDEF to do this. RECOVER will never back out changes for objects that are not being recovered.

4.4 APPEND

The APPEND YES/NO option of the CREATE TABLE and ALTER TABLE statement might be a very helpful new table trait.

The APPEND processing option instructs DB2 to ignore clustering during SQL insert and online LOAD processing. Rather than attempting to insert rows in cluster-preserving order, rows are appended at the end of the table or appropriate partition. For range-partitioned table spaces, the appropriate partition is the one dictated by the value in the row's partitioning column. For partition-by-growth table spaces, the appropriate partition is any partition with space at the end.

You can benefit from the APPEND processing option even if you still rely on having a clustering index defined on your table. You can insert or LOAD your data quickly and regain the clustering order later by running REORG.

You cannot use the APPEND option for:

- LOB tables
- XML tables
- Tables residing in work files

Catalog table SYSIBM.SYSTABLES has got a new column named APPEND where it stores a 'Y' if you used the keyword APPEND to create or alter your table and a 'N' if APPEND is not used. The default value for the APPEND keyword is 'N'.

4.5 Conditional Restart Log Truncation by Timestamp

Sometimes you are forced to restore your whole DB2 subsystem to a previous point in time. The only method to do this without ending up with inconsistent data is to use a conditional restart with log truncation. Up to DB2 V8, you were forced to specify a specific RBA or LRSN at which the log truncation was to be performed. Those conditional restarts are initiated via the creation of so-called conditional restart control records using the DSNJU003 stand-alone utility. Up to now you had to use either keyword ENDRBA or ENDLRSN to specify the log truncation point.

There are at least two reasons that force you to use a conditional restart with ENDRBA or ENDLRSN:

- A disaster occurred. In this case, you would specify the highest ENDRBA that is still available on the remaining logs in order to minimize the loss of data.
- You want to back out changes of multiple applications. This might primarily be the case in test environments in ERP systems. If this is what is driving you to restore your system to a previous point in time, then it is most likely that you know the date and time to which you would like to restore, but you do not know exactly which RBA represents the correct log point. There are means to approximately find those RBAs and many of you currently use those methods, but it might be easier if you could just specify a time stamp at which the log is to be truncated.

Starting with DB2 V9, you can now specify time stamps for both normal conditional restart and for the RESTORE system utility. It is now allowed to use the existing keyword ENDTIME on the CRESTART statement. In addition to that, a new keyword SYSPITRT has been added as an extension to the existing SYSPITR keyword for the RESTORE system utility. The default time is GMT.

You can specify a time stamp for both ENDTIME and SYSPITRT. DSNJU003 converts the time stamp to a store-clock (STCK) value and uses the first 6 bytes plus any data sharing LRSN delta to generate an LRSN or STCK value for log truncation. The time stamp value has the following format:

```
yyyydddhhmmsst
```

yyyy	This indicates the year. Allowed values are 1989–2099.
ddd	This indicates the day of the year. Valid values are 0–365 and 0–366 in leap years.
hh	This indicates the hour. Valid values are 0–23.
mm	This indicates the minutes. Valid values are 0–59.
ss	This indicates the seconds. Valid values are 0–59.
t	This indicates tenth of a second.

An example for September 14, 2006, 12:36:00 p.m. is:

2006257141236000

4.5.1 ENDTIME usage

Attention: You must use GMT to specify ENDTIME and SYSPITRT.

The scenario in Figure 4-21 on page 104 shows two checkpoints and one unit of recovery running between them. We first point out the difference between GMT and your local time. The majority of time stamps that you see when you print the log using the DSNJU004 utility, for example, represent the GMT. In our example, the first checkpoint that started at RBA 1E427090 has been written at 01:03:06 on September 14, 2006 GMT, which was 18:03:06 on September 13, 2006 in San Jose, California. The application UR with URID 1E444444 started a little bit later and has been completed at RBA 1E450000. Some time later you find that this UR has done something wrong. Very many page sets have be changed by this UR. To back out those changes, you must reset the whole DB2 subsystem to a point in time prior to URID 1E444444. You might know that everything was still okay with your system at 18:43 local time. As a result, you create your conditional restart control record with ENDTIME 20062570143000. Note that since you must use GMT, this is now much later than 18:43 on even the following day.

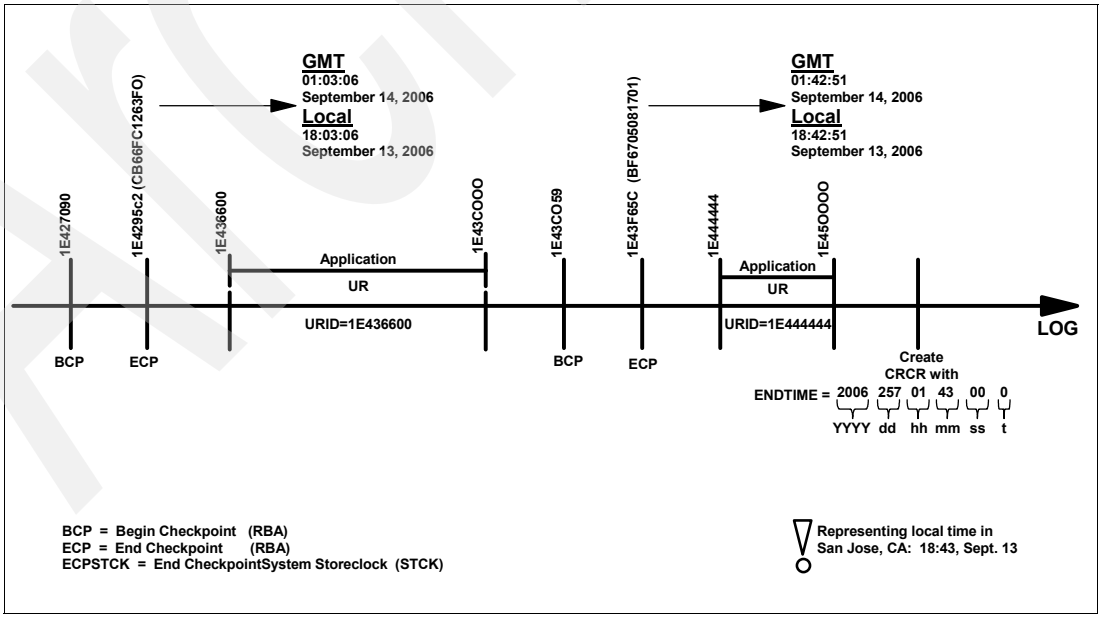


Figure 4-21 ENDTIME in GMT

The DSNJU003 job might look like that shown in Figure 4-22.

```
//DSNTLOG EXEC PGM=DSNJU003
//STEPLIB  DD  DISP=SHR,DSN=DB9B9.SDSNEXIT
//         DD  DISP=SHR,DSN=DB9B9.SDSNLOAD
//SYSUT1   DD  DISP=OLD,DSN=DB9BU.BSDS01
//SYSUT2   DD  DISP=OLD,DSN=DB9BU.BSDS02
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSIN    DD  *
CRESTART CREATE,ENDTIME=20062570143000
```

Figure 4-22 DSNJU003 CRESTART with ENDTIME

We printed the BSDS after the successful creation of the conditional restart control record. The resulting active record looks like that shown in Figure 4-23.

```
CRCR IDENTIFIER  0007
     USE COUNT   0
     RECORD STATUS
         CRCR ACTIVE
         CRCR NOT USED
     PROCESSING STATUS
         FORWARD = YES
         BACKOUT = YES
     STARTRBA                        NOT SPECIFIED
     ENDRBA                          NOT SPECIFIED
     ENDLRSN                         NOT SPECIFIED
     ENDTIME                         20062570143000
     TRUNCATION STCK (FROM ENDTIME)  BF6704FD5D10
     EARLIEST REQUESTED RBA          000000000000
     FIRST LOG RECORD RBA            000000000000
     ORIGINAL CHECKPOINT RBA         000000000000
     NEW CHECKPOINT RBA (CHKPTRBA)   00001E43991C
     END CHECKPOINT RBA              00001E43BE28
     CRCR CREATED            01:44:33 SEPTEMBER 14, 2006
     TIME OF CHECKPOINT      01:42:51 SEPTEMBER 14, 2006
     RESTART PROGRESS                 STARTED     ENDED
                                      =======     =====
         CURRENT STATUS REBUILD          NO         NO
         FORWARD RECOVERY PHASE          NO         NO
         BACKOUT RECOVERY PHASE          NO         NO
```

Figure 4-23 Active CRCR with ENDTIME usage

When you restart your DB2 subsystem with the CRCR shown in Figure 4-23 being active, you can see the messages shown in Figure 4-24. As you can see from the bold letters, it now indicates the ENDTIME and calculated STCK value rather than the ENDRBA value as it would be if you used the ENDRBA keyword on CRESTART.

Archived

```
DSNY001I  -DB9B SUBSYSTEM STARTING
DSNJ127I  -DB9B SYSTEM TIMESTAMP FOR BSDS= 06.256 21:43:28.81
DSNJ255I  -DB9B DSNJW306 CONDITIONAL RESTART ENDTIME  132
OF 20062570143000 WILL RESULT IN LOG TRUNCATION AT STCK BF6704FD5D10
016 DSNJ245I  -DB9B CONDITIONAL RESTART RECORD INDICATES TRUNCATION AT STCK
BF6704FD5D10.  REPLY Y TO CONTINUE,
         N TO CANCEL
```

Figure 4-24 ENDTIME usage DB2 start message in MSTR address space

In a data sharing environment, all log records that are greater than the specified LRSN will be truncated. In a non-data sharing environment, a STCK value will be calculated from your specified end time, which is then used to identify the log records that are to be truncated as a result of your conditional restart. In fact, all log records with an LRHTIME (Log Record Header Time) value greater than the generated STCK value will be truncated. If you refer to Figure 4-23, the calculated STCK value for the specified ENDTIME 20062570143000 is BF6704FD5D10. Having the above explanation in mind, this means that all log records whose LRHTIME is greater than BF6704FD5D10 are discarded in our example.

4.5.2 SYSPITRT usage

The SYSPITRT usage is very similar to ENDTIME. The format of the time stamp that you specify on your CRESTART statement for the DSNJU003 utility is the same as explained above. Refer to Figure 4-25 for a sample DSNJU003 CRESTART with SYSPITRT utility statement.

```
//DSNTLOG EXEC PGM=DSNJU003
//STEPLIB  DD  DISP=SHR,DSN=DB9B9.SDSNEXIT
//         DD  DISP=SHR,DSN=DB9B9.SDSNLOAD
//SYSUT1   DD  DISP=OLD,DSN=DB9BU.BSDS01
//SYSUT2   DD  DISP=OLD,DSN=DB9BU.BSDS02
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSIN    DD  *
CRESTART CREATE,SYSPITRT=20062570143000
```

Figure 4-25 DSNJU003 CRESTART with SYSPITRT

Refer to Figure 4-26 to see how an active CRCR with SYSPITRT looks. As you can see from the bold letters, using this CRCR places the DB2 subsystem in SYSTEM LEVEL RECOVERY MODE. Refer to *Disaster Recovery with DB2 UDB for z/OS,* SG24-6370, to learn more about the general usage of the BACKUP and RESTORE system utilities.

Archive

```
CRCR IDENTIFIER  0008
     USE COUNT   0
     RECORD STATUS
         CRCR ACTIVE
         CRCR NOT USED
     PROCESSING STATUS
         FORWARD = YES
         BACKOUT = YES
     SYSPITR SYSTEM LEVEL RECOVERY MODE RESTART
     STARTRBA                         NOT SPECIFIED
     ENDRBA                           NOT SPECIFIED
     ENDLRSN                          NOT SPECIFIED
     ENDTIME                          20062562033000
     TRUNCATION STCK (FROM ENDTIME)   BF66BFB30570
     EARLIEST REQUESTED RBA           000000000000
     FIRST LOG RECORD RBA             000000000000
     ORIGINAL CHECKPOINT RBA          000000000000
     NEW CHECKPOINT RBA (CHKPTRBA)    00001E403CC4
     END CHECKPOINT RBA               00001E406C24
     CRCR CREATED             21:19:50 SEPTEMBER 14, 2006
     TIME OF CHECKPOINT       20:23:50 SEPTEMBER 13, 2006
     RESTART PROGRESS                   STARTED     ENDED
                                        =======     =====
         CURRENT STATUS REBUILD           NO          NO
         FORWARD RECOVERY PHASE           NO          NO
         BACKOUT RECOVERY PHASE           NO          NO
```

Figure 4-26 CRCR with SYSPITRT

4.6 REFRESH for EARLY code

Ever since DB2 was first developed about 25 years ago, the activation of a new Early code was always a little bit cumbersome, because an IPL was necessary for this activation. Starting with DB2 9 for z/OS, an IPL is no longer needed. Instead, you can make use of the new REFRESH DB2 command whose syntax is shown in Figure 4-27.

```
REFRESH DB2,EARLY
```

Figure 4-27 REFRESH command

You can only issue this command while DB2 is shut down. If you attempt to execute this command for a DB2 subsystem that is currently started, message DSNY003I, which is also shown in Figure 4-28, informs you of the failure of this command.

```
 )REFRESH DB2,EARLY
 DSNY003I  ) SUBSYSTEM IS ALREADY ACTIVE
 DSN9023I  ) DSNYSCMD 'REFRESH DB2' ABNORMAL COMPLETION
```

Figure 4-28 REFRESH - Error example

The successful execution of this command reloads the EARLY load modules and rebuilds the EARLY control block. On the SYSLOG you can see a message that is similar to the one displayed in Figure 4-29.

```
-DB9B REFRESH DB2,EARLY
DSN3100I -DB9B DSN3UR00 - SUBSYSTEM DB9B READY FOR START COMMAND
DSN3117I DB9B MAINTENANCE LEVELS    651
  CSECT     DATE      APAR        CSECT     DATE      APAR
  DSN3UR00  09/15/06  UK18085     DSN3RDMP  02/02/07  UK21823
  DSNAPRHX  01/14/07  UK21248     DSN3RRXF  01/14/07  UK21248
  DSN3RRSX  12/08/06  UK20427     DSN3AC0X  07/19/06  UK16439
  DSN3EC0X  07/19/06  UK16439     DSNAET03  03/31/06  09.05
  DSNAET04  03/31/06  11.14       DSNAPRH0  03/31/06  11.14
  DSNVRMTR  03/31/06  14.13       DSNVRSRB  03/31/06  14.13
  DSNZPARS  03/31/06  16.30       DSN3CL0X  03/31/06  16.38
  DSN3DEQ0  03/31/06  16.39       DSN3ECMS  03/31/06  16.39
  DSN3ENQ0  03/31/06  16.40       DSN3EST0  03/31/06  16.41
  DSN3RIB   03/31/06  16.43       DSN3RS0X  03/31/06  18.29
  DSN3RTR0  03/31/06  18.29       DSN3SPRX  03/31/06  18.29
```

Figure 4-29 REFRESH - Log information

Those of you who are operating with DB2 subsystems in data sharing mode must remember that this command only has member-scope. As a consequence, the -REFRESH command needs to be executed for every instance where a unique recognition character for the -START command has been defined.

In terms of authorizations, no DB2 authority is required. However, you can execute the command only from a z/OS console with the START command capability.

4.7 RENAME INDEX

There is an ongoing effort to give you more and more possibilities to change object characteristics without the need to drop and recreate the object. Starting with DB2 9 in NFM you can use the extended RENAME statement to rename existing indexes.

Refer to Figure 4-30 for the syntax diagram.

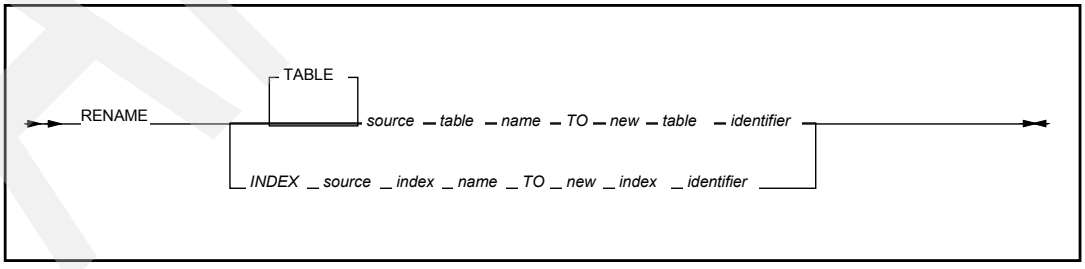


Figure 4-30 RENAME INDEX syntax

You can specify the source index name with implicit or explicit qualifier, but can not specify a qualifier for the new index name. The qualifier of the source index name is used to qualify the new name for the index.

As for any other object, an index is unambiguously identified within a database by its OBID. Renaming the index does not change this number. Plans and packages identify indexes used in their access paths by those IDs. As a result, your plans and packages will not be invalidated when you rename an index.

Tip: If you have a static DDL statement referencing the index in the application, after RENAME INDEX runs, the old index name is deleted from the catalog. When the application is rebound and the DDL statement still contains the old index name, the statement will be re-processed. The old index name cannot be found in the catalog and therefore the rebind will fail. To perform a successful rebind, you must modify the index name in the application.

In contrast to that, the index names are used for the identification of indexes that are used by statements that are stored in the dynamic statement cache. Due to this, statements in the dynamic statement cache will be invalidated if they ought to use the index that you renamed.

4.8 RENAME COLUMN

With DB2 Version 8, a lot of additional possibilities to change table and index schema have been added. With DB2 9 under some circumstances you are now able to rename columns of an existing table without having the need to drop and recreate the object. A new clause, RENAME COLUMN, is added to the ALTER TABLE statement. If ALTER TABLE RENAME COLUMN succeeds, all the existing objects associated with this column, such as table, indexes, RI, auxiliary tables, auxiliary indexes, foreign keys, and statistic information will function the same way as before renaming the column.

RENAME COLUMN results in the invalidation of any plan or package that is dependent on the table whose column is being renamed. Any attempt to execute the invalidated plan or package will trigger an automatic rebinding of the plan or package.

The simple syntax for altering a column name is:

```
ALTER TABLE tablename RENAME COLUMN oldcolumname TO newcolumnname
```

4.8.1 Effects of ALTER TABLE RENAME COLUMN

As mentioned before, all plans and packages referencing the table that hosts the renamed column are invalidated and result in an automatic rebind the next time that you use those plans or packages. We now have to discuss what influences the outcome of the automatic rebind attempt and which additional actions you might need to take.

Scenario 1 - The plan or package has not referenced the source column

If the plan or package does not reference the column that you have renamed, it is unaffected by your change. In this case there is no need for you to modify the plan or package, and the rebind will be successful.

Scenario 2 - The plan or package has referenced the source column

If the plan or package has referenced the source column the rebind will not be successful. The result of the attempt to automatically rebind your applications will indicate that the renamed column no longer exists in the table. In this case, you must modify, recompile, and rebind your applications.

Scenario 3 - Source column referenced and column added

The third scenario is not that common, but could occur. Let us assume that you have changed a column name first and have added a new column using the old column name afterwards. In this case the source column is referenced, but the rebind can be successful nevertheless. Refer to Example 4-2 for a detailed description of the scenario.

Example 4-2 Source column referenced and equally named column added afterwards

```
CREATE TABLE MYTABLE (MYCOL1 INT);
INSERT INTO TABLE MYTABLE VALUES (1);
SELECT MYCOL1 FROM MYTABLE; (this is the DML in the package MYPACKAGE)
-> the result of the query is 1

ALTER TABLE MYTABLE RENAME MYCOL1 TO MYCOL2;
ALTER TABLE MYTABLE ADD COLUMN MYCOL1 (C1 VARCHAR(10);
-> MYPACKAGE is invalidated

INSERT INTO TABLE MYTABLE (MYCOL1) VALUES ('ABCD')
-> execute MYPACKAGE will result in an automatic rebind and the automatic
rebind will succeed. However, execute the package will render 'ABCD'
instead of 1
```

Note: Although we do not think that this is a very common situation, you must be careful if you act as described in Example 4-2. Even though your programs might continue to run with return code zero, the results might not be the ones that you intended to produce.

4.8.2 Usage restrictions

There are a couple of restrictions that you must keep in mind when you plan to rename a column:

- The RENAME COLUMN keyword cannot be used in one statement with any other ALTER TABLE options.
- The RENAME COLUMN keyword cannot be repeated within one ALTER TABLE statement.
- ALTER TABLE RENAME COLUMN is not allowed when the source column is referenced in a view. If you try to do so, you will end up in SQLCODE -750. If you want to rename a column for a table that has views defined on it, you must drop the views first. This means that if not all of your table columns are referenced by a view, you are allowed to rename all unreferenced columns.
- If you try to run ALTER TABLE RENAME COLUMN for a table that has a trigger defined on it, the attempt results in SQLCODE -750. You must drop the trigger, rename the column, and recreate the trigger afterwards. This is the case for all columns (even those columns that are not specified in the trigger cannot be renamed).
- You cannot use ALTER TABLE RENAME COLUMN if the table is referenced in a Materialized Query Table (MQT).
- ALTER TABLE RENAME COLUMN is not allowed if the source column has a check constraint or a field procedure defined on it.
- ALTER TABLE RENAME COLUMN is not allowed if the table containing the column has a valid or edit procedure defined on it. You cannot drop edit and field procedures (you can never use the ALTER TABLE statement to rename those columns).

- ALTER TABLE RENAME COLUMN is not allowed if the source column has an index defined on it. You must drop all indexes that are referencing the specific column in order to be able to rename the column.
- You cannot rename a column if there is an index on expression or spatial index depends on the column.
- You cannot alter any column of the DB2 catalog.

4.9 UPDATE SCHEMA

This enhancement supports changing schema, creator, owner, and VCAT name of objects using the CATMAINT utility. This is a system-wide function, meaning that all objects are altered from one schema name to another. Creator, owner, and VCAT names are also updated in a similar manner.

Three new options, SCHEMA, OWNER, and VCAT, are added to the CATMAINT utility. Owner, creator, and schema names can be changed using the new SCHEMA option. Indexes, table spaces, and storage groups can be altered using the VCAT new option to use different integrated catalog facility catalog (or VCATNAME).

Ownership of objects can be changed to a role using the OWNER option. These options are only available in new-function mode and require installation SYSADM authority. After the CATMAINT utility completes, all of the existing objects whose owner, schema, or VCATNAME are changed will function in the same way as before.

Figure 4-31 shows the new syntax for the CATMAINT utility.

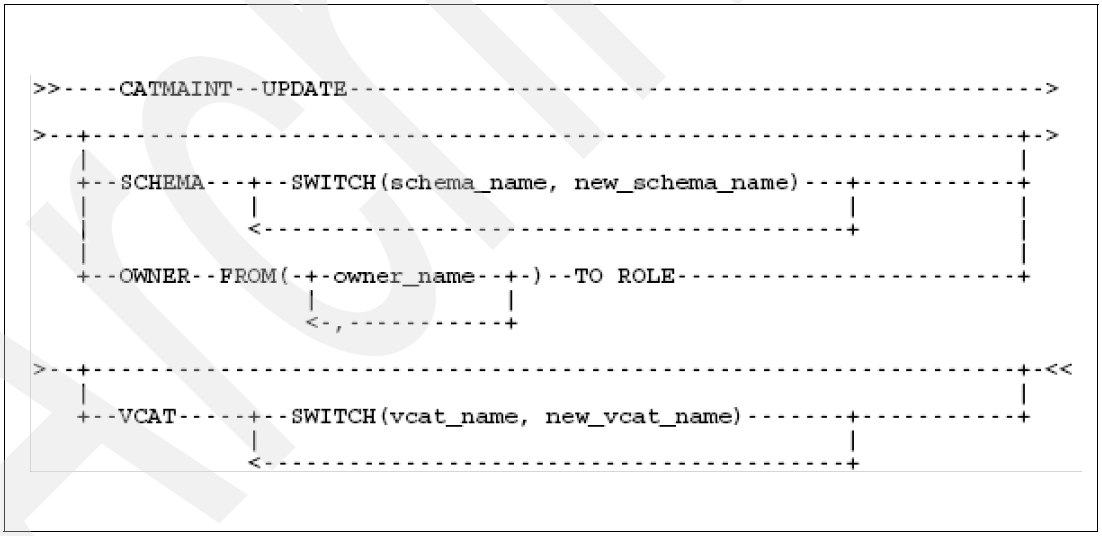


Figure 4-31 Update schema syntax diagram

SCHEMA SWITCH(schema_name,new_schema_name)

This process changes owner, creator, and schema of database objects, plans, and packages. This process updates every owner, creator, or schema name in the catalog and directory that matches the 'schema_name'. All grants made by or received by the original owner are changed to the new owner. You can change multiple names by repeating the SWITCH keyword, but you cannot specify the same name more than once. The names must be 8 bytes or less in EBCDIC representation. 'SYSIBM' is not allowed.

'schema_name' is a string that identifies the existing owner, creator, or schema to be changed. It is ignored if it does not identify any owner, creator, or schema names. It cannot identify any schema(qualifier) of any object on which triggers, views, SQL functions, and materialized query tables are dependent. It cannot be referenced in check condition in any check constraints. Ownership of objects is not changed if the owner is a role.

'new_schema_name' specifies the new name for the owner, creator, or schema. The name cannot be a schema that qualifies existing objects.

OWNER FROM (owner_name, owner_name2...) TO ROLE

Change the ownership of objects from a user to a role. The user running this utility has to be running under a trusted context with a role and this role becomes the owner. Privileges held on the object are transferred from the original owner to the role. The original user can be the grantor or grantee, and the original owners no longer have any privileges on the object. If the 'owner_name' does not own any object, it is ignored. OWNER FROM and SCHEMA SWITCH are mutually exclusive. You cannot specify both clauses in the same CATMAINT UPDATE statement.

Ownership of roles is also changed. However, if the associated trusted context role is owned by the 'owner_name', the ownership of this role is not changed. A role cannot be owned by itself.

VCAT SWITCH (vcat_name,new_vcat_name)

This changes the catalog name used by storage groups, user indexes, and table spaces. This function is similar to the ALTER TABLESPACE USING VCAT statement for changing the catalog name. You need to move the data for the affected indexes or table spaces to the data set on the new catalog in a separate step. The *DB2 Version 9.1 for z/OS Administration Guide,* SC18-9840, documents procedures for moving DB2 data sets. You can change multiple names by repeating the SWITCH keyword, but you cannot specify the same name more than once. The names must be 8 bytes or less in EBCDIC representation. This clause has no effects on the system indexes and table spaces in DSNDB06/DSNDB01, since the catalog name is maintained in DSNZPARM.

'vcat_name' identifies the integrated catalog facility catalog currently used by user-managed data sets for indexes, table spaces, and storage groups. This is the catalog to be switched from.

'new_vcat_name' specifies the new integrated catalog facility catalog to be used by storage groups or user-managed data sets for indexes and table spaces.

Each name can be enclosed by single quotes so that any non-alphanumeric characters can be specified.

Here is a summary of the restrictions:

- New VCAT and schema names cannot be longer than 8 bytes in EBCDIC representation.
- SWITCH(A,B), SWITCH(A,C) is not allowed, because the original name is specified more than once.
- SWITCH(A,B), SWITCH(B,C) is not allowed, because the same name is specified as the original and target(new).
- OWNER FROM (A,B,C,A) is not allowed, because the owner name is specified more than once.
- SWITCH(SYSIBM,A) and OWNER FROM(SYSIBM) are not allowed.

- OWNER clause and SCHEMA clause cannot be specified in the same CATMAINT statement.
- If OWNER clause is specified, the CATMAINT statement has to be run by a user with the "TO" ROLE, and this ROLE must be defined with the ROLE AS OBJECT OWNER clause.
- The DB2 system has to be started with ACCESS(MAINT). Other members have to be stopped.
- If the source schema name appears in the PATHSCHEMA catalog column, the target schema name replaces the source, but the resulting string cannot exceed 2048 bytes.
- The source schema name cannot be referenced in check condition in any check constraints.

4.9.1 Considerations

When the schema name of an object is changed, any plans or packages that are dependent on the object are invalidated. Automatic rebind occurs when the invalidated plan or package is executed. Rebind may not be successful if the object is referenced in the application explicitly with the original schema name. In this case, the application should be modified.

No access to catalog is allowed while CATMAINT UPDATE SCHEMA / OWNER / VCAT is running. Any access to the catalog receives a resource unavailable message. You need to start the DB2 subsystem in ACCESS(MAINT) before you run this utility. If you have a data sharing group, you should stop all the other members in the group. If the utility is terminated prematurely during the update process, the catalog remains unavailable until the utility is resubmitted and completed successfully. If the utility is terminated prematurely because of existing errors in the catalog or directory, you may need to correct them using the REPAIR utility before resubmitting the job.

You should create backups of the catalog and directory before running this utility.

4.10 Toleration of DBET errors during DB2 restart

When you restart DB2, one important part of restart phase two, Current Status Rebuild, is to rebuild the database exception table (DBET). The DBET is rebuilt from associated log records starting at the last system checkpoint up to the last available log information. In data sharing, the DBET states are rebuilt from the DBET information stored in the SCA for member restart. For group restart when the SCA information is not available, DB2 rebuilds the DBET states from the logs of each of the DB2 members. Once restart phase two is finished, DB2 starts with the forward and backward log recovery. It is possible that DB2 must add more information to the DBET during those two phases. This is, for example, the case when DB2 tries to access some page sets to roll back work of in-flight or in-abort URs, but the page sets are not available for whatever reason. In this case, DB2 does not stop its restart process, but adds those pages to LPL. In Data Sharing DB2 puts an object needed for log apply into GRECP if it was dependent on a group buffer pool that was damaged or deallocated.

As you can see, the DBET states allow the DB2 restart to tolerate various errors. If, for some reason, DB2 cannot correctly set the DBET states due to a DBET error, DB2 cannot complete the restart process. The loss of a DBET state is a data integrity exposure and cannot be ignored.

Starting with DB2 V9, the way DB2 handles DBET errors during restart of the DB2 subsystem has changed in order to reduce problems due to DBET abends. DB2 now tolerates the DBET

error and puts the page set into a restrictive exception state instead. This exception state requires the object to be recovered.

This behavior is only available during log apply of DB2 restart.

Pending states

Depending on the type of objects that would need an entry in the DBET, DB2 now sets either of the three pending states:

- RECP - for table spaces
- RBDP - physical or logical index partitions
- PSRBD - non-partitioned secondary indexes

In addition to that, DB2 puts a new advisory pending state DBETE on the object for which the DBET error has been tolerated. As for all advisory states, this state by itself does not restrict you from doing anything with the objects. The restriction that lies on those objects is from the recover and rebuild pending states mentioned above. Once you have successfully completed the recover or rebuild utility for those objects, the DBETE state is also removed from that page set. Although we do *not* recommend this, you can also reset the restrictive states and the advisory state DBETE using the command -START DATABASE ACCESS(FORCE).

Attention: The reason for setting DBETE and any of the three restrictive states is that data integrity cannot be guaranteed for those objects. If you use -START DATABASE ACCESS(FORCE) you must be aware that data integrity is no longer guaranteed for these objects.

4.10.1 Steps taken when DBET errors occur

When a DBET error occurs, DB2 issues message DSNI046I to the console to indicate that an object has been involved in that error. The message contains the LRSN or RBA of the log record that was being applied when the error occurred. In addition to that, DB2 takes a dump for the DBET abend and puts the corresponding page set in a recovery exception state and the DBETE state. Once a DBET error is tolerated during restart, DB2 will not attempt a further log apply for the data in the table space or index space.

If, for whatever reason, DB2 cannot place the associated page set in recovery or rebuild pending state, DB2 reacts as it used to do up to DB2 V8 (that is, it abends).

4.11 Miscellaneous availability and serviceability enhancements

The following enhancements of DB2 9 address improved availability by eliminating the source of outages or alternatively minimizing the scope of the outages.

4.11.1 Add keyword RECOVERBEFORE to RECOVER utility

Let us assume that you have accidently lost one of more of the least recent image copies for one table space or for a given table space set. Even though the image copy data sets physically no longer exist, SYSIBM.SYSCOPY nevertheless contains corresponding rows. To speed up the recovery of those page sets, it would be helpful to be able to tell DB2 to not try to allocate the lost image copies, but to work with older, still existing image copy data sets right from the beginning.

With DB2 9, the Recover utility has been enhanced in this respect by providing a new RESTOREBEFORE (RBA or LRSN) parameter option.

RESTOREBEFORE allows you to specify an RBA or LRSN value, which tells DB2 to not use any Image Copy SYSCOPY entry that has a higher START_RBA than the specified value. The recover utility will then use an older full image copy as a recovery base for the object and merge any subsequent incremental image copies prior to the specified RBA value and then apply the log records to restore the object to the current or any specific point in time.

Refer to Figure 4-32 for some usage scenarios of the RECOVERBEFORE keyword.

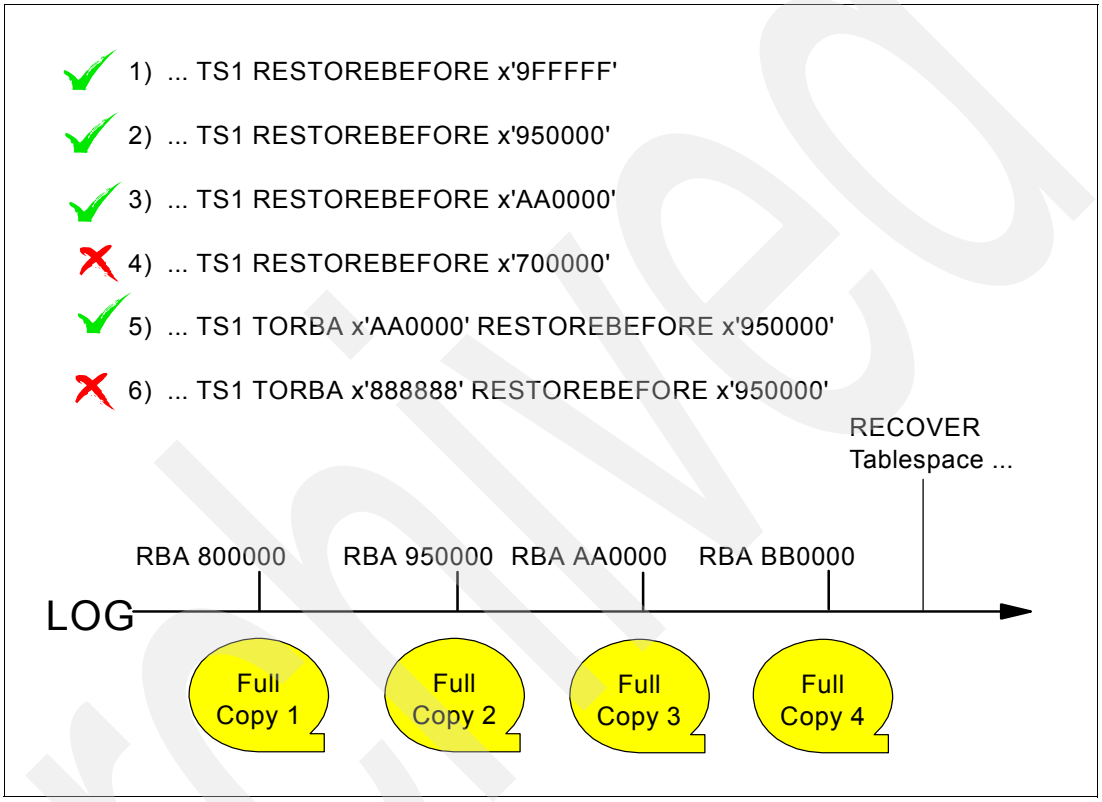


Figure 4-32 RESTOREBEFORE usage scenarios

This enhancement is available for you starting with compatibility mode.

4.11.2 Timeout or deadlock condition documented in utility output

When a deadlock or timeout condition occurs, starting with V9, DB2 now issues additional information that can be gathered from IFCID 172 trace records. Figure 4-33 show a sample for that additional information that used to be shown on the SYSLOG or MSTR address space only in former releases.

Archived

```
DSNU000I    046 21:04:46.07 DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = TEMP
DSNU1044I   046 21:04:46.18 DSNUGTIS - PROCESSING SYSIN AS EBCDIC
DSNU050I    046 21:04:46.30 DSNUGUTC -  COPY TABLESPACE DSN00170.TESTIMPL
DSNT500I    046 21:10:46.31 DSNUGBAC - RESOURCE UNAVAILABLE
                    REASON 00C200EA
                    TYPE 00000200
                    NAME DSN00170.TESTIMPL
```

Figure 4-33 Timeout condition information in utility output

4.11.3 Julian date and time stamp in utility output

In order to make it easier to analyze utility runs on your DB2 subsystems, messages DSNUxxxx that are sent to SYSPRINT dataset job output now include the Julian date and time stamp following the message number, displaying the local time in the format of:

```
DDD HH:MM:SS.TT
```

```
DSNU000I    046 21:04:46.07 DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = TEMP
DSNU1044I   046 21:04:46.18 DSNUGTIS - PROCESSING SYSIN AS EBCDIC
DSNU050I    046 21:04:46.30 DSNUGUTC -  COPY TABLESPACE DSN00170.TESTIMPL
DSNT500I    046 21:10:46.31 DSNUGBAC - RESOURCE UNAVAILABLE
                    REASON 00C200EA
                    TYPE 00000200
                    NAME DSN00170.TESTIMPL
```

Figure 4-34 Julian date and time stamp in utility output

As you can now easily see from the copy utility output shown in Figure 4-34, the timeout condition occurred exactly six minutes after the copy utility had been started.

In a data sharing environment, the data sharing member name follows the time stamp. Also, utility messages that are issued by utility subtasks may not necessarily come out in order. The added date and time stamp to utility messages can be very useful in diagnosing situations involving DB2 utilities.

This enhancement is already available in compatibility mode.

Archive

4.11.4 DSN1PRNT output for broken pages formatted

When you used DSN1PRNT with the FORMAT option prior to DB2 9 and the utility found a suspicious page, which appeared to be in error, the utility did not give you the formatted output for this page even though it might have been possible, because just one byte somewhere is this page was not correct. This behavior is now different. If DSN1PRNT detects a broken page, it nevertheless tries to format the page. If it works, you will get the formatted output followed by the unformatted page, as shown in Figure 4-35.

```
PAGE: # 00000002 ----------------------------------------------------------------
DATA PAGE:  PGCOMB='EF'X  PGLOGRBA='0000253ACD03'X  PGNUM='00000002'X  PGFLAGS='
            PGFREE='002B'X  PGFREEP=3953  PGFREEP='0F71'X  PGHOLE1='0000'X  PGMA
            PGNANCH=49
PGTAIL:  PGIDFREE='00'X  PGEND='N'
ID-MAP FOLLOWS:
01  0014 0065 00B8 0105 0152 01A1 01EF 0242
09  0295 02E9 033A 038D 03E0 0430 0483 04D9
11  052C 057B 05C9 0619 0669 06BB 070E 075D
19  07AD 07FB 084D 089B 08EA 093B 098A 09D5
21  0A24 0A75 0AC4 0B0F 0B62 0BAE 0C02 0C54
29  0CA8 0CF6 0D43 0D94 0DE7 0E34 0E81 0ED0
31  0F1E

RECORD:  XOFFSET='0014'X  PGSFLAGS='02'X  PGSLTH=81  PGSLTH='0051'X  PGSOBD='000
F0F0F0F0 F1F0C900 C1F0F000 F3F9F7F8 00196501 0100D7D9 C5E24040 40400080   000010
1200C600 19330814 00F00527 500000F0 00100000 00F00042 2000003E 0047C3C8   ..F...
D9C9E2E3 C9D5C5C8 C1C1E2                                                  RISTIN
...
...
...
RECORD:  XOFFSET='0F1E'X  PGSFLAGS='02'X  PGSLTH=83  PGSLTH='0053'X  PGSOBD='000
F0F0F0F0 F9F0E600 C5F1F100 F5F4F9F8 00197008 1500D4C1 D5C1C7C5 D9400080   000090
1000C600 19410515 00F00297 500000F0 00060000 00F00023 8000003E 0044C5C9   ..F...
D3C5C5D5 C8C5D5C4 C5D9E2D6 D5                                             LEENHE

***  PAGE IN ERROR - FORMATTED CONTENTS MAY BE INCONSISTENT OR TRUNCATED
0000 EF000025 3ACD0300 00000200 002B0F71   00003131 02005100 0001F0F0 F0F0F1F0
0020 C900C1F0 F000F3F9 F7F80019 65010100   D7D9C5E2 40404040 00801200 C6001933
0040 081400F0 05275000 00F00010 000000F0   00422000 003E0047 C3C8D9C9 E2E3C9D5
0060 C5C8C1C1 E2020053 000002F0 F0F0F0F2   F0D300C2 F0F100F3 F4F7F600 19731010
0080 00D4C1D5 C1C7C5D9 40008012 00D40019   48020200 F0041250 0000F000 08000000
00A0 F0003300 00003E00 45D4C9C3 C8C1C5D3   E3C8D6D4 D7E2D6D5 02004D00 0003F0F0
```

Figure 4-35 Broken page output in DSN1PRNT

The utility ends with RC 4 to indicate to you that there might be something wrong.

4.11.5 More messages during LOGAPPLY phase of recover utility

In order to provide recovery progress status for the RECOVER utility during the LOGAPPLY phase, the -DISPLAY UTILITY now displays the START RBA / LRSN, END RBA / LRSN, and the last COMMIT point LRSN / RBA for the RECOVER utility in the LOGAPPLY phase, and also the elapsed time based on the starting time of the LOGAPPLY phase.

That additional information helps you to estimate for the amount of the log processing that has been completed for the recovery job. A sample record providing you with the mentioned information is shown in Figure 4-36.

```
DSNU116I DSNUCALA - RECOVER LOGAPPLY PHASE DETAILS: STARTING TIME =
14:02:01 START RBA = xxxx START LRSN = yyyy END RBA = wwww END LRSN = zzzz
LAST COMMITTED RBA = rrrr LRSN = cccc ELAPSED TIME = 00:01:47
```

Figure 4-36 -DIS UTILITY sample output for LOGAPPLY phase of Recover

4.11.6 Avoid copypending restrictive state when broken page detected

Starting with DB2 9, when you run the COPY utility with the CHECKPAGE option, DB2 no longer sets the table space or index space into COPY pending state when a broken page is detected. Instead, the copy utility performs validity checking for each page, one page at a time, and if it finds an error, COPY issues message DSNU518I, which identifies the broken page and the type of error. A sample error message is shown in Figure 4-37.

```
DSNU518I  -DB9B 047 14:08:10.84 DSNUBASA -  TABLESPACE DSN00170.TESTIMPL DSNUM
1 CONTAINS BROKEN PAGE  X'00000002',
ERROR CODE X'0C30'
```

Figure 4-37 Sample error message, issued by COPY when broken page is detected

If more than one error exists in a page, only the first error will be identified. In addition, when COPY has detected a broken page, TTYPE, which is a new column in SYSIBM.SYSCOPY that will be set to 'B' (that is,)broken page. Following this, for the table space shown in Figure 4-37, the SYSCOPY entry contains the values shown in Figure 4-38.

```
DBNAME    TSNAME          DSNUM  ICTYPE  ICDATE  START_RBA    FILESEQNO  DEVTYPE
---------+---------+---------+---------+---------+---------+---------+---------+
DSN00170  TESTIMPL            0  C       070215  ....îò               0
DSN00170  TESTIMPL            0  F       070216  ...q6&               0  3390
DSN00170  TESTIMPL            0  T       070216  ...º.ñ               0

....
....
JOBNAME   AUTHID    OLDEST_VERSION  LOGICAL_PART  LOGGED  TTYPE     INSTANCE
---+---------+---------+---------+---------+---------+---------+---------+--
                                0             0  Y                        1
PAOLOR7A  PAOLOR7               0             0  Y                        1
PAOLOR7A  PAOLOR7               0             0  Y       B                1
```

Figure 4-38 SYSIBM.SYSCOPY - new TTYPE column

COPY will continue checking the remaining pages in the table space or index space after it finds an error. If an error is detected, the object table space or index space will *not* be put in COPY pending state, as it used to be until (and including) DB2 V8. Instead, a SYSCOPY record with an ICTYPE of 'T' will be inserted into SYSIBM.SYSCOPY. This prevents subsequent incremental image copies from being taken. CHECKPAGE is now the default option for the COPY utility and cannot be turned off.

4.11.7 New ability to cancel DB commands

DB2 9 provides you with the ability to cancel database commands that have been issued from the console.

We have issued a -STOP DATABASE (DSN0170) SPACE(TESTIMP*) command while some other transactions were still operating on objects in this table space. As a result, the page set status is changed from RW to STOPP, as shown in Figure 4-39.

```
DSNT360I  -DB9B **********************************
DSNT361I  -DB9B *  DISPLAY DATABASE SUMMARY
                *    GLOBAL
DSNT360I  -DB9B **********************************
DSNT362I  -DB9B     DATABASE = DSN00170  STATUS = RW
                    DBD LENGTH = 4028
DSNT397I  -DB9B
NAME     TYPE PART  STATUS            PHYERRLO PHYERRHI CATALOG  PIECE
-------- ---- ----- ---------------- -------- -------- -------- -----
TESTIMPL TS    0001 STOPP
******* DISPLAY OF DATABASE DSN00170 ENDED      **********************
DSN9022I  -DB9B DSNTDDIS 'DISPLAY DATABASE' NORMAL COMPLETION
***
```

Figure 4-39 STOPP after -STOP DB command

A new keyword extension exists for the -DISPLAY THREAD(*) command. The extension is the TYPE(SYSTEM) keyword. The result of this command is shown in Figure 4-40. As you can see, the STOP command can easily be identified. The system agent command has token 93 associated to it, which makes it now possible for you to cancel this system thread using the usual -CANCEL THREAD(93) DB2 command.

```
NAME     ST A   REQ ID          AUTHID   PLAN     ASID TOKEN
DB9B     N  *     0 020.STOPDB09 SYSOPR           0087    93
V490-SUSPENDED 07047-12:21:55.19 DSNTLSUS +00000646 UK20421
V494-IRLM 02 0C00000F01B5000300000000
V501-COMMAND EXECUTING:             -STOP DATABASE(DSN00170)SPACE(TES
                      TIMP*)
```

Figure 4-40 -DIS THREAD(*) TYPE(SYSTEM) command output

Although the thread command completes correctly and the system thread to stop the database is removed, you will still see STOPP state on your database object.

Important: The advantage over DB2 V8 is that now you can use the -START DB command to change status STOPP to RW.

When you tried to interrupt or reverse a STOPP status prior to V9, you ended up in a resource unavailable condition. Once the system tried to finish the requested STOP, communicating this to you using the STOPP restrictive state, you did not have a chance to get rid of it without allowing DB2 to stop the requested database object.

This enhancement is available in new-function mode only.

Archived

5

Data sharing

This chapter describes the improvements to data sharing. The main focus is on availability, although there are also performance improvements.

The high-availability improvements are achieved by a combination of performance, usability, and availability enhancements.

The following topics are the significant availability improvements and all relate to restarting a failed DB2 member:

- Data sharing logging improvement
- Index improvements
- Reduction in LOB locks
- Locking constraint relief
- Improved group buffer pool write performance
- Improved WLM routing based on DB2 health
- Improved workload balancing within the same LPAR
- Group buffer pool dependency removal by command
- Open dataset ahead of use via command
- Enhanced messages when unable to get p-locks
- Initiating Auto GRECP recovery at the end of restart
- Deferring the updates of SYSLGRNX till after end of restart
- Opening data sets earlier in restart processing
- Allowing table level retained locks to support postponed abort URs
- Simplification of the special open processing

Recovery to a consistent point in time, described in 4.3, “Recovery to point in time with consistency” on page 100, is even more important in a data sharing environment.

© Copyright IBM Corp. 2007. All rights reserved.

5.1 Data sharing logging improvement

Prior to DB2 V9, successive log records produced on the same DB2 member always had different LRSNs. In order to achieve this DB2 would re-drive the pulling of the *store clock* (STCK) to produce the LRSN if it was the same as the previous one. With processor speeds continually increasing, especially with the z9 technology, this became increasingly likely. Each time the STCK was re-driven in this way, CPU cycles were effectively lost. Note that while DB2 is doing the re-driving of the pulling of the STCK, the log latch is maintained. Holding the log latch in this way aggravates the log latch contention.

With this enhancement it is now only necessary for a given DB2 member to produce unique LRSNs when the log records are for the same data page. In V9, any LRSN update spins are done without holding the log latch, so other work is not blocked. This saves both CPU cycles and reduces log latch contention.

5.2 Index improvements

The following improvements to indexes are particularly beneficial in a data-sharing environment. They are covered elsewhere within this book.

- Index compression and greater than 4 KB pages for indexes.

  See 3.8, “Index compression” on page 66.

- Sequential key insert performance improvement.

  See 13.2, “Enhanced sequential key insert” on page 495.

- Ability to randomize key giving less contention.

  See 13.14, “Performance scalability” on page 532.

- Detection of unused indexes.

  Date when the index is used for SELECT, FETCH, searched UPDATE, searched DELETE, or enforce referential integrity constraints with dynamic or static SQL is now maintained in RTS column SYSINDEXSPACESTATS.LASTUSED.

  As a reminder, if an index is used by a static SQL statement, it shows up as BTYPE='I' in SYSPACKDEP (package or SYSPLANDEP for DBRM bound directly into a plan).

- A DPSI could be used to support affinity routing and eliminating data sharing overhead on that partition. The potential to do this is now increased, as a key can now be unique within the partition.

5.3 Reduction in LOB locks

Changes to LOB locking detailed in 13.12, “LOBs performance improvements” on page 528, mean that LOB data must be externalised for GBP-dependent LOB table spaces before locks are released. Also, the index on the LOB table needs to be externalized if GBP-dependant. LOBs do not have to be externalized to disk for non-GBP dependent LOBs. For GBP-dependant LOBs the externalization can be to either DASD or the GBP. The latter is usually the faster. Users should therefore consider using the GBPCACHE CHANGED option for LOB table spaces containing smaller LOBS.

Note that this function is brought in with locking protocol 3. To be enabled the data sharing group must be quiesed and restarted, once in New Function Mode.

5.4 Locking constraint relief

With z/OS 1.7 the XES lock limit on the number of locks IRLM can use (about 3.5 million) in the CF structure has been increased. Prior to this enhancement, XES places all local resource information for a lock structure connector in a single 2 G data space (and similarly, places all global resource information for a connector in a different single 2 G data space). This limits the number of XES locks that can be obtained and held by a lock structure connector (local data space limit). APAR OA03194 provides support for the allocation and use of 16 additional 2 G data spaces for use as the local data space. The APAR applies to V8 as well as V9.

5.5 Improved group buffer pool write performance

Prior to V9, batched GBP writes involved copying the pages to be written to the GBP to contiguous storage, the address of which was passed to XES. XES is now passed a list of addresses corresponding to the buffer pool pages. The copying of pages is now avoided, improving performance.

For group buffer pools that are duplexed, DB2 V9 eliminates cross invalidations as a result of the secondary being updated.

5.6 Improved WLM routing based on DB2 health

Each DB2 has a new health monitor, to detect whether DB2 is becoming storage constrained. Each DB2 periodically informs WLM as to health of DB2. When DB2 is storage constrained, the health will be less than 100%. WLM will attempt to route work away from subsystems that are less than 100%, hopefully giving them a chance to recover.

5.7 Improved workload balancing within the same LPAR

Due to virtual storage constraints, some installations have grown their data sharing groups. Instead of adding new LPARs for each additional DB2, they have decided to run a second DB2 on existing LPARs used by the group. When the group attachment is used from foreground TSO or batch, the connection is directed to the first available subsystem that is defined in IEFSSNxx member in SYS1.PARMLIB. This means that workload balancing is particularly difficult for batch.

In V9 the behavior is changed so that connections are made on a round-robin basis. This applies to DSN, CAF, RRS, and utilities. This functionality can optionally be achieved in both V7 and V8 via a USERMOD.

Archived

5.8 Group buffer pool dependency removal by command

There may be periods when work against a particular table space is being performed only from one member in the group (for example, during batch processing). Under these circumstances it is desirable not to not to incur the data sharing overhead. DB2 V9 has introduced a new command to allow the removal of group buffer pool dependency for specified objects. The new command is:

```
-ACCESS DB(dbname) SPACE(spacename) PART(n) MODE(NGBPDEP)
```

The command should be run on the member on which work is to continue or be scheduled.

DB2 will perform the following:

1. Drain all readers and writers on all members other than that on which the command is entered.
2. If '1' above is successful, then the writers on the system on which the command was entered will be drained, assuming that the object is not already *pseudo* closed.

If the drains fail the following message is issued:

```
DSNI048I mod-name CLAIMERS EXIST FOR DATA BASE dbname, SPACE NAME tsname,
   PART partno. GROUP BUFFERPOOL DEPENDENCY CANNOT BE REMOVED.
```

The -DISPLAY DATABASE CLAIMERS command could be used to identify who is blocking the drains.

If the drains are successful then the member on which the command was issued will convert the object to non-group buffer pool dependant, including the castout of any changed pages. The object is eligible immediately to go group buffer pool dependant, should there be subsequent accesses from the other members.

In addition to table spaces, the command is valid for index spaces as well.

The STARTDB authority, either explicit or implicit, is required for the -ACCESS command.

5.9 Open dataset ahead of use via command

The first thread to access a given object will drive the physical open of the data set. This will give an elongated response time for that thread. For response-sensitive transactions, a new command is available to pre-open or prime the data set:

```
-ACCESS DB(dbname) SPACE(spacename) PART(n) MODE(OPEN)
```

The command, which requires STARTDB authority, can be used for both table spaces and index spaces. The command works in both data sharing and non-data sharing. The command is local in scope within data sharing.

5.10 Enhanced messages when unable to get p-locks

The ABEND04E with reason code 00C20255 was a generic "Unable to obtain a P-Lock" condition. This will be enhanced once in NFM to be more specifically "P-lock failures involving another DB2 member holding the lock which has not responded to the P-lock request".

These additional reason codes have been added to indicate other p-lock issues:

- 00C2026A - Unable to obtain a P-lock because another DB2 member is in the process of shutting down
- 00C2026B - Unable to obtain a P-lock because another DB2 member has had logging suspended due to the active logs being full or an outstanding SET LOG SUSPEND command
- 00C2026C - Unable to obtain a P-lock because another DB2 member holding the lock encountered an abend in its P-lock exit

5.11 Initiating Auto GRECP recovery at the end of restart

In DB2 V5, the automatic recovery of group buffer pool recovery pending objects was introduced. This catered for a failure of the GBP and for total connectivity by *all* members being lost, but did not help when there were failures of the DB2 members as well. This is effectively the case in many disaster recovery scenarios. The DB2s are restarting having not been taken down cleanly and also with the GBPs missing.

DB2 V9 will also initiate auto GRECP recovery on DB2 restart when a GBP failure has occurred. While this new functionality applies to both a group restart and a normal restart, it is the group restart scenario that makes this particularly beneficial to some disaster recovery situations. GBP duplexing within one location is not sufficient to avoid the DR implications (that is, no GBPs available at the recovery site).

Situation in DB2 V8

Unless GBP duplexing is in effect, if all DB2s lose connectivity to the GBP or the GBP structure fails, then DB2 performs damage assessment and marks affected objects as GRECP.

Recovery of GRECP marked objects is done from information in the logs. If AUTOREC YES has been set via the ALTER GROUPBUFFERPOOL command, then DB2 will perform an automatic recovery. If AUTOREC NO was set or DB2 has failed, then START DATABASE commands need to be issued. Many users have their own code to automate this and try to spread the command and, hence, work, around the group.

Situation in DB2 V9

For total loss of connectivity or GBP structure failures, DB2 performs automatic recovery when AUTOREC YES is set, as per DB2 V8. Additionally, when a GBP structure has been lost, DB2 will initiate automatic GRECP recovery once it has restarted. Each DB2 will only initiate recovery for GRECP for group buffer pool dependant objects that it had an update interest in at the time it came down (R/W state). AUTOREC is used to control this functionality, so must be set to YES to get automatic GRECP recovery on restart.

When DB2 initiates automatic GRECP recovery after restart, it will try and acquire a conditional drain for each GRECP object. If DB2 is successful in acquiring the drain, it will perform the recovery from the log. Note that all the drained objects are recovered by one start command. The reason for this is two fold. Firstly, this results in just one scanning of the log. Secondly, and more importantly, the objects do not need to be sequenced (that is, the catalog and directory objects do not need to be done first). Fast log apply is always used, as this is part of restart.

If the drain of an object is not successful, a DSNI005I message is issued and the object remains in GRECP. A possible reason for not successfully draining is that there are

outstanding retained locks because another DB2 is either stopped or in the early stages of restarting. If this is the case, once that DB2 completes its own restart, it too will initiate automatic GRECP recovery.

From an operational perspective, -START DB2 commands for all failed members should be issued as soon as possible. The amount of recovery work performed by an individual DB2 member will depend on GBP dependencies across the group. The last member could do a significant amount if the workload is fully balanced across the group.

The following are exceptions when automatic GRECP recovery will not be used:

- DB2 is restarted with the DEFER ALL option.
- DB2 is restarted in system level PITR mode.
- DB2 is started in tracker mode.
- DB2 is restarted in Restart Light mode.

5.12 Deferring the updates of SYSLGRNX till after end of restart

The closing off of SYSLGRNX records is important for the performance of recoveries. In DB2 V8 non-GBP-dependant objects that were opened but not involved in the restart log apply phases (forward log recovery and backward log recovery) have SYSLNGRNX entries closed off at the end of restart. These updates at the end of restart could contend with other member updates, resulting in extended restart times.

In DB2 V9 the updating of non GBP dependent objects SYSLGRNX entries is deferred beyond restart, therefore allowing restart to complete quicker. The updating of the SYSLGRNX entries are now triggered by the first system checkpoint following restart.

Note that this benefits DB2 restart in a non data sharing environment as well.

5.13 Opening data sets earlier in restart processing

During the forward log recovery phase of restart, any pagesets for which log records are found will need to be opened. The number of pagesets needing to be opened could number in the thousands. DB2 sets a limit of 40 tasks for the parallel open or close of VSAM data sets, so data set opens could still amount to a significant part of the elapsed time of restart.

During restart, fast log apply is always enabled. Fast log apply involves reading the log and sorting records. Finally, the updates are done by one or more apply tasks, with list prefetches being done to read the appropriate pages. Synchronous opens are done as required.

This enhancement identifies pagesets not yet opened during the reading of the logs and schedules an asynchronous open. As a result of scheduling these asynchronous opens, which are conditional, less time will be spent waiting on data set opens. In the case where the open has not occurred by the time the apply task needs access to the data set, then a normal open is performed and the asynchronous open will not happen, as it is conditional.

This function will also benefit non-data sharing environments.

This feature is always active unless

- DB2 is in tracker mode.
- DB2 is started in Restart light mode.
- DB2 is restarted in System Level PITR mode.
- DB2 is restarted with the DEFER ALL option.

5.14 Allowing table level retained locks to support postponed abort URs

DB2 during system checkpoints stores information about each modified object by an uncommitted UR. This information, which is stored in the UR summary checkpoint record, is kept at the table space and partition level, but not at the table level. The information in the checkpoint is used for two purposes:

- Determine how far back in the log it needs to go to completely back out the UR.
- Assist in building the retain lock for postponed abort objects.

The enhancement in V9 is to now maintain in the checkpoint information at table level so that each table in a segmented table space can be tracked independently. With each table now having its own lock, an application is not blocked from using other tables within a multi-table table space, due to one table being subject to a postponed abort.

Note that although each retained table lock will be purged as the backout of the last postponed abort is completely done for that table, the AREST state applies to the table space. The AREST state is not changed until all tables in the table space have had any postponed aborts resolved.

5.15 Simplification of the special open processing

During the forward and backward log phases of restart, data sets are opened if there are log records to apply. During the opening of these data sets the retained page set p-locks are reacquired and converted to active pageset p-locks. These will be held until data set closure.

Not every GBP-dependant object at the time of the DB2 failure will, however, have log records to apply. Without the presence of log records to apply, these objects will not be opened, and consequently the retained p-locks would remain. To avoid this situation DB2 processes these objects at the end of restart, in what is referred to as *special open* processing. The pageset p-locks will be purged if it is still in retained state. The special open process is trying to ensure that the pageset p-lock is reacquired for a GBP-dependent object by one updating member to ensure that changed data might still in GBP for this object (that is, need to castout before the object can be converted to non-GBP dependent). The special open will open the data set if it is the last updating member.

When DB2 reacquires the pageset p-locks, it first takes what is known as a conversion lock. This is to prevent a p-lock state from being immediately changed by another DB2. These p-lock states are stored by DB2. It is important that states are accurately recorded and the conversion locks perform the serialisation to ensure this. In effect, this is a lock around the taking of a lock. The reason this serialisation is needed is that the p-lock exit could be triggered the moment the p-lock is reacquired (for example, if the reacquired state is IX, but is immediately changed to SIX before the lock state is stored inside DB2). DB2 could end up with the wrong state being stored.

There is, however, lock contention around taking these conversion locks, which can prolong restart times. The enhancement in V9 is to remove the need for conversion locks during the special open.

To remove the need for conversion locks, DB2 needs an alternative serialisation technique.

When DB2 manipulates the p-locks, IRLM provides DB2 with a sequence number. DB2 now stores this sequence number along with the lock state. When DB2 is updating the p-lock

state, it can avoid doing the update if the sequence number already stored is greater than the one associated with the state it is trying to update. In this way it will not regress the state.

Conversion locks are still used outside of special open processing.

5.16 Coupling Facility Control Code (CFCC) Level 15

Coupling Facility Control Code Level 15 is being made available on System z9 EC and BC. The enhancement includes increasing the allowable tasks in the Coupling Facility (CF) from 48 to 112.

When migrating CF levels, lock, list, and cache structure sizes may need to be increased to support the new function. This adjustment can impact the system when it allocates structures or copies structures from one coupling facility to another at different CF levels. The Coupling Facility Structure Sizer tool is designed to size structures for you, and takes into account the amount of space needed for the current CFCC levels.

Archived

Part 2

Application functions

This part contains the chapters that deal with functions generally associated to the application functions of DB2 9 for z/OS.

People tend to think only of older interfaces like JCL, Assembler, COBOL, CLIST, and TSO when referring to the mainframe. Although these languages continue to play a critical role, the fastest growing way of accessing DB2 for z/OS data today is via modern application development environments such as Java. Application developers do not need to be aware that the database resides on the mainframe, because the application languages that are supported by DB2 are fully portable.

DB2 9 delivers dozens of SQL enhancements that are needed by applications to ease development and porting. DB2 9 delivers a new native XML data type and pureXML capabilities, which allow companies to integrate their XML data with relational data in an efficient and high performance manner to provide superior business agility with lower development costs. Other new data types (large integer, decimal float, and spatial) extend application capabilities and flexibility. The Developer Workbench is a new application development environment to improve programmer productivity. DB2 9 native SQL stored procedures improve performance and manageability.

Note that DB2 supports more new languages than just Java. Examples are Ruby, Perl, PHP, Python, and TOAD for DB2.

Moving to more common clients across DB2 for luw, IDS, and DB2 for z/OS is a significant change for customers, as it helps us deliver better consistency across the line, better quality, and improved productivity for us and for customers.

© Copyright IBM Corp. 2007. All rights reserved.

Because of the large amount of application-related enhancements, we have divided them across several chapters. XML and e-business have each a dedicated chapter. The SQL contents have been slit between two chapters — one more related to the SQL language, the other one to application-enabling infrastructure changes.

This part contains the following chapters:

- Chapter 6, “SQL” on page 131
- Chapter 7, “Application enablement” on page 167
- Chapter 8, “XML” on page 263
- Chapter 9, “e-business and connectivity” on page 335

Archived

6

SQL

DB2 9 for z/OS delivers functional changes in SQL that increase productivity of programmers through family compatibility and leadership. Version 8 of DB2 for z/OS took big steps toward improving consistency of SQL across the DB2 family by providing more functions that are common between DB2 for Linux, UNIX, and Windows. More SQL functions are moved from the unique set to the common set improving programming productivity. V9 also introduces several new SQL functions that are firsts in the DB2 family.

This chapter provides a description of the major SQL-related enhancements introduced by DB2 9 for z/OS regarding data types and DML. These enhancements will help improve the productivity of SQL application programmers and allow better portability of applications.

This chapter discusses the following:

- New data types: BIGINT, BINARY, VARBINARY, DECFLOAT
- INSTEAD OF triggers
- MERGE
- SELECT FROM MERGE/UPDATE/DELETE
- ORDER BY and FETCH FIRST in subselect
- TRUNCATE
- INTERSECT and EXCEPT
- Built-in functions

© Copyright IBM Corp. 2007. All rights reserved.

6.1 New data types: BIGINT, BINARY, VARBINARY, DECFLOAT

DB2 9 introduces new data types BIGINT, BINARY, VARBINARY, and DECFLOAT. We examine their characteristics in the following sections.

6.1.1 BIGINT

SQL BIGINT data type (8 bytes of storage for integer values) is introduced for compatibility with languages such as Java, C, and C++, as well as SQL standard. For optimal performance, we recommend mapping the Java data types used in the application to the DB2 column data types. The main reason is to provide for efficient predicate processing. This also minimizes data type conversion cost and facilitates portability.

BIGINT is an exact numeric data type capable of representing 63-bit integers. It extends a set of currently supported exact numeric data types (SMALLINT and INTEGER) and is compatible with all numeric data types.

The BIGINT scalar function returns a big integer representation of a number or string representation of a number. See Example 6-1.

Example 6-1 BIGINT examples

```
SELECT BIGINT(12345.6)
FROM SYSIBM.SYSDUMMY1;
---Returns 12345

SELECT BIGINT('00123456789012')
FROM SYSIBM.SYSDUMMY1;
---Returns 123456789012
```

The existing scalar functions CHAR, DIGITS, LENGTH, MOD, MULTIPLY_ALT, POWER™, and VARCHAR have been extended to support BIGINT data type.

6.1.2 BINARY

BINARY is a fixed-length binary string (1 to 255 bytes) and VARBINARY is a variable-length binary string (1 to 32704 bytes). BINARY and VARBINARY data types extend current support of binary strings (BLOB), and are compatible with BLOB data type.

They are not compatible with character string data types.

- Improvement for binary data over the character string FOR BIT DATA
- Can use CAST specification to change FOR BIT DATA character string into BINARY string
- There is a difference in padding characters:

  ```
  [VAR]CHAR Padded with spaces (X'40' for EBCDIC,
                                X'20' for ASCII and Unicode)
  BINARYPadded with hex zeros (X'00')
  VARBINARYNot padded, even during comparison
  ```

  Two binary strings are equal only if the lengths are identical. If two strings are equal up to the length of the shorter string length, the shorter string is considered less than the longer string, even when the remaining bytes in the longer string are hex zeros.

Examples:

```
X'4100' < X'410000'
X'4100' < X'42'
X'4100' = X'4100'
X'4100' > X'41'
X'4100' > X'400000'
```

The BINARY scalar function returns a BINARY (fixed-length binary string) representation of a string. The following examples assume EBCDIC encoding of the input literal strings:

```
SELECT BINARY(‘‘,1)
FROM SYSIBM.SYSDUMMY1
```

Returns a fixed length binary string with a length attribute 1 and a value BX'00'.

```
SELECT BINARY('KBH',5)
FROM SYSIBM.SYSDUMMY1
```

Returns a fixed length binary string with a length attribute 5 and a value BX'D2C2C80000'.

The VARBINARY scalar function returns a VARBINARY (varying-length binary string) representation of a string. The following examples assume EBCDIC encoding of the input literal strings:

```
SELECT VARBINARY(‘‘)
FROM SYSIBM.SYSDUMMY1
```

Returns a varying length binary string with a length attribute 1, actual length 0, and a value of empty string.

```
SELECT VARBINARY('KBH',5)
FROM SYSIBM.SYSDUMMY1
```

Returns a varying length binary string with a length attribute 5, actual length 3, and a value BX'D2C2C8'.

The existing scalar functions INSERT, LEFT, LTRIM, POSSTR (POSITION does not support), REPEAT, REPLACE, RIGHT, RTRIM, STRIP, and SUBSTR have been extended to support BINARY and VARBINARY data types.

Note: To ease the migration of existing applications, altering CHAR FOR BIT DATA or VARCHAR FOR BIT DATA column data types to BINARY or VARBINARY data types is allowed (even though they are not considered to be compatible). If there is an index defined on that column, the index is placed in RBDP. Altering BINARY or VARBINARY data types to CHAR FOR BIT DATA or VARCHAR FOR BIT DATA is not allowed.

6.1.3 DECFLOAT

DB2 9 adds native support of the Decimal Floating Point (DECFLOAT) data type. It enables DECFLOAT data to be stored or loaded into DB2 tables, and it allows manipulation of DECFLOAT data. The rules for manipulation of DECFLOAT are similar to those for Packed Decimal. This means that DECFLOAT processing deals with exact numbers, not numerical approximations of IEEE Floating Point.

A decimal floating-point value (DECFLOAT) is an IEEE 754r number with a decimal point. The position of the decimal point is stored in each decimal floating-point value. The maximum precision is 34 digits. The range of a decimal floating point number is either 16 or 34 digits of precision, and an exponent range of respectively 10-383 to 10+384 or 10-6143 to 10+6144.

In addition to the finite numbers, decimal floating point numbers are able to represent one of the following named special values:

- Infinity - a value that represents a number whose magnitude is infinitely large.
- Quiet NaN - a value that represents undefined results and does not cause an invalid number condition. NaN is not a number.
- Signaling NaN - a value that represents undefined results that will cause an invalid number condition if used in any numerical operation.

When a number has one of these special values, its coefficient and exponent are undefined. The sign of an infinity is significant. It is possible to have both positive and negative infinity. The sign of a NaN has no meaning for arithmetic operations. INF can be used in place of INFINITY.

DECFLOAT (or a distinct type based on DECFLOAT) cannot be used for primary key, unique key, a foreign key or parent key, an IDENTITY column, a column in the partitioning key (PARTITION BY RANGE), a column used for index on expression, and a column has FIELDPROC.

The scalar functions COMPARE_DECFLOAT, DECFLOAT, DECFLOAT_SORTKEY, NORMALIZE_DECFLOAT, QUANTIZE, and TOTALORDER have been introduced.

DECFLOAT is currently supported in Java, Assembler, and REXX™ languages.

This format will be generated by the hardware instruction on machines that have this instruction available. On machines where this instruction is not available, the conversion will be done in software by DB2. The current implementation takes advantage of z9 hardware millicode for arithmetic and DECFLOAT(34)<--> DECFLOAT(16) cast operations.

With the introduction of these data types, the numeric data types are categorized as follows:

- Exact numerics: binary integer and decimal

  Binary integer includes small integer, large integer, and big integer. Binary numbers are exact representations of integers. Decimal numbers are exact representations of real numbers. Binary and decimal numbers are considered exact numeric types.

- Decimal floating point

  Decimal floating point numbers include DECFLOAT(16) and DECFLOAT(34), which are capable of representing either 16 or 34 significant digits.

- Approximate numerics: floating-point

Floating-point includes single precision and double precision. Floating-point numbers are approximations of real numbers and are considered approximate numeric types.

See *DB2 Version 9.1 for z/OS SQL Reference,* SC18-9854, for details, special registers, and built-in functions related to DECFLOAT.

See 11.5.2, “Rounding of DECFLOAT by UNLOAD and LOAD” on page 399, for options when loading DECFLOAT data.

String representation

Decimal floating-point variables can be defined in Java. Values whose data types are small integer, large integer, big integer, floating-point, decimal, and decimal floating-point are stored in an internal form that is transparent to the user of SQL. However, string representations of numeric values can be used in some contexts. A valid string representation of a numeric value must conform to the rules for numeric constants. See “Decimal floating-point constants”

on page 135. The encoding scheme in use determines what type of strings can be used for string representation of numeric values. For ASCII and EBCDIC, a string representation of a numeric value must be a character string. For UNICODE, a string representation of a numeric value can be either a character string or a graphic string. Thus, the only time a graphic string can be used for a numeric value is when the encoding scheme is UNICODE.

When a decimal, decimal floating-point, or floating-point number is cast to a string (for example, using a CAST specification), the implicit decimal point is replaced by the default decimal separator character that is in effect when the statement is prepared.

When a string is cast to a decimal, decimal floating-point, or floating-point value (for example, using a CAST specification), the default decimal separator character in effect when the statement was prepared is used to interpret the string.

Decimal floating-point constants

A decimal floating-point constant specifies a decimal floating-point number as two numbers separated by an E. The first number can include a sign and a decimal point. The second number can include a sign but not a decimal point.

The value of the constant is the product of the first number and the power of 10 specified by the second number. It must be within the range of DECFLOAT(34). The number of characters in the constant must not exceed 42.

Excluding leading zeros, the number of digits in the first number must not exceed 34, and the number of digits in the second must not exceed 4.

A constant specified as two numbers separated by E is a decimal floating-point constant only if the value lies outside the range of a floating-point constant.

A constant specified as a number that does not contain an E, but has more than 31 digits, is also considered a decimal floating-point constant.

In addition to numeric constants, the following reserved keywords can be used to specify decimal floating-point special values: INF, INFINITY, NAN, and SNAN, where INF and INFINITY represent infinity, NAN represents quiet not-a-number, and SNAN represents signaling not-a-number. The keywords may be preceded by an optional sign (+ or -) and may be any combination of upper-case and lower-case letters. SNAN results in a warning or exception when used in a numerical operation. NAN does not. SNAN can be used in non-numerical operations without causing a warning or exception, for example, in the VALUES list of an INSERT or as a constant compared in a predicate.

When referenced in a predicate:

```
-NAN < -SNAN < -INFINITY < -0 < 0 < INFINITY < SNAN < NAN
```

Decimal to DECFLOAT

When a decimal number is assigned to a DECFLOAT column or variable, the number is converted to the precision (16 or 34) of the target. Leading zeros are eliminated. Depending on the precision and scale of the decimal number and the precision of the target, the value might be rounded to fit.

For static SQL statements, the ROUNDING option of the BIND subcommand or of the CREATE PROCEDURE or ALTER PROCEDURE statement for a version of a native SQL procedure determines the rounding mode.

For dynamic SQL statements, the special register CURRENT DECFLOAT ROUNDING MODE determines the rounding mode.

Integer to decimal

When an integer is assigned to a decimal column or variable, the number is converted first to a temporary decimal number and then, if necessary, to the precision and scale of the target. The precision and scale of the temporary decimal number is 5,0 for a small integer, 11,0 for a large integer, or 19,0 for a big integer.

Integer to DECFLOAT

When an integer is assigned to a DECFLOAT column or variable, the number is converted first to a temporary decimal number and then to DECFLOAT. The precision and scale of the temporary decimal number is 5,0 for a small integer, 11,0 for a large integer, or 19,0 for a big integer. The decimal number is then converted to DECFLOAT using the rules for Decimal to DECFLOAT.

Floating point to DECFLOAT

When a single or double precision floating-point number is assigned to a DECFLOAT column or variable, the number is first converted to a temporary string representation of the floating point number. The string representation of the number is then converted to DECFLOAT.

DECFLOAT to integer

When a DECFLOAT is assigned to a binary integer column or variable, the fractional part of the number is lost.

The necessary number of trailing zeros is added, or the necessary number of trailing digits is eliminated.

If one number is DECFLOAT and the other number is integer, decimal, single precision floating-point, or double precision floating-point, the comparison is made with a temporary copy of the other number, which has been converted to DECFLOAT. If one number is DECFLOAT(16) and the other number is DECFLOAT(34), the DECFLOAT(16) value is converted to DECFLOAT(34) before the comparison. Additionally, the DECFLOAT data type supports both positive and negative zero. Positive and negative zero have different binary representations, but the equal (=) predicate will return true for comparisons of positive and negative zero.

The DECFLOAT data type also supports the specification of negative and positive NaN (quiet and signaling), and negative and positive infinity.

From an SQL perspective, infinity = infinity, NaN = NaN, and sNaN = sNaN.

The following rules are the comparison rules for these special values:

- Infinity compares equal only to infinity of the same sign (positive or negative).
- NaN compares equal only to NaN of the same sign (positive or negative).
- sNaN compares equal only to sNaN of the same sign (positive or negative).
- The ordering among the different special values is as follows:

```
-NAN < -SNAN < -INFINITY < 0 < INFINITY < SNAN <NAN
```

6.2 INSTEAD OF triggers

With the drive to have more and more logic enforced and supported by the database rather than the applications, there is a need to look for a way to introduce new logic for both read and write access but without the need to have the applications be aware of where this logic is

enforced. You use views for read access control, but the only option for write access control, in the simplest environment, is to use triggers on the base table. This is because when a view is read only, it cannot be used on INSERT, UPDATE, or DELETE activity. Since there is no way to influence read access on the base table, you are forced to build awareness into the applications that read access goes against one object (for example, the view) and write access goes against another (for example, the base table). This greatly increases the amount of work to introduce new logic into the system and also leaves the integrity of the system dependent on the logic within the applications.

INSTEAD OF triggers provide a mechanism to unify the target for all read/write access by an application while permitting separate and distinct actions to be taken for the individual read and write actions.

INSTEAD OF triggers are triggers that are processed instead of the update, delete, or insert operation that activates the trigger. Unlike other forms of triggers that are defined only on tables, INSTEAD OF triggers can only be defined on views.

Views are not deletable, updatable, or insertable only if they are read-only. INSTEAD OF triggers, provide an extension to the updatability of views. Using INSTEAD OF triggers, the requested insert, update, or delete operation against the view gets replaced by the trigger logic, which performs the operation on behalf of the view. This happens transparently to the application, which believes all operations are performed against the view. See Example 6-2.

Example 6-2 Example of INSTEAD OF trigger

```
CREATE TABLE WEATHER (CITY VARCHAR(25), TEMPF DECIMAL(5,2));
CREATE VIEW CELSIUS_WEATHER (CITY, TEMPC) AS
  SELECT CITY, (TEMPF-32)/1.8 FROM WEATHER

CREATE TRIGGER CW_INSERT INSTEAD OF INSERT ON CELSIUS_WEATHER
  REFERENCING NEW AS NEWCW FOR EACH ROW MODE DB2SQL
  INSERT INTO WEATHER VALUES (NEWCW.CITY, 1.8*NEWCW.TEMPC+32)

CREATE TRIGGER CW_UPDATE INSTEAD OF UPDATE ON CELSIUS_WEATHER
  REFERENCING NEW AS NEWCW OLD AS OLDCW FOR EACH ROW MODE DB2SQL
  UPDATE WEATHER W SET W.CITY = NEWCW.CITY, W.TEMPF = 1.8*NEWCW.TEMPC+32
  WHERE W.CITY = OLDCW.CITY
```

Table WEATHER stores temperature values in Fahrenheit. View CELSIUS_WEATHER is meant for users who prefer to work in Celsius instead of Fahrenheit. The INSTEAD OF trigger is used on the CELSIUS_WEATHER view to convert Celsius values to Fahrenheit values and then insert the Fahrenheit value into the WEATHER table.

This example includes INSTEAD OF triggers for INSERT and UPDATE operations. Consider the impact of each of these statements.

```
INSERT INTO CELSIUS_WEATHER VALUES ('DALLAS', 40)
```

The SELECT statement associated with the view CELSIUS_WEATHER would normally cause the view to be read only (because of the calculations). The presence of the INSTEAD OF INSERT trigger makes the view insertable. The values provided in the INSERT statement are available to the trigger in the new transition variables (just like other insert triggers).

In this case the trigger body wants to insert the row into the base table WEATHER, so it must do the conversion before storing the value for TEMPC. If a query had been used instead of a VALUES clause on the INSERT statement, the trigger body would be processed for each row from the query.

```
UPDATE CELSIUS_WEATHER SET TEMPC=72 WHERE CITY='DALLAS'
```

Since the view is not updatable (because of the calculations), such a view would not normally allow an UPDATE. As a result, the INSTEAD OF UPDATE trigger is activated so the view is not used directly. The new transition variables represent the values that are being changed by the triggering UPDATE statement. The old transition variables represent the values in the row of the view before the updates would be applied. For the old transition variable, consider it as the value from the row of the view that qualifies based on the search condition in the triggering UPDATE statement. For example:

```
SELECT * FROM CELSIUS_WEATHER WHERE CITY = 'DALLAS'
```

A restriction where the column list cannot be specified for the INSTEAD OF UPDATE trigger results in the body of the trigger being written to actually update all of the values in the row. If the triggering UPDATE does not change a column, the new and old transition variables are the same for that column. For the specific example given, the WEATHER table is updated since there is a row in the table for city DALLAS.

6.2.1 Considerations

You should be aware of the following considerations when you use INSTEAD OF triggers:

- Only one INSTEAD OF trigger is allowed for each type of operation (INSERT, UPDATE, DELETE) per view.
- The WHEN clause cannot be specified for an INSTEAD OF TRIGGER.
- FOR EACH STATEMENT must not be specified for an INSTEAD OF TRIGGER.
- The UPDATE OF column list cannot be specified.
- Must not specify a view (that is, the subject view of the INSTEAD OF TRIGGER definition) where any of the following conditions are true:
  - The view is defined with the WITH CASCADED CHECK option (a symmetric view).
  - The view on which a symmetric view has been defined.
  - The view references data that is encoded with different encoding schemes or CCSID values.
  - The view has a LOB or XML column.
  - The view has columns that have FIELDPROCs.
  - All of the underlying tables of the view are catalog tables or created global temporary tables.
  - The view has other views that are dependent on it.
- The OLD and NEW *correlation-name* variables cannot be modified in an INSTEAD OF trigger.
- If the result table of a cursor is read-only, the cursor is read-only. The cursor that references a view with INSTEAD OF TRIGGER is read-only since positioned UPDATE and positioned DELETE statements are not allowed using those cursors.
- If a view is specified as the target of the SELECT FROM UPDATE/DELETE/INSERT statement or the MERGE statement, the view must not be defined with any INSTEAD OF TRIGGER.

6.2.2 Authorization considerations

In defining a trigger on a view, the privilege set that is defined below must include SYSADM authority or each of the following:

- The SELECT privilege on any table or view to which the search condition of triggered action refers
- The necessary privileges to invoke the triggered SQL statements in the triggered action
- The authorization to define a trigger on the view, which must include at least one of the following:
  - Ownership of the view on which the trigger is defined
  - SYSCTRL authority

When an INSTEAD OF trigger is dropped, the associated privilege is revoked from anyone that possesses the privilege as the result of an implicit grant that occurred when the trigger was created. Also, certain packages may be marked invalid. For example:

- An AFTER trigger TR1 body contains an UPDATE view V1 statement.
- View V1 has an INSTEAD OF UPDATE trigger TR2.
- DROP TRIGGER TR2 causes TR1's package to be invalidated because TR1 depends on TR2.

Dropping a view that has INSTEAD OF triggers causes the INSTEAD OF triggers and their associated packages to be dropped as well.

6.2.3 Catalog changes

The following are the changes to the catalog tables in support of INSTEAD OF triggers:

- 'I' for TRIGTIME column in SYSTRIGGERS
- Non-zero values for OBID and DBID columns in SYSTABLES for a view that has an INSTEAD OF trigger defined
- New value of 'E' for BTYPE column in SYSPLANDEP and SYSPACKDEP to reflect dependency on INSTEAD OF trigger

6.3 MERGE

The MERGE statement updates a target (a table or view, or the underlying tables or views of a fullselect) using the specified input data. Rows in the target that match the input data are updated as specified, and rows that do not exist in the target are inserted. Updating or inserting a row in a view results in an update or insert of the row in the tables on which the view is based, if no INSTEAD OF trigger is defined for the operation on this view. If an INSTEAD OF trigger is defined, DB2 issues SQLcode -7008 (SQLSTATE 55019).

This statement can be embedded in an application program or issued interactively. It is an executable statement that can be dynamically prepared.

You can update existing data and insert new data in a single operation by using the MERGE statement. This operation is useful when you want to update a table with a set of rows, some of which are changes to existing rows and some of which are new rows. This is a multi-row merge. A typical use will have many rows to merge. As with multi-row insert, we use arrays to provide the multiple rows of data to merge into the table.

Archived

For example, an application might request a set of rows from a database, allow a user to modify the data through a GUI, and then store the modified data in the database. Some of this modified data is updates to existing rows, and some of this data is new rows. You can do these update and insert operations in one step.

To update existing data and insert new data, specify a MERGE statement with the WHEN MATCHED and WHEN NOT MATCHED clauses. These clauses specify how DB2 handles matched and unmatched data. If DB2 finds a matching row, that row is updated. If DB2 does not find a matching row, a new row is inserted.

Figure 6-1 shows an example of the use of MERGE statement.

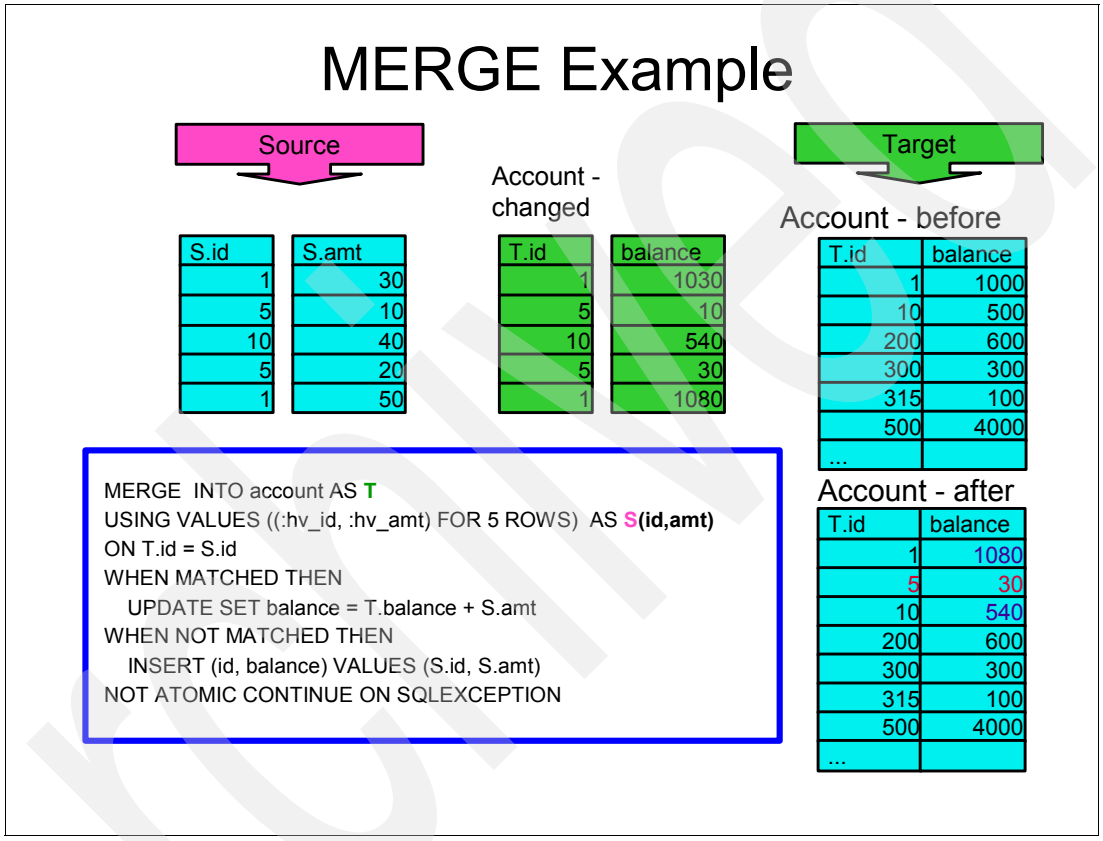


Figure 6-1 MERGE example

A row inserted into the target is immediately available for update. A row updated is immediately available for more updates in the same statement. Therefore, notice a new row for ID 5 is inserted and subsequently updated. The row for ID 1 is updated twice, and the row for ID 10 is updated once. Other rows remain unchanged.

NOT ATOMIC CONTINUE ON SQL EXCEPTION specifies that, regardless of the failure of any particular source row, the MERGE statement will not undo any changes that are made to the database by the successful merge of other rows from the host variable arrays. Merge will be attempted for rows that follow the failed row. However, the minimum level of atomicity is at least that of a single source row (that is, it is not possible for a partial merge to complete), including any triggers that might run as a result of the MERGE statement.

The GET DIAGNOSTICS statement can be used immediately after the MERGE statement to check which input rows fail during the merge operation. The GET DIAGNOSTICS statement information item, NUMBER, indicates the number of conditions that are raised. The GET

DIAGNOSTICS condition information item, DB2_ROW_NUMBER, indicates the input source rows that cause an error.

6.3.1 EXPLAIN for MERGE

The statement in Example 6-3 is executed under SPUFI.

Example 6-3 MERGE example

```
EXPLAIN PLAN SET QUERYNO = 1 FOR
MERGE INTO ACCOUNT AS T
USING (VALUES(?,?) FOR 5 ROWS) AS S(ID,AMT)
ON T.ID = S.ID
WHEN MATCHED THEN
 UPDATE SET BALANCE = T.BALANCE + S.AMT
WHEN NOT MATCHED THEN
 INSERT (ID,BALANCE) VALUES(S.ID,S.AMT)
   NOT ATOMIC CONTINUE ON SQLEXCEPTION;
```

The following SELECT statement is issued against the PLAN_TABLE:

```
SELECT QBLOCKNO,QBLOCK_TYPE,PLANNO,SUBSTR(TNAME,1,2) AS TNAME,
       TABLE_TYPE,JOIN_TYPE,METHOD,ACCESSTYPE
FROM PLAN_TABLE WHERE QUERYNO = 1
ORDER BY QUERYNO, QBLOCKNO, PLANNO;
```

Figure 6-2 shows the result of the sample EXPLAIN for MERGE statement.

Sample EXPLAIN for MERGE

QBLOCKNO	QBLOCK_TYPE	PLANNO	TNAME	TABLE_TYPE	JOIN_TYPE	METHOD	ACCESSTYPE
1	MERGE	1	S	B (1*)		0	V (2*)
1	MERGE	2	AC	T	L	1 (3*)	R (4*)
2	UPDATE	0	AC	T		0	
3	INSERT	0	AC	T		0	

1* : TABLE_TYPE "B" is already supported in V8.
2* : ACCESSTYPE "V" is already supported in V8.
3* : Since we are doing "update in place", only Nested Loop Join is considered.
4* : Since we are doing "update in place", if an index column is being updated, the index is not considered for the table access to avoid problems.

RID access ("I" with prefetch="L") is not considered.
Sparse index access ("T") is not considered.
There is no parallel support for MERGE.

Figure 6-2 Sample EXPLAIN for MERGE

Notice the following:

- New QBLOCK_TYPE for MERGE.

  QBLOCK_TYPE for UPDATE and INSERT are already supported in V8.
- QBLOCKNO for MERGE is 1, for UPDATE is 2 and for INSERT is 3.
- TABLE_TYPE “B” indicates that data is inside the “Buffers for SELECT from INSERT, SELECT from UPDATE, SELECT from MERGE, or SELECT from DELETE statement”.
- MERGE is equivalent to performing “source table” left outer join “target table”.
- ACCESSTYPE shows the method of accessing the table. It is "V" accesstype, indicating a Virtual access.
- "S" is a "source" table. It is really a list of hostvar arrays.

6.3.2 Authorization

The privileges that are held by the privilege set that is defined below must include at least one of the following privileges:

- SYSADM authority
- Ownership of the table
- If the search condition contains a reference to a column of the table or view, the SELECT privilege for the referenced table or view
- If the insert operation is specified, the INSERT privilege for the table or view
- If the update operation is specified, at least one of the following privileges is required:
  - The UPDATE privilege for the table or view
  - The UPDATE privilege on each column that is updated
  - If the right side of the assignment clause contains a reference to a column of the table or view, the SELECT privilege for the referenced table or view

If the insert operation or assignment clause includes a subquery, the privileges that are held by the privilege set must also include at least one of the following privileges:

- SYSADM authority
- The SELECT privilege on every table or view that is identified in the subquery
- Ownership of the tables or views that are identified in the subquery

Privilege set

If the statement is embedded in an application program, the privilege set is the privileges that are held by the owner of the plan or package. If the statement is dynamically prepared, the privilege set is determined by the DYNAMICRULES behavior in effect (run, bind, define, or invoke).

6.4 SELECT FROM MERGE/UPDATE/DELETE

DB2 V8 introduced SELECT FROM INSERT. DB2 9 provides support for SELECT FROM MERGE, SELECT FROM UPDATE, and SELECT FROM DELETE.

Archived

6.4.1 Selecting values as you merge: SELECT FROM MERGE

You can select values from rows that are being merged by specifying the MERGE statement in the FROM clause of the SELECT statement. When you merge one or more rows into a table, you can retrieve:

- The value of an automatically generated column such as a ROWID or identity column
- Any default values for columns
- All values for a merged row, without specifying individual column names
- Calculated values based on the changes to merged rows

You want to use the FINAL TABLE clause with SELECTFROM MERGE statements. The FINAL TABLE consists of the rows of the table or view after the merge occurs.

Figure 6-3 shows how to extend the MERGE statement shown in Figure 6-1 on page 140 to retrieve the values inserted or updated.

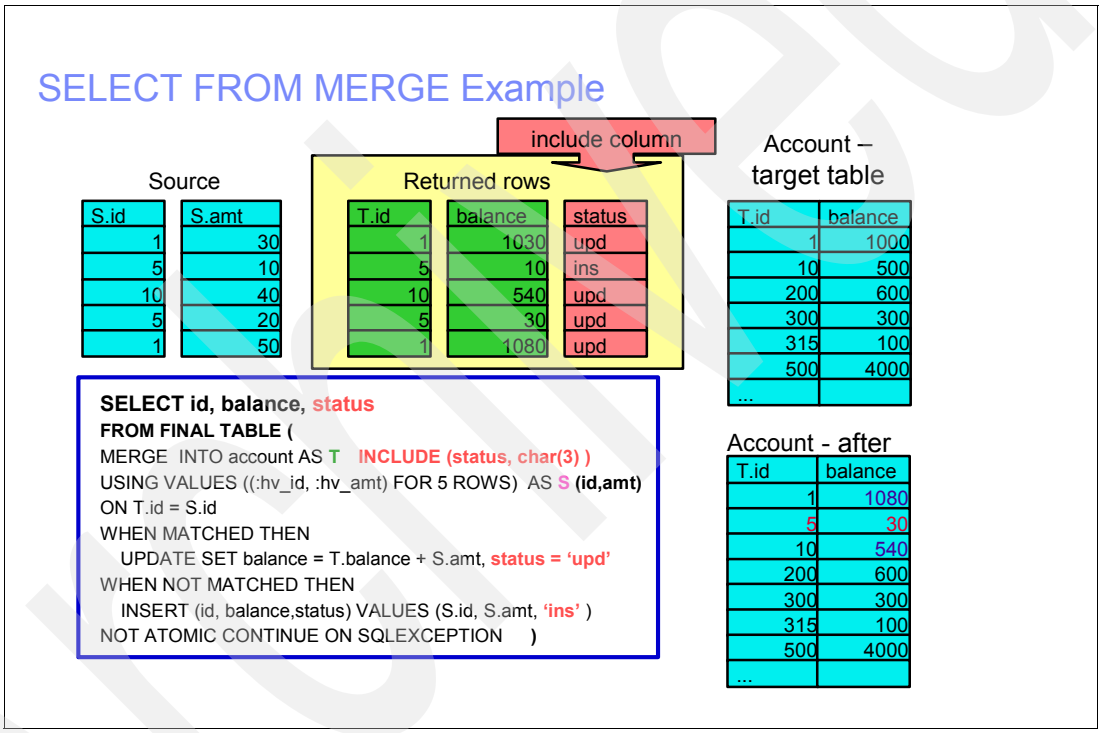


Figure 6-3 SELECT from MERGE example

The include-column clause has been introduced with the SQL data change statements (DELETE/INSERT/UPDATE/MERGE). The include-column allows you to specify one or a list of additional columns on the select list. The included columns are appended to the end of the list of columns that is identified by target table. The INCLUDE can be specified only if the DELETE/INSERT/UPDATE/MERGE statement is nested in the FROM clause of a SELECT statement.

We discuss another scenario where SELECT FROM MERGE is used.

Example

Suppose that you need to input data into the STOCK table, which contains company stock symbols and stock prices from your stock portfolio. Some of your input data refers to companies that are already in the STOCK table, and some of the data refers to companies that you are adding to your stock portfolio. If the stock symbol exists in the SYMBOL column,

you need to update the PRICE column. If the company stock symbol is not yet in the STOCK table, you need to insert a new row with the stock symbol and the stock price. Furthermore, you need to add a new value DELTA to your output to show the change in stock price. Suppose that the STOCK table contains the data that is shown in Table 6-1.

Table 6-1 STOCK table before SELECT FROM MERGE statement

SYMBOL	PRICE
XCOM	95.00
YCOM	24.50

Now, suppose that :hv_symbol and :hv_price are host variables that contain updated data that corresponds to the data that is shown in Table 6-1.

Table 6-2 shows the host variable data for stock activity.

Table 6-2 Host variable arrays of stock activity

hv_symbol	hv_price
XCOM	97.00
NEWC	30.00
XCOM	107.00

NEWC is new to the STOCK table, so its symbol and price need to be inserted into the STOCK table. The rows for XCOM reflect stock price changes, so these values need to be updated in the STOCK table. Also, the output needs to show the change in stock prices as a DELTA value.

The SELECT FROM MERGE statement shown in Example 6-4 updates the price of XCOM, inserts the symbol and price for NEWC, and returns an output that includes a DELTA value for the change in stock price.

Example 6-4 SELECT FROM MERGE

```
SELECT SYMBOL, PRICE, DELTA
FROM FINAL TABLE (
   MERGE INTO STOCK AS S INCLUDE (DELTA DECIMAL(5,2))
   USING (:hv_symbol, :hv_price) FOR :hv_nrows ROWS AS R (SYMBOL, PRICE)
   ON S.SYMBOL = R.SYMBOL
   WHEN MATCHED      THEN UPDATE SET DELTA = R.PRICE - S.PRICE, PRICE=R.PRICE
   WHEN NOT MATCHED THEN INSERT (SYMBOL, PRICE, DELTA)
      VALUES (R.SYMBOL, R.PRICE, R.PRICE)
      NOT ATOMIC CONTINUE ON SQLEXCEPTION);
```

The INCLUDE clause specifies that an additional column, DELTA, can be returned in the output without adding a column to the STOCK table. The UPDATE portion of the MERGE statement sets the DELTA value to the differential of the previous stock price with the value set for the update operation. The INSERT portion of the MERGE statement sets the DELTA value to the same value as the PRICE column.

Archived

The MERGE statement results in the following changes to the STOCK table:

- Processing the first row of data in the host variable arrays, a row for company XCOM already exists in the STOCK table, so an UPDATE is performed and the value for the PRICE column is set to 97.00.
- Processing the second row of data in the host variable arrays, a row for company NEWC does not yet exist in the STOCK table, so an INSERT is performed and the value for PRICE is set to 30.00.
- Processing the last row of data in the host variable arrays, a row for company XCOM exists in the STOCK table, so another UPDATE is performed on that row and the PRICE column is set to 107.00 (thereby overlaying the prior value of 97.00 that was set earlier during the MERGE statement).

After completion of the MERGE statement, the STOCK table contains the data shown in Table 6-3.

Table 6-3 STOCK table after SELECT FROM MERGE statement

SYMBOL	PRICE
XCOM	107.00
YCOM	24.50
NEWC	30.00

Notice that there is a single row for company XCOM and that the value of the PRICE column is 107.00. The value of 107.00 for that row is the value specified in the second set of input data in the host variable arrays for company XCOM.

The output of the SELECT FROM MERGE statement, as shown in Example 6-5, includes both updates to XCOM and a DELTA value for each output row.

Example 6-5 Output of SELECT FROM MERGE statement

```
SYMBOL      PRICE      DELTA
------------------------------
XCOM         97.00       2.00
NEWC         30.00      30.00
XCOM        107.00      10.00
```

6.4.2 Selecting values as you update: SELECT FROM UPDATE

You can select values from rows that are being updated by specifying the UPDATE statement in the FROM clause of the SELECT statement. When you update one or more rows in a table, you can retrieve:

- The value of an automatically generated column such as a ROWID or identity column
- Any default values for columns
- All values for an updated row, without specifying individual column names

Archived

In most cases, you want to use the FINAL TABLE clause with SELECT FROM UPDATE statements. The FINAL TABLE consists of the rows of the table or view after the update occurs. For example, suppose that all designers for a company are receiving 5% raises. You can use the following SELECT FROM UPDATE statement to increase the salary of each designer by 5% and to retrieve the total increase in salary for the company:

```
SELECT SUM(SALARY) INTO :salary FROM FINAL TABLE
   (UPDATE EMP SET SALARY = SALARY * 1.05
    WHERE JOB = 'DESIGNER');
```

To retrieve row-by-row output of updated data, use a cursor with a SELECT FROM UPDATE statement. For example, suppose that all designers for a company are receiving a 30% increase in their bonus. You can use the following SELECT FROM UPDATE statement to increase the bonus of each designer by 30% and to retrieve the bonus for each designer:

```
DECLARE CS1 CURSOR FOR
 SELECT LASTNAME, BONUS FROM FINAL TABLE
   (UPDATE EMP SET BONUS = BONUS * 1.3
    WHERE JOB = 'DESIGNER');
FETCH CS1 INTO :lastname, :bonus;
```

You can use the INCLUDE clause to introduce a new column to the result table but not add the column to the target table. For example, suppose that sales representatives received a 20% increase in their commission. You need to update the commission (COMM) of sales representatives (SALESREP) in the EMP table and you need to retrieve the old commission and the new commission for each sales representative. You can use the following SELECT FROM UPDATE statement to perform the update and to retrieve the required data:

```
DECLARE CS2 CURSOR FOR
SELECT LASTNAME, COMM, old_comm FROM FINAL TABLE
   (UPDATE EMP INCLUDE(old_comm DECIMAL (7,2))
    SET COMM = COMM * 1.2, old_comm = COMM
    WHERE JOB = 'SALESREP');
```

6.4.3 Selecting values as you delete: SELECT FROM DELETE

You can select values from rows that are being deleted by specifying the DELETE statement in the FROM clause of the SELECT statement. When you delete one or more rows in a table, you can retrieve:

- Any default values for columns
- All values for a deleted row, without specifying individual column names
- Calculated values based on deleted rows

When you use a SELECT FROM DELETE statement, you must use the FROM OLD TABLE clause to retrieve deleted values. The OLD TABLE consists of the rows of the table or view before the delete occurs. For example, suppose that a company is eliminating all operator positions and that the company wants to know how much salary money it will save by eliminating these positions. You can use the following SELECT FROM DELETE statement to delete operators from the EMP table and to retrieve the sum of operator salaries:

```
SELECT SUM(SALARY) INTO :salary FROM OLD TABLE
(DELETE FROM EMP
 WHERE JOB = 'OPERATOR');
```

To retrieve row-by-row output of deleted data, use a cursor with a SELECT FROM DELETE statement. For example, suppose that a company is eliminating all analyst positions and that the company wants to know how many years of experience each analyst had with the

company. You can use the following SELECT FROM DELETE statement to delete analysts from the EMP table and to retrieve the experience of each analyst:

```
DECLARE CS1 CURSOR FOR
SELECT YEAR(CURRRENT DATE - HIREDATE) FROM OLD TABLE
   (DELETE FROM EMP
    WHERE JOB = 'ANALYST');
FETCH CS1 INTO :years_of_service;
```

If you need to retrieve calculated data based on the data that you delete but not add that column to the target table, for example, suppose that you need to delete managers from the EMP table and that you need to retrieve the salary and the years of employment for each manager. You can use the following SELECT FROM DELETE statement to perform the delete operation and to retrieve the required data:

```
DECLARE CS2 CURSOR FOR
SELECT LASTNAME, SALARY, years_employed FROM OLD TABLE
   (DELETE FROM EMP INCLUDE(years_employed INTEGER)
    SET years_employed = YEAR(CURRENT DATE - HIREDATE)
    WHERE JOB = 'MANAGER');
```

6.5 ORDER BY and FETCH FIRST in subselect

In DB2 V8, the ORDER BY and FETCH FIRST n ROWS ONLY clauses are only allowed at the statement level as part of select-statement or a SELECT INTO statement. That is, you can specify the clauses as part of select-statement and write:

```
SELECT * FROM T ORDER BY c1 FETCH FIRST 1 ROW ONLY
```

But you cannot specify the clauses within the fullselect and write:

```
INSERT INTO T1
(SELECT * FROM T2 ORDER BY c1 FETCH FIRST 1 ROW ONLY)
```

Assume that you have a huge table of which you want only the first 2000 rows sorted in a particular order. You would code a SELECT statement using the FETCH FIRST and ORDER BY clauses. The bad news is the sort is done before the fetch. This would cause a huge sort for no reason. The work around is to code this using a temp table, which is a lot more work than a simple select.

DB2 V9 allows specification of these clauses as part of subselect or fullselect.

Example 1

Using the DSN8910.EMPPROJACT table, find the project numbers that have an employee whose salary is in the top three salaries for all employees. Order the result by project number:

```
SELECT T1.EMPNO, T1.PROJNO
FROM DSN8910.EMPPROJACT T1
WHERE T1.EMPNO IN
   (SELECT T2.EMPNO
   FROM DSN8910.EMP T2
   ORDER BY SALARY DESC
   FETCH FIRST 3 ROWS ONLY)
ORDER BY T1.PROJNO;

---------+---------+---------+---------+--
EMPNO   PROJNO
```

```
---------+---------+---------+---------+--
000010  AD3100
000010  MA2100
000110  MA2100
000010  MA2110
```

The above can be verified by executing the following statements:

```
SELECT T2.EMPNO,T2.SALARY
FROM DSN8910.EMP T2
ORDER BY SALARY DESC ;
---------+---------+---------+--
EMPNO        SALARY
---------+---------+---------+--
000010     52750.00
200010     46500.00
000110     46500.00
000020     41250.00
000050     40175.00
..............
..............
..............
000310     15900.00
200310     15900.00
000290     15340.00

SELECT T1.EMPNO,T1.PROJNO
FROM DSN8910.EMPPROJACT T1
ORDER BY EMPNO;
---------+---------+-------
EMPNO   PROJNO
---------+---------+-------
000010  MA2100
000010  MA2110
000010  AD3100
000020  PL2100
000030  IF1000
000030  IF2000
000050  OP1000
000050  OP2010
000070  AD3110
000090  OP1010
000100  OP2010
000110  MA2100
........................
........................
........................
000330  OP2012
000340  OP2013
000340  OP2013
```

Archived

Example 2

Assume that table T1 has columns C1 and C2 to hold integer data. Also assume that table T2 has columns C1 and C2 to hold integer data and is populated. It is now possible to code the following INSERT statement:

```
INSERT INTO T1
SELECT * FROM T2
ORDER BY C1;
```

6.5.1 Returning fullselect results in the order of the subquery result table

When you specify SELECT FROM INSERT, SELECT FROM UPDATE, SELECT FROM DELETE, or SELECT FROM MERGE, you can preserve the order of the derived table by specifying the ORDER OF clause with the ORDER BY clause. These two clauses ensure that the result rows of a fullselect follow the same order as the result table of a subquery within the fullselect.

You can use the ORDER OF clause in any query that uses an ORDER BY clause, but the ORDER OF clause is most useful with queries that contain a set operator, such as UNION.

Example 1

The following example retrieves the following rows:

- Rows of table T1 in no specified order
- Rows of table T2 in no specified order

The example query then performs a UNION ALL operation on the results of the two subqueries. "ORDER BY 1" is applied to the UNION ALL result. The ORDER BY ORDER OF UTABLE clause in the query specifies that the fullselect result rows are to be returned in the same order as the result rows of the UNION ALL statement.

```
SELECT * FROM
   (SELECT C1,C2 FROM T1
      UNION ALL
    SELECT C1,C2 FROM T2
    ORDER BY 1
   ) AS UTABLE
ORDER BY ORDER OF UTABLE;
If tables T1 and T2 have 2 rows with values 1 and 3 for C1, and values 2 and 4
for C2, this would result in
---------+---------+---------+
         C1          C2
---------+---------+---------+
          1           2
          1           2
          3           4
          3           4
```

Archived

Example 2

The following example joins data from table T1 to the result table of a nested table expression. The nested table expression is ordered by the second column in table T2. The ORDER BY ORDER OF TEMP clause in the query specifies that the fullselect result rows are to be returned in the same order as the nested table expression.

```
SELECT T1.C1, T1.C2, TEMP.Cy, TEMP.Cx
FROM T1, (SELECT T2.C1, T2.C2 FROM T2 ORDER BY 2) as TEMP(Cx, Cy)
WHERE Cx = T1.C1
ORDER BY ORDER OF TEMP;
```

Alternatively, you can produce the same result by explicitly stating the ORDER BY column TEMP.Cy in the fullselect instead of using the ORDER OF syntax:

```
SELECT T1.C1, T1.C2, TEMP.Cy, TEMP.Cx
FROM T1, (SELECT T2.C1, T2.C2 FROM T2 ORDER BY 2) as TEMP(Cx, Cy)
WHERE Cy = T1.C1
ORDER BY TEMP.Cy;
```

6.5.2 Other considerations

In some cases, the ORDER BY clause may be ignored in a subquery if there are no side effects caused by ignoring it. The following are examples of cases where the ORDER BY clause may *not* be ignored:

- If the subquery also contains a FETCH FIRST n ROWs
- If the subquery also contains a user-defined function with EXTERNAL ACTION or MODIFIES SQL DATA in the ORDER BY clause

Push down of FETCH FIRST n ROW to subselect and fullselect can improve performance. The database manager ceases processing the query once it has determined the number of rows specified in the FETCH first n ROW.

6.6 TRUNCATE

With V8, to empty a table you have to either do a mass delete (that is, use DELETE FROM *table-name* without the WHERE clause) or use the LOAD utility with REPLACE REUSE and LOG NO NOCOPYPEND.

If there is a delete trigger on the table, using the DELETE statement requires you to DROP and subsequently recreate the deleted trigger to empty the table without firing that trigger. LOAD REPLACE works on a table space level instead of on a table level. You cannot empty a specific table if the belonging table space contains multiple tables.

The TRUNCATE statement addresses these problems. The TRUNCATE statement deletes all rows for either base tables or declared global temporary tables. The base table can be in a simple table space, a segmented table space, a partitioned table space, or a universal table space. If the table contains LOB or XML columns, the corresponding table spaces and indexes are also truncated.

Archived

6.6.1 TRUNCATE statement

The syntax of the TRUNCATE statement is shown in Figure 6-4.

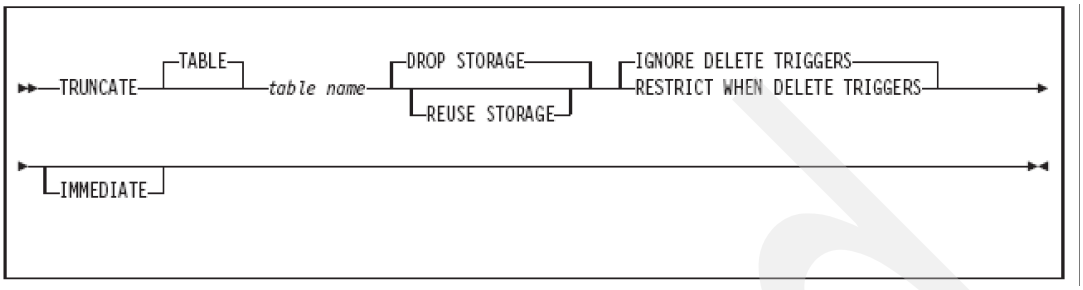


Figure 6-4 Syntax of TRUNCATE statement

The table-name must identify only a table. If the table-name is a base table of a table space, all tables that are defined under the table (for example, auxiliary LOB table spaces and XML table spaces), and all of their associated indexes are also truncated.

By default, all storage that is allocated for the table is released and made available for use for the same table or any other table that resides in the table space. The deallocated space is always available for reuse by all tables in the table space. However, if you specify REUSE STORAGE, all storage that is allocated for the table is emptied, but continues to be allocated for the table. REUSE STORAGE is ignored for a table in a simple table space and the statement is processed as though DROP STORAGE is specified.

By default, any delete triggers that are defined for the table are not activated by the truncate operation. However, if you specify RESTRICT WHEN DELETE TRIGGERS, an error is returned if delete triggers are defined on the table.

By default, a ROLLBACK statement can undo the truncate operation. However, if you specify IMMEDIATE, the truncate operation is processed immediately and cannot be undone. If the IMMEDIATE option is specified:

- The table must not contain any uncommitted updates.
- In the case of a table in a simple table space, the table space must not contain any uncommitted updates or the truncate operation fails. Also, if any uncommitted CREATE or ALTER statement has been executed, the truncate operation fails.
- The truncated table is immediately available for use in the same unit of work. Although a ROLLBACK statement is allowed to execute after a TRUNCATE statement, the truncate operation is not undone, and the table remains in a truncated state. For example, if another data change operation is done on the table after the TRUNCATE IMMEDIATE statement and then the ROLLBACK statement is executed, the truncate operation is not undone, but all other data change operations are undone.
- If the table is in a segmented table space or a universal table space, TRUNCATE IMMEDIATE allows deallocated spaces to be reclaimed immediately for subsequent insert operations in the same unit of work without committing the truncate operation.

Example 1

Empty an unused inventory table regardless of any existing triggers and return its allocated space:

```
TRUNCATE TABLE INVENTORY DROP STORAGE IGNORE DELETE TRIGGERS;
```

Example 2

Empty an unused inventory table regardless of any existing delete triggers but preserve its allocated space for later reuse:

```
TRUNCATE TABLE INVENTORY REUSE STORAGE IGNORE DELETE TRIGGERS;
```

Example 3

Empty an unused inventory table permanently (a ROLLBACK statement cannot undo the truncate operation when the IMMEDIATE option is specified) regardless of any existing delete triggers and preserve its allocated space for immediate use:

```
TRUNCATE TABLE INVENTORY REUSE STORAGE IGNORE DELETE TRIGGERS IMMEDIATE;
```

6.6.2 Using the TRUNCATE statement with multilevel security

When a user with a valid security label uses a TRUNCATE statement to delete all data from a table with row-level security enabled, DB2 compares the security label of the user to the security label of each row. The delete proceeds according to the following rules:

- If the security label of the user and the security label of the row are equivalent, the row is deleted.
- If the security label of the user dominates the security label of the row, the user's write-down privilege determines the result of the DELETE statement:
  - If the user has write-down privilege or write-down control is not enabled, the row is deleted.
  - If the user does not have write-down privilege and write-down control is enabled, the row is not deleted.
- If the security label of the row dominates or is disjoint with the security label of the user, the row is not deleted.
- If the row cannot be deleted as a result of the security label verification, the TRUNCATE statement fails.

6.6.3 Other considerations

The following considerations may be of interest if applicable to your environment:

- The truncate operation cannot be executed if the table is a parent table in a defined referential constraint. If the referential integrity constraint exists, the TRUNCATE statement is restricted regardless of whether the child table contains rows.
- There are no new restrictions if the TRUNCATE statement is used on a table with the Change Data Capture (CDC) attribute.
- If a VALIDPROC is defined for the table, the truncate operation needs to verify the validity of each row in the table.

6.7 INTERSECT and EXCEPT

The UNION, EXCEPT, and INTERSECT clauses specify the set operators union, difference, and intersection. UNION is already supported. For DB2 family compatibility, DB2 9 introduces EXCEPT and INTERSECT.

Archived

To combine two or more SELECT statements to form a single result table, use one of the following key words:

- UNION

  This returns all of the values from the result table of each SELECT statement. If you want all duplicate rows to be repeated in the result table, specify UNION ALL. If you want redundant duplicate rows to be eliminated from the result table, specify UNION or UNION DISTINCT.

- EXCEPT

  This returns all rows from the first result table (R1) that are not also in the second result table (R2). If you want all duplicate rows from R1 to be contained in the result table, specify EXCEPT ALL. If you want redundant duplicate rows in R1 to be eliminated from the result table, specify EXCEPT or EXCEPT DISTINCT.

  EXCEPT and EXCEPT ALL are alternatives to using subqueries to find *orphan* rows (that is, rows that are not picked up by inner joins).

- INTERSECT

  This returns rows that are in the result table of both SELECT statements. If you want all duplicate rows to be contained in the result table, specify INTERSECT ALL. If you want redundant duplicate rows to be eliminated from the result table, specify INTERSECT or INTERSECT DISTINCT.

Rules for columns:

- R1 and R2 must have the same number of columns, and the data type of the nth column of R1 must be compatible with the data type of the nth column of R2.
- R1 and R2 must not include columns having a data type of CLOB, BLOB, DBCLOB, XML, or a distinct type that is based on any of these types. However, this rule is not applicable when UNION ALL is used with the set operator.
- If the nth column of R1 and the nth column of R2 have the same result column name, the nth column of the result table of the set operation has the same result column name. Otherwise, the nth column of the result table of the set operation is unnamed.
- To order the entire result table, specify the ORDER BY clause at the end. Qualified column names cannot be used in the ORDER BY clause when the set operators are specified.

Duplicate rows

Two rows are duplicates if the value in each column in the first row is equal to the corresponding value of the second row. For determining duplicates, two null values are considered equal.

The DECFLOAT data type allows for multiple bit representations of the same number. For example, 2.00 and 2.0 are two numbers with the same coefficient, but different exponent values. So if the result table of UNION contains a DECFLOAT column and multiple bit representations of the same number exist, the one returned is unpredictable.

Operator precedence

When multiple set operations are combined in an expression, set operations within parentheses are performed first. If the order is not specified by parentheses, set operations are performed from left to right with the exception that all INTERSECT operations are performed before any UNION or any EXCEPT operations.

Archive

Results of set operators

Table 6-4 illustrates the results of all set operations, with rows from result tables R1 and R2 as the first two columns and the result of each operation on R1 and R2 under the corresponding column heading.

Table 6-4 Example of UNION, EXCEPT, and INTERSECT on result tables R1 and R2

Rows in R1	Rows in R2	Result of UNION ALL	Result of UNION DISTINCT	Result of EXCEPT ALL	Result of EXCEPT DISTINCT	Result of INTERSECT ALL	Result of INTERSECT DISTINCT
1	1	1	1	1	2	1	1
1	1	1	2	2	5	1	3
1	3	1	3	2		3	4
2	3	1	4	2		4	
2	3	1	5	4			
2	3	2		5			
3	4	2					
4		2					
4		3					
5		3					
		3					
		3					
		3					
		4					
		4					
		4					
		5					

Attention: The result of the EXCEPT set operator can be different if you switch the tables. The results shown in Table 6-4 are from the execution of the following statement:

```
SELECT C1 FROM R1 EXCEPT ALL SELECT C1 FROM R2 and
SELECT C1 FROM R1 EXCEPT DISTINCT SELECT C1 FROM R2
```

The execution of the following statements (by switching the tables R1 and R2) results in three rows with values 3 3 3 for the EXCEPT ALL and no rows are returned for EXCEPT DISTINCT:

```
SELECT C1 FROM R2 EXCEPT ALL SELECT C1 FROM R1 and
SELECT C1 FROM R2 EXCEPT DISTINCT SELECT C1 FROM R1
```

Archived

Example 1

Assume that tables T1 and T2 exist, each containing the same number of columns named C1, C2, and so on. The following example of the EXCEPT operator produces all rows that are in T1 but not in T2, with redundant duplicate rows removed:

```
SELECT * FROM T1 EXCEPT DISTINCT SELECT * FROM T2
```

If no NULL values are involved, this example returns the same result as:

```
SELECT DISTINCT * FROM T1
WHERE NOT EXISTS
(SELECT * FROM T2 WHERE T1.C1 = T2.C1 AND T1.C2 = T2.C2 AND ...)
```

Where the subquery contains an equal predicate for each pair of columns that exists in both tables.

Example 2

Assume that the tables T1 and T2 exist. The following example of INTERSECT operator produces all rows that are in both tables T1 and T2, with redundant duplicate rows removed:

```
SELECT * FROM T1 INTERSECT DISTINCT SELECT * FROM T2
```

If no NULL values are involved, this example returns the same results as:

```
SELECT DISTINCT * FROM T1
WHERE EXISTS
(SELECT * FROM T2
WHERE T1.C1 = T2.C2 AND T1.C2 = T2.C2 AND ...)
```

Where the subquery contains an equal predicate for each pair of columns that exists in both tables.

6.8 Built-in functions

The following is a list in alphabetical order of the Scalar functions introduced. Scalar functions related to the new data types BIGINT, BINARY, VARBINARY, and DECFLOAT are not included here. We provide details for some of these in this section.

- ASCII_CHR
- ASCII_STR
- COLLATION_KEY
- DECRYPT_BINARY
- DIFFERENCE
- EBCDIC_CHR
- EBCDIC_STR
- EXTRACT
- LOCATE_IN_STRING
- LPAD
- MONTHS_BETWEEN
- NORMALIZE_STRING
- OVERLAY
- RID
- RPAD
- SOUNDEX
- TIMESTAMPADD
- TIMESTAMPDIFF
- TIMESTAMP_FORMAT

Archived

- TIMESTAMP_ISO
- UNICODE
- UNICODE_STR
- VARCHAR_FORMAT

The following Aggregate functions have been introduced:

- CORRELATION
- COVARIANCE
- COVARIANCE_SAMP

6.8.1 ASCII_CHR

The ASCII_CHR function returns the character that has the ASCII code value that is specified by the argument. The schema is SYSIBM. The syntax is shown in Figure 6-5.

Figure 6-5 ASCII_CHR syntax

`Expression` is an expression that returns a built-in data type of BIGINT, INTEGER, or SMALLINT.

The result of the function is a fixed length character string encoded in the SBCS ASCII CCSID (regardless of the setting of the MIXED option in DSNHDECP). The length of the result is 1. If the value of expression is not in the range of 0 to 255 (0 to 127 if the SBCS ASCII CCSID for this system is CCSID 367), the null value is returned.

The result can be null. If the argument is null, the result is the null value.

CHR can be specified as a synonym for ASCII_CHR.

Example

Select the euro symbol in CCSID 923:

```
SELECT HEX(ASCII_CHR(164))
FROM SYSIBM.SYSDUMMY1 ;
Results in value A4 being displayed.
```

Select the Euro symbol in CCSID 1252:

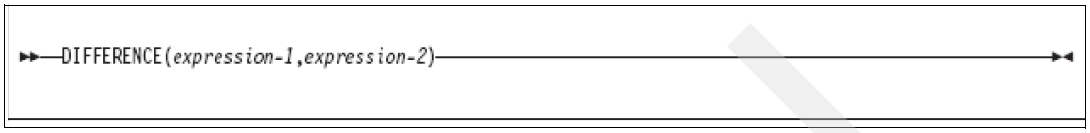
```
SELECT HEX(ASCII_CHR(128))
FROM SYSIBM.SYSDUMMY1 ;
Results in value 80 being displayed.
```

In both cases, the Euro symbol is displayed, but because the Euro symbol is located at different code points for the two CCSIDs, the input value is different.

Archived

6.8.2 DIFFERENCE

The DIFFERENCE function returns a value from 0 to 4 that represents the difference between the sounds of two strings based on applying the SOUNDEX function to the strings.



A value of 4 is the best possible sound match. The schema is SYSIBM. Each of the expression-1 or expression-2 arguments must return a value that is a built-in numeric, character string, or graphic string data type that is not a LOB. A numeric argument is cast to a character string before the function is evaluated. The data type of the result is INTEGER. If any argument is null, the result is the null value.

Example 1

Find the DIFFERENCE and SOUNDEX values for 'CONSTRAINT' and 'CONSTANT':

```
SELECT DIFFERENCE('CONSTRAINT','CONSTANT'),
   SOUNDEX('CONSTRAINT'),
   SOUNDEX('CONSTANT')
FROM SYSIBM.SYSDUMMY1;
```

This example returns the values 4, C523, and C523. Since the two strings return the same SOUNDEX value, the difference is 4 (the highest value possible).

Example 2

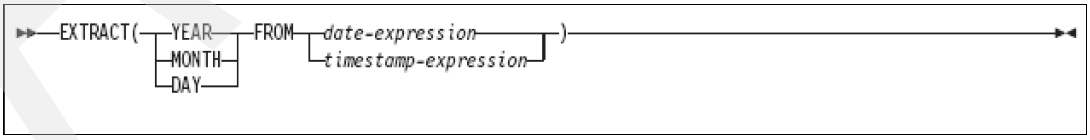
Find the DIFFERENCE and SOUNDEX values for 'CONSTRAINT' and 'CONTRITE':

```
SELECT DIFFERENCE('CONSTRAINT','CONTRITE'),
   SOUNDEX('CONSTRAINT'),
   SOUNDEX('CONTRITE')
FROM SYSIBM.SYSDUMMY1;
```

This example returns the values 2, C523, and C536. In this case, the two strings return different SOUNDEX values, and hence, a lower difference value.

6.8.3 EXTRACT date values

The EXTRACT date values function returns a portion of a date or time stamp based on its arguments. if the argument is null, the result is the null value. The schema is SYSIBM.



YEAR specifies that the year portion of date-expression or timestamp-expression is returned. The result is identical to the YEAR scalar function.

MONTH specifies that the month portion of date-expression or timestamp-expression is returned. The result is identical to the MONTH scalar function.

Archive

DAY specifies that the day portion of date-expression or timestamp-expression is returned. The result is identical to the DAY scalar function.

date-expression An expression that returns the value of either a built-in date or a built-in character string data type. If date-expression is a character or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid character-string or graphic-string representation of a date.

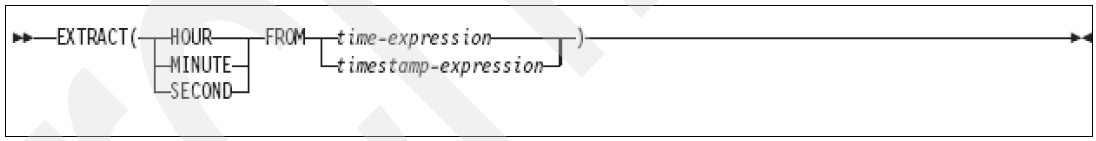
timestamp-expression An expression that returns the value of either a built-in time stamp or a built-in character string data type. If timestamp-expression is a character or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid character-string or graphic-string representation of a time stamp.

For example:

```
SELECT BIRTHDATE, EXTRACT (DAY FROM BIRTHDATE) AS DAY,
                  EXTRACT (MONTH FROM BIRTHDATE) AS MONTH,
                  EXTRACT (YEAR FROM BIRTHDATE) AS YEAR
  FROM DSN8910.EMP
  WHERE LASTNAME = 'HAAS' ;
---------+---------+---------+---------+---------+---------
BIRTHDATE           DAY         MONTH         YEAR
---------+---------+---------+---------+---------+---------
1933-08-14           14           8         1933
```

6.8.4 EXTRACT time values

The EXTRACT time values function returns a portion of a time or a time stamp based on its arguments. If the argument is null, the result is the null value. The schema is SYSIBM.



HOUR specifies that the hour portion of time-expression or timestamp-expression is returned. The result is identical to the HOUR scalar function.

MINUTE specifies that the minute portion of time-expression or timestamp-expression is returned. The result is identical to the MINUTE scalar function.

SECOND specifies that the second portion of time-expression or timestamp-expression is returned. The result is identical to the SECOND scalar function.

time-expression An expression that returns the value of either a built-in time or built-in character string data type. If time-expression is a character or graphic string, it must not be a CLOB or DBCLOB and its value must be a valid string representation of a time.

timestamp-expression An expression that returns the value of either a built-in time stamp or a built-in character string data type. If timestamp-expression is a character or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid string representation of a time.

Archive

The data type of the result of the function depends on the part of the datetime value that is specified:

- If YEAR, MONTH, DAY, HOUR, or MINUTE is specified, the data type of the result is INTEGER.
- If SECOND is specified, the data type of the result is DECIMAL(8,6). The fractional digits contains microseconds.

For example:

```
SELECT '12:30:50' AS TIME,EXTRACT(HOUR FROM '12:30:50') AS HOURS,
                          EXTRACT(MINUTE FROM '12:30:50') AS MINUTES,
                          EXTRACT(SECOND FROM '12:30:50') AS SECONDS
FROM SYSIBM.SYSDUMMY1;
---------+---------+---------+---------+---------+---------+---------
TIME            HOURS      MINUTES     SECONDS
---------+---------+---------+---------+---------+---------+---------
12:30:50           12           30    50.000000
```

6.8.5 MONTHS_BETWEEN

The MONTHS_BETWEEN function returns an estimate of the number of months between expression-1 and expression-2. The schema is SYSIBM.

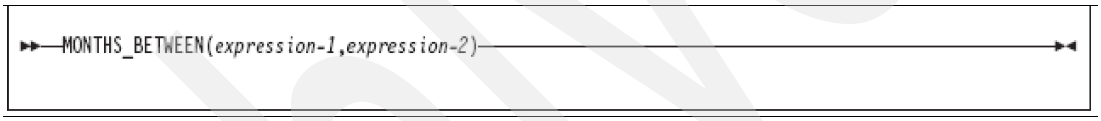


Figure 6-6 expression-1,expression-2

Expressions return a value of any of the following built-in data types: a date, a time stamp, a character string, or a graphic string. If either expression is a character or graphic string, it must not be a CLOB or DBCLOB, and its value must be a valid string representation of a date or time stamp with an actual length that is not greater than 255 bytes.

If expression-1 represents a date that is later than expression-2, the result is positive. If expression-1 represents a date that is earlier than expression-2, the result is negative. If expression-1 and expression-2 represent dates with the same day of the month, or the last day of the month, the result is a whole number. Otherwise, the result is calculated based on a 31-day month, and the result that is returned represents the difference between expression-1 and expression-2 as a fractional number.

The result of the function is a DECIMAL(31,15). If either argument is null, the result is the null value.

Example

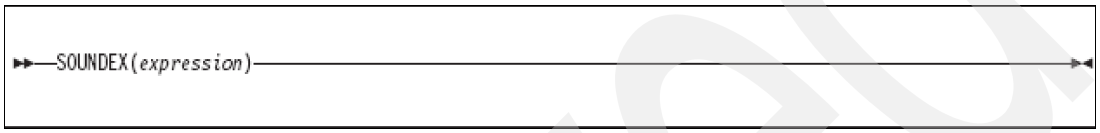
The following example calculates the months between two dates:

```
SELECT MONTHS_BETWEEN ('2007-01-17','2007-02-17')
   AS MONTHS_BETWEEN
FROM SYSIBM.SYSDUMMY1;
---------+---------+---------+---------+---------+---------+---------
MONTHS_BETWEEN
---------+---------+---------+---------+---------+---------+---------
-1.00000000000000
```

```
SELECT MONTHS_BETWEEN ('2007-02-20','2007-01-17')
AS MONTHS_BETWEEN
FROM SYSIBM.SYSDUMMY1;
---------+---------+---------+---------+---------+---------+---------
MONTHS_BETWEEN
---------+---------+---------+---------+---------+---------+---------
1.096774193548387
```

6.8.6 SOUNDEX

The SOUNDEX function returns a 4-character code that represents the sound of the words in the argument. The result can be used to compare with the sound of other strings. The schema is SYSIBM.



The argument expression must return a value of any built-in numeric, character, or graphic string data type that is not a LOB. A numeric, mixed character, or graphic string value is cast to a Unicode SBCS character string before the function is evaluated.

The data type of the result is CHAR(4). If the argument can be null, the result can be null. If the argument is null, the result is the null value.

The CCSID of the result is the Unicode SBCS CCSID.

The SOUNDEX function is useful for finding strings for which the sound is known but the precise spelling is not. It makes assumptions about the way that letters and combinations of letters sound that can help to search for words with similar sounds. The comparison of words can be done directly or by passing the strings as arguments to the DIFFERENCE function.

Example

Use the SOUNDEX function to find a row where the sound of the LASTNAME value closely matches the phonetic spelling of 'Loucesy':

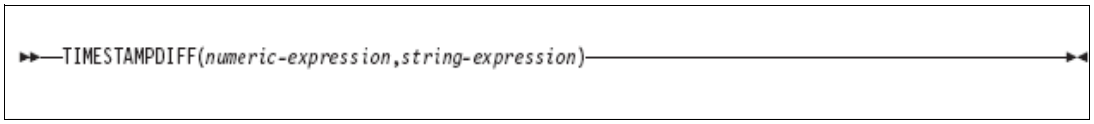
```
SELECT EMPNO, LASTNAME
FROM DSN910.EMP
WHERE SOUNDEX(LASTNAME) = SOUNDEX('Loucesy');
```

This example returns the following row:

```
000110 LUCCHESI;
```

6.8.7 TIMESTAMPDIFF

The TIMESTAMPDIFF function returns an estimated number of intervals of the type defined by the first argument, based on the difference between two time stamps. The schema is SYSIBM.



The argument numeric-expression is an expression that returns a value that is a built-in SMALLINT or INTEGER data type. The value specifies the interval that is used to determine the difference between two time stamps. Table 6-5 lists the valid values for numeric-expression.

Table 6-5 Specifying the type of time interval

Numeric expression (INTEGER or SMALLINT	Value
1	Microseconds
2	Seconds
4	Minutes
8	Hours
16	Days
32	Weeks
64	Months
128	Quarters
256	Years

The argument string-expression is the equivalent of subtracting two time stamps and converting the result to a string of length 22. The argument must be an expression that returns a value of a built-in character string or a graphic string data type that is not a LOB. If the supplied argument is a graphic string, it is first converted to a character string before the function is executed. Table 6-6 lists the valid input value ranges for string-expression.

Table 6-6 Valid input value ranges for string-expression and equivalent Datetime formats

Valid numeric ranges for string-expression	Equivalent Datetime
0–999999	Microseconds
0–59	Seconds
0–59	Minutes
0–24	Hours
0–30	Days
0–11	Months
0000–9998	Years

The result of the function is an integer. If either argument is null, the result is the null value.

The following assumptions are made when converting the information in the second argument, which is a time stamp duration, to the interval type that is specified in the first argument, and can be used in estimating the difference between the time stamps:

- One year has 365 days.
- One year has 52 weeks.
- One year has 12 months.
- One month has 30 days.
- One week has 7 days.

Archived

- One day has 24 hours.
- One hour has 60 minutes.
- One minute has 60 seconds.

For example, if the number of days (interval 16) is requested for a difference in time stamps for '1997-03-01-00.00.00' and '1997-02-01-00.00.00', the result is 30. The difference between the time stamps is one month, so the assumption that one month has 30 days applies.

Example

The following statement estimates the age of an employee whose last name is Haas in months and returns that value as AGE_IN_MONTHS:

```
SELECT LASTNAME, BIRTHDATE,
  TIMESTAMPDIFF(64,CAST(CURRENT_TIMESTAMP - TIMESTAMP_ISO(BIRTHDATE)
       AS CHAR(22))) AS AGE_IN_MONTHS
  FROM DSN8910.EMP
  WHERE LASTNAME = 'HAAS' ;
---------+---------+---------+---------+---------+---------+---------
LASTNAME         BIRTHDATE   AGE_IN_MONTHS
---------+---------+---------+---------+---------+---------+---------
HAAS             1933-08-14            881
```

6.8.8 TIMESTAMP_FORMAT

The TIMESTAMP_FORMAT function returns a time stamp. The schema is SYSIBM. The syntax is shown in Figure 6-7.

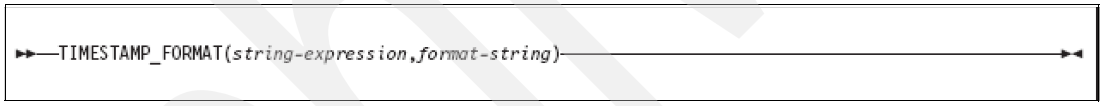


Figure 6-7 Time stamp syntax

The syntax is:

string-expression An expression that returns a value of any built-in character string data type or graphic string data type, other than a CLOB or DBCLOB, with a length attribute that is not greater than 255 bytes. Leading and trailing blanks are removed from the string, and the resulting substring is interpreted as a time stamp using the format that is specified by format-string.

format-string A character string constant with a length that is not greater than 255 bytes. format-string contains a template of how string-expression is to be interpreted as a time stamp value. Leading and trailing blanks are removed from the string, and the resulting substring must be a valid template for a time stamp.

The valid formats that can be specified for the function are:

- 'YYYY-MM-DD'
- 'YYYY-MM-DD-HH24-MI-SS'
- 'YYYY-MM-DD-HH24-MI-SS-NNNNNN'

Archived

Instead of using - as the separator, you can also use . / , : ; and blank. These separators can be used in any combination (for example, 'YYYY/MM-DD HH:MM.SS').

DD	Day of month (01–31)
HH24	24 hours of day (00.24. When the value is 24, the minutes and seconds must be 0.)
MI	Minutes (00.59)
MM	Month (01–12, January = 01)
NNNNNN	Microseconds (000000-999999)
SS	Seconds (00.59)
YYYY	4-digit year

The result of the function is a time stamp. If the argument is null, the result is the null value.

TO_DATE can be specified as a synonym for TIMESTAMP_FORMAT.

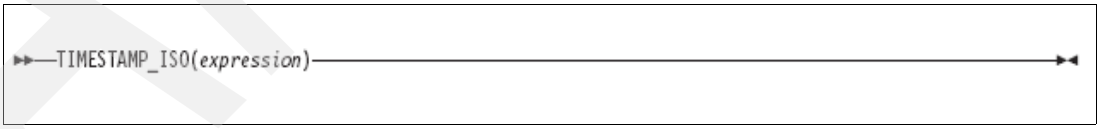
Example

Retrieve the time stamp value of CREATEDTS from SYSIBM.SYSSTOGROUP using the character string format specified. However, only the entries made in the catalog table on or after September 1, 2006 are required.

```
SELECT SUBSTR(NAME,1,8) AS NAME,
       VARCHAR_FORMAT(CREATEDTS,'YYYY-MM-DD-HH24:MI:SS') AS CREATED_TS
FROM SYSIBM.SYSSTOGROUP
WHERE CREATEDTS >=
TIMESTAMP_FORMAT('2006-09-01 00:00:00','YYYY-MM-DD HH24:MI:SS');
---------+---------+---------+
NAME      CREATED_TS
---------+---------+---------+
TESTSG    2006-09-01-17:36:04
SABISG    2006-09-08-16:27:43
NOVOL     2006-09-15-15:52:49
```

6.8.9 TIMESTAMP_ISO

The TIMESTAMP_ISO function returns a time stamp value that is based on a date, a time, or a time stamp argument. The schema is SYSIBM.



The argument is an expression that returns a value of one of the following built-in data types: a time stamp, a date, a time, a character string, or a graphic string. If expression is a date, TIMESTAMP_ISO inserts a value of zero for the time and microseconds parts of the time stamp. If expression is a time, TIMESTAMP_ISO inserts the value of CURRENT DATE for the date part of the time stamp and a value of zero for the microseconds part of the time stamp. If expression is a character or graphic string, it must not be a CLOB or DBCLOB and its value must be a valid string representation of a date, a time, or a time stamp. The result of the function is a time stamp. If the argument is null, the result is the null value.

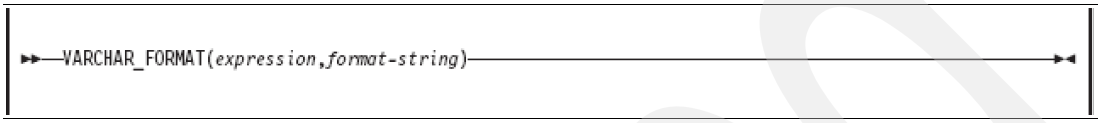
Archived

The CAST specification should be used for maximum portability. For example, the following returns the value '2007-02-14-00.00.00.000000':

```
SELECT TIMESTAMP_ISO(DATE('2007-02-14'))
FROM SYSIBM.SYSDUMMY1
```

6.8.10 VARCHAR_FORMAT

The VARCHAR_FORMAT function returns a character representation of a time stamp in the format that is indicated by format-string. The schema is SYSIBM.



The variables are:

expression An expression returns a value of a built-in time stamp data type.

format-string A character string constant with a maximum length that is not greater than 255 bytes. format-string contains a template of how expression is to be formatted.

A valid format string can contain a combination of the format elements listed in Table 6-7 on page 164. Two format elements can optionally be separated by one or more valid separator characters (- . / , : ; and blank). Separator characters may also be specified at the start or end of format-string. The elements of format-string must be specified in upper case.

Table 6-7 shows the valid formats that can be specified for the function.

Table 6-7 Format string values

format-string	Value
CC	Century (00–99). If the last two digits of the four-digit year are zero, the result is the first two digits of the year. Otherwise, the result is the first two digits of the year plus one.
D	Day of week (1–7).
DD	Day of month (01–31).
DDD	Day of year (001–366).
FF*n*	Fractional seconds (0–999999). The number *n* is used to specify the number of digits to include in the value returned. Valid values for n are integers 1 through 6. The default is 6.
HH	Hour of the day (01–12).
HH12	Hour of the day (01–12).
HH24	Hour of day (00–24). When 24, minutes and seconds must be 0.
ID	ISO day of week where 1 is Monday and 7 is Sunday (1–7).
IW	ISO week of year (1–53). The week starts on Monday and includes 7 days. Week 1 is the first week of the year to contain a Thursday, which is equivalent to the first week of the year to contain January 4.

format-string	Value
IYYY	ISO year (0000–9999). The last four digits of the year based on the ISO week that is returned.
J	Julian date.
MI	Minutes (00–59).
MM	Month (01–12) where January is 01.
MON	Three-character abbreviation of the month in English (JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, or DEC).
NNNNNN	Microseconds (0–999999).
Q	Quarter (1–4) where the months January through March return 1.
RR	RR behaves the same way as YY for the VARCHAR_FORMAT function.
SS	Seconds (00–59).
SSSSS	Seconds since previous midnight (00001–86400).
WW	Week of the year (1–53), where week starts on January 1 and ends on January 7.
YY	Year (00–99). Last two digits of the year.
YYYY	Year (0000–9999). The last four digits of the year.

The result is the varying-length character string that contains the argument in the format that is specified by format-string. format-string also determines the length attribute and actual length of the result. If either argument is null, the result is the null value.

The CCSID of the result is determined from the context in which the function is invoked.

TO_CHAR can be specified as a synonym for VARCHAR_FORMAT.

Example

Retrieve the time stamp value of CREATEDTS from SYSIBM.SYSSTOGROUP using the character string format specified:

```
SELECT SUBSTR(NAME,1,8) AS NAME,
       VARCHAR_FORMAT(CREATEDTS,'YYYY-MM-DD-HH24:MI:SS')
FROM SYSIBM.SYSSTOGROUP;
---------+---------+---------+
NAME      CREATED_TS
---------+---------+---------+
SYSDEFLT  2006-08-29-10:16:15
DSN8G910  2006-08-29-11:17:06
TESTSG    2006-09-01-17:36:04
SABISG    2006-09-08-16:27:43
NOVOL     2006-09-15-15:52:49
```

Archived

7

Application enablement

This chapter describes DB2 9 for z/OS functions, not strictly confined to SQL, which provide infrastructure support for new applications. See Chapter 6, “SQL” on page 131, for topics more strictly related to SQL language changes.

This chapter discusses the following:

- Automatic creation of objects
- Index on expression
- Optimistic concurrency control and update detection
- Cultural sort and case insensitive comparison
- CURRENT SCHEMA
- Skip locked rows
- LOB file reference
- FETCH CONTINUE
- Spatial support
- Allow ALTER TABLE ... LONG VARCHAR TO VARCHAR
- RENAME COLUMN
- RENAME INDEX
- SMS constructs
- Native SQL procedures
- Nested compound statements in SQL stored procedures
- New stored procedure related special registers
- DB2 and DSN commands for native SQL procedures
- Changes to DB2-supplied dynamic SQL processing applications
- Differences from external SQL procedures
- Unified Debugger

© Copyright IBM Corp. 2007. All rights reserved.

7.1 Automatic creation of objects

In DB2 V8, when you use the CREATE TABLE statement without specifying an associated table space and database, DB2 implicitly creates a default-sized table space in default database DSNDB04 for you. To make your life easier, DB2 V9 provides additional ways to help you create tables and associated objects quicker. DB2 V9 has the capability to create the following objects implicitly depending on how you code your CREATE TABLE SQL statement:

- Database
- Table space
- Enforcing primary key index
- Enforcing unique key index
- ROWID index if the ROWID column is defined as GENERATED BY DEFAULT
- LOB table space, auxiliary table, auxiliary index

7.1.1 Implicit database

If you do not specify the IN clause on your CREATE TABLE statement, DB2 will now not only create an implicit table space for you using database DSNDB04 — starting with DB2 9, DB2 also creates an implicit database for you or it uses a previously implicitly created database. DB2 uses the range from DSN00001 to DSN60000 as naming convention for implicitly created databases. The maximum number of database per DB2 subsystem has been increased from 32767 to 65271.

Creation of implicit databases

When you type your first CREATE TABLE statement in V9 NFM without specifying the IN clause, DB2 implicitly creates database DSN00001 for you. The second created database is DSN00002, and so on until DSN60000 is reached. No more implicit database will be added from now on. Instead, DB2 wraps around and uses existing implicitly created databases. Each of those databases can contain multiple table spaces and tables.

The characteristics of implicitly created databases are shown in Table 7-1.

Table 7-1 Characteristics of an implicitly created database

Field	Value
Name	DSNXXXXX, where XXXXX= 00001 - 60000
BUFFERPOOL	BP0, BP8K0, BP16K0, BP32K Default values. Changeable through DSNZPARM update.
INDEXBP	IDXBPOOL setting in DSNZPARM
STOGROUP	SYSDEFLT
value in column IMPLICIT of SYSIBM.SYSDATABASE	'Y'
ENCODING_SCHEME	DEFAULT: DSNHDECP setting
SBCS_CCSID	DEFAULT: DSNHDECP setting
DBCS_CCSID	DEFAULT: DSNHDECP setting
MIXEC_CCSID	DEFAULT: DSNHDECP setting

Usage of implicit databases

There are some special rules that apply to implicitly created databases and the associated naming conventions:

- You cannot explicitly create a database following the naming conventions of implicitly created databases. If you try to do so, you will end up in error SQLCODE -20074. The given error message states that you are not allowed to create a database using DSN as the first three characters. This error message is misleading. It is possible to create databases whose names start with DSN. DB2 just prevents you from creating databases that start with DSN and are followed by five numeric values.
- You can use the ALTER DATABASE statement to change the characteristics of the implicit created database if you have the proper privileges. The creator of implicitly created databases is SYSIBM. Therefore the authorization ID that issued the CREATE TABLE statement that led to the creation of the database might not have the proper privileges.
- You cannot specify the name of an implicitly created database in the IN clause of a CREATE TABLE of CREATE TABLESPACE statement.

Figure 7-1 shows you exactly how DB2 reuses those implicit databases. In DB2 V8, you were allowed to use 'DSN' followed by a 5-digit number as the database name. Due to that, it is possible that after migration to V9, you already have databases defined in your subsystem that use names that are now reserved to implicit databases.

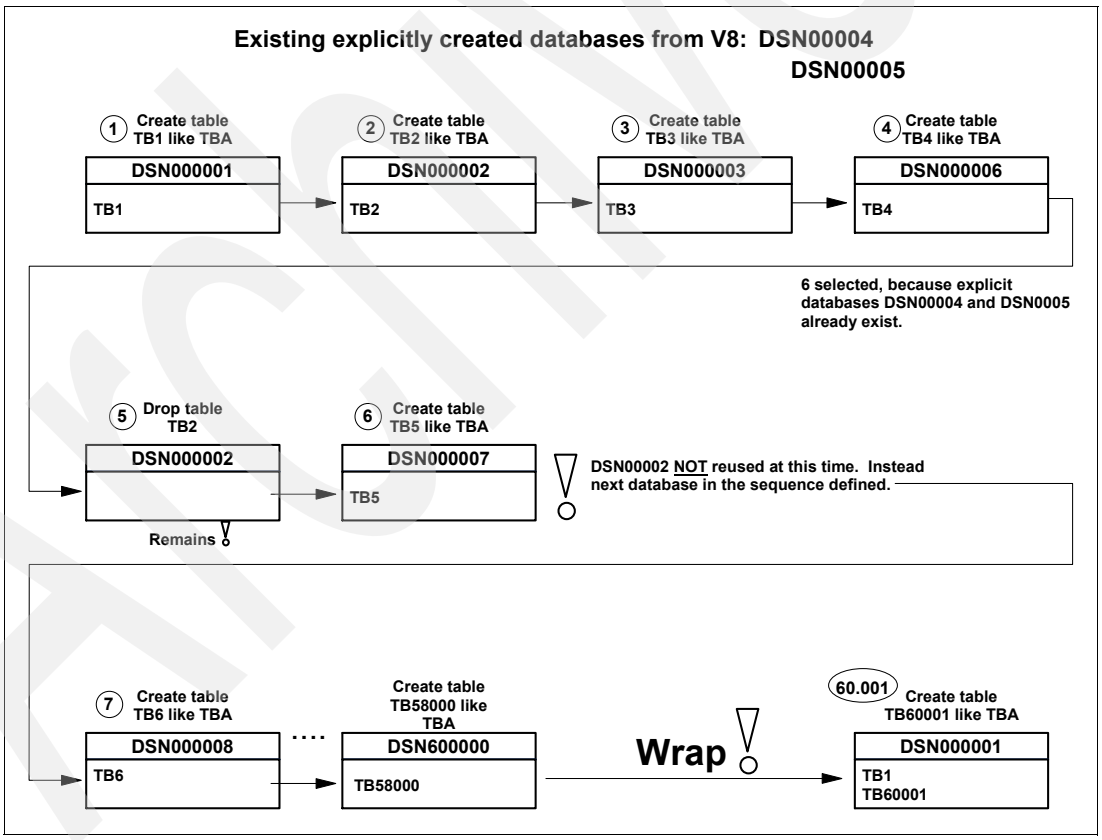


Figure 7-1 Reuse of implicitly created databases

We assume that databases DSN00004 and DSN00005 are explicit databases that have been migrated from V8 to V9.

1. Create table TB1 without using the IN clause on the CREATE TABLE statement. As a result, DB2 creates database DSN00001 implicitly for this table.

2. Create table TB2 without using the IN clause on the CREATE TABLE statement. DSN00001 is not used to store a second table now. Instead, the database name is incremented by one and DB2 implicitly creates database DSN00002.
3. Again and for all following CREATE TABLE statements, the IN clause is not used. The database name is incremented by 1 again and database DSN00003 is implicitly generated.
4. Starting with the creation of table TB4, the numbering of the table and the database diverges. Since DSN00004 already exists, DB2 continues to check whether DSN00005 is available for being created. Since this one also exists, DB2 must increment the number one more time, and finally implicitly creates database DSN00006.
5. You drop table TB2 now. Database DSN00002 that hosted this table is not implicitly dropped as well. It continues to exist as an empty database.
6. If you create a new table TB5 subsequently, DB2 does not reuse database DSN00002, which has just become empty. Instead, it continues incrementing the database names by 1 and creates the next database in row, which is DSN00007.
7. If you continuously create new tables without using the IN clause, DB2 also continuously creates new implicit databases until it finally reaches DSN60000. Once this number is reached, DB2 starts wrapping around.
8. The next CREATE TABLE statement, which could be, for example, for table TB60001, again starts using the implicit databases with the counter reset to 1. Following this, DB2 creates table TB60001 in DSN00001, which has been used in the past for hosting TB1. DSN00001 now contains two tables.

The whole process continues over and over again, incrementing the number or database that is to be used by 1 after every CREATE TABLE without an IN clause in it.

Attention: if you do not specify SEGSIZE, NUMPARTS, or MAXPARTITIONS, the implicitly created table space is created with SEGSIZE 4 and LOCKSIZE ROW.

Note: If you create a table space explicitly using the CREATE TABLESPACE statement, but you do not specify a database name, DSNDB04 is used as the hosting database.

As you can see from the Figure 7-1 on page 169 and from the description above, DB2 continues to create new implicit databases for you until you reach DSN60000. You should monitor the size of your DSNDB06.SYSDBASE table space as well as the usage of your EDM DBDC cache.

Tip: If you want to prevent users from being able to create implicit databases, you must revoke the CREATETB privilege on DSNDB04 from PUBLIC.

7.1.2 Implicit table space

DB2 has always supported the implicit creation of table spaces as part of the creation of a table. The structure of the created table space used to be a simple table space. Starting with DB2 V9, all implicitly created table spaces will be segmented table spaces.

Note: You can no longer create simple table spaces. However, DB2 continues supporting the existing simple table spaces that were created prior to V9.

While you are operating in V9 compatibility mode, all implicitly created table spaces will have:

- SEGSIZE 4
- LOCKSIZE ROW

Once you have migrated your DB2 subsystem to V9 NFM, all implicitly created table spaces will be allocated with:

- SEGSIZE 4
- DSSIZE = 4G
- MAXPARTITIONS = 256
- LOCKSIZE = ROW
- LOCKMAX=SYSTEM

As a consequence, all implicitly created table spaces in DB2 V9 are *partition by growth table spaces*. Refer to 4.1.1, “Partition-by-growth table spaces” on page 74 to learn more about the characteristics of these table spaces.

Note: You cannot use the CREATE TABLE..... statement with PARTITION BY syntax without specifying a table space. Due to this, implicitly created table spaces can never be range-partitioned table spaces or regular partitioned table spaces. Refer to 4.1.2, “Range-partitioned universal table spaces” on page 80, for a more detailed description of this table space species.

Influence of system parameters

New system parameters have been added to DB2 V9 that you can use to influence DB2’s behavior when it creates implicit table spaces.

IMPDSDEF - Define data sets

This parameter corresponds to the DEFINE YES/NO on the CREATE TABLESPACE or CREATE INDEX statement. Depending on which value you choose to set in your DSNZPARM, DB2 either defines the underlying data set when you implicitly create a table space or not. The setting of this parameter is also used for implicitly created enforcing primary key, unique, and ROWID indexes. The default value is YES.

Note: This value is not used for implicitly created LOB or XML table spaces, auxiliary indexes, or DocID indexes. The underlying VSAM clusters are always created in these cases.

IMPTSCMP - Use data compression

This new parameter corresponds to COMPRESS YES/NO on the CREATE TABLESPACE statement. That is, whenever you create a table space implicitly, the setting of this parameter controls whether your table space will be compressed. The default is COMPRESS NO.

Note: This new parameter is also used for implicit table spaces that are created in the default database DSNDB04. This is a change in behavior compared to what we had in DB2 V8.

Default buffer pools

You can choose default buffer pools for implicit objects per page size. The associated ZPARMS are:

- TBSBPOOL for 4 KB pages
- TBSBP8K for 8 KB pages
- TBSBP16K for 16 KB pages
- TBSBP32K for 32 KB pages
- IDXBPOOL for indexes

DB2 chooses a specific buffer pool during creation of the implicit object depending on the record size. When the maximum record size reaches approximately 90% of the capacity of the smaller page size, DB2 chooses the next larger page size. This 90% rule is applied in order to reduce the likeliness of having to change the page size later in case you extend existing or add new columns. Table 7-2 shows the approximate maximum record size used for the calculation of page sizes.

Table 7-2 Maximum record size calculation for different buffer pools

Page size	Maximum record size calculation
4 KB	3640
8 KB	7310
16 KB	14680
32 KB	32768

We are telling you about these thresholds that are used for the automatic selection of page sizes because it is your responsibility to activate or adjust your buffer pool sizes accordingly.

TBSBPLOB

This new ZPARM allows the user to specify the default buffer pool to use for LOB table space that is created implicitly and for LOB table space that is created explicitly without the BUFFERPOOL clause. The USE privilege is required on the specified buffer pool in order to use it by default. The default for this ZPARM is BP0.

TBSBXML

This ZPARM allows the user to specify the default buffer pool to use for the XML table space that is created implicitly. The USE privilege is required on the specified buffer pool in order to use it by default.

Let us now look at some possible scenarios in this context

Scenario 1

You plan to create a table with a record length of 3500 bytes. You specify that DB2 should use a database that has been created using buffer pool BP8K1. You do not explicitly create the underlying table space but let DB2 create it implicitly for you.

As a result, your table would be created in an implicitly created table space using the default buffer pool for 8 KB pages. That is, DB2 does not use the minimal possible pages size, but inherits the one of the hosting database.

Scenario 2

You plan to create a table with a record length of 10000 bytes. You specify that DB2 should use a database that has been created using buffer pool BP8K1. You do not explicitly create the underlying table space but let DB2 create it implicitly for you.

In this case, the inherited page size is not large enough to store a 10000-byte row. In this case, DB2 would now allocate a 16 KB table space according to the information shown in Table 7-2 on page 172.

Scenario 3

You create a table with record length 3500 and force DB2 to create the hosting table space implicitly. Following the information in Table 7-2 on page 172, DB2 decides to create a 4-K table space. After a certain time, you ALTER the width of some columns and the record length becomes larger than 3640 bytes and smaller than 4 K. Even if in the initial step DB2 would now have chosen a page size of 8 KB, the ALTER statement works fine without any changes to the table space.

If you try to alter the record length to a value larger than 4 K, you will end up in -670, telling you the record length the page size limit exceeds.

7.1.3 Implicit system required objects

As mentioned in the introduction to this topic, DB2 V9 has been enhanced so that you can now create tables and associated objects automatically that would have required additional manual creation steps in order to complete the table definition in V8. The system-required objects are always created automatically if the table space hosting the table that requires additional objects is created implicitly. That is, if you specify a table space name in your CREATE TABLE statement, DB2 does not create the system required objects. In this case your table definition is considered incomplete until you manually create those missing objects. The automatic creation of system-required objects is only dependent upon the implicit creation of the hosting table space. It does not matter whether this table space is created in an implicit database or in DSNDB04.

Enforcing primary key index

An enforcing primary key index is necessary in order to complete the table definition if you code CREATE TABLE.... columname NOT NULL PRIMARY KEY. DB2 automatically creates a new index for you when the above-mentioned prerequisites are fulfilled.

The names of those implicitly created indexes are a combination of the table name that they are created on and some additional randomly generated characters.

Enforcing unique key index

An enforcing unique key index is necessary in order to complete the table definition if you code CREATE TABLE.... columname NOT NULL UNIQUE. DB2 automatically creates a new index for you when the above prerequisites are fulfilled.

LOB table space, auxiliary table, auxiliary index

In prior DB2 releases, there was limited support for the automatic creation of system-required LOB-related objects depending on the CURRENT RULES register.

CURRENT RULES is a data server register where you can specify whether certain SQL statements are executed in accordance with DB2 rules or the rules of the SQL standard. The default setting for SQLRULES is ’DB2’. The valid values are 'DB2' and 'STD'. If the server is not the local DB2, the initial value of the register is 'DB2'. Otherwise, the initial value is the

same as the value of the SQLRULES bind option. You can change the value of the register by executing the statement SET CURRENT RULES.

When you issued your DDL containing LOB columns with special register CURRENT RULES set to ‘STD’, DB2 created the LOB table space, the auxiliary table, and the auxiliary index automatically.

The creation was only dependent on the CURRENT RULES setting. That is, the associated objects have always been created automatically, independently of whether the base table space was created implicitly or explicitly.

In DB2 V9, if the base table is created implicitly, all those LOB-related objects are created automatically. This is turn is now independent from the CURRENT RULES setting. DB2 still supports SET CURRENT RULES = 'STD' automatic creation. The automatic creation does not apply for explicitly created table spaces.

ROWID index

Starting with DB2 V9, if you create a table with one of the columns being defined as ROWID GENERATED BY DEFAULT, the necessary indexes are now also automatically created for you.

Attention: You cannot drop any of these automatically created implicit system required objects. The only way to remove them from your DB2 subsystem is to drop the containing objects such as primary key, unique key, LOB column, and ROWID column.

7.1.4 Authorization considerations

To create an implicit table, the following four authorities must all be held explicitly or implicitly, either at database or system level. The database that controls the creation of implicit tables is DSNDB04, even though the table will reside in either a new database or an existing one in the range DSN00001 to DSN60000.

- Create table privilege in DSNDB04.
- Create table space privilege in DSNDB04.
- Use of the default storage group.
- Use of the buffer pool the underlying table space will default to.

Note: It is not sufficient to have database authorities on database DSN00001 (even if this is the database the table would end up in).

Authorization of existing implicit tables

To access an implicit table directly, one of the following authorities is needed:

- Ownership of the table
- Select, update authority, and so on, on the specific table
- DBADM on DSNDB04
- DBADM on the actual database the table resides in (see note below about control via RACF® exit)
- SYSADM

Authorization to implicit table spaces

This is for commands such as STOP DATABASE(dsn00020) SPACE(mytable) or utilities:

- The required authority (for example, IMAGCOPY) on DSNDB04 (This could be implicit in DBCNTL, for example.)
- The required authority (for example, DISPLAYDB) on the actual database the table space is in (see note below about control via RACF exit) (This could be implicit in DBADM, for example.)
- SYSADM or SYSCTRL
- The special case for implicit table spaces, the owner of the table space

Authorization using the actual database of an implicit table or table space

For installations using RACF to control authorization for DB2 objects, authorities cannot be obtained via the actual database the table or table space resides in. RACF will use DSNDB04. RACF will honor the ownership of the table space to allow commands or utilities.

Note: A user with an authority on DSNDB04 with the grant option cannot grant a database authority on the databases in the range DSN00001 to DSN60000.

7.2 Index on expression

Index on expression allows you to create an index on a general expression. You can enhance your query performance if the optimizer chooses the index created on the expression.

Index on expression is one of the possible types of extended indexes. Other extended indexes are spatial index and XML value index.

Use index on expression when you want an efficient evaluation of queries involving a column-expression. In contrast to simple indexes where index keys are composed by concatenating one or more table columns that you specify, the index key values are not exactly the same as values in the table columns. The values have been transformed by the expressions that you specify.

If the index is defined using an expression, the uniqueness is enforced against the values that are stored in the index, not against the original values of the columns.

Figure 7-2 shows the CREATE INDEX statement syntax to create index on expression.

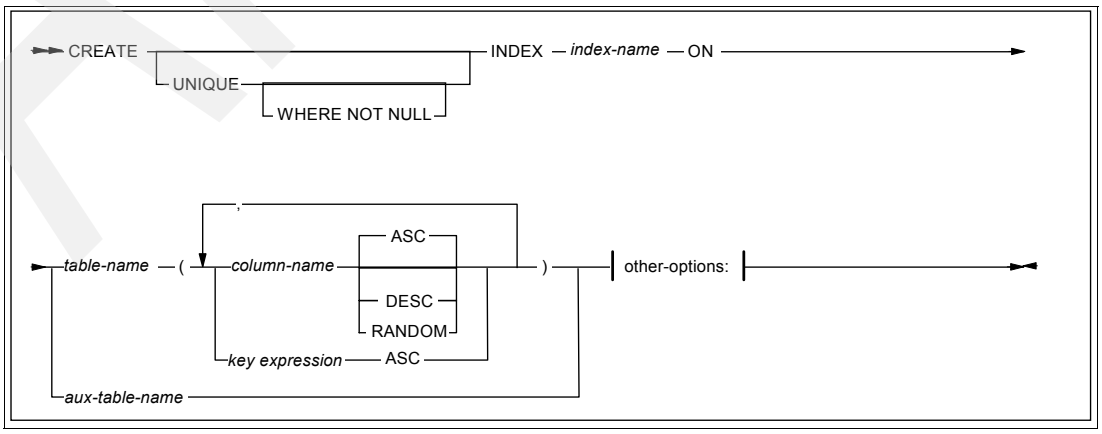


Figure 7-2 CREATE INDEX statement for index on expression

An index on expression cannot be a clustering index.

If the index is created using an expression, the EXECUTE privilege is required on any user-defined function that is invoked in the index expression.

ASC puts the index entries in ascending order by the column.

RANDOM puts the index entries in a random order by the column. DESC puts the index entries in descending order by the column.

Neither RANDOM nor DESC can be specified if the ON clause contains key-expression.

7.2.1 Examples

Assume that the following table is created:

```
CREATE TABLE employee (
   id INTEGER NOT NULL,
   lastname VARCHAR(20) NOT NULL,
   firstname VARCHAR(20) NOT NULL,
   salary DEC(15,2) NOT NULL,
   bonus FLOAT)
```

1. Create the following index on expression:

   ```
   CREATE INDEX upper_empname ON employee
   (UPPER(lastname, 'EN_US'), UPPER(firstname, 'EN_US'), id)
   ```

   The following entries are populated in the catalog tables after the index upper_empname is created successfully:

   – The qualified names of the index are shown in the SYSIBM.SYSINDEXES table with IX_EXTENSION_TYPE = 'S' to indicate that the index is an index on expression.

   – The entries shown in Table 7-3 are populated into the SYSIBM.SYSKEYS table to show the columns on which the index depends. *colno* here means the source column via which the index key target is calculated.

Table 7-3 SYSIBM.SYSKEY for index upper_empname

colname	colno	ordering
ID	1	blank
LASTNAME	2	blank
FIRSTNAME	3	blank

   – The entries shown in Table 7-4 are populated into the new SYSIBM.SYSKEYTARGETS catalog table to show the key target that forms the index key entry.

Table 7-4 SYSIBM.SYSKEYTARGETS for index upper_name

keyseq	colno	datatypeid	length	derived_from
1	0	448 (varchar)	20	UPPER(LASTNAME,'EN_US)
2	0	448 (varchar)	20	UPPER(FIRSTNAME,'EN_US)
3	1	496 (integer)	4	ID

2. Create the following index on expression on employee total compensation using the columns salary and bonus:

```
CREATE INDEX total_compensation ON employee (salary + bonus)
```

Note that the index total_compensation is created based on two columns (salary and bonus), but it has only one key-target, which is generated by the scalar expression "salary + bonus". In this case, the entries shown in Table 7-5 are populated into SYSIBM.SYSKEYS to show that this index depends on the salary column and the bonus column.

Table 7-5 SYSIBM.SYSKEYS for index total_compensation

colname	colno	ordering
SALARY	4	blank
BONUS	5	blank

Since this index has only one key-target, the entry in SYSIBM.SYSTARGETS in Table 7-6 shows the result of the expression.

Table 7-6 SYSIBM.SYSKEYTARGETS for index total_compensation

keyseq	colno	datatypeid	length	derived_from
1	0	480 (float)	8	SALARY + BONUS

3. Create the following unique index:

```
CREATE UNIQUE INDEX empname ON employee
(SUBSTR(firstname,1,1) CONCAT '. ' CONCAT lastname);
```

If an index is created as a UNIQUE index, the uniqueness is enforced against the values stored in the index, not the values stored in the table columns. In this case, DB2 may return a duplicate key error even if it appears that there is no duplicate value in the table columns.

DB2 returns a duplicate key error when you try to insert an employee name Monica Smith into the employee table if the table already contains an employee name Michael Smith. In this case, both Monica Smith and Michael Smith produce same index entry M. Smith even though they appear to be different in the table.

7.2.2 Considerations on index on expression

You should be aware of the following considerations:

- Errors evaluating the expressions for an index

  Errors that occur during the evaluation of an expression for an index are returned when the expression is evaluated. This can occur on an UPDATE, INSERT, DELETE statement, SELECT from a DML data change statement, or utilities such as REBUILD INDEX, CREATE INDEX, or REORG TABLESPACE. For example, the evaluation of the expression 10 / column_1 returns an error if the value in column_1 is 0. The error is returned during CREATE INDEX processing if the table is not empty and contains a row with a value of zero in column_1. The error might also be returned during the processing of the insert or update operation when a row with a value of zero in column_1 is inserted or updated.

- Result length of expressions that return a string type

  If the result data type of key-expression is a string type and the result length cannot be calculated at bind time, the length is set to the maximum allowable length of that data type

or the largest length that DB2 can estimate. In this case, the CREATE INDEX statement can fail because the total key length might exceed the limit of an index key.

For example, the result length of the expression REPEAT('A', CEIL(1.1)) is VARCHAR(32767) and the result length of the expression SUBSTR(DESCRIPTION,1,INTEGER(1.2)) is the length of the DESCRIPTION column. Therefore, a CREATE INDEX statement that uses any of these expressions as a key-expression might not be created because the total key length might exceed the limit of an index key.

7.2.3 Restrictions

The key-expression that you specify has the following restrictions:

- Each key-expression must contain as least one reference to a table column.
- All references to table columns must be unqualified. Referenced columns cannot be LOB, XML, or DECFLOAT data types or a distinct type that is based on one of these data types. Referenced columns cannot include any FIELDPROCs or a SECURITY LABEL.
- Key-expression must not include the following:
  - A subquery
  - An aggregate function
  - A not deterministic function
  - A function that has an external action
  - A user-defined function
  - A sequence reference
  - A host variable
  - A parameter marker
  - A special register
  - A CASE expression
- If key-expression references a cast function, the privilege set must implicitly include EXECUTE authority on the generated cast functions for the distinct type.
- If key-expression references the LOWER or UPPER functions, the input string-expression cannot be FOR BIT DATA.
- If key-expression references the TRANSLATE function, the function invocation must contain the to-string argument.
- The same expression cannot be used more than one time in the same index.
- The data type of the result of the expression (including the intermediate result) cannot be a LOB, XML, or DECFLOAT value.
- The CCSID encoding schema of the result of a key-expression must be the same encoding scheme as the table.

  The maximum length of the text string of each key-expression is 4000 bytes after conversion to UTF-8. The maximum number of key-expression in an index on expression is 64.

7.3 Optimistic concurrency control and update detection

Optimistic concurrency control, also known as *optimistic locking,* represents a faster, more scalable locking alternative to database locking for concurrent data access. It minimizes the time for which a given resource is unavailable for use by other transactions. When an application uses optimistic concurrency control, locks are obtained immediately before a read

operation and released immediately. Update locks are obtained immediately before an update operation and held until the end of the transaction. Optimistic concurrency control uses the RID and a row change token to test whether data has been changed by another transaction since the last read operation.

Because DB2 can determine when a row was changed, it can ensure data integrity while limiting the time that locks are held. With optimistic concurrency control, DB2 releases the row or page locks immediately after a read operation. DB2 also releases the row lock after each FETCH, taking a new lock on a row only for a positioned update or delete to ensure data integrity.

To safely implement optimistic concurrency control, you must establish a ROW CHANGE TIMESTAMP column with a CREATE TABLE statement or an ALTER TABLE statement. The column must be defined as:

```
NOT NULL GENERATED ALWAYS FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP or
NOT NULL GENERATED BY DEFAULT FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP
```

After you establish a ROW CHANGE TIMESTAMP column, DB2 maintains the contents of this column. When you want to use this change token as a condition when making an update, you can specify an appropriate condition for this column in your WHERE clause.

7.3.1 Optimistic concurrency control and cursor stability isolation

For packages and plans that contain updatable static scrollable cursors, ISOLATION(CS) lets DB2 use optimistic concurrency control. DB2 can use optimistic concurrency control to shorten the amount of time that locks are held in the following situations:

- Between consecutive fetch operations
- Between fetch operations and subsequent positioned update or delete operations

DB2 cannot use optimistic concurrency control for dynamic scrollable cursors. With dynamic scrollable cursors, the most recently fetched row or page from the base table remains locked to maintain position for a positioned update or delete.

Figure 7-3 and Figure 7-4 on page 180 show processing of positioned update and delete operations with static scrollable cursors without optimistic concurrency control and with optimistic concurrency control.

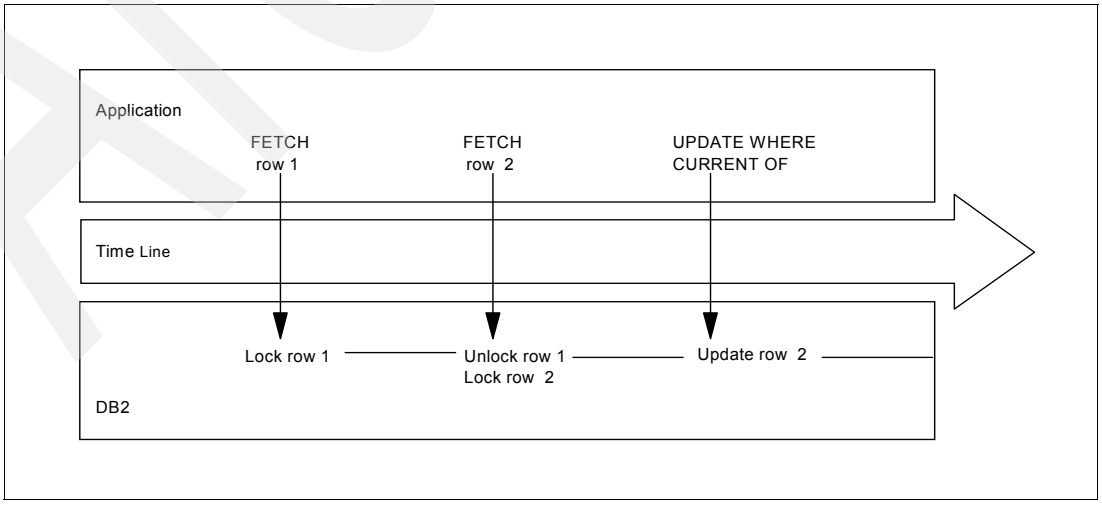


Figure 7-3 Positioned updates and deletes without optimistic concurrency control

Without optimistic concurrency control, the lock taken at the first FETCH is held until the next FETCH. The lock taken at the last FETCH is held until commit/rollback or the end of the transaction.

Optimistic concurrency control consists of the following:

- When the application requests a fetch operation to position the cursor on a row, DB2 performs the following steps:
  a. Locks that row
  b. Executes the FETCH
  c. Releases the lock
- When the application requests a positioned update or delete operation on the row, DB2 performs the following steps:
  a. Locks the row
  b. Reevaluates the predicate to ensure that the row still qualifies for the result table

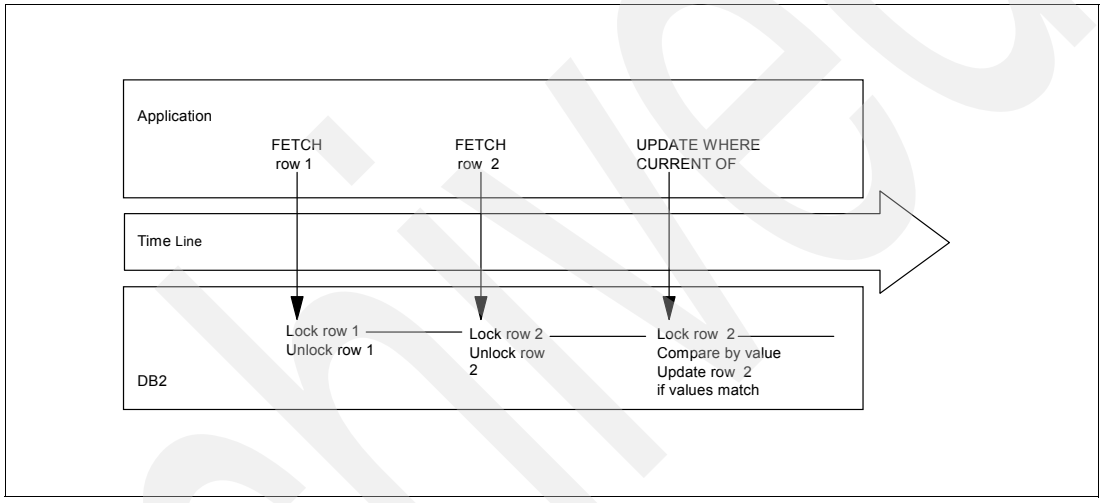


Figure 7-4 Positioned updates and deletes with optimistic concurrency control

7.3.2 Determining when a row was changed

If a table has a ROW CHANGE TIMESTAMP column, you can determine when a row was changed. You can include a ROW CHANGE TIMESTAMP column in a table either with the CREATE TABLE or ALTER TABLE statement, using the FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP clause.

This clause specifies that the column is a time stamp and the values are generated by DB2. DB2 generates a value for the column for each row as a row is inserted, and for any row for which any column is updated. The value that is generated for a ROW CHANGE TIMESTAMP column is a time stamp that corresponds to the time of the insert or update of the row. If multiple rows are inserted or updated with a single statement, the value of the ROW CHANGE TIMESTAMP column might be different for each row. You must specify NOT NULL with the ROW CHANGE TIMESTAMP column.

When you add a ROW CHANGE TIMESTAMP column to an existing table, the initial value for existing rows is not stored at the time of the ALTER statement. DB2 places the table space in an advisory-REORG pending state. When the REORG utility is subsequently run, DB2 generates the values for the ROW CHANGE TIMESTAMP column in all existing rows and then removes the advisory-REORG pending status. These values will not change unless the row is updated.

Archived

Example

Create a table EMP_INFO that contains a phone number and address for each employee. Include a ROW CHANGE TIMESTAMP column in the table to keep track of the modification of employee information:

```
CREATE TABLE EMP_INFO
(EMPNO CHAR(6)NOT NULL,
 EMP_INFOCHANGE
  NOT NULL GENERATED ALWAYS FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP,
 EMP_ADDRESS VARCHAR(300),
 EMP_PHONENO CHAR(4),
 PRIMARY KEY (EMPNO))
```

GENERATED ALWAYS specifies that DB2 always generates a value for the column when a row is inserted into the table. This is the recommended values unless you are using data propagation.

If you specify GENERATED BY DEFAULT, DB2 generates a value for the column when a row is inserted into the table, unless an explicit value is specified.

7.3.3 Finding the rows that were changed within a specified period of time

To find the rows that were changed within a specified period of time, specify the ROW CHANGE TIMESTAMP expression in the predicate of your SQL statement.

Example 1

Suppose that the EMP_INFO table has a ROW CHANGE TIMESTAMP column and that you want to return all of the rows that have changed in the last 30 days. The following query returns all of those rows:

```
SELECT * FROM EMP_INFO
WHERE ROW CHANGE TIMESTAMP FOR EMP_INFO <= CURRENT TIMESTAMP AND
ROW CHANGE TIMESTAMP FOR EMP_INFO >= CURRENT TIMESTAMP - 30 days
```

Example 2

Suppose that you want to return all of the rows in the EMP_INFO table that have changed since 9:00 a.m. January 1, 2006. The following query returns all of those rows:

```
SELECT * FROM EMP_INFO
WHERE ROW CHANGE TIMESTAMP FOR EMP_INFO >= '2006-01-01-09.00.00'
```

Example 3

Suppose that you issue the following statements to create, populate, and alter a table:

```
CREATE TABLE T1 (C1 INTEGER NOT NULL);

INSERT INTO T1 VALUES (1);

ALTER TABLE T1 ADD COLUMN C2 NOT NULL
  GENERATED ALWAYS FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP;
```

If a qualifying row does not have a value for the ROW CHANGE TIMESTAMP column, DB2 returns the time that the page in which that row resides was updated. Because the ROW CHANGE TIMESTAMP column was added after the data was inserted, the following statement returns the time that the page was last modified:

```
SELECT T1.C2 FROM T1 WHERE T1.C1 = 1;
```


Archived

Assume that you then issue the following statement:

```
INSERT INTO T1(C1) VALUES (2);
```

Assume that this row is added to the same page as the first row. The following statement returns the time that value 2 was inserted into the table:

```
SELECT T1.C2 FROM T1 WHERE T1.C1 = 2;
```

Because the row with value 1 still does not have a value for the ROW CHANGE TIMESTAMP column, the following statement returns the time that the page was last modified, which in this case is the time that value 1 was inserted:

```
SELECT T1.C2 FROM T1 WHERE T1.C1 = 1;
```

Note that if the table does not have a ROW CHANGE TIMESTAMP column, DB2 returns all rows on each page that has had any changes within the given time period. In this case, your result set can contain rows that have not been updated in the given time period if other rows on that page have been updated or inserted.

7.3.4 Considerations for tables containing a ROW CHANGE TIMESTAMP column

A ROW CHANGE TIMESTAMP column that is defined as GENERATED ALWAYS should not be specified in an INSERT statement unless the OVERRIDING USER VALUE clause is specified to indicate that any specified value is ignored and a unique system-generated value is inserted.

The correct INSERT statements for the sample table above are:

- In case the column is omitted in the change

```
INSERT INTO EMP_INFO (EMPNO, EMP_ADDRESS, EMP_PHONENO)
VALUES(
'000001','ADDRESS 000001', '0001');
```

- In case the column is included in the change

```
INSERT INTO EMP_INFO
OVERRIDING USER VALUE
VALUES(
'000001','2007-04-17-15.56.12.906693',
'ADDRESS 000001', '0001');
```

You can also specify IMPLICITLY HIDDEN, which indicates that the column is not visible in the result for SQL statements unless you explicitly refer to the column by name. For example, assuming that the table T1 includes a column that is defined with the IMPLICITLY HIDDEN clause, the result of a SELECT * would not include the implicitly hidden column. However, the result of a SELECT statement that explicitly refers to the name of the implicitly hidden column would include that column in the result table. IMPLICITLY HIDDEN must not be specified for all columns of a table.

To create an index that refers to a ROW CHANGE TIMESTAMP column in the table, values must already exist in the column for all rows. Values are stored in ROW CHANGE TIMESTAMP columns whenever a row is inserted or updated in the table. If the ROW CHANGE TIMESTAMP column is added to an existing table that contains rows, the values for the ROW CHANGE TIMESTAMP column are not materialized and stored at the time of the ALTER TABLE statement. Values are materialized for these rows when they are updated or when a REORG or a LOAD REPLACE utility is run on the table or table space.

When a table is identified in the LIKE clause and the table contains a ROW CHANGE TIMESTAMP column, the corresponding column of the new table inherits only the data type of the ROW CHANGE TIMESTAMP column. The new column is not considered as a generated column.

You cannot use the ROW CHANGE TIMESTAMP column with temporary tables.

For the LOAD utility, the ROW CHANGE TIMESTAMP column has similar restrictions and characteristics as ROWID columns.

7.4 Cultural sort and case insensitive comparison

This enhancement addresses the following requirements:

- In national languages such as Czech, Slovak, Hungary, and so on, national alphabets have special two character characters that are not correctly sorted in the current DB2 sorting algorithm. For example, in the Czech language the character *ch* is considered as one character (not two) and is sorted in this order:

  a....b....c.....d.....e.....f.....g.....h, ch, i.....j.....

  and not:

  a....b....c.....ch.....d..... (as it is now in DB2 sort result)

  So the letter ch should be correctly located between *h* and *i* and not between *c* and *d*.

- As the Web is becoming more integrated into the mainframe architecture and data is stored in multiple platforms, the old upper case displays are becoming less desirable for the business and end users. From a usability and ease-of-use standpoint the preference is to see data in mixed case. This is an issue when the character data displayed is stored on the host and is part of a query predicate. In order to maximize query efficiency it is best practice to store all data in upper case. This makes the searches easy and indexable but makes it very difficult to then reformat the display into a mixed format. Searching on mixed case data is very problematic since a like predicate and a scalar function are needed, which makes the access path anything but optimal. This enhancement makes it possible to store data as mixed case and run case insensitive searches, which do not require table scans to return the results. For example, Minnesota, MINNESOTA, and minnesota are returned by a query without having to do something like where upper(:hv) = upper(column).

7.4.1 z/OS support for Unicode

DB2 uses the z/OS support for Unicode for this purpose. z/OS support for Unicode offers character conversion, case conversion, normalization, and collation. Within character conversion, characters are converted from one coded character set identifier (CCSID) to another. Case conversion allows conversion to upper or lower case. Normalization allows the decomposition or composition of Unicode encoding input.

Collation® allows for culturally correct comparisons between two Unicode strings. It can also provide a sort key for one or two input Unicode strings for later use in binary comparisons. z/OS support for Unicode consists of two main components:

- The infrastructure that provides the conversion environment
- The conversion services that use the conversion environment

For more information refer to the publication *z/OS Support for Unicode: Using Conversion Services*, SA22-7649.

7.4.2 Collation

Collation is the general term for the process and function of determining the sorting order of strings of characters. It is a key function in computer systems. Whenever a list of strings is presented to users, they are likely to want it in a sorted order so that they can easily and reliably find individual strings. Thus, it is widely used in user interfaces. It is also crucial for the operation of databases, not only in sorting records but also in selecting sets of records with fields within given bounds.

However, collation is not uniform. It varies according to language and culture. Germans, French, and Swedes sort the same characters differently. It may also vary by specific application. Even within the same language, dictionaries may sort differently from phone books or book indexes. For non-alphabetic scripts such as East Asian ideographs, collation can be either phonetic or based on the appearance of the character. Collation can also be commonly customized or configured according to user preference, such as ignoring punctuation or not, putting uppercase before lowercase (or lowercase before uppercase), and so on.

Linguistically correct searching also needs to use the same mechanisms. Just as *v* and *w* sort as though they were the same base letter in Swedish, a loose search should pick up words with either one of them.

It is important to ensure that collation meets user expectations as fully as possible. For example, in the majority of Latin languages, ø sorts as an accented variant of o, meaning that most users would expect ø alongside o. However, there are a few languages (Norwegian and Danish, for example) that sort ø as a unique sorting element after z. Sorting "Søren" after "Sylt" in a long list (as would be expected in Norwegian or Danish) will cause problems if the user expects ø as a variant of o. A user will look for "Søren" between "Sorem" and "Soret", will not see it in the selection, and will assume that the string is missing, although in fact it has sorted in a completely different location. In matching, the same can occur, which can cause significant problems.

To address the complexities of language-sensitive sorting, a multilevel comparison algorithm is employed. In comparing two words, for example, the most important feature is the base character, such as the difference between an A and a B. Accent differences are typically ignored if there are any differences in the base letters. Case differences (uppercase or lowercase) are typically ignored if there are any differences in the base or accents. Punctuation is variable. In some situations a punctuation character is treated like a base character. In other situations, it should be ignored if there are any base, accent, or case differences. There may also be a final, tie-breaking level, whereby if there are no other differences at all in the string, the (normalized) code point order is used.

Figure 7-5, “Comparison levels” on page 185 shows comparison levels.

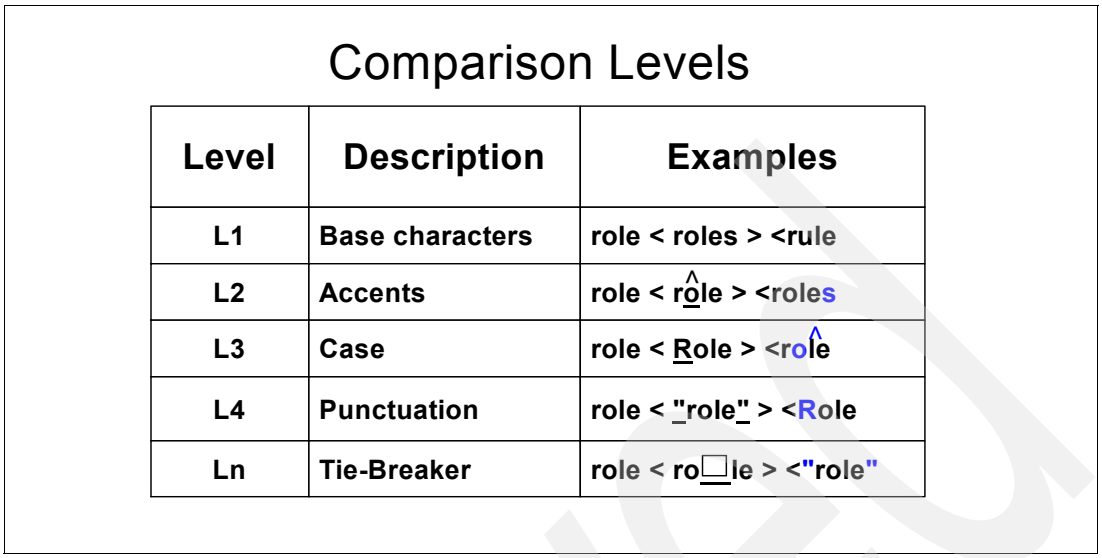


Figure 7-5 Comparison levels

These examples are in English. The levels may correspond to different features in other languages. Notice that in each example for levels L2 through Ln, the differences on that level (indicated by the underlined characters) are swamped by the stronger-level differences. For example, the L2 example shows that difference between an o and an accented o is swamped by an L1 difference (the presence or absence of an s). In the last example, the small rectangle represents a format character, which is otherwise completely ignorable.

The core concept is that the primary level (L1) is for the basic sorting of the text, and the non-primary levels (L2..Ln) are for tweaking other linguistic elements in the writing system that are important to users in ordering, but less important than the order of the basic sorting. In practice, not all of these levels may be needed, depending on the user preferences or customizations.

These levels are also referred to as primary, secondary, tertiary, and quaternary.

7.4.3 The Unicode Collation Algorithm

The Unicode Collation Algorithm (UCA) provides a specification for how to compare two Unicode strings while remaining in conformance with the requirements of The Unicode Standard. The UCA also supplies the Default Unicode Collation Element Table (DUCET), which is data specifying the default collation order for all Unicode characters. This table is designed so that it can be tailored to meet the requirements of different languages and customizations.

Briefly stated, the UCA takes an input Unicode string and a Collation Element Table, containing mapping data for characters. It produces a sort key, which is an array of unsigned 16-bit integers. Two or more sort keys so produced can then be binary-compared to give the correct comparison between the strings for which they were generated.

7.4.4 DB2 implementation

DB2 aims to provide basic national language support in countries where U.S. English is not the national language. Cultural sort is part of advanced national language support. DB2

introduces a new COLLATION_KEY function in addition to some changes to the LOWER and UPPER functions in support of Cultural sort.

7.4.5 COLLATION_KEY built-in function

Figure 7-6 shows the syntax for the COLLATION_KEY function. The schema is SYSIBM.

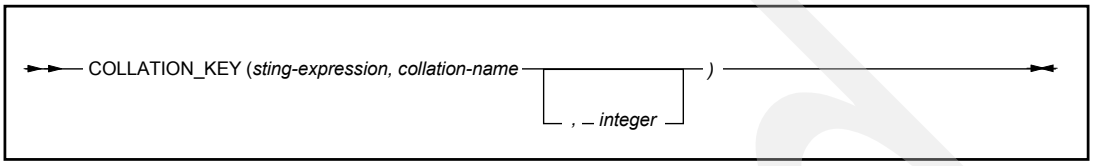


Figure 7-6 COLLATION_KEY

The COLLATION_KEY function returns a varying-length binary string that represents the collation key of the string-expression in the specified collation-name.

The result of COLLATION_KEY on one string can be compared in binary form with the result of COLLATION_KEY on another string to determine their order within the specified collation-name. For the comparison to be meaningful, the results of the COLLATION_KEY must be from the same collation-name.

string-expression

This is an expression that returns a character or graphic string that is not a LOB for which the collation key is to be determined. If string-expression is CHAR or VARCHAR, it must not be FOR BIT DATA. If string-expression is not in Unicode UTF-16 (CCSID 1200), it is converted to Unicode UTF-16 before the corresponding collation key is obtained. The length of string-expression must not exceed 32704 bytes of the UTF-16 representation.

collation-name

This is a string constant or a string host variable that is not a binary string, CLOB, or DBCLOB. collation-name specifies the collation to use when determining the collation key. If collation-name is not an EBCDIC value, it is converted to EBCDIC. The length of collation-name must be between 1 and 255 bytes of the EBCDIC representation.

Archive

The value of the host variable must not be null. If the host variable has an associated indicator variable, the value of the indicator variable must not indicate a null value. The collation-name must be left justified within the host variable. It must also be padded on the right with blanks if the length is less than that of the host variable and the host variable is a fixed length CHAR or GRAPHIC data type. The tables in Figure 7-7 show some supported values and abbreviations for the collation keywords.

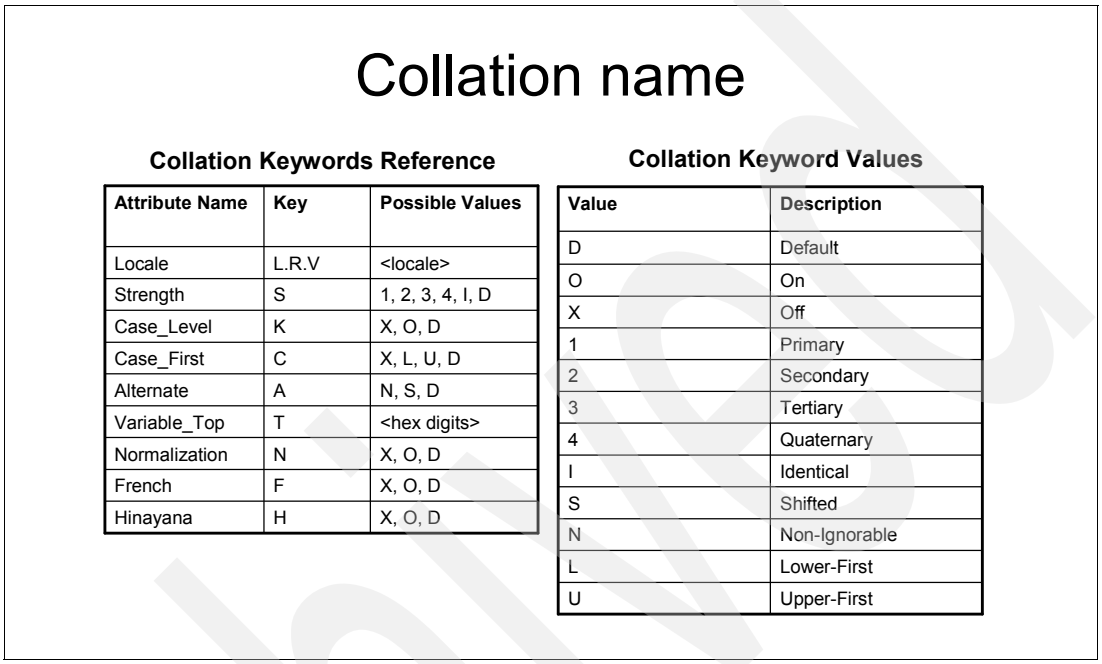


Figure 7-7 Collation name

The examples in Figure 7-8 show keywords using the above specifications.

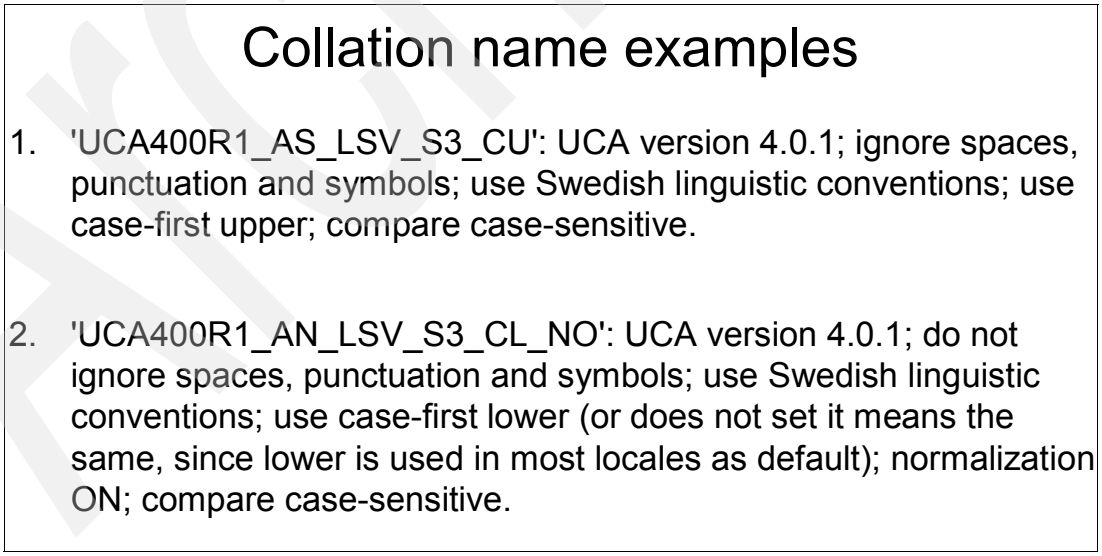


Figure 7-8 Collation name examples

integer

This is an integer value that specifies the length attribute of the result. If specified, the value must be an integer constant between 1 and 32704.

If the length is not specified, the length attribute of the result is determined as shown in Table 7-7.

Table 7-7 Determining the length attribute

string-expression	Result length attribute
CHAR(n) or VARCHAR(n)	MIN (VARBINARY(12n), 32704)
GRAPHIC(n) or VARGRAPHIC(n)	MIN (VARBINARY(12n), 32704)

Regardless of whether the length is specified, the length of the collation key must be less than or equal to the length attribute of the result. The actual result length of the collation key is approximately six times the length of string-expression, where the length of string-expression is in Unicode byte representation. For a certain collation-name such as UCA410_LKO_RKR (for Korean collation) the default length attribute of the result, 12n, might not be large enough and an error will be returned. To avoid such an error, the length attribute of the result must be explicitly specified to a larger constant. For the proper length attribute of the result, refer to the publication *z/OS Support for Unicode: Using Conversion Services*, SA22-7649, for information about target buffer length considerations for collation services.

If the first argument can be null, the result can be null. If the first argument is null, the result is the null value.

The COLLATION_KEY function uses Unicode Collation Services in z/OS to return the collation key. Unicode Collation Services support two collation versions:

- UCA400R1. This Collation version supports Unicode standard character suite 4.0.0 and uses Normalization Service under 4.0.1 Unicode character suite.
- UCA410. This Collation version supports Unicode standard character suite 4.1.0 and uses Normalization Service under 4.1.0 Unicode character suite.

If Unicode Collation Services are not available when the COLLATION_KEY function is run, an error is returned.

Example 1

The following query orders the employees by their surnames using the default Unicode Collation Algorithm V4.0.1 (UCA), ignoring spaces, punctuation, and symbols, using Swedish linguistic conventions, and not comparing case:

```
SELECT FIRSTNME, LASTNAME
  FROM DSN8910.EMP
  ORDER BY COLLATION_KEY(LASTNAME, 'UCA400R1_AS_LSV_S2');
```

Example 2

The following query uses the COLLATION_KEY function on the columns LASTNAME and SALES_PERSON to obtain the sort keys from the same collation name in order to do a culturally correct comparison. It finds the departments of employees in Quebec:

```
SELECT E.WORKDEPT
  FROM EMPLOYEE AS E INNER JOIN SALES AS S
  ON COLLATION_KEY(E.LASTNAME, 'UCA400R1_LFR') =
           COLLATION_KEY(S.SALES_PERSON, 'UCA400R1_LFR')
  WHERE S.REGION = 'Quebec';
```

Example 3

Create an index EMPLOYEE_NAME_SORT_KEY for table EMPLOYEE based on built-in function COLLATION_KEY with collation name 'UCA410_LDE' tailored for German.

```
CREATE INDEX EMPLOYEE_NAME_SORT_KEY
  ON EMPLOYEE (COLLATION_KEY(LASTNAME,'UCA410_LDE', 600),
               COLLATION_KEY(FIRSTNAME, 'UCA410_LDE', 600),
               ID);
```

Note: Unicode Collation and Normalization services were upgraded so that users can customize according to some specific collation rules. You need z/OS 1.8 or PTF UA32691for open APAR OA16037 for V1.7.

7.4.6 Extended UPPER and LOWER built-in functions

The UPPER and LOWER built-in functions, as well as the LOCALE LC_CTYPE special register, were enhanced in DB2 V8 to allow DB2 to perform casting on Unicode characters according to the Unicode 4.0 standard. Prior to this enhancement, the UPPER and LOWER functions work with Language Environment® Locales for EBCDIC data, but not for Unicode data. You must set the LOCALE LC_CTYPE special register to get the enhanced behavior.

The argument of the UPPER or LOWER function is a string. The alphabetic characters of the argument are translated to upper case characters (if function UPPER is used) or lowercase characters (if function LOWER is used) based on the value of the LC_CTYPE locale in effect for the statement.

The UPPER and LOWER functions are extended in V9 to include optionally the specification of the locale name and length attribute of the result. Some examples of locales include Fr_BE, Fr_FR@EURO, En_US, and Ja_JP. For information about locales and their naming conventions, refer to the publication *z/OS C/C++ Programming Guide*, SC09-4765. For Unicode data, there are three options for locale-name:

- Blank - simple casting on A–Z, a-z, and full-width Latin lowercase letters a-z and full-width Latin uppercase letters A-Z. Characters with diacritics are not affected.
- "UNI" - If the value "UNI" is specified, casting will use both the "NORMAL" and "SPECIAL" casing capabilities as described in z/OS Support for Unicode: Using Conversion Services.
- A locale - In this case, locale-specific casing will be performed using the "LOCALE" casing capabilities as described in z/OS Support for Unicode: Using Conversion Services.

If locale name is not specified, the locale is determined by special register CURRENT LOCALE LC_CTYPE, as in DB2 V8. If the UPPER or LOWER function is referenced in an index that is based on an expression, locale-name must be specified.

7.5 CURRENT SCHEMA

DB2 V8 introduced CURRENT SCHEMA special register and related SET CURRENT SCHEMA statement. The restriction in DB2 V8 is that the CREATE statements are not allowed when the value of the CREATE SCHEMA is different from the value in CURRENT SQLID.

DB2 V9 removes this restriction.

The CURRENT SCHEMA, or equivalently CURRENT_SCHEMA, special register specifies the schema name used to qualify unqualified database object references in dynamically prepared SQL statements. The data type is VARCHAR(128).

For information about when the CURRENT SCHEMA is used to resolve unqualified names in dynamic SQL statements and the effect of its value, see 7.5.1, “Unqualified alias, index, JAR, sequence, table, trigger, and view names” on page 190.

The CURRENT SCHEMA special register contains a value that is a single identifier without delimiters.

The initial value of the special register is the value of CURRENT SQLID at the time the connection is established. The initial value of the special register in a user-defined function or procedure is inherited from the invoking application (initial value when INHERIT SPECIAL REGISTERS option is specified) or the value of CURRENT SCHEMA when the routine is entered (initial value when DEFAULT SPECIAL REGISTERS option is specified). The routine can also use the SET statement to modify.

The value of the special register can be changed by executing the SET SCHEMA statement. The value of CURRENT SCHEMA is the same as the value of CURRENT SQLID unless a SET SCHEMA statement has been issued specifying a different value. After a SET SCHEMA statement has been issued in an application, the values of CURRENT SCHEMA and CURRENT SQLID are separate. Therefore, if the value of CURRENT SCHEMA needs to be changed, a SET SCHEMA statement must be issued.

Example: Set the schema for object qualification to ’D123’:

```
SET SCHEMA = 'D123'
```

The one-part and two-part names in every FROM clause and the one-part and two-part qualifiers of column names are expanded into a fully qualified form.

For example, if a dynamic SQL statement uses FROM Q, and DYNAMICRULES run behavior (RUN) is in effect, Q is expanded to S.A.Q, where S is the value of CURRENT SERVER and A is the value of CURRENT SCHEMA. If DYNAMICRULES bind behavior (BIND) is in effect instead, A is the plan or package qualifier as determined during the bind process or the qualifier for the native SQL procedure as determined when the procedure was defined. This step is later referred to as *name completion*. An error occurs if the first part of every name (the location) is not the same.

7.5.1 Unqualified alias, index, JAR, sequence, table, trigger, and view names

Unqualified alias, index, JAR, sequence, table, trigger, and view names are implicitly qualified by the default schema. The default schema is determined as follows:

- For static SQL statements, the default schema is the identifier specified in the QUALIFIER option of the BIND subcommand or the CREATE PROCEDURE or ALTER PROCEDURE statement (for a native SQL procedure). If this option is not in effect for the plan, package, or native SQL procedure, the default schema is the authorization ID of the owner of the plan, package, or native SQL procedure.
- For dynamic SQL statements, the behavior as specified by the combination of the DYNAMICRULES option and the run-time environment determines the default schema.
  - If run behavior applies, the default schema is the schema in the CURRENT SCHEMA special register. Run behavior is the default.
  - If bind behavior applies, the default schema is the identifier that is implicitly or explicitly specified in the QUALIFIER option, as explained above for static SQL statements.

– If define behavior applies, the default schema is the owner of the function or stored procedure (the owner is the definer).
– If invoke behavior applies, the default schema is the authorization ID of the invoker of the function or stored procedure.

Exception: For bind, define, and invoke behavior, the default schema of PLAN_TABLE, DSN_STATEMNT_TABLE, and DSN_FUNCTION_TABLE (output from the EXPLAIN statement) is always the value in special register CURRENT SQLID.

7.5.2 Summary

Figure 7-9 summarizes how CURRENT SCHEMA works in DB2 V9.

For alias, auxiliary table, created global temporary table, table and view

Object being created	Schema Qualifier of new object	Owner of new object
name	Value of CURRENT SCHEMA	Value of CURRENT SQLID
qualifier.name	qualifier	qualifier
…qualifier.name	qualifier	qualifier

For user-defined distinct type, user-defined function, procedure, sequence or trigger: SQL PATH is used to determine the schema name, same as V8.

Object being created	Schema Qualifier of new object	Owner of new object
name	Value of CURRENT SCHEMA	Value of CURRENT SQLID
qualifier.name	qualifier	Value of CURRENT SQLID
…qualifier.name	qualifier	Value of CURRENT SQLID

For most objects CURRENT SCHEMA value is the qualifier for dynamically CREATEd objects for which no qualifier is specified on the CREATE. CURRENT SQLID remains the object owner.

Figure 7-9 CURRENT SCHEMA in DB2 V9

7.6 Skip locked rows

The SKIP LOCKED DATA option allows a transaction to skip rows that are incompatibly locked by other transactions. Because the SKIP LOCKED DATA option skips these rows, the performance of some applications can be improved by eliminating lock wait time. However, you should use the SKIP LOCKED DATA option only for applications that can reasonably tolerate the absence of the skipped rows in the returned data. If your transaction uses the SKIP LOCKED DATA option, it does not read or modify data that is held by locks.

Important: When DB2 skips data because of the SKIP LOCKED DATA option, it does not issue a warning. Even if only a subset of the data that satisfies a query is returned or modified, the transaction completes as though no data was skipped. To use the SKIP LOCKED DATA option, you must be willing to accept inconsistent results.

Recommendation: For applications and transactions that require fast results and data that is somewhat accurate, consider using SKIP LOCKED DATA to increase concurrency. For applications and transactions that require accurate data, do not use the SKIP LOCKED DATA option.

To use the SKIP LOCKED DATA option, specify the clause in a SELECT, SELECT INTO, PREPARE, searched UPDATE, or searched DELETE statement. You can also use the SKIP LOCKED DATA option with the UNLOAD utility.

Isolation levels for SKIP LOCKED DATA

You can use the SKIP LOCKED DATA option with cursor stability (CS) isolation and read stability (RS) isolation. However, you cannot use SKIP LOCKED DATA with uncommitted read (UR) or repeatable read (RR) isolation levels. For UR and RR, DB2 ignores the SKIP LOCKED DATA clause.

Lock sizes for SKIP LOCKED DATA

The SKIP LOCKED DATA option works only with row locks and page locks. If you specify SKIP LOCKED DATA for a transaction with row level locking, incompatibly locked rows are skipped. If you specify SKIP LOCKED DATA for a transaction with page level locking, all rows on pages with incompatible locks are skipped.

In general, the SKIP LOCKED DATA clause does not apply to table, partition, LOB, XML, or table space locks. When LOCKSIZE TABLE or LOCKSIZE TABLESPACE is specified for a table space or when a lock is escalated to a gross table, partition, or table space lock, DB2 ignores the SKIP LOCKED DATA clause.

Lock mode compatibility for SKIP LOCKED DATA

Lock mode compatibility for transactions that use the SKIP LOCKED DATA option is the same as lock mode compatibility for other page-level and row-level locks. However, when incompatible locks are held, a transaction that uses the SKIP LOCKED DATA option does not wait for the locks to be released and skips the locked data instead.

Example

Suppose that application A holds an s lock on a row that process B also wants to access. The query in process B specifies SKIP LOCKED DATA. The outcome of process B depends on the mode of lock that it acquires. If process B requires a compatible s or u lock, process B can access the row on which application A holds an s lock. If process B requires an incompatible x lock, process B cannot access the locked row. Because the SKIP LOCKED DATA option is specified, that row is skipped and the results of process B are returned without that row.

Examples of results with SKIP LOCKED DATA

With some queries that use the SKIP LOCKED DATA clause, you can receive unexpected or inconsistent results.

Archive

Example

Suppose that a table EXTABLE exists in a table space with row-level locking, or in a table space with page-level locking, and the rows of the table are distributed across several pages. EXTABLE has two columns, C1 and C2, that contain the following data:

```
C1   C2
1    AAAA
2    BBBB
3    CCCC
4    DDDD
```

Next, suppose that a transaction issues the following update statement:

```
UPDATE EXTABLE SET C1 = 99 WHERE C1 < 3;
```

Suppose that a second transaction issues the following SELECT statement before the previous transaction commits:

```
SELECT COUNT (*) FROM EXTABLE WHERE C2>='AAAA' SKIP LOCKED DATA;
```

If you do not have an index defined on C2, DB2 returns a count of 2 because DB2 skips the two rows that are locked by the uncommitted UPDATE statement.

However, you cannot always expect DB2 to skip this data.

Attention: The difference between uncommitted data and skip locked rows is the following:

- In the case of uncommitted data, uncommitted rows can show up.
- In the case of skip locked rows, only committed rows show up, although not all committed rows may show up.

7.7 LOB file reference

Multimedia application environments rely on many types of large data objects. Those objects can be large text documents, X-ray images, audio messages, pictures, and many other types of images.

The data types provided by DB2, such as VARCHAR or VARBINARY (V9), are not large enough to hold a large amount of data because of their limit of 32 KB. Support for large data objects (LOBs) is based on the set of data types that were introduced with DB2 V6.

DB2 provides LOBs that support strings of up to 2 GB in size, well beyond the 32 KB supported by a varchar column. Techniques for storing and retrieving these huge amounts of data have also been created within DB2.

The LOB data types allow you to store directly in DB2 tables objects in size of up to 2 GB, and 65,536 Terabytes (TB) per LOB column. Their characteristics depend on the way they are stored in your DB2 subsystem:

- Character Large Objects (CLOBs)
- Binary Large Objects (BLOBs)
- Double-byte Character Large Objects (DBCLOBs)

For its structure support, DB2 uses the ROWID data type.

DB2 also provides host variables and data types that are used for manipulating LOBs:

- LOB locators
- LOB file reference variables

Generally, non-LOB columns are stored in a base table. LOBs belong to a base table and are related to it, but they are not stored in the same table with the other non-LOB columns, they are stored in a LOB table space. The table, where the rows of a LOB column live and that is embedded in the LOB table space, is called an auxiliary table.

Each LOB column in a base table requires its own LOB table space, so LOB columns are also separated from other LOB columns belonging to the same base table. Each LOB table space and auxiliary table contains the values of the same LOB column. It is important to know that each page can only contain a LOB or a portion of a LOB. It never contains values of two LOBs or two LOB columns.

If the base table is partitioned, every LOB column in every partition has its own LOB table space. This means up to 4,096 LOB table spaces per LOB column if the base table has 4,096 partitions.

The LOB column itself is referred to as an auxiliary column, because it is not stored in the base table. The rows in the base table are associated with the LOB columns residing in the auxiliary table in the LOB table space using the ROWID as a pointer from the base table to the auxiliary table. In order to quickly locate the proper LOB value in the auxiliary table, an auxiliary index must be created on the auxiliary table. This index includes the ROWID as its only column. Additionally, a LOB indicator is also stored in the base table.

Figure 7-10 shows a summary picture of all the objects mentioned here and how they work together.

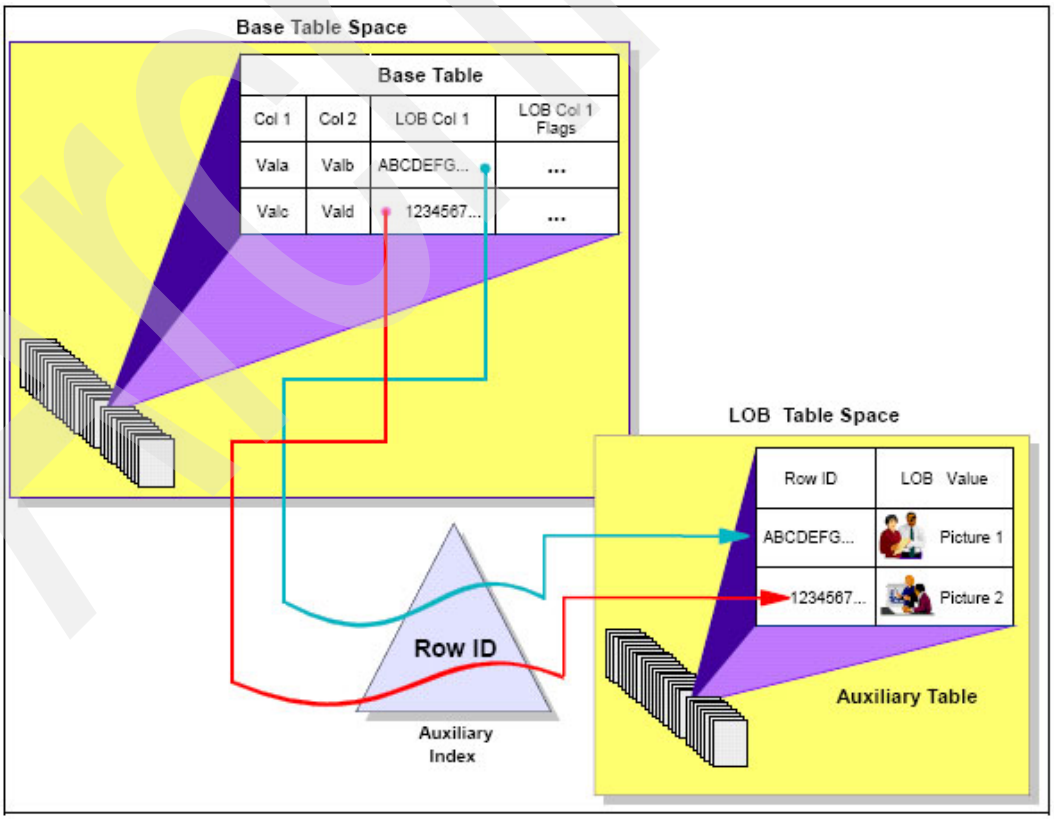


Figure 7-10 Association between base table, ROWID, auxiliary table, and LOB table space

An important reason for the separation of LOB columns is performance. Assuming table space scans on the base table, LOB values do not have to be processed during these scans. Probably most of the scanning time would be spent in scanning LOB columns if the LOB columns resided in the same table as the non-LOB columns.

Based on the nature of LOBs, they can be larger than the biggest available page size, which is still 32 KB in DB2 9 and is also valid for LOB table spaces. Therefore, a single LOB value can span pages.

Because each data set of a LOB table space can grow up to 64 GB, and there can be up to 254 data sets per table space, the maximum size of a non-partitioned LOB table space is 16,256 GB (16 TB) in total. Because the number of partitions can grow up to 4,096 in the base table, there are 4,096 LOB table spaces, each holding up to 16 TB of data as a maximum size. This gives a grand total of 4,096 x 16 TB = 65,536 TB available for a single column of LOB data.

A single database can hold a maximum of 65,535 objects, or to be more specific, X'FFFF' object identifiers (OBDs).

Regardless of the number of partitions, you use one OBD for each auxiliary relationship per LOB column. Furthermore, DB2 uses five OBDs per LOB column per partition. Therefore, this gives us a maximum of three LOB columns for a 4,096 partition table, not exceeding 65,535 OBDs. According to these values:

```
3 + n + 5np <= 65535
```

with *n* as the number of LOB columns in your base table and p as the number of partitions. The formula gives you the number of partitions and LOB columns that can reside inside one database.

The values are summarized in Table 7-8.

Table 7-8 Maximum number of LOB columns by partitions

Partitions	Data sets	Maximum number of LOB columns
250	12,250	48
1,000	13,000	12
4,096	16,384	3

7.7.1 LOB locators

In general, when the application is accessing data, DB2 has to deal with the materialization of data on the layers involved: auxiliary storage, main storage, data buffers, and the movement of data across them. The materialization of LOB values consists of placing LOB data in contiguous storage. Because LOB values can be very large, DB2's actions are inspired by the objective of avoiding materialization of LOB data until it becomes absolutely necessary. Furthermore, if the application is running on a client machine, distinct from the database server, the transfer of LOB values from the server to the client adds a sizable time and resource consumption. Application programs typically process LOB values a piece at a time, rather than as a whole. For all of those cases where an application does not need (or want) the entire LOB value stored in application memory, the application can reference a LOB value using the large object locator (LOB locator) and avoid materialization.

A LOB locator is a host variable with a value that represents a single LOB instance in the database server. You can consider locators roughly as cursors that reference remote data.

Locators, and the structures associated to them, are contained in virtual storage blocks allocated by DB2 in the DBM1 address space. The LOB data referenced by the locators is allocated by DB2 in memory structures, which are dynamically created for the duration of the request.

7.7.2 Purpose of LOB locators

A LOB locator is a value generated by DB2 when a LOB string is assigned to a host variable that was previously specified as a LOB locator in your application program. Every LOB locator, also called host-identifier, is a special type of SQL object materialized as a 4-byte token used to represent the LOB. This representation allows the value of the large object strings to be contained in the host-identifier, rather than the actual string of bytes of the large object.

Figure 7-11 provides an example of LOB locator usage.

A LOB locator is an association between a host variable and its corresponding LOB value in DB2 at a point in time. The application only accesses the locator while the entire LOB value resides in DB2 and is not propagated to the application program. Using this technique, an application does not need to acquire a buffer large enough to contain the entire LOB value. Instead, the application deals only with LOB locators, therefore largely reducing the amount of resources to be allocated. The definition and usage are not mandatory. However, considerations on performance soon lead you to the conclusion that it might be a better idea for the applications to use them, because locators dramatically reduce the movement of data between the different address spaces involved in LOB data management, as well as greatly reducing connectivity issues.

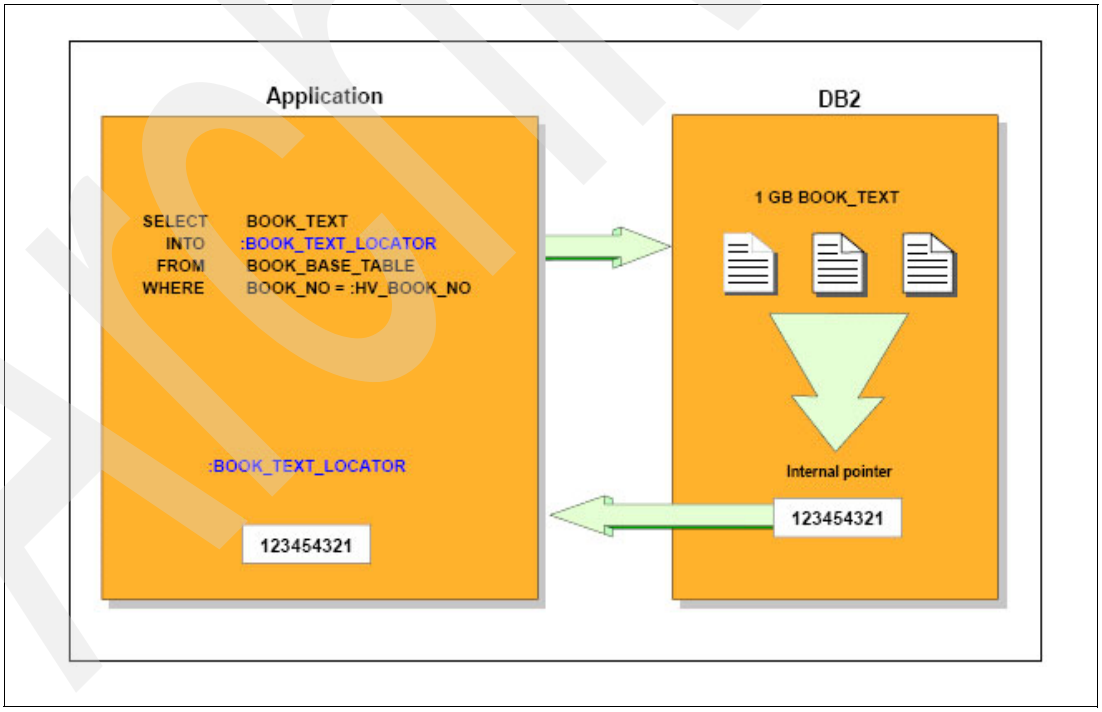


Figure 7-11 Assigning a LOB locator for a LOB value

7.7.3 Different types of LOB locators

Each LOB format (BLOB, CLOB, and DBCLOB) has its own type of locator, so currently there are three different types of LOB locators:

- Binary Large Object locator, which is associated with large binary strings
- Character Large Object locator, which is associated with large character strings
- Double-byte Character Large Object locator, which is associated with large graphic strings

The definition of a LOB locator depends on the language you choose for developing programs for LOB usage. Example 7-1 shows the syntax for defining LOB locators in COBOL.

Example 7-1 Host variable definitions for LOB locators in COBOL

```
01 BLOB-LOCATOR USAGE IS SQL TYPE IS BLOB-LOCATOR.
01 CLOB-LOCATOR USAGE IS SQL TYPE IS CLOB-LOCATOR.
01 DBCLOB-LOCATOR USAGE IS SQL TYPE IS DBCLOB-LOCATOR.
```

The DB2 precompiler converts the locator structure into the COBOL structure, as reported in Example 7-2.

Example 7-2 What the DB2 precompiler makes of LOB locators

```
01 BLOB-LOCATOR PIC S9(9) COMP.
01 CLOB-LOCATOR PIC S9(9) COMP.
01 DBCLOB-LOCATOR PIC S9(9) COMP.
```

The locator 4-byte value is then stored in a host variable. The program, as already shown in Figure 7-11 on page 196, can use it to refer to a LOB value. Even if every LOB locator shows up identically for all definitions of host variables, DB2 knows the associated LOB type and does not let you use a locator with a different type of LOB. If you define a CLOB locator and try to use it for a BLOB, SQLCODE -171 is issued.

You can only use LOB locators inside an application program. You cannot deal with them interactively. This means that you cannot use them, for instance, with SPUFI or QMF.

LOB locator disadvantages

There are some disadvantages for using LOB locators. Their coding is pretty cumbersome, and in the long run, they can cause problems in the application. The code becomes complicated and hard to maintain. The need to chain the locators when dealing with extremely large objects and limited storage in a user address space can cause the program to abend due to lack of storage in a user address space.

When no complicated manipulations on LOBs are required, or the application is not for usage in a distributed environment, we recommend that you program using LOB file reference variables introduced in DB2 V9.

7.7.4 LOB file reference variables

The purpose of file reference variables is to import or export data between a LOB column and an external file outside of the DB2 system. In the past, if you used a host variable to materialize the entire LOB in the application, your application would not only need adequate storage but would also incur poor performance, because the file I/O time would not be overlapped with any DB2 processing or network transfer time.

Archived

Locator variables used in conjunction with the SUBSTR function can be used to overlap the file I/O time with DBM1 processing time or network transfer time and also to avoid materializing the whole LOB in the application. However, there is still some CPU overhead to transfer pieces of the LOB between DBM1 and the application.

LOB file reference variables accomplish the same function using less CPU time and avoiding the use of any application storage for the LOBs. LOB file references are also easier to use than locator variables. LOB file reference variables are supported within a DB2 for z/OS system or in a distributed configuration between DB2 for z/OS subsystems.

In DB2 V9, three new SQL host variables have been introduced:

- BLOB_FILE
- CLOB_FILE
- DBCLOB_FILE

File reference host variables can be used in applications to insert a LOB from a file into a DB2 table or to select a LOB from a DB2 table into a file. They can be used to update a LOB from a file as well. When you use a file reference variable, you can select or insert an entire LOB value without contiguous application storage to contain the entire LOB. In other words, LOB file reference variables move LOB values from the database server to an application or from an application to the database server without going through the application's memory. Furthermore, LOB file reference variables bypass the host language limitation on the maximum size allowed for dynamic storage to contain a LOB value.

7.7.5 DB2-generated file reference variable constructs

For each LOB file reference variable that an application declares, DB2 generates an equivalent construct that uses the host language data types. When an application refers to a LOB file reference variable, the application must use the equivalent construct that DB2 generates. If the construct is not used, the DB2 precompiler issues an error. The construct describes properties of the file, as shown in Table 7-9.

Table 7-9 DB2-generated construct

Data type	BLOB, CLOB, or DBCLOB. This property is specified when the variable is declared using the BLOB_FILE, CLOB_FILE, or DBCLOB_FILE data type.
File name	This property must be specified by the application program at run time. The file name property can have the following values: ► The complete path name of the file. We recommend this. ► A relative file name. If a relative file name is provided, it is appended to the current path of the client process. A file should be referenced only once in a file reference variable.
File name length	This property must be specified by the application program at run time.

File options	An application must assign one of the file options to a file reference variable before the application can use that variable. File options are set by the INTEGER value in a field in the file reference variable construct. One of the following values must be specified for each file reference variable: ► Input (from application to database): SQL-FILE-READ. A regular file that can be opened, read, and closed. ► Output (from database to application): SQL-FILE-CREATE. If the file does not exist, a new file is created. If the file already exists, an error is returned. SQL-FILE-OVERWRITE. If the file does not exist, a new file is created. If the file already exists, it is overwritten. SQL-FILE-APPEND. If the file does not exist, a new file is created. If the file already exists, the output is appended to the existing file.
Data length	The length, in bytes, of the new data written to the file, provided by the DB2 server.

7.7.6 Language support for LOB file reference variables

In this section we examine options when using file reference variables.

Encoding scheme

The encoding scheme CCSID of the file name is based on the application's encoding scheme. The CCSID of the LOB (contents of the file) can be set by the application by using the SQL DECLARE host variable CCSID statement if it is different from the application encoding scheme. DB2 performs any character conversion required prior to inserting LOB data into a DB2 table or placing a LOB into a file.

Programming support for LOB file reference variables

You can declare a LOB file reference variable or a LOB file reference array for applications that are written in C, C++, COBOL, PL/I, Assembler, and REXX. No implementation for file reference variables is required in JCC type 2 or type 4 drivers. Both JCC Type 2 and Type 4 drivers have the capability to read a file to be used in its *stream* (byte/character) methods. The LOB file reference variables do not contain LOB data. They represent a file that contains LOB data. Database queries, updates, and inserts can use file reference variables to store or retrieve column values. As with other host variables, a LOB file reference variable can have an associated indicator variable.

The definition of a LOB file reference variable depends on the language you choose for developing programs for LOB usage. Example 7-3 shows the syntax for defining LOB file reference variables in COBOL.

Example 7-3 Host variable definition for BLOB file reference variable in COBOL

```
01 MY-BLOB-FILE SQL TYPE IS BLOB-FILE.
```

The DB2 precompiler converts the declaration into the COBOL structure, as reported in Example 7-4.

Example 7-4 What the DB2 precompiler makes of BLOB file reference variable

```
01 MY-BLOB-FILE.
49 MY-BLOB-FILE-NAME-LENGTH PIC S9(9) COMP-5.
49 MY-BLOB-FILE-DATA-LENGTH PIC S9(9) COMP-5.
49 MY-BLOB-FILE-FILE-OPTION PIC S9(9) COMP-5.
49 MY-BLOB-FILE-NAME PIC X(255).
```

Table 7-10 shows the precompiler generated file option constant declarations. You can use these constants to set the file option variable when you use file reference host variables.

Table 7-10 File option constants

Constant name	Constant value
SQL-FILE-READ	2
SQL-FILE-CREATE	8
SQL-FILE-OVERWRITE	16
SQL-FILE-APPEND	32

NULL indicator support for file reference variables

Like all other host variables, a LOB file reference variable can have an associated indicator variable. Indicator variables for LOB file reference behave differently from indicator variables of normal host variables. Because the file name can never be NULL for either input or output, a negative indicator variable value indicates that the LOB value represented by the file reference variable is NULL.

When a NULL value of a LOB column is returned from the database, the indicator variable is set and the file associated with the variable is not opened by DB2.

When a NULL value is set by an application to place a LOB file reference into a DB2 LOB column, the file is not opened for writing.

ODBC API support of LOB file reference

Two new APIs are added to support LOB file reference.

The new APIs are:

- SQLBindFileToCol

  SQLBindFileToCol is used to bind a LOB column in a result set to a file reference allowing the column data to be transferred into a file when the row is fetched.

- SQLBindFileToParam

  SQLBindFileToParam is used to bind a parameter marker to a file reference allowing the data from the file to be transferred into a LOB column.

An ODBC application can use either the statement attributes or the keyword CURRENTAPPENSCHEM to override the default CCSID setting. If both are specified, the statement attributes override the setting of CURRENTAPPENSCHEM in the INI file. ODBC converts the input file name to the application's encoding scheme before passing it to DB2 for processing.

File local/client support

The DB2 support of the LOB file reference is on the local side (non-distributed environment) or on the client side (distributed environment) in a DB2 to DB2 (both z/OS) connection. The file referenced by a LOB file must reside on the system on which the program is running, or it must be accessible by the system on which the program is running. If a remote stored procedure that issues an SQL statement that uses a LOB file reference is called, the file must reside on the system on which the stored procedure is running or be accessible by the system on which a stored procedure is running.

A LOB file reference cannot be used in the stored procedure or user-defined function as an input or output parameter. If a LOB file reference is used in the stored procedure or user-defined function as an input or output parameter, SQLCODE -104 is issued.

DB2 9 for z/OS SQL supports sequential files (DSORG=PS), Hierarchical File System (HFS) files, Partition Data Set (PDS) files, and Partition Data Set Extended (PDSE) files.

Utilities support of LOB file reference

LOAD and UNLOAD utilities support the capability of LOB file reference variables introduced in DB2 V9, and support is retrofitted to V7 and V8 with APARs PK10278 and PK22910.

Restriction: Utilities do not support sequential (BSAM or QSAM) data sets for LOB file reference variables.

Archive

DSNZPARMs affected

DB2 9 introduces the new parameter MAXOFILR (MAX OPEN FILE REFS) to control the maximum number of data sets that can be open concurrently for processing of the LOB file reference. The value setting for the parameter appears on the installation panel DSNTIPE, as shown in Figure 7-12.

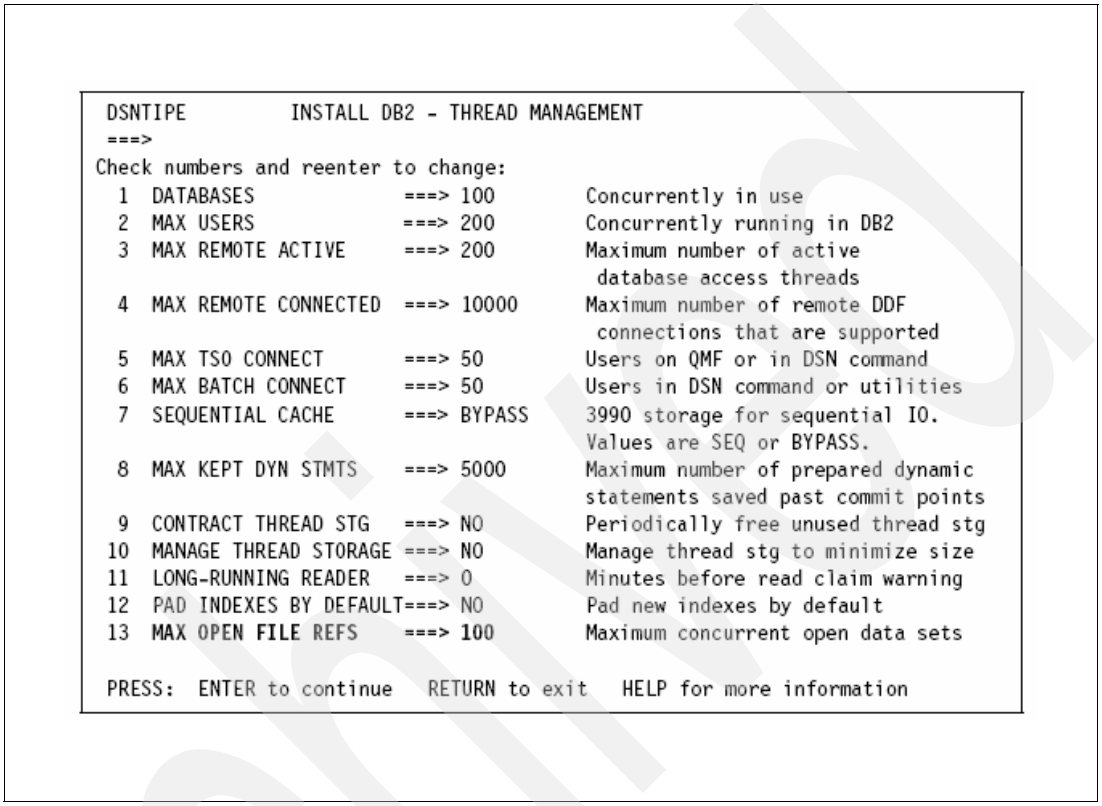


Figure 7-12 DSNTIPE installation panel

Though the default value is 100, the highest number in the range is also effectively limited to the setting of the CTHREAD (MAX USERS) parameter. This limitation exists because a thread can hold at most one file open using file references. With z/OS 1.7, DSMAX can be 64 KB and even get up to 100 KB, as noted below.

Important: APARs PK13287 and PK29281 and z/OS 1.7 GRS SPE allow support of up to 100,000 open data sets.

Because some storage needs to be allocated for file processing, the parameter also has some effect on the consumption of storage below the 2 GB bar.

7.8 FETCH CONTINUE

This enhancement introduces extensions to the FETCH SQL statement that provide a convenient method for applications to read from tables that contain LOB or XML columns, when the actual length of the LOB or XML value is not known or is so large that the application cannot materialize the entire LOB in memory. The declared maximum length for a LOB column is frequently much larger than the typical LOBs that are inserted into those columns. Prior to this enhancement, applications that used embedded SQL to read from

tables containing LOB columns typically had to declare or allocate a piece of storage that was equal in size to the maximum defined storage size for the LOB column. This can cause a shortage of virtual memory in certain configurations. These extensions to the FETCH statement are the WITH CONTINUE clause and the CONTINUE clause.

LOB locators are one way to avoid having to pre-allocate space for the entire LOB, but they have some problems as well, including slower performance in some cases, excessive resource consumption at the server, and more complex coding.

SQL Driver programs such as the JDBC driver (when using type-2 connectivity), the native ODBC driver, or even DSNTEP2 and SPUFI that connect locally to DB2 are especially susceptible to virtual storage constraint problems when handling LOBs of unknown length, particularly in environments that support multiple concurrent user connections.

This problem is compounded in V9 with the introduction of XML objects. There is no defined maximum for XML columns. The DB2 architectural limit is 2 GB, but that limit is impractical for use in declaring or allocating a buffer.

This enhancement allows an application to do a FETCH against a table that contains LOB or XML columns, using a buffer that might not be large enough to hold the entire LOB or XML value. If any of the fetched LOB or XML columns do not fit, DB2 returns information about which columns were truncated and what the actual length is. To enable this behavior on the FETCH, the application must add the WITH CONTINUE clause. The application is then able to use that actual length information to allocate a larger target buffer, and execute a FETCH statement with the CONTINUE clause, to retrieve the remaining data for those columns. Alternatively, the application can be coded to handle large LOB or XML values by *streaming* the data. That is, it can use a fixed-size buffer and, following the initial FETCH, perform repeated FETCH CONTINUE operations to retrieve the large LOB or XML value, one piece at a time. Note that the application is not required to consume all LOB or XML objects in their entirety. It can FETCH the next row at any time, even if truncated columns remain in the current row.

7.8.1 Example using dynamic SQL

This example uses dynamic SQL and manipulation of the SQLDA by the application (as opposed to precompiler-generated SQLDA manipulation). In this example, the application uses, at most, two fetch operations to retrieve the LOB and XML values. On the first fetch operation, it fetches these columns into a moderately sized buffer. In cases where that buffer is not large enough, it receives accurate length information from DB2 so it can then allocate the appropriate amount of storage and then retrieve the remaining, unreturned data for the truncated columns. Consider the following pseudocode and description:

Assume that a table exists that was created as:

```
CREATE TABLE T1 (C1 INT, C2 CLOB(100M), C3 CLOB(32K), C4 XML);
```

And assume that a row in T1 exists where C1 contains a valid integer, C2 contains 100 MB of data, C3 contains 32 KB, and C4 contains XML.

```
[1] EXEC SQL DECLARE CURSOR1 CURSOR FOR DYNSQLSTMT1;
    EXEC SQL PREPARE DYNSQLSTMT1 FROM 'SELECT * FROM T1';
[2] EXEC SQL DESCRIBE DYNSQLSTMT1 INTO DESCRIPTOR :SQLDA;
[3] EXEC SQL OPEN CURSOR1;
[4] Prepare for FETCH:
      Allocate data buffers (32K for each CLOB, XML item)
      Set data pointers and lengths in SQLDA.
[5] EXEC SQL FETCH WITH CONTINUE CURSOR1 INTO DESCRIPTOR :SQLDA;
```

```
[6] if truncation occurred on any LOB or XML column
      loop through each column
          if column is LOB or XML and was truncated
              allocate larger buffer area for any truncated columns, move first
               piece larger area
              reset data pointers, length fields in SQLDA
          endif
      endloop
[7]   EXEC SQL FETCH CURRENT CONTINUE CURSOR1 INTO DESCRIPTOR :SQLDA
      endif
      Work with returned data
      ...
[8]  EXEC SQL FETCH WITH CONTINUE CURSOR1 INTO DESCRIPTOR SQLDA;.
[9]  EXEC SQL CLOSE CURSOR1;
```

Description:

1. The application declares a cursor for a dynamic SQL statement, then prepares a SELECT statement that retrieves LOB and XML columns of different sizes.
2. The application DESCRIBEs the statement. This populates the SQLDA with initial data type and length information.
3. The application opens the cursor.
4. The application prepares for the FETCH by allocating storage to receive each of the output columns. For the LOB and XML columns it allocates 32767 bytes. This is an arbitrary size. A larger or smaller size could be used. This example assumes that the programming language being used allows for dynamic storage allocation. The application then completes the SQLDA setup in preparation for the FETCH. It sets the SQLDATA pointers to point at each allocated buffer area and sets the SQLLONGLEN field to 32767. It can optionally set each SQLDATALEN field to point at a 4-byte length field to hold the LOB output length. For any XML columns, it must change the data type and set the "XML" bit within the SQLNAME field.
5. The application issues the FETCH request using the new WITH CONTINUE clause to tell DB2 to manage LOB and XML truncation on output differently, as described below. After the FETCH, the buffers contain the complete data for C1 (the integer) and C3 (it fits in the 32 KB buffer). Since the data for C2 and C4 is greater than 32 KB, truncation occurs for these two columns. The FETCH returns a truncation warning — SQLWARN1 is set to 'W', and SQLCODE +20141 may be returned.

Because the FETCH CONTINUE flag is on for truncated columns C2 and C4, DB2 performs the following actions:

1. The amount of data written to the data buffer equals the length specified by the SQLLONGLEN field in the secondary SQLVAR minus possibly 4 bytes depending on whether SQLDATALEN is NULL. The remaining data remains materialized (cached) at the server, and it can be retrieved by the application using FETCH CURRENT CONTINUE immediately following the current FETCH. If the data contains multi-byte characters, a partial character may result at the truncation point because the data is truncated on a byte boundary.
2. The required buffer length is reported in one of two places. If the SQLDATALEN field in the secondary SQLVAR is not NULL, it contains a pointer to a 4-byte long buffer that contains the required buffer length in bytes (even for DBCLOB). If the SQLDATALEN field is NULL, the required buffer length (in bytes for CLOB and BLOB, in characters for DBCLOB) is stored in the first 4 bytes of the buffer pointed by the SQLDATA field in the base SQLVAR. A required buffer length is the length of buffer space required to hold the entire data value.

Therefore, it includes the amount of data already written to the data buffer. For this example, assume that the SQLDATALEN pointer is not null.

3. The application checks the result of the FETCH and processes the returned data. If any data has been truncated, then the SQLWARN1 field in the SQLCA is set to 'W'. (In some cases SQLCODE +20141 is returned, but checking SQLWARN1 is the best way to check for truncation.) In this example, we know that truncation has occurred. So the application loops through each output column to find the truncated columns. For the LOB and XML columns, it does this by comparing the value pointed at by SQLDATALEN with the value in SQLLONGLEN in the corresponding secondary SQLVAR. If the SQLDATALEN value is greater, truncation has occurred. The application then uses the value pointed at by SQLDATALEN to allocate a new buffer. It then copies the first piece of data into the new buffer and resets the SQLDATA pointer to point just past that new data piece. SQLLONGLEN is then set to the new buffer length minus the length of the first chunk (32767 in this case).

   Alternatively, the application could have set the SQLDATALEN pointer to zero. In that case, the processing would be similar, except that DB2 would place the actual length into the first four bytes of that data output area pointed at by SQLDATA.

4. The application issues a FETCH CURRENT CONTINUE. DB2 then processes the SQLDA, ignores SQLVARs for columns that are neither LOB nor XML, and finds that there is data cached for C2 and C4. DB2 then writes the data to the provided host variables in the same way that it would for a normal FETCH operation, but begins at the truncation point.

   The application then processes the returned data in the data buffers. In this example, the application allocated the buffer sizes for the FETCH CURRENT CONTINUE to be successful. However, if one of the data buffers was still too small, DB2 would again set the truncation warnings and lengths as described for the FETCH WITH CONTINUE step.

5. This FETCH operation just fetches the next row of data in the result set. In this example, the application consumed all of the LOB and XML data and did not leave any truncated data. But if it had, this FETCH would make that data unavailable. The application is not required, when using this continue capability, to consume all of the truncated data. When the cursor is moved to the next row, or closed, that data is then unavailable. Steps 4 through 8 could be repeated here until the application does not request any more rows, the application closes the cursor, or in the case of non-scrolling cursors, there are no more rows of the result set available.

6. The application closes the cursor. Similarly, if there had been any truncated LOB or XML columns that had not been fully fetched, DB2 would discard the remaining data.

7.8.2 Example (using static SQL)

This example uses static SQL and direct references to host variables in the FETCH statements (no SQLDA manipulation). This is typical for the static SQL programming model. This example also shows the use of *streaming* the LOB data. That is, the application does not attempt to fully materialize the entire LOB or XML data items in its memory. Rather, it processes the data one piece at a time. In this case, it writes it to a file.

Assume that this application works with the same table shown in 7.8.1, “Example using dynamic SQL” on page 203, but to simplify the example, it selects only one column, C2, which contains a 10 MB CLOB in a column defined with a maximum size of 100 MB.

```
[1] EXEC SQL BEGIN DECLARE SECTION
      DECLARE CLOBHV SQL TYPE IS CLOB(32767);
    EXEC SQL END DECLARE SECTION;
[2] EXEC SQL DECLARE CURSOR1 CURSOR FOR SELECT C2 FROM T1;
```

```
[3] EXEC SQL OPEN CURSOR1;
[4] EXEC SQL FETCH WITH CONTINUE CURSOR1 INTO :CLOBHV;
[5] if (sqlcode >= 0) + sqlcode <> 100
      loop until LOB is completely fetched (no truncation occurred - compare
                                 returned length to provided buffer length)
         write current piece of data to output file
         if the column was truncated
[6]           EXEC SQL FETCH CURRENT CONTINUE CURSOR1 INTO :CLOBHV;
         endif
      endloop
     endif
[7]  EXEC SQL CLOSE CURSOR1;
```

Description:

1. The application declares a CLOB host variable that it uses to fetch the CLOB into.
2. The application declares a cursor for a static SQL SELECT statement that retrieves one CLOB column from the table.
3. The application opens the cursor.
4. The application issues the FETCH request. It uses the WITH CONTINUE clause on the FETCH to enable subsequent FETCH CURRENT CONTINUE operations. The precompiler generates the code that sets up the appropriate indicators in the RDI parameter block.

   DB2 sees that FETCH WITH CONTINUE was specified on and processes column C2 accordingly.

   The amount of data written to the data buffer equals the length specified by the SQLLONGLEN field in the secondary SQLVAR minus 4 bytes. The remaining data remains materialized (cached) at the server, and can be retrieved by the application using FETCH CURRENT CONTINUE immediately following the current FETCH. If the data contains multi-byte characters, a partial character may result at the truncation point because the data is truncated on a byte boundary.

   The precompiler-generated code does not use the SQLDATALEN field, so the required buffer length is reported (in bytes for CLOB and BLOB, in characters for DBCLOB) in the first 4 bytes of the buffer pointed by the SQLDATA field in the base SQLVAR. The required buffer length is the length of buffer space required to hold the entire data value. Therefore, it includes the amount of data already written to the data buffer.
5. The application checks for a successful fetch and then enters a loop in which it writes the buffer contents out to an external file, and then checks if truncation occurred. To check for truncation, it first checks the SQLWARN1 field to see whether it is set to 'W'. If so, that means that at least one column was truncated. To check each column, the application must compare the length returned in the first 4 bytes of the output data with the length of the buffer that it provided (this will still be set in SQLLONGLEN). If there was truncation, it executes the FETCH CURRENT CONTINUE statement to get the next piece of data. This is repeated until the LOB or XML column is completely fetched. The check for truncation involves comparing the integer value in the first 4 bytes of the data buffer with the length of the input host variable.
6. When doing the FETCH CURRENT CONTINUE, the application uses a direct host variable reference in the INTO clause. If there had been other host variables in the original SELECT list, those would have to be specified in the INTO clause as well.

   To process the FETCH CURRENT CONTINUE statement, DB2 writes data to the output host variables in the same way that FETCH does, but beginning at the truncation point. DB2 only writes out data for LOB or XML columns that were previously truncated. Other

columns are ignored. The application processes the returned data in the data buffers. In this case, the application allocated the required sizes for the FETCH CURRENT CONTINUE to be successful. However, if the LOB data buffer is still too small, DB2 would again set the truncation warnings and lengths as described on the FETCH step. One difference is that the length returned in the first 4 bytes on the FETCH CURRENT CONTINUE statement is equal to the length of the data from the truncation point to the end.

7. After the loop, the application closes the cursor. If there had been truncated columns with unfetched data remaining, it would have been discarded.

7.8.3 Scrollable and allocated cursors

The FETCH CURRENT CONTINUE functionality can be used with scrollable cursors as well. The FETCH operation can specify WITH CONTINUE even for backward, relative, and absolute fetches. Following such FETCH operations, the FETCH CURRENT CONTINUE statement can retrieve any truncated data.

A FETCH CURRENT CONTINUE statement can reference an allocated cursor associated with a stored procedure result set. Likewise, the FETCH against an allocated cursor can use the WITH CONTINUE clause.

7.8.4 Restrictions

When using FETCH WITH CONTINUE, DB2 only reserves truncated data for result set columns of type BLOB, CLOB, DBCLOB, or XML, and only when the output host variable data type is the appropriate LOB data type.

FETCH CURRENT CONTINUE is not supported with multi-row fetch.

If an application uses FETCH WITH CONTINUE, and there remains truncated data after the FETCH, the application cannot perform any intervening operations on that cursor before performing the FETCH CURRENT CONTINUE. If intervening operations are performed, the truncated data is lost.

FETCH CURRENT CONTINUE functionality is not supported for non-LOB and non-XML columns that have been truncated. For example, if a VARCHAR column is truncated, the WITH CONTINUE clause is effectively ignored and data truncation is treated as always — SQLWARN1 is set and truncated data is discarded.

7.9 Spatial support

You can use IBM Spatial Support for DB2 for z/OS to generate and analyze spatial information about geographic features, and to store and manage the data on which this information is based.

A geographic feature is anything in the real world that has an identifiable location, or anything that could be imagined as existing at an identifiable location. A feature can be:

- An object (for example, a river, forest, or range of mountains)
- A space (for example, a safety zone around a hazardous site, or the marketing area serviced by a particular business)
- An event that occurs at a definable location (for example, an auto accident that occurred at a particular intersection, or a sales transaction at a specific store)

Features exist in multiple environments. For example, the objects river, forest, mountain range belong to the natural environment. Other objects, such as cities, buildings, and offices, belong to the cultural environment. Still others, such as parks, zoos, and farmland, represent a combination of the natural and cultural environments.

Examples of spatial information are:

- Locations of geographic features on the map (for example, longitude and latitude values that define where cities are situated)
- The location of geographic features with respect to one another (for example, points within a city where hospitals and clinics are located, or the proximity of the city's residences to local earthquake zones)
- Ways in which geographic features are related to each other (for example, information that a certain river system is enclosed within a specific region, or that certain bridges in that region cross over the river system's tributaries)
- Measurements that apply to one or more geographic features (for example, the distance between an office building and its lot line, or the length of a bird preserve's perimeter)

Spatial information, either by itself or in combination with traditional relational data, can help you with such activities as defining the areas in which you provide services, and determining locations of possible markets. For example, suppose that the manager of a county welfare district needs to verify which welfare applicants and recipients actually live within the area that the district services. IBM Spatial Support for DB2 for z/OS can derive this information from the serviced area's location and from the addresses of the applicants and recipients.

Or suppose that the owner of a restaurant chain wants to do business in nearby cities. To determine where to open new restaurants, the owner needs answers to questions such as: Where in these cities are concentrations of clientele who typically frequent my restaurants? Where are the major highways? Where is the crime rate lowest? Where are the competition's restaurants located? IBM Spatial Support for DB2 for z/OS can produce information to answer these questions. Furthermore, front-end tools, though not required, can play a part. For example, a visualization tool can put information produced by IBM Spatial Support for DB2 for z/OS, such as the location of concentrations of clientele and the proximity of major highways to proposed restaurants, in graphic form on a map. Business intelligence tools can put associated information, like names and descriptions of competing restaurants, in report form.

7.9.1 IBM Spatial support for DB2 for z/OS

IBM Spatial Support for DB2 for z/OS provides a set of spatial data types that you can use to model real-world entities, such as the locations of customers, the boundaries of parks, and the path of cable lines. You can manipulate spatial data by using spatial functions, which you can invoke from within an SQL statement. Also, you can create indexes on spatial data, which can be used by DB2 to optimize spatial query performance.

Figure 7-13 to Figure 7-17 on page 212 show details of this support.

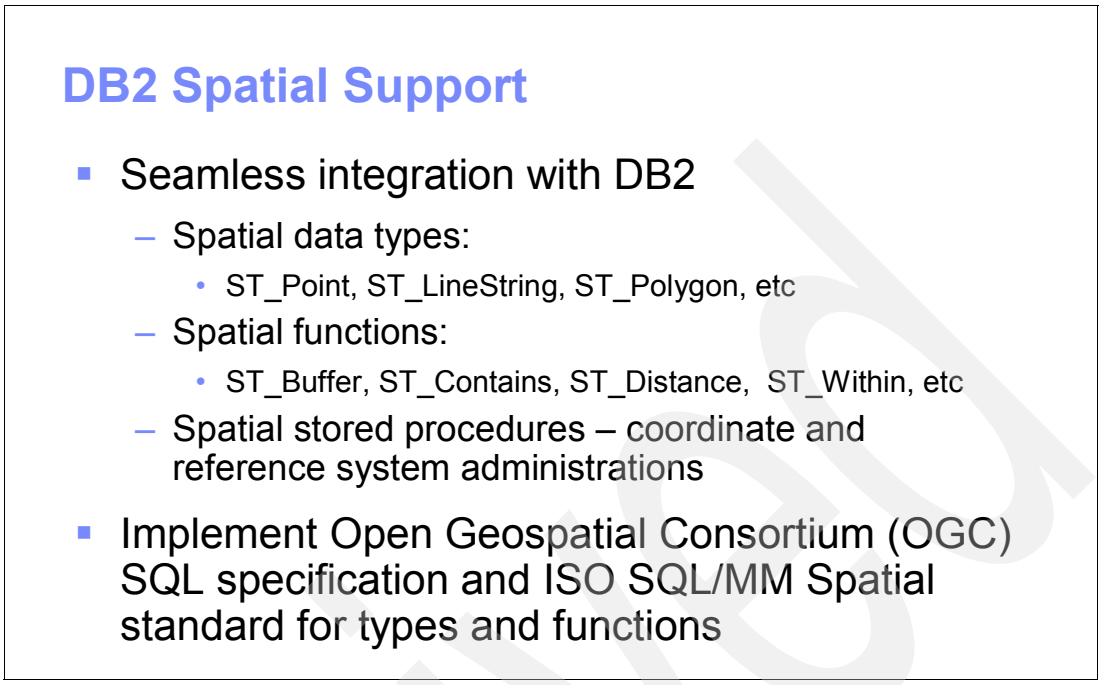


Figure 7-13 DB2 Spatial Support (1 of 2)

There is seamless integration of spatial data with DB2. A geographic feature can be represented by one or more data items in a row of a table. A data item is the value or values that are stored in the column of a relational table. For example, consider office buildings and residences. In Figure 7-14 each row of the BRANCHES table represents a branch office of a bank. Similarly, each row of the CUSTOMERS table, taken as a whole, represents a customer of the bank. However, a subset of each row — specifically, the data items that constitute a customer's address — represent the customer's residence.

BRANCHES

ID	NAME	ADDRESS	CITY	POSTAL CODE	STATE_PROV	COUNTRY
937	Airzone-Multern	92467 Airzone Blvd	San Jose	95141	CA	USA

CUSTOMERS

ID	LAST NAME	FIRST NAME	ADDRESS	CITY	POSTAL CODE	STATE_PROV	COUNTRY	CHECKING	SAVINGS
59-6396	Kriner	Endela	9 Concourt Circle	San Jose	95141	CA	USA	A	A

*** The names and addresses in both tables are fictional ***

Figure 7-14 Data representing geographic features

Archived

A subset of the above data — the values that denote the branches' and customers' addresses — can be translated into values from which spatial information is generated. For example, as shown in Figure 7-14 on page 209, one branch office's address is 92467 Airzone Blvd., San Jose, CA 95141, USA. A customer's address is 9 Concourt Circle, San Jose, CA 95141, USA. IBM Spatial Support for DB2 for z/OS can construct a ST_POINT column object by using the geocoded x and y coordinate values. Figure 7-15 shows the BRANCHES and CUSTOMERS tables with new columns that are designated to contain such values. In each table, the LOCATION column contains coordinates that correspond to the addresses.

BRANCHES

ID	NAME	ADDRESS	CITY	POSTAL CODE	STATE_PROV	COUNTRY	LOCATION
937	Airzone-Multern	92467 Airzone Blvd	San Jose	95141	CA	USA	

CUSTOMERS

ID	LAST NAME	FIRST NAME	ADDRESS	CITY	POSTAL CODE	STATE_PROV	COU...	LOCATION	CH...	SA...
59-6396	Kriner	Endela	9 Concourt Circle	San Jose	95141	CA	USA		A	A

Figure 7-15 Tables with spatial columns added

Because spatial information is derived from the data items stored in the LOCATION column, these data items are referred to as spatial data.

The simplest spatial data item consists of a single coordinate pair that defines the position of a single geographic location. A more extensive spatial data item consists of several coordinates that define a linear path that a road or river might form. A third kind consists of coordinates that define the boundary of an area (for example, the boundary of a land parcel or flood plain).

Spatial data types

Each spatial data item is an instance of a spatial data type. The data type for coordinates that mark a single location is ST_Point. The data type for coordinates that define a linear path is ST_LineString. The data type for coordinates that define the boundary of an area is ST_Polygon. These types, together with the other spatial data types, are structured types that belong to a single hierarchy.

Archived

Spatial functions

You can use spatial functions to generate spatial data from input data. For example, suppose that the bank whose branches are defined in the BRANCHES table wants to know how many customers are located within five miles of each branch. Before the bank can obtain this information from the database, it needs to define the zone that lies within a specified radius around each branch. An IBM Spatial Support for DB2 for z/OS function, ST_Buffer, can create such a definition. Using the coordinates of each branch as input, ST_Buffer can generate the coordinates that demarcate the perimeters of the zones. Figure 7-16 shows the BRANCHES table with information that is supplied by ST_Buffer.

BRANCHES

ID	NAME	ADDRESS	CITY	POSTAL CODE	STATE_PROV	COUNTRY	LOCATION	SALES_AREA
937	Airzone-Multern	92467 Airzone Blvd	San Jose	95141	CA	USA	1653 3094	1002 2001, 1192 3564, 2502 3415, 1915 3394, 1002 2001

*** Coordinates in SALES_AREA are not actual ***

Figure 7-16 Table that includes new spatial data derived from existing spatial data

The coordinates in the SALES_AREA column were derived by the ST_Buffer function from the coordinates in the LOCATION column. Like the coordinates in the LOCATION column, those in the SALES_AREA column are simulated. They are not actual.

In addition to the ST_Buffer function, IBM Spatial Support for DB2 for z/OS provides several other functions that derive new spatial data from existing spatial data.

Spatial stored procedures

After your DB2 subsystem is enabled for spatial support, you are ready to set up the resources that you need in order to use spatial data.

Among these resources are a coordinate system to which spatial data conforms and a spatial reference system, which defines the extent of the geographical area that is referenced by the data.

IBM Spatial Support for DB2 for z/OS provides several stored procedures. You can invoke any of these stored procedures, either implicitly or explicitly, to administer the coordinate and reference systems.

Open Geospatial Consortium (OGC)

This is a non-profit, international, voluntary consensus standards organization that is leading the development of standards for geospatial and location based services (http://www.opengeospatial.org).

Archived

ISO SQL/MM

ISO SQL/MM is the effort to standardize extensions for multi-media and application-specific packages in SQL. SQL is extended to manage data like texts, still images, and spatial data, or to perform data mining.

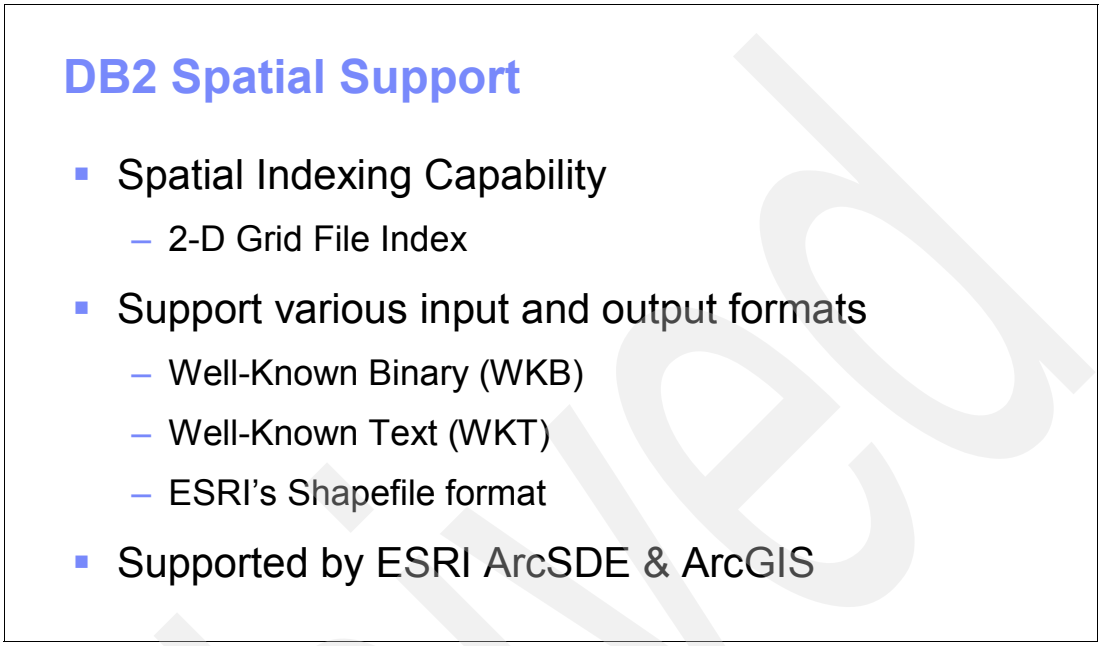


Figure 7-17 DB2 Spatial Support (2 of 2)

Spatial indexes

Good query performance is related to having efficient indexes defined on the columns of the base tables in a database.

The performance of the query is directly related to how quickly values in the column can be found during the query. Queries that use an index can execute more quickly and can provide a significant performance improvement.

Spatial queries are typically queries that involve two or more dimensions. For example, in a spatial query you might want to know whether a point is included within an area (polygon). Due to the multidimensional nature of spatial queries, the DB2 native B-tree indexing is inefficient for these queries.

Spatial queries use a type of index called a spatial grid index. The indexing technology in IBM Spatial Support for DB2 for z/OS utilizes grid indexing, which is designed to index multidimensional spatial data, to index spatial columns. IBM Spatial Support for DB2 for z/OS provides a grid index that is optimized for two-dimensional data.

Supported data formats

IBM Spatial Support for DB2 for z/OS supports several industry standard spatial data formats, such as well-known text (WKT) representation, well-known binary (WKB) representation, and shape representation.

The OpenGIS Consortium *Simple Features for SQL* specification defines the well-known text representation to exchange geometry data in ASCII format. This representation is also referenced by the ISO *SQL/MM Part: 3 Spatial* standard.

Archived

The OpenGIS Consortium *Simple Features for SQL* specification defines the well-known binary representation for geometries. This representation is also defined by the International Organization for Standardization (ISO) *SQL/MM Part: 3 Spatial* standard. The basic building block for well-known binary representations is the byte stream for a point, which consists of two double values. The byte streams for other geometries are built using the byte streams for geometries that are already defined.

Shape representation is a widely used industry standard defined by ESRI, which designs and develops the world's leading geographic information system (GIS) technology. For a full description of shape representation, see the document at the ESRI Web site at:

`http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf`

ArcSDE software is part of the ESRI ArcGIS family of software products that integrates geographic information query, mapping, spatial analysis, and editing within a multiuser enterprise DBMS environment. ArcSDE allows you to manage spatial data in DB2.

Spatial analysis capability has applications across a wide variety of industries. Figure 7-18 shows examples of several applications.

Figure 7-18 Examples of spatial applications

Given an industry segment and a little imagination, it is easy to come up with applications for a spatially aware DBMS.

Archive

Figure 7-19 lists the steps to follow to use spatial support.

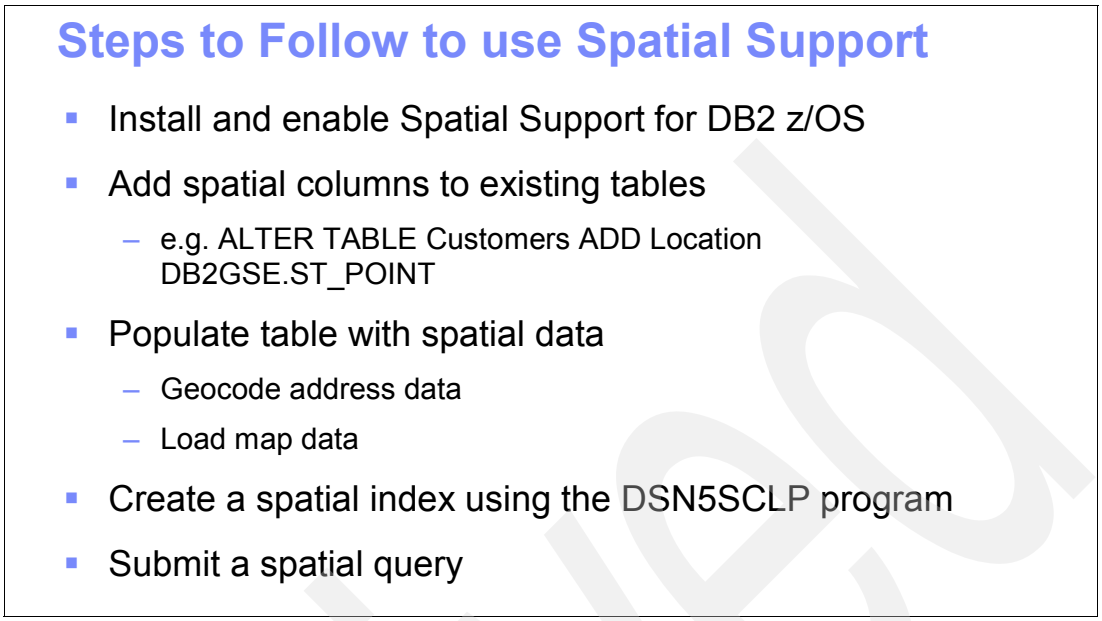


Figure 7-19 Steps to follow to use spatial support

The steps (for example, that the insurance company would follow) are:

1. Enable the database for spatial.

   Spatially enabling a database adds the spatial data types (points, lines, polygons, and so on) and functions (distance, contains, within, intersect) to the database. It also identifies the available coordinate systems and other spatial meta-data.

2. Enable a table for spatial.

   Spatially enabling a table is the process of identifying the spatial column in the table and describing (optionally) how it is generated (the source address fields for the geocoder, for example). You would need to create an UDF to call an external Web services geocoder and triggers to maintain the spatial columns.

3. Create a spatial index.

   DSN5SCLP is an ODBC program that you can use to invoke IBM Spatial Support for DB2 for z/OS stored procedures for administrative tasks. You can use several commands with the DSN5SCLP program. Use the **`create_idx`** command to create a spatial grid index on a spatial column to help optimize spatial queries.

4. Submit the queries.

   Since these queries are in vanilla SQL (albeit using spatial functions) this step can be done with any SQL generating or accepting application.

For details refer to *IBM Spatial Support for DB2 for z/OS User's Guide and Reference*, GC19-1145.

7.10 Allow ALTER TABLE ... LONG VARCHAR TO VARCHAR

Prior to V9, if you have LONG VARCHAR or LONG VARGRAPHIC columns in a table, it is not possible to change the data type of those columns. If you have a requirement to change the

Archived

data type of such columns, you have to unload the data, drop the table, create the table using the new column definitions, and reload the data into the new table.

In V9, you can alter the data type of LONGVARCHAR and LONG VARGRAPHIC columns to VARCHAR and VARGRAPHIC, respectively, using the ALTER TABLE ALTER COLUMN syntax, and specify the length stored in the catalog table for the LONG VARCHAR or LONG VARGRAPHIC column.

Example

Assume that you have migrated a table LONGTB that has LONG VARCHAR column C. You want to change the data type of this column to VARCHAR.

You must first look at column LENGTH in the SYSIBM.SYSCOLUMNS catalog table for column C. Use the value stored in the LENGTH column in the following statement:

```
ALTER TABLE LONGTB ALTER COLUMN C SET DATA TYPE VARCHAR(length).
```

If you do not specify the value stored in column LENGTH in the ALTER TABLE statement, you receive the following message:

```
DSNT408I SQLCODE = -190, ERROR:  THE ATTRIBUTES SPECIFIED FOR THE COLUMN C ARE
NOT COMPATIBLE WITH THE EXISTING COLUMN DEFINITION
```

Note that COLTYPE in SYSIBM.SYSCOLUMNS has LONGVAR for the LONG VARCHAR column and LOGVARG for the LONG VARGRAPHIC column.

However, for a table created in V9, COLTYPE in SYSIBM.SYSCOLUMNS has VARCHAR for the LONG VARCHAR column and VARGRAPHIC for the LONG VARGRAPHIC column, and so this enhancement is relevant only for migrated tables.

7.11 RENAME COLUMN

With DB2 9, under some circumstances you are now able to rename columns of an existing table without having the need to drop and recreate the object. A new clause, RENAME COLUMN, is added to the ALTER TABLE statement. You can read an in-depth description of this new feature of the ALTER TABLE statement in 4.8, “RENAME COLUMN” on page 109.

7.12 RENAME INDEX

This is an ongoing effort to give you more possibilities to change object characteristics without the need to drop and recreate the object. One next piece has been added to the functionality of DB2 V9. Starting with DB2 NFM you can use the RENAME INDEX, extended RENAME statement, to rename existing indexes. You can refer to 4.7, “RENAME INDEX” on page 108.

7.13 SMS constructs

There has been a requirement for being able to use SMS storage classes with DB2 defined data sets and to pass SMS type parameters, such as storage class, management class, or data class, on DB2 STOGROUP commands. Prior to DB2 9, you could only use these parameters on explicit IDCAMS defines. The only way to use those specifications was to specify this in your SMS ACS routings. Since ACS routines filter on data set names, these

routines tend to become very lengthy if you start defining a lot of different combinations for different data sets.

DB2 now allows you to utilize SMS storage classes on the CREATE STOGROUP statement. This increases the flexibility of handling these data sets, while it minimizes the manual effort involved. You can now manage large *hot* DB2 data sets using DFSMS.

7.13.1 CREATE STOGROUP

Figure 7-20 shows the extended syntax diagram for the CREATE STOGROUP statement.

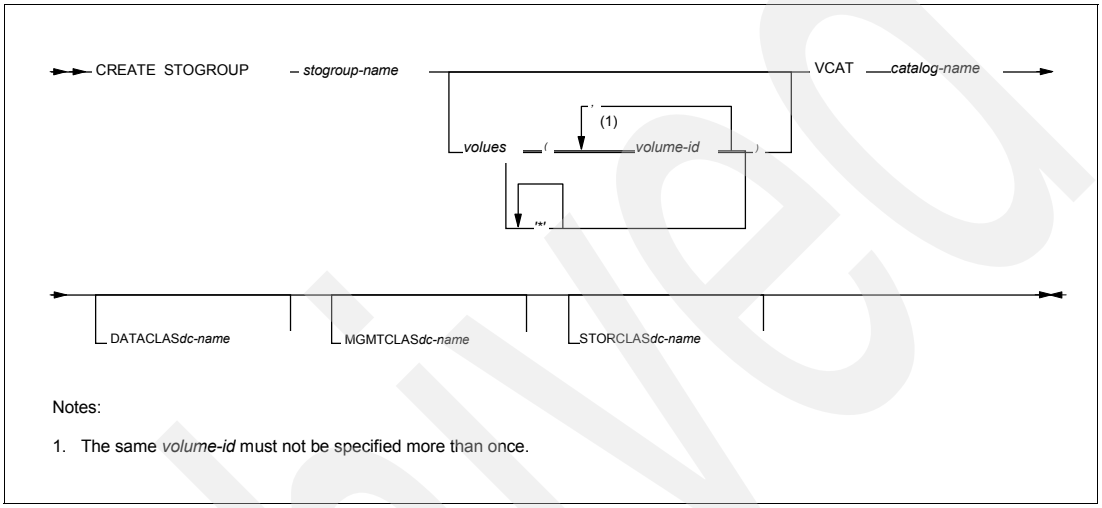


Figure 7-20 CREATE STOGROUP syntax

There are three new keywords in the syntax. You can specify just one, two or even all three of them, on one CREATE STOGROUP statement:

- DATACLAS dc-name

  SMS data classes influence characteristics such as the data set control block (DCB), striping, extended format usage, extended addressability usage, and so on.

- MGMTCLAS mc-name

  Management classes define the data set's frequency of volume backups, migration requirement, and so on.

- STORCLAS sc-name

  Among other things, storage classes can, for example, define guaranteed space requirements.

For a detailed description of the SMS classes, see *z/OS DFSMS Implementing System-Managed Storage*, SC26-7407-02.

If you use one of these keywords on your CREATE STOGROUP statement, you can now omit the VOLUMES keyword. If you omit the VOLUMES keyword and your data sets associated with the storage group are not SMS-managed, the creation of your page sets will fail.

If you omit the VOLUMES parameter, the volume selection and the number of volumes allocated for the DB2 managed data set will be completely controlled by SMS. This means that the primary space allocation of your data sets may cross multi-volumes even for a non-guaranteed and non-striped data set. Also, in addition to the primary volume, DB2 managed data sets may now possess a list of candidate volumes in the ICF catalog. As a

consequence, if an end of volume condition is returned to DB2 while it tries to extend a data set, DB2 will not issue ALTER ADDVOLUME commands to retry the extend.

Important: The maximum number of volumes that could have been used to span an SMS managed data set used to be 59. Depending on what you specify in your data class, you can now run into an end of volume condition even though there are empty volumes available in the storage group associated to your data set.

If you specify the VOLUMES parameter, DB2 continues to pass the VOLUMES parameter along with the new SMS class parameters when it creates the data sets via the AMS DEFINE command. The result is that DB2 overrides the volume count attribute that might exist in the SMS data class. If DB2 encounters an end of volume condition, it tries to extend the data set using ALTER ADDVOLUMES. DB2 continues doing that until it has reached the maximum number of 59 volumes for one data set. If the primary allocation cannot be satisfied, DB2 continues with two additional approaches with half the secondary quantity and one tenth of it.

DB2 does not check the existence of the data class, management class, or storage class that you specify on your CREATE STOGROUP statement. This is the same behavior as it always was for the VCAT and VOLUMES parameters. Later, when the STOGROUP is used to allocate a data set, the specified classes are passed to DFSMS, which does the actual work. If you do not specify all of the classes, the ACS routine that is associated to your data set name provides the data set characteristics.

7.13.2 ALTER STOGROUP

The new parameters DATACLAS, MGMTCLAS, and STORCLAS have also been added to the ALTER STOGROUP SQL statement, as shown in Figure 7-21.

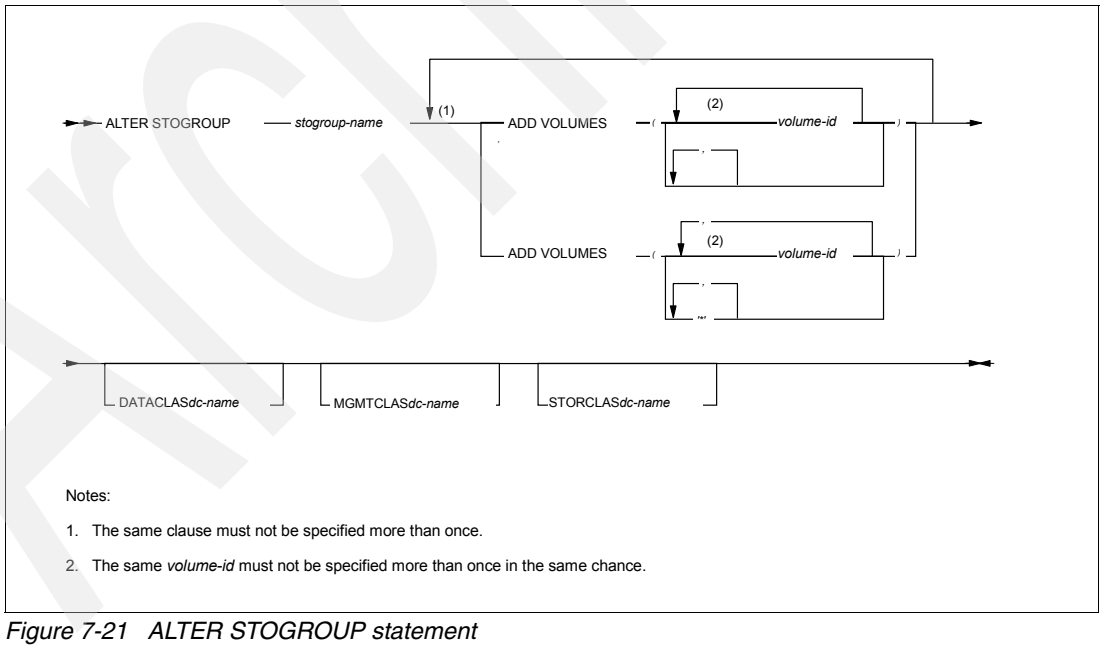


Figure 7-21 ALTER STOGROUP statement

If the data set associated with the STOGROUP is not SMS-manages, then one of the parameters ADD VOLUMES and REMOVE VOLUMES must be specified for ALTER STOGROUP. Neither ADD VOlUMES nor REMOVE VOLUMES is required when any of the SMS classes is specified.

When you alter SMS class names of a DB2 STOGROUP, this does not affect the existing data sets. However, if you run the REORG, RECOVER, or LOAD REPLACE utility, DB2 deletes the associated data set and redefines it using the new description of the storage group. It might also happen that REORG requires a new partition for a Partition by growth table space. This partition will now be created using the changed storage group characteristics.

7.13.3 Catalog changes

To reflect the usage of the news parameters in the DB2 catalog, table SYSIBM.SYSSTOGROUP has three new columns, DATACLAS, MGMTCLAS, and STORCLAS. They contain the name of the SMS classes that you have used on your CREATE or ALTER STOGROUP statement.

If you do not use those new parameters, DB2 uses the management class and storage class assigned to the corresponding ACS routine. In this case the above-mentioned columns contain blanks.

7.14 Native SQL procedures

With versions prior to V9, when you create an SQL procedure, the SQL procedure requires SQL code and C code. The logic in your SQL procedure body has to be translated into C code, compiled, and finally exists as a compiled C program outside of your DB2 catalog. Starting with V9, we have external SQL stored procedures, which still require a C compiler and still exist as external load modules outside of DB2, and we also have native SQL procedures that do not require the use of a C program.

Note that the enhancements that V9 has provided to SQL procedures are all applicable only to the native SQL procedures. The following sections all apply to native SQL procedures only:

- 7.15, “Nested compound statements in SQL stored procedures” on page 238
- 7.16, “New stored procedure related special registers” on page 249
- 7.17, “DB2 and DSN commands for native SQL procedures” on page 251
- 7.18, “Changes to DB2-supplied dynamic SQL processing applications” on page 252
- 7.19, “Differences from external SQL procedures” on page 255

With V9 new function mode, when you create a native SQL stored procedure, its procedural statements are now converted to a native representation that is stored in the DB2 catalog and directory, as it is done with other SQL statements. The parameter list and procedure options are stored in the database catalog tables as in the prior releases. When you call a native SQL procedure, DB2 loads the native representation from the catalog and the DB2 engine executes the procedure.

Authorization

In order to be able to use the CREATE PROCEDURE SQL statement, you must have at least one of the following:

- The CREATEIN privilege on the schema that you are using
- SYSADM or SYSCTRL authority

If the authorization ID matches the schema name, it implicitly has the CREATEIN privilege on the schema. Refer to *DB2 Version 9.1 for z/OS SQL Reference,* SC18-9854, for detailed information regarding authorization.

7.14.1 CREATE PROCEDURE syntax for native SQL procedures

Figure 7-22, Figure 7-23 on page 220, and Figure 7-24 on page 221 show the complete CREATE PROCEDURE syntax for native SQL procedures.

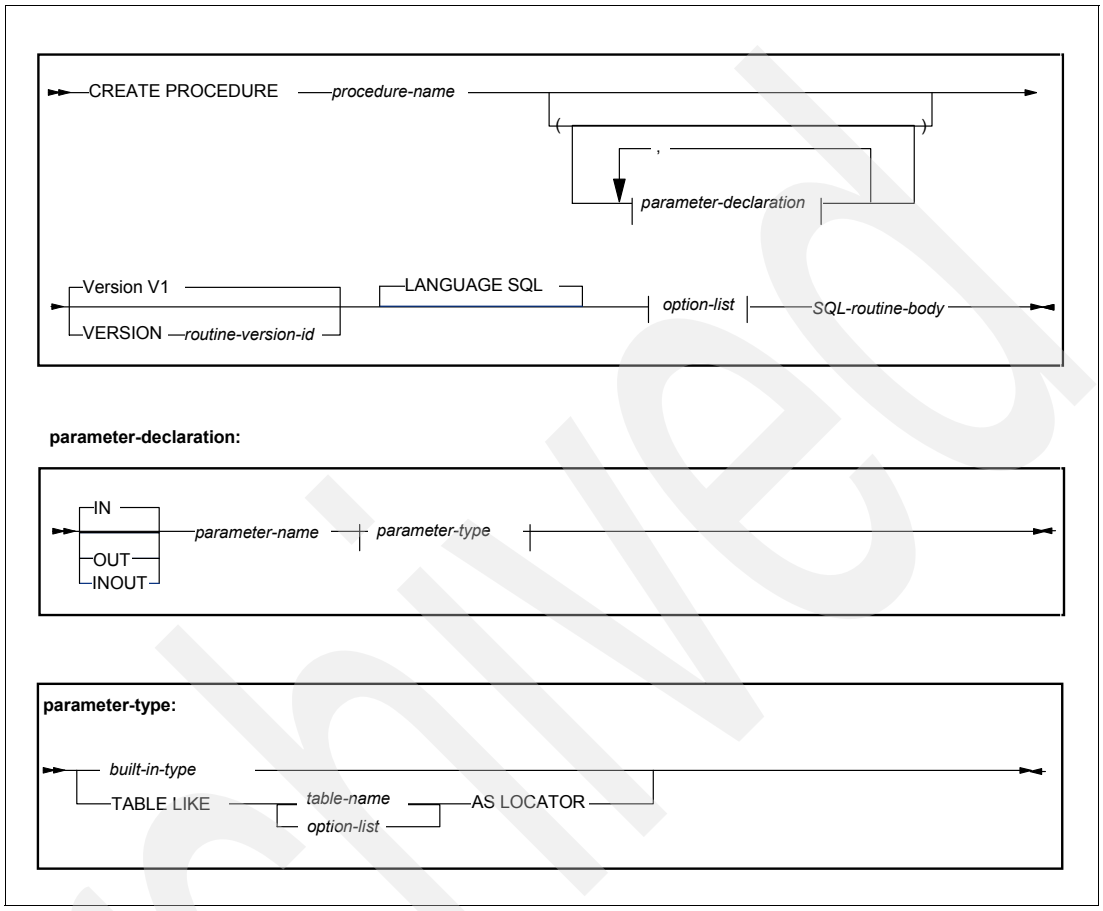


Figure 7-22 Start of CREATE PROCEDURE syntax

Archive

Figure 7-23 shows the continuation of the CREATE PROCEDURE statement with the built-in type.

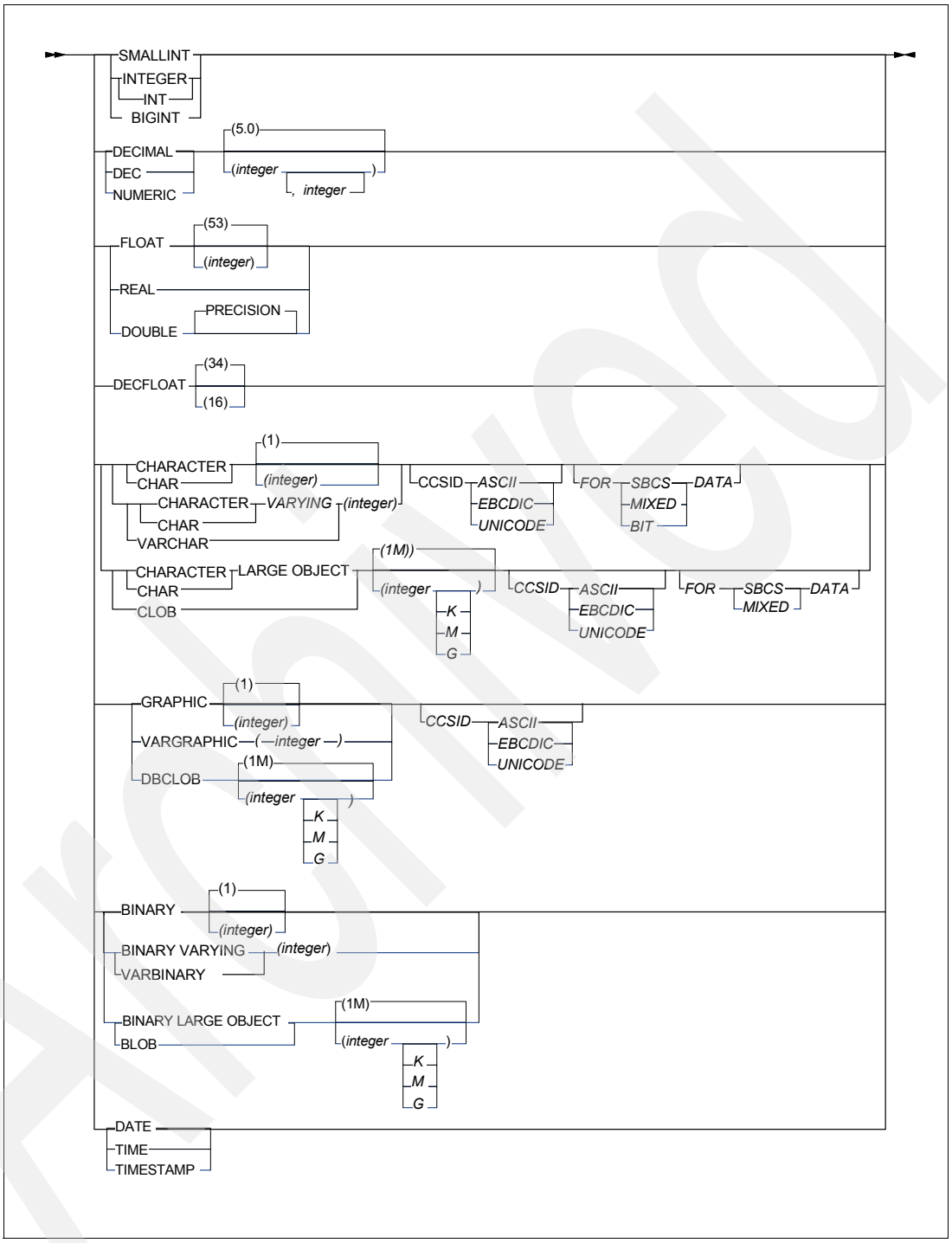


Figure 7-23 CREATE PROCEDURE built-in-type

Archived

Figure 7-24 shows the last part of the CREATE PROCEDURE syntax with the option list.

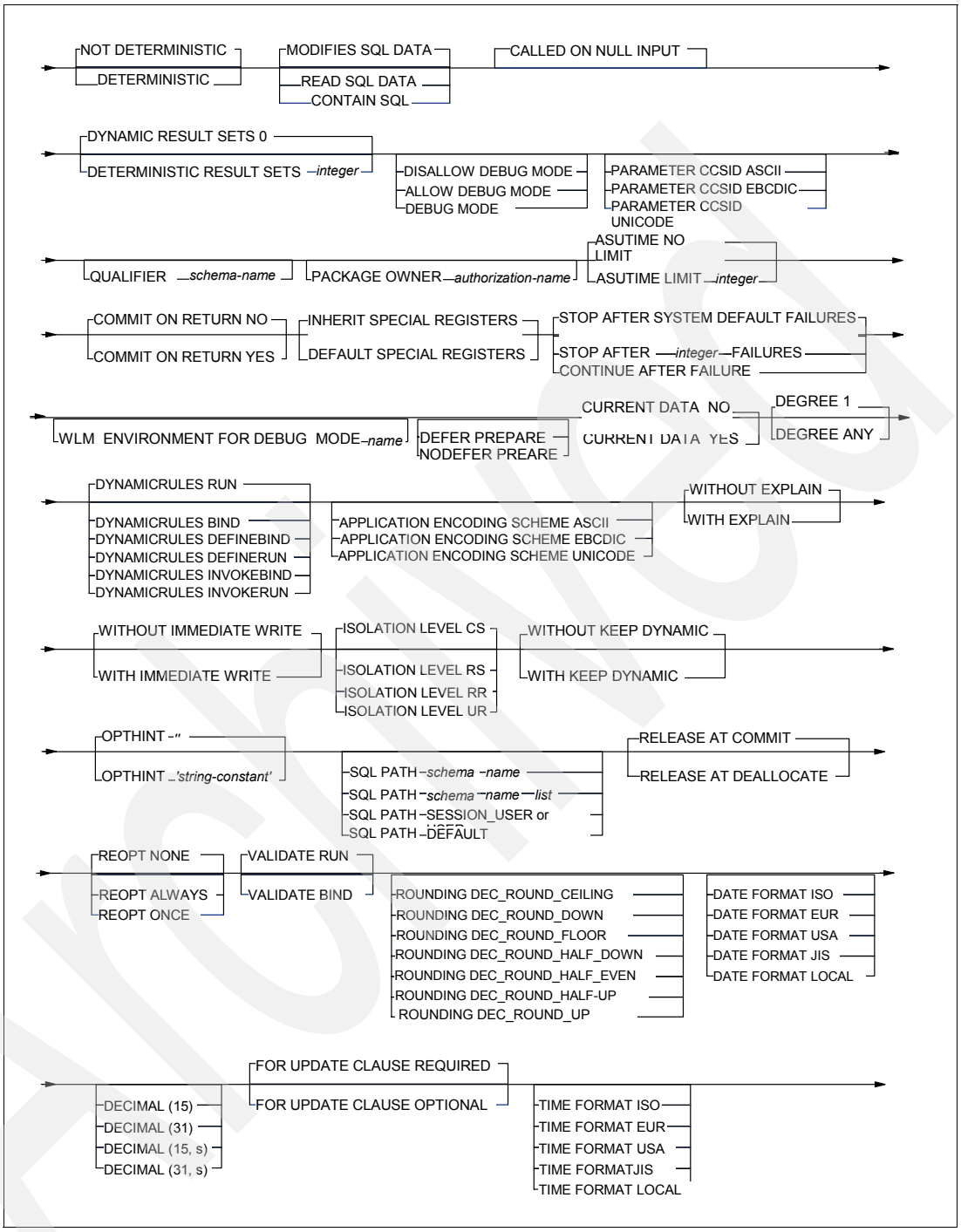


Figure 7-24 CREATE PROCEDURE option list

In comparison to what we had and still have for external SQL procedures, the most keywords have been added to the part shown in Figure 7-24. In this section, we only cover those keywords that are new or have some changed rules. Additionally, most of the parameters are basically the same as those used in BIND and REBIND. Refer to either *DB2 Version 9.1 for z/OS SQL Reference,* SC18-9854-00, or *DB2 Version 9.1 for z/OS Command Reference,* SC18-9844-00, for a more detailed description of those commonly used options.

Archive

LANGUAGE SQL

For native SQL procedures, keyword LANGUAGE SQL is now optional.

VERSION

This specifies the version identifier for the first version of the procedure that is to be generated. A routine-version-id can be up to 124 UTF-8 bytes. You can use an ALTER PROCEDURE statement with the ADD VERSION clause or the BIND DEPLOY command to create additional versions of the procedure. V1 is the default version identifier. Refer to 7.14.2, “ALTER PROCEDURE syntax for native SQL procedures” on page 226, to learn more about adding versions.

ALLOW DEBUG MODE, DISALLOW DEBUG MODE, DISABLE DEBUG MODE

With this option, you can specify whether the procedure can be run in debugging mode. The default is determined using the value of the CURRENT DEBUG MODE special register.

- ALLOW DEBUG MODE specifies that the procedure can be run in debugging mode.
- DISALLOW DEBUG MODE specifies that the procedure cannot be run in debugging mode. You can use an ALTER PROCEDURE statement to change this option to ALLOW DEBUG MODE for the initial version of the procedure.
- DISABLE DEBUG MODE specifies that the procedure can never be run in debugging mode. The procedure cannot be changed to specify ALLOW DEBUG MODE or DISALLOW DEBUG MODE once the procedure has been created or altered using DISABLE DEBUG MODE. To change this option, create a version of the procedure using the desired option and make that version the active version. Do not specify DISABLE DEBUG MODE if WLM ENVIRONMENT FOR DEBUG MODE.

Important: If you use either ALLOW DEBUG MODE or DISALLOW DEBUG MODE, you must have specified a valid value for WLMENV in DSNTIJUZ. You should also make sure that your WLM environment is set up properly so that debugging really can be invoked. You have to provide a valid value for the WLMENV in the DSNZPARM even if you specify DISALLOW DEBUG MODE on your CREATE statement, because with ALTER PROCEDURE you have the chance to switch between ALLOW and DISALLOW later on. If you know for sure at creation time already that you are not going to use debugging at all, you should use the DISABLE DEBUG MODE option.

QUALIFIER schema-name

Use this keyword to specify the implicit qualifier that is used for unqualified names of tables, views, indexes, and aliases that are referenced in the procedure body. The default value is the same as the default schema.

PACKAGE OWNER authorization-name

This specifies the owner of the package that is associated with the procedure. The SQL authorization ID of the process is the default value. The owner must have the privileges that are required to execute the SQL statements that are contained in the routine body. The value of PACKAGE OWNER is subject to translation when sent to a remote system.

WLM ENVIRONMENT FOR DEBUG MODE name

This specifies the workload manager (WLM) application environment that is used by DB2 when you are debugging the procedure. The name of the WLM environment is an SQL identifier. If you do not specify WLM ENVIRONMENT FOR DEBUG MODE, DB2 uses the default WLM-established stored procedure address space specified at installation time on panel ’Routine parameters panel: DSNTIPX’.

Note: As for every other stored procedure, you must have the appropriate authority for the WLM application environment to define a procedure that is to run in a specified WLM application environment.

Do not specify WLM ENVIRONMENT FOR DEBUG MODE when DISABLE DEBUG MODE is specified.

DEFER PREPARE or NODEFER PREPARE

This parameter specifies whether to defer preparation of dynamic SQL statements that refer to remote objects or to prepare them immediately. The default depends on the value in effect for the REOPT option. If REOPT NONE is in effect the default is NODEFER PREPARE. Otherwise, the default is DEFER PREPARE.

CURRENT DATA

This specifies whether to require data currency for read-only and ambiguous cursors when the isolation level of cursor stability is in effect. CURRENTDATA also determines whether block fetch can be used for distributed, ambiguous cursors.

DEGREE

This specifies whether to attempt to run a query using parallel processing to maximize performance.

DYNAMICRULES

This specifies the values that apply, at run time, for the following dynamic SQL attributes:

- The authorization ID that is used to check authorization
- The qualifier that is used for unqualified objects
- The source for application programming options that DB2 uses to parse and semantically verify dynamic SQL statements
- Whether dynamic SQL statements can include GRANT, REVOKE, ALTER, CREATE, DROP, and RENAME statements

In addition to the value of the DYNAMICRULES clause, the runtime environment of a native SQL procedure controls how dynamic SQL statements behave at run time. The combination of the DYNAMICRULES value and the runtime environment determines the value for the dynamic SQL attributes. That set of attribute values is called the dynamic SQL statement behavior. The following values can be specified:

RUN This specifies that dynamic SQL statements are to be processed using run behavior. RUN is the default.

BIND This specifies that dynamic SQL statements are to be processed using bind behavior.

DEFINEBIND This specifies that dynamic SQL statements are to be processed using either define behavior or bind behavior.

DEFINERUN This specifies that dynamic SQL statements are to be processed using either define behavior or run behavior.

INVOKEBIND This specifies that dynamic SQL statements are to be processed using either invoke behavior or bind behavior.

INVOKERUN This specifies that dynamic SQL statements are to be processed using either invoke behavior or run behavior.

Archived

APPLICATION ENCODING SCHEME

This specifies the default encoding scheme for SQL variables in static SQL statements in the procedure body. The value is used for defining an SQL variable in a compound statement if the CCSID clause is not specified as part of the data type, and the PARAMETER CCSID routine option is not specified. This option also affects the content of the data that is returned by the SQL statement DESCRIBE. DB2 will return column names, label names, or both (if requested) in the specified application encoding scheme.

WITH EXPLAIN or WITHOUT EXPLAIN

This specifies whether information will be provided about how SQL statements in the procedure will execute.

WITH IMMEDIATE WRITE or WITHOUT IMMEDIATE WRITE

This specifies whether immediate writes are to be done for updates that are made to group buffer pool dependent page sets or partitions. This option is only applicable for data sharing environments. The IMMEDWRITE subsystem parameter has no affect on this option.

ISOLATION LEVEL

This specifies how far to isolate the procedure from the effects of other running applications.

KEEP DYNAMIC

This specifies whether DB2 keeps dynamic SQL statements after commit points.

OPTHINT

This specifies whether query optimization hints are used for static SQL statements that are contained within the body of the procedure.

SQL PATH

This specifies the SQL path that DB2 uses to resolve unqualified user-defined distinct type, function, and procedure names in the procedure body. The default value is “SYSIBM,” “SYSFUN,” “SYSPROC”. and procedure-schema, where procedure-schema is the schema qualifier for the procedure that is the target of the statement.

RELEASE AT

This specifies when to release resources that the procedure uses: either at each commit point or when the procedure terminates.

REOPT

This specifies whether DB2 will determine the access path at run time by using the values of SQL variables or SQL parameters, parameter makers, and special registers.

VALIDATE RUN or VALIDATE BIND

This specifies whether to recheck, at run time, errors of the type OBJECT not FOUND and NOT AUTHORIZED that are found during bind or rebind. The option has no effect if all objects and needed privileges exist.

VALIDATE RUN This specifies that if needed objects or privileges do not exist when the CREATE PROCEDURE statement is processed, warning messages are returned, but the CREATE PROCEDURE statement succeeds. The DB2 subsystem rechecks for the objects and privileges at run time for those SQL statements that failed the checks during processing of the CREATE PROCEDURE statement. The authorization checks the use of the authorization ID of the owner of the procedure. VALIDATE RUN is the default.

Archived

VALIDATE BIND This specifies that if needed objects or privileges do not exist at the time the CREATE PROCEDURE statement is processed, an error is issued and the CREATE PROCEDURE statement fails.

ROUNDING

This specifies the desired rounding mode for manipulation of DECFLOAT data. The default value is taken from the DEFAULT DECIMAL FLOATING POINT ROUNDING MODE in DECP.

- DEC_ROUND_CEILING

  This specifies that numbers are rounded towards positive infinity.

- DEC_ROUND_DOWN

  This specifies that numbers are rounded towards 0 (truncation).

- DEC_ROUND_FLOOR

  This specifies that numbers are rounded towards negative infinity.

- DEC_ROUND_HALF_DOWN

  This specifies that numbers are rounded to nearest. If equidistant, round down.

- DEC_ROUND_HALF_EVEN

  This specifies that numbers are rounded to nearest. If equidistant, round so that the final digit is even.

DATE FORMAT

This specifies the date format for result values that are string representations of date or time values.

DECIMAL(15), DECIMAL(31), DECIMAL(15,s), or DECIMAL(31,s)

This specifies the maximum precision that is to be used for decimal arithmetic operations.

FOR UPDATE CLAUSE

This specifies whether the FOR UPDATE clause is required for a DECLARE CURSOR statement if the cursor is to be used to perform positioned updates.

TIME FORMAT

This specifies the time format for result values that are string representations of date or time values.

Compatibilities

For compatibility with previous versions of DB2, you can still specify the clauses listed below. DB2 will ignore those statements and issue SQLCODE +434 as a warning.

- LANGUAGE SQL
- STAY RESIDENT
- PROGRAM TYPE
- RUN OPTIONS
- NO DBINFO
- COLLID or NO COLLID
- SECURITY
- PARAMETER STYLE GENERAL WITH NULLS

Note: If you specify WLM ENVIRONMENT *wlmname* without the FOR DEBUG keyword, DB2 issues an error SQLCODE.

Archived

Additionally to the above-mentioned keywords that DB2 ignores, you can continue to use the following:

- For the DYNAMIC RESULT SETS keyword, you can also:
  - Omit the DYNAMIC keyword.
  - Use SET instead of SETS.
- For DETERMINISTIC you can use NOT VARIANT instead.
- For NOT DETERMINISTIC you can use VARIANT instead.

Operational hint

As we explained before, native stored procedures are not translated to C code, compiled, and bound afterwards. Instead, native SQL stored procedures are compiled into runtime structures and bound at creation time. As a result, the complete CREATE PROCEDURE statement, including all the programming logic, is stored as package in your DB2 catalog table SYSIBM.SYSPACKAGE. On average, those packages are significantly larger than packages of stored procedures written in either any other programming language or those from external SQL stored procedures.

Attention: We encourage you to track the size of SYSIBM.SYSPACKAGE, since this might grow faster once you start using native SQL procedures extensively. Also keep in mind that your packages are loaded into your EDMPOOL at execution time. Since those packages might be significantly larger than other packages, you might run into EDMPOOL short of space conditions if you do not monitor its usage.

7.14.2 ALTER PROCEDURE syntax for native SQL procedures

The ALTER PROCEDURE statement changes the definition of an SQL procedure at the current server. You can change the procedure options, parameter names, and routine body, and you can define and maintain additional versions of the procedure using the ALTER PROCEDURE statement.

You can embed this statement in an application program or issue it interactively. It is an executable statement that can be dynamically prepared only if DYNAMICRULES run behavior is implicitly or explicitly specified. If your ALTER statement contains either a REPLACE VERSION or a ADD VERSION clause, the statement can only be dynamically prepared.

ALTER PROCEDURE syntax

Figure 7-25, Figure 7-26 on page 228, and Figure 7-27 on page 229 show the complete ALTER PROCEDURE syntax for native SQL stored procedures. You have a few more options here compared to the ALTER PROCEDURE for external SQL procedures.

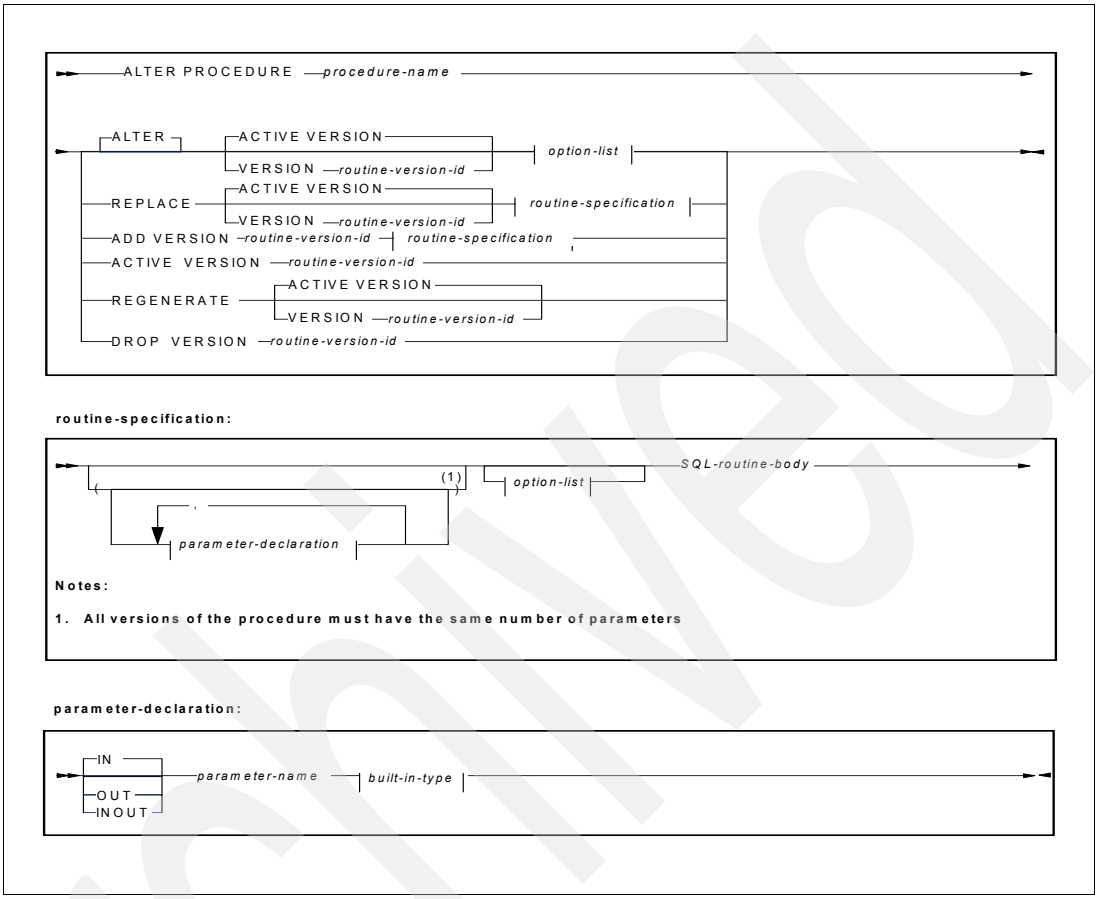


Figure 7-25 Start of ALTER PROCEDURE syntax

Archived

Figure 7-26 shows the continuation of the ALTER PROCEDURE statement with the built-in type.

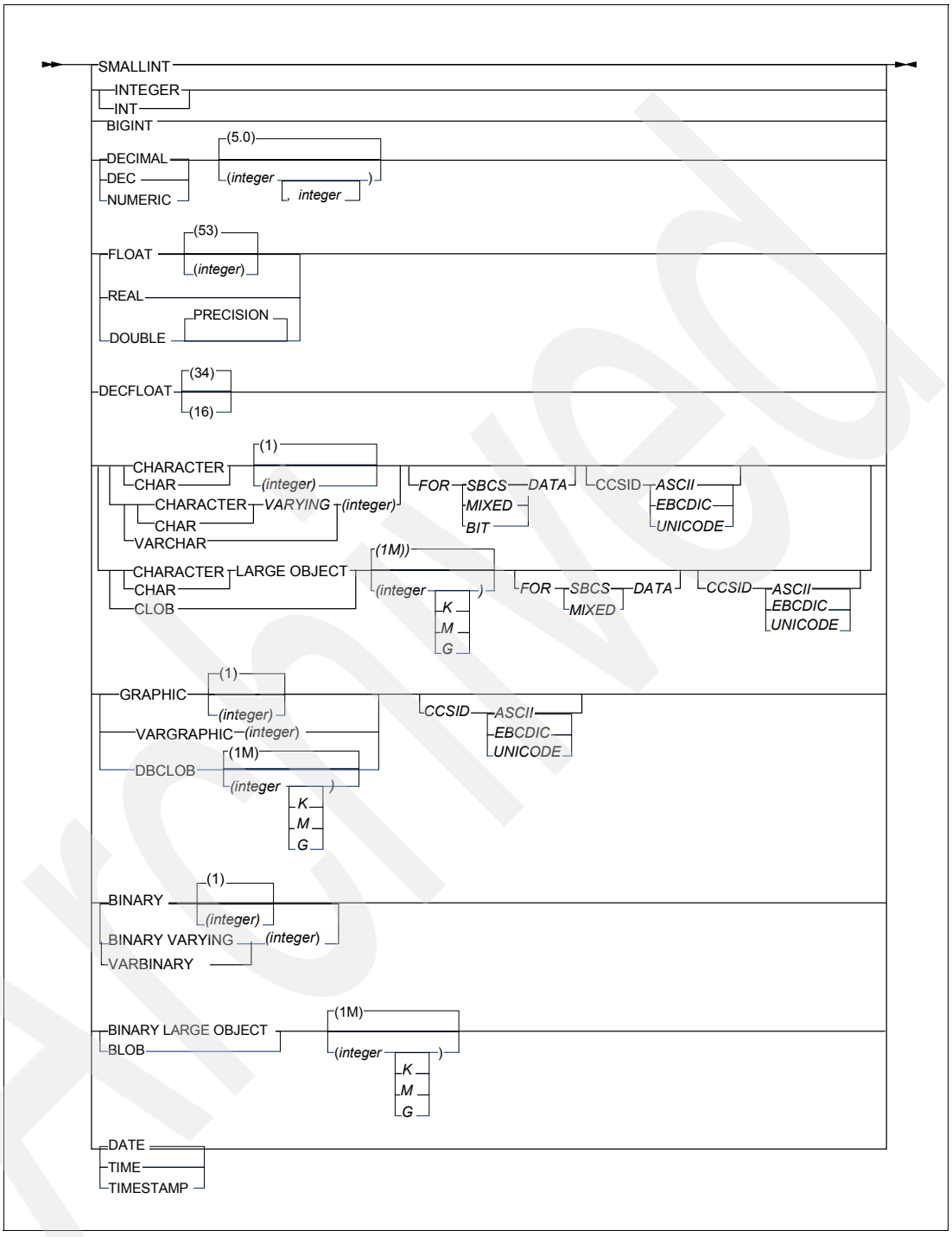


Figure 7-26 ALTER PROCEDURE built-in type

Archive

Figure 7-27 shows the last part of the CREATE PROCEDURE syntax with the option list.

Most of the options are the same as described at 7.14.1, “CREATE PROCEDURE syntax for native SQL procedures” on page 219. However there are a few keywords in the ALTER PROCEDURE statement that we have not described yet. These keywords are described hereafter.

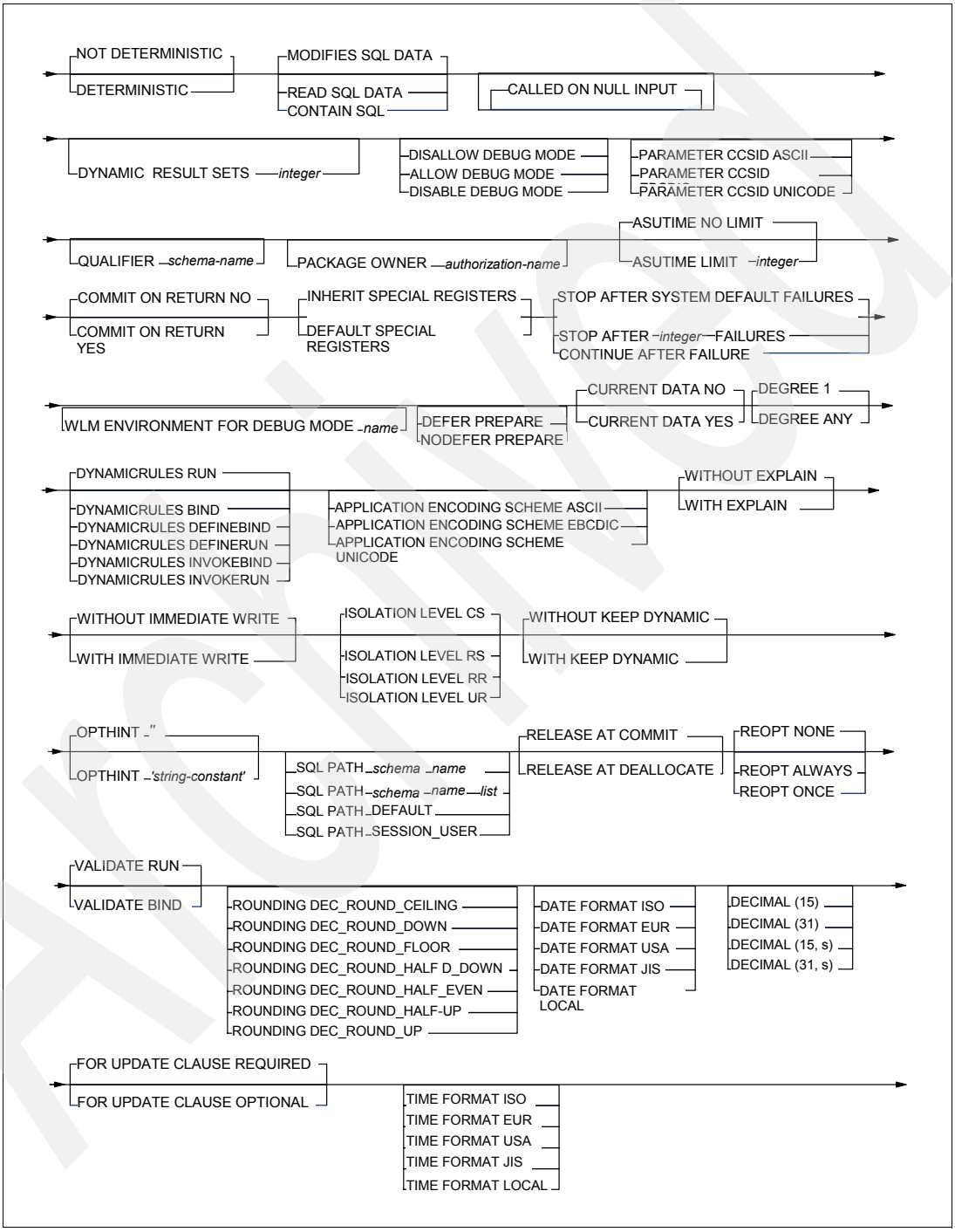


Figure 7-27 ALTER PROCEDURE option-list

Archived

ALTER ACTIVE VERSION

This identifies the version of the procedure that is to be changed, replaced, or regenerated depending on whether you specify the ALTER, REPLACE, or REGENERATE keyword.

- ACTIVE VERSION

  This specifies that the currently active version of the procedure is to be changed, replaced, or regenerated.

- VERSION routine-version-id

  Identifies the version of the procedure that is to be changed, replaced, or regenerated. routine-version-id is the version identifier that is assigned when the version is defined. routine-version-id must identify a version of the specified procedure that exists at the current server.

REPLACE ACTIVE VERSION

This specifies that a version of the procedure is to be replaced. When you bind the replaced version of the procedure, this might result in a new access path even if the routine body is not changed. When you replace a procedure, the data types, CCSID specifications, and character data attributes (FOR BIT/SBCS/MIXED DATA) of the parameters must be the same as the attributes of the corresponding parameters for the currently active version of the procedure. For options that are not explicitly specified, the system default values for those options are used, even if those options were explicitly specified for the version of the procedure that is being replaced.

ADD VERSION

This specifies that a new version of the procedure is to be created. *routine-version-id* is the version identifier for the new version of the procedure. *routine-version-id* must not identify a version of the specified procedure that already exists at the current server. When a new version of a procedure is created, the comment that is recorded in the catalog for the new version will be the same as the comment that is in the catalog for the currently active version. When you add a new version of a procedure the data types, CCSID specifications, and character data attributes (FOR BIT/SBCS/MIXED DATA) of the parameters must be the same as the attributes of the corresponding parameters for the currently active version of the procedure. The parameter names can differ from the other versions of the procedure. For options that are not explicitly specified, the system default values will be used.

ACTIVATE VERSION

This specifies the version of the procedure that is to be the currently active version of the procedure. *routine-version-id* is the version identifier that is assigned when the version of the procedure is defined. The version that is specified with routine-version-id is the version that will be invoked by the CALL statement, unless the value of the CURRENT ROUTINE VERSION special register overrides the currently active version of the procedure when the procedure is invoked. routine-version-id must identify a version of the procedure that already exists at the current server.

REGENERATE VERSION

This regenerates a version of the procedure. When DB2 maintenance is applied that changes how an SQL procedure is generated, the procedure might need to be regenerated to process the maintenance changes.

REGENERATE automatically rebinds, at the local server, the package for the SQL control statements for the procedure and rebinds the package for the SQL statements that are included in the procedure body. If a remote bind is also needed, you must explicitly use the BIND PACKAGE COPY command for all of the remote servers.

REGENERATE is different from a REBIND PACKAGE command where the SQL statements are rebound (that is, to generate better access paths for those statements), but the SQL control statements in the procedure definition remain the same.

DROP VERSION

This drops the version of the procedure that is identified with *routine-version-id*. routine-version-id is the version identifier that is assigned when the version is defined. routine-version-id must identify a version of the procedure that already exists at the current server and must not identify the currently active version of the procedure. Only the identified version of the procedure is dropped.

When only a single version of the procedure exists at the current server, you can use the DROP PROCEDURE statement to drop the procedure. A version of the procedure for which the version identifier is the same as the contents of the CURRENT ROUTINE VERSION special register can be dropped if that version is not the currently active version of the procedure.

7.14.3 Sample versioning and deployment scenarios

After all these theoretical explanations regarding the CREATE and ALTER procedure statement for native SQL procedures, we show you two scenarios for versioning and deployment of these stored procedures.

Tip: If you create SQL procedures using SPUFI, you should make sure that the SQL terminator on panel CURRENT SPUFI DEFAULTS is set to any value other than a semicolon. You must separate statements within your SQL stored procedure using the semicolon. If a semicolon is also supposed to separate SQL statements, you will run into SQLCODE -104 stating that you have coded a wrong statement.

Scenario for versioning

The first statement creates a native SQL procedure. As you can see from Figure 7-28, the CREATE PROCEDURE statement creates version V1, which is also the active version.

Note: A routine version ID is an SQL identifier of up to 124 UTF-8 bytes that designates a version of a routine.

```
CREATE PROCEDURE
    UPDATE_BALANCE
     (IN CUSTOMER_NO INTEGER,
      IN AMOUNT DECIMAL(9,2))
    VERSION V1
    LANGUAGE SQL
    READS SQL DATA
 BEGIN
  DECLARE CUSTOMER_NAME CHAR(20);
  SELECT CUSTNAME
    INTO CUSTOMER_NAME
    FROM ACCOUNTS
    WHERE CUSTNO = CUSTOMER_NO;
 END
```

Figure 7-28 CREATE PROCEDURE UPDATE_BALANCE V1

If we invoke this stored procedure via the SQL CALL statement now, V1 of UPDATE_BALANCE is called and executed. The CALL statement embedded in an application program is:

```
CALL UPDATE_BALANCE (99,200.00);
```

Tip: If you do not want to write an application program for calling your stored procedure, you can do this via the Developer Workbench, which comes with DB2 9 for LUW and the DB2 for z/OS Client Management Package and replaces the Development Center. This tool is used to handle similar development and testing functionality for DB2 UDB V8 for LUW.

Let us now use the ALTER PROCEDURE statement, as described in 7.14.2, “ALTER PROCEDURE syntax for native SQL procedures” on page 226, to alter the existing procedure definition, as shown in Figure 7-29. As you can see, we are changing two things with just one ALTER PROCEDURE statement:

- Version number from V1 to V2.
- Instead of selecting rows from table ACCOUNTS, we have now specified an UPDATE statement within the procedure body.

```
ALTER PROCEDURE UPDATE_BALANCE
ADD VERSION V2
  (IN CUSTOMER_NO INTEGER,
   IN AMOUNT INTEGER)
   MODIFIES SQL DATA
 BEGIN
   UPDATE ACCOUNTS
     SET BAL = BAL + AMOUNT
     WHERE CUSTNO = CUSTOMER_NO;
 END
```

Figure 7-29 ALTER PROCEDURE ADD VERSION example

You have now created a second version of your native SQL stored procedure UPDATE_BALANCE. If you look at your DB2 catalog table SYSIBM.SYSROUTINES, column version now indicates that you have two versions defined for this procedure. To obtain which version is the active one, you must also refer to column ACTIVE. For our procedure, V1 is still the active version, and therefore contains value ‘Y’ in this column.

If you finally want to make V2 the active version that will be used with your next CALL statement, you must now issue a second ALTER PROCEDURE statement with the ACTIVATE option as in:

```
ALTER PROCEDURE UPDATE_BALANCE ACTIVATE VERSION V2
```

Starting from this moment, DB2 will execute procedure UPDATE_BALANCE V2 when it is called.

Archive

Note: The ALTER PROCEDURE ACTIVATE statement initiates a change in the DB2 catalog so that starting from the successful execution of this statement, DB2 considers another procedure version as being the active version, and reflects this by setting Y in the ACTIVE column of SYSIBM.SYSROUTINES table.

However, you can also use the new special register CURRENT ROUTINE VERSION to interactively ask DB2 to execute a routine version other than the one marked as active in the DB2 catalog. Refer to “CURRENT ROUTINE VERSION” on page 250 for a more detailed description of this new special register.

Scenario for deployment

The following scenario shows you how to create, test, and deploy multiple versions of an SQL stored procedure from one server to another. When done, you can change a version’s logic again, test it, and redeploy it again for general use.

New DEPLOY keyword on BIND command

DEPLOY is a new BIND option. The syntax for the DEPLOY option is:

```
BIND PACKAGE ..... DEPLOY(collection-id.package-id) COPYVER(version-id) ...
```

You can use DEPLOY when the target DB2 subsystem is also a DB2 for z/OS subsystem which is operating with a compatible PTF level. If the PTF level is not compatible, SQLCODE -20249 is issued.

If you specify ACTION(ADD) for a version that does not exist at the target location, DB2 creates or adds a new version of the native SQL procedure and its associated package while keeping the source native SQL procedure’s logic. DB2 adds a new version if a native SQL procedure with the same target name already exists.

If you specify ACTION(REPLACE), DB2 replaces the version specified in COPYVER. If you specify REPLVER, the version ID must be the same as the COPYVER version ID or DB2 returns TSO error message DSNE977E.

For the example shown in the next few figures starting with Figure 7-30, we used SANJOSE as the name of the current server and BERLIN as the name of the remote server.

Figure 7-30 shows the create statement for our native SQL stored procedure.

```
CREATE PROCEDURE TEST.UPDATE_BALANCE
      (IN CUSTOMER_NO INTEGER,
       IN AMOUNT DECIMAL(9,2))
       VERSION V1
       LANGUAGE SQL
       READS SQL DATA
       QUALIFIER PAOLOR7
BEGIN
   DECLARE CUSTOMER_NAME CHAR(20);
   SELECT CUSTNAME
        INTO CUSTOMER_NAME
        FROM ACCOUNTS
        WHERE CUSTNO = CUSTOMER_NO;
END
```

Figure 7-30 CREATE PROCEDURE SAMPLE for deployment on server SANJOSE

The next step in the deployment process is to create a new SQL procedure named BERLIN.PRODUCTION.UPDATE_BALANCE from the existing SQL procedure that we have just created (that is, SANJOSE.TEST.UPDATE_BALANCE).

As you can see from Figure 7-31, both procedures have the same version V1. One difference is that procedure BERLIN.PRODUCTION.UPDATE_BALANCE has qualifier PAOLOR8. The example shown below is considered being a remote bind on location BERLIN.

```
BIND PACKAGE(BERLIN.PRODUCTION)
     DEPLOY(TEST.UPDATE_BALANCE)
     COPYVER(V1)
     ACTION(ADD)
     QUALIFIER(PAOLOR8)
```

Figure 7-31 Bind Package statement with DEPLOY option

Now that V1 of TEST.UPDATE BALANCE has successfully been deployed to BERLIN.PRODUCTION, we can, for example, add a new version of UPDATE_BALANCE at the SANJOSE server, as shown in Figure 7-32; run function tests to make sure that it works properly; and deploy the new version to BERLIN once we are satisfied with it.

```
ALTER PROCEDURE TEST.UPDATE_BALANCE
   ADD VERSION V2
       (IN CUSTOMER_NO INTEGER,
        IN AMOUNT DECIMAL(9,2))
        MODIFIES SQL DATA
        BEGIN
          UPDATE ACCOUNTS
            SET BAL = BAL + AMOUNT
            WHERE CUSTNO = CUSTOMER_NO;
        END
```

Figure 7-32 ALTER PROCEDURE add V2

Refer to the bind statement shown in Figure 7-33, which we used to add version V2 of procedure BERLIN.PRODUCTION.UPDATE_BALANCE from SANJOSE.TEST.UPDATE_BALANCE.

```
BIND PACKAGE(BERLIN.PRODUCTION)
     DEPLOY(TEST.UPDATE_BALANCE)
     COPYVER(V2)
     ACTION(ADD)
     QUALIFIER(PAOLOR8)
```

Figure 7-33 Bind package statement with deploy option for V2

7.14.4 GET DIAGNOSTICS

The GET DIAGNOSTICS SQL statement has been enhanced for use with native SQL stored procedures.

DB2_LINE_NUMBER, which is the new keyword on the GET DIAGNOSTICS statement, returns the line number where an error is encountered in parsing a dynamic statement. Also,

it returns the line number where an error is encountered in parsing, binding, or executing a CREATE or ALTER statement for a native SQL procedure.

DB2_LINE_NUMBER also returns the line number when a CALL statement invokes a native SQL procedure and the procedure returns with an error. This information is not returned for an external SQL procedure. This value will only be meaningful if the statement source contains new line control characters.

7.14.5 FOR SQL control statement

The FOR SQL statement executes one or multiple statements for each row in a table. You define the cursor by specifying a SELECT list that describes the columns and rows selected. The statements within the FOR statement are executed for each row selected. The SELECT list must consist of unique column names, and the table that is specified in the FROM clause of select-statement must exist when the procedure is created. Figure 7-34 shows the syntax.

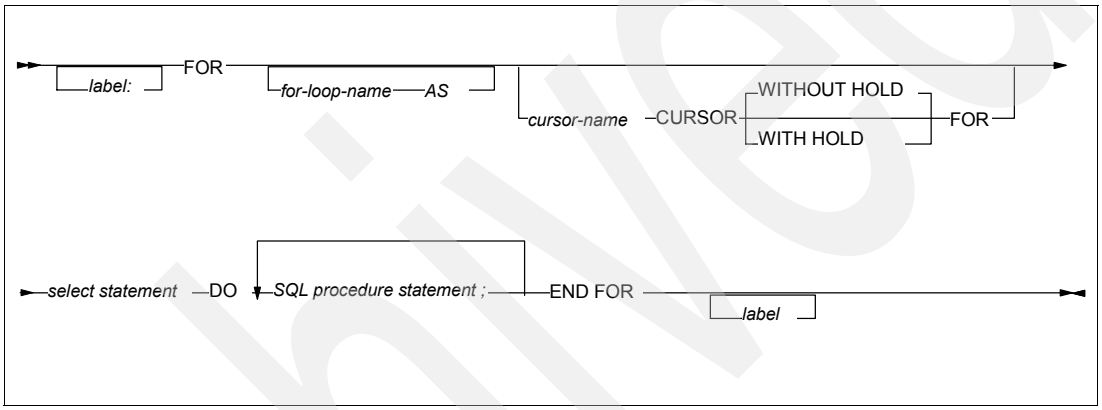


Figure 7-34 FOR statement syntax

Refer to Figure 7-35 for a very simple usage sample. For every row that is selected from table accounts, operation x = x + balance is performed.

```
DECLARE X INTEGER DEFAULT 0;
FOR L1 AS SELECT balance FROM accounts DO
  SET X = X + balance;
END FOR;
```

Figure 7-35 FOR statement - very simple usage sample

Archive

For your reference we have also listed a more complicated example that utilizes additional parts or the syntax shown in Figure 7-34 on page 235. Refer to Figure 7-36 for a more complete example.

```
BEGIN
  DECLARE fullname CHAR(40);
  FOR v1 AS
         c1 CURSOR FOR
    SELECT firstname, midinit, lastname FROM employee
      DO
         SET
           fullname = lastname CONCAT ', '
                               CONCAT firstname
                               CONCAT ' ' CONCAT midinit;
         INSERT INTO TNAMES VALUES ( fullname );
  END FOR;
END;
```

Figure 7-36 FOR statement - utilizing more syntax parts

7.14.6 New catalog table SYSIBM.SYSENVIRONMENT

The SYSIBM.SYSENVIRONMENT table stores environment information utilized for index on expressions as well as native SQL stored procedures. The environment information consists of a set of options and special registers. Column ENVID in SYSIBM.SYSENVIRONMENT is a unique value for each environment information recorded.

In order to combine the information in SYSIBM.SYSROUTINES with those additional ones stored in SYSIBM.SYSENVIRONMENT, a new column TEXT_ENVID is added to SYSIBM.SYSROUTINES to reference this environment. To give you a better understanding of which information DB2 stores in this new catalog table, you can refer to its structure and column description, which we added for you in Table 7-11.

Table 7-11 SYSIBM.SYSENVIRONMENT catalog table

Column name	Data type	Description
ENVID	INTEGER NOT NULL	Environment identifier
CURRENT_SCHEMA	VARCHAR(128) NOT NULL	The current schema
RELCREATED	CHAR(1) NOT NULL	The release when the environment information is created
PATHSCHEMAS	VARCHAR(2048) NOT NULL	The schema path
APLLICATION_ENCODING_CCSID	INTEGER NOT NULL	The CCSID of the application environment
ORIGINAL_ENCODING_CCSID	INTEGER NOT NULL	The original CCSID of the statement text string
DECIMAL_POINT	CHAR(1) NOT NULL	The decimal point indicator: C Comma P Period

Column name	Data type	Description
MIN_DIVIDE_SCALE	CHAR(1) NOT NULL	The minimum divide scale: N The usual rules apply for decimal division in SQL Y Retain at least three digits to the right of the decimal point after any decimal division
STRING_DELIMITER	CHAR(1) NOT NULL	The string delimiter that is used in COBOL string constants: A Apostrophe(') Q Quote (“)
SQL_STRING_DELIMITER	CHAR(1) NOT NULL	The SQL string delimiter that is used in string constants: A Apostrophe(') Q Quote(“)
MIXED_DATA	CHAR(1) NOT NULL	Uses mixed DBCS data: N No mixed data Y Mixed data
DECIMAL_ARITHMETIC	CHAR(1) NOT NULL	The rules that are to be used for CURRENT PRECISION and when both operands in a decimal operation have a precision of 15 or less: 1 DEC15 specifies that the rules do not allow a precision greater than 15 digits 2 DEC31 specifies that the rules allow a precision of up to 31 digits
DATA_FORMAT	CHAR(1) NOT NULL	The date format: I ISO - yyyy-mm-dd J JIS - yyyy-mm-dd U USA - mm/dd/yyyy E EUR - dd.mm.yyyy L Locally defined by an installation exit routine
TIME_FORMAT	CHAR(1) NOT NULL	The time format: I ISO - hh.mm.ss J JIS - hh.mm.ss U USA - hh:mm AM or hh:mm PM E EUR - hh.mm.ss L Locally defined by an installation exit routine
FLOAT_FORMAT	CHAR(1) NOT NULL	The floating point format. I IEEE floating point format S System /390 floating point format
HOST_LANGUAGE	CHAR(8) NOT NULL	The host language: * ASM * C * CPP * IBMCOB * PLI * FORTRAN

Archive

Column name	Data type	Description
CHARSET	CHAR(1) NOT NULL	The character set: A Alphanumeric
FOLD	CHAR(1) NOT NULL	FOLD is only applicable when HOST_LANGUAGE is C or CPP. Otherwise FOLD is blank. N Lower case letters in SBCS ordinary identifiers are not folded to uppercase Y Lower case letters in SBCS ordinary identifiers are folded to uppercase blank Not applicable
IBMREQD	CHAR(1) NOT NULL	A value of Y indicates that the row came from the basic machine-readable material (MRM) tape.
ROUNDING	CHAR(1) NOT NULL WITH DEFAULT	The rounding mode that is used when arithmetic and casting operations are performed on DECFLOAT data: C ROUND_CEILING D ROUND_DOWN F ROUND_FLOOR G ROUND_HALF_DOWN E ROUND_HALF_EVEN H ROUND_HALF_UP U ROUND_UP

Note that a same environment can be associated with multiple native SQL procedures or with multiple versions of a native SQL procedure.

7.15 Nested compound statements in SQL stored procedures

With the introduction of native SQL stored procedures on DB2 for z/OS, nested compound statements are supported. A compound statement groups other statements together into an executable block (delimited by BEGIN and END). SQL variables can be declared within a compound statement.

Up to DB2 V8, the body of an SQL procedure could contain only a single compound statement, which could contain other SQL statements, except for another compound statement, or a single SQL procedure statement other than the compound statement. Thus, you could not nest compound statements within an SQL procedure. Additionally, this meant that a condition handler could not contain a compound statement.

Starting with DB2 V9, you can now use:

- A compound statement within a condition handler
- Nested compound statements to define different scopes for SQL variables, cursors, condition names, and condition handlers

A compound statement is easily recognized as starting and ending with the keywords BEGIN and END. You can provide a label on the BEGIN statement to identify the code block.

You can use this label to qualify names of SQL variables that are defined within a compound statement, while cursor names must be unique within a procedures. These labels are very useful with nested compound statements. It is possible that you have an SQL procedure that contains both nested compound statements and compound statements that are at the same level. In Figure 7-37 compound statement OUTERMOST contains two other compound statements, INNER1 and INNER2. Additionally, the INNER1 compound statement contains another compound statement, INNERMOST.

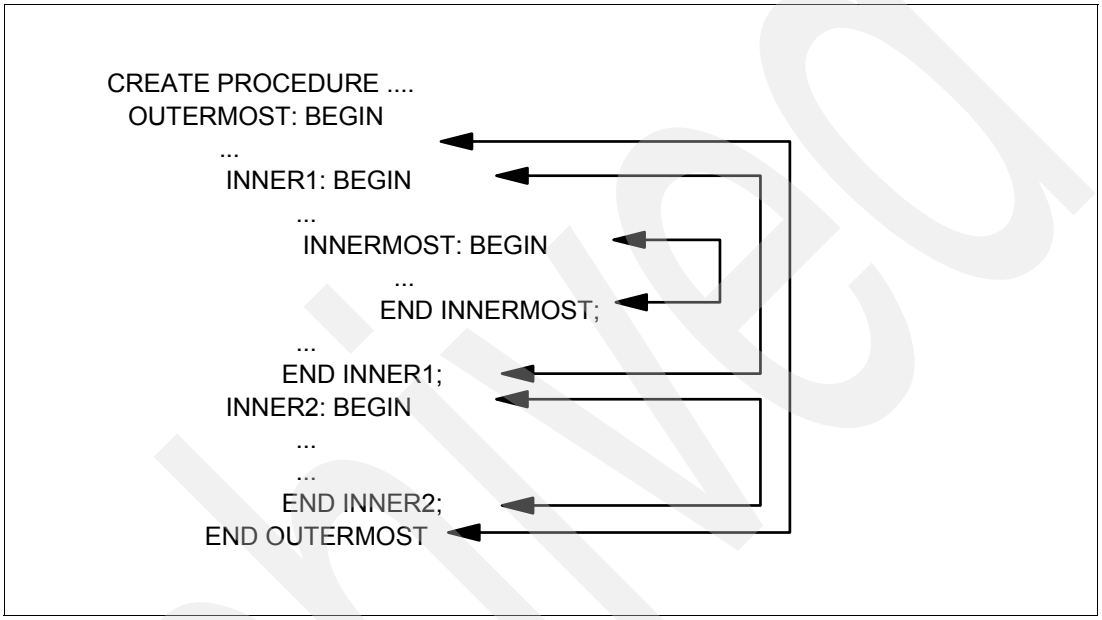


Figure 7-37 Nested compound statement

7.15.1 Using a compound statement within a condition handler declaration

With support for nested compound statements, you can now use a compound statement within the declaration of a condition handler. Up to DB2 V8, the action of a condition handler could only be a single SQL procedure statement. With DB2 V9, the SQL statement can be a compound statement which in turn encloses one or more other SQL statements. This enables you to use multiple statements within the body of a condition handler by enclosing all of them within the compound statement.

Archive

In the example shown in Figure 7-38, we capture the SQLSTATE value and set a local flag to TRUE all within the body of the condition handler.

```
BEGIN
  DECLARE SQLSTATE CHAR(5);
  DECLARE PRVSQLSTATE CHAR(5) DEFAULT '00000';
  DECLARE EXCEPTSTATE INT;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
    BEGIN
      SET PRVSQLSTATE = SQLSTATE;
      SET EXCEPTSTATE = 1;
    END;
END
```

Figure 7-38 Compound statement in condition handler

7.15.2 Using statement labels with nested compound statements

You can use nested compound statements within an SQL procedure to define the scope of SQL variable declarations, condition names, and condition handlers. Each compound statement has its own defined scope, and can be defined to have a label. It is possible to define multiple labels, each having the same name, within a procedure as long as they are defined in different scopes. A label name must be unique within the compound statement for which it is defined, including any labels defined within compound statements that are nested within that compound statement. That is, a label specified on a compound statement must be unique from other labels within the same scope. However, compound statements that are not nested can have the same name. Also, a label must not be the same as the name of the procedure.

Archive

You can reference a label in another statement such as the GOTO, LEAVE, and ITERATE SQL Procedure language control statements. A label can only be referenced within the compound statement in which it is defined, including any compound statements that are nested within that compound statement. Figure 7-39 illustrates the use of statement labels and their scope. As you can see, a reference to OUTER 1 from inside INNER1 compound statement is okay, because label OUTER1 is on a higher level than INNER1. In contrast to that, the statement of label NOWAY is not allowed and leads to an error, because the target label INNER2 is not within the scope of the LEAVE statement.

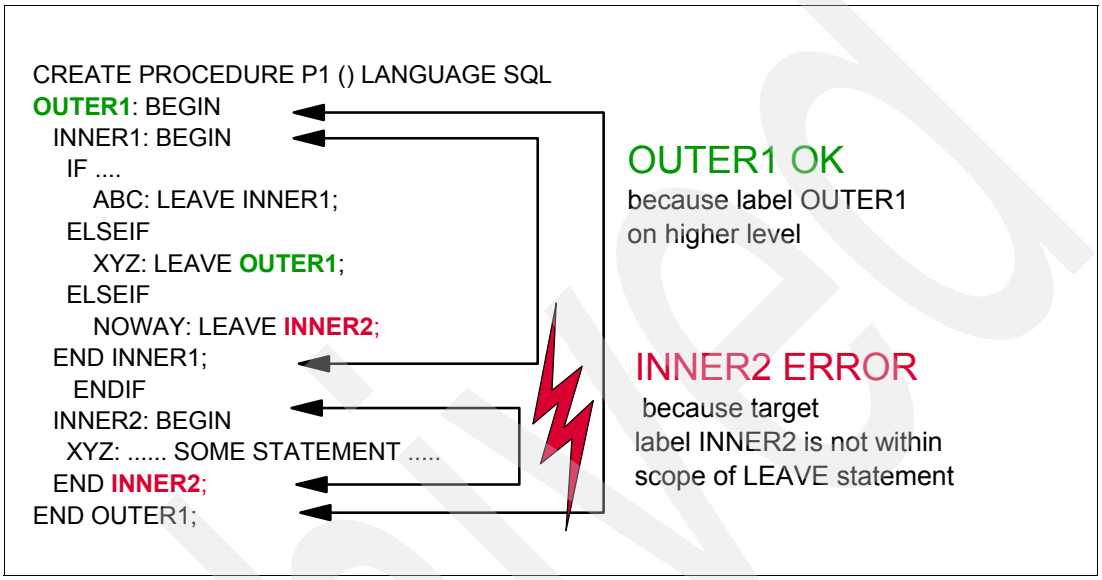


Figure 7-39 Scoping label names

Archived

Let us have a look at more examples to get used to what is allowed and what is not allowed with regards to usage of labels in nested compound statements. See Figure 7-40.

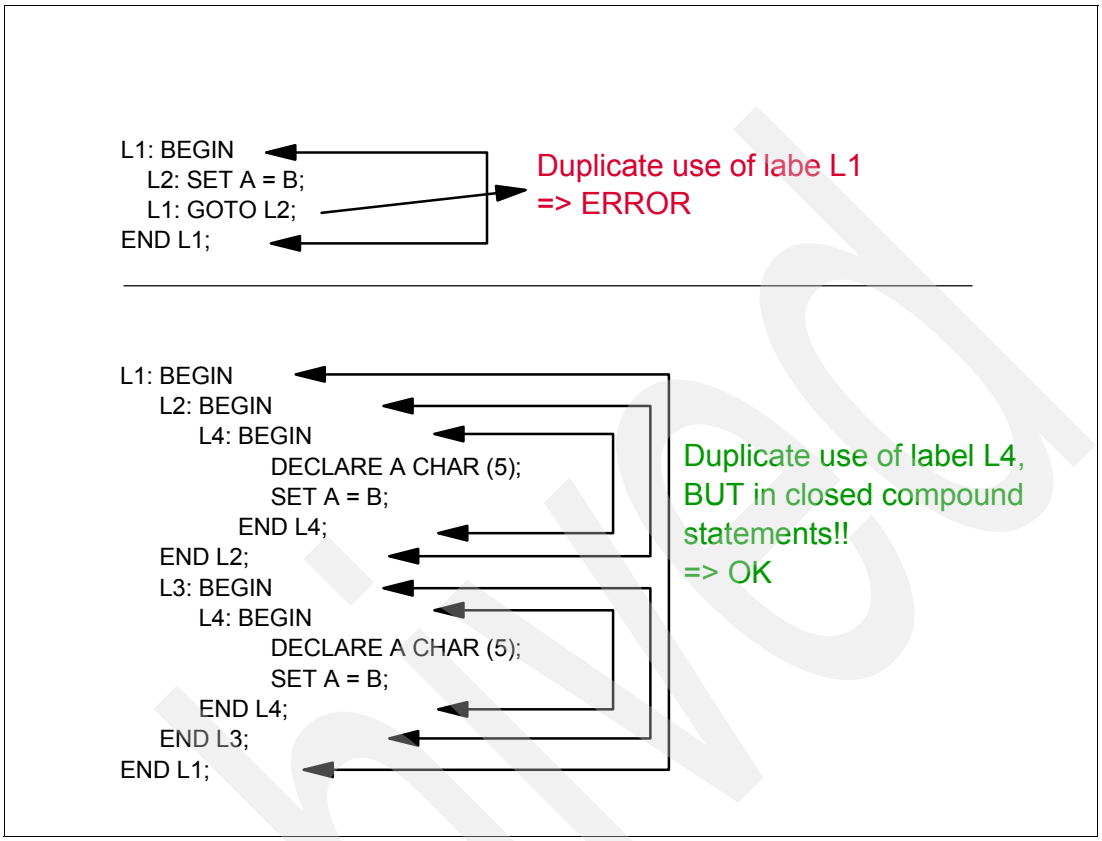


Figure 7-40 Correct and incorrect label usage

In the upper part of Figure 7-40, the usage of label L1 is invalid, because it is in the same scope as the outer label L1. The correct usage of nested labels is shown in the lower part of Figure 7-40. Here we used label L4 twice as well, but since they are not in the same scope, you do not receive an error in this case.

7.15.3 Scoping SQL variable declarations

One more situation in which you can use nested compound statement is the definition of SQL variable names within an SQL stored procedure. An SQL variable is declared within a compound statement. With the use of nested compound statements, it is possible for you to define multiple SQL variables with the same name within an SQL stored procedure.

When you use non-unique variable names in an SQL stored procedure, each SQL variable must be declared within a different scope. An SQL variable name must be unique within the compound statement in which it is declared, excluding any declarations in compound statements that are nested within that compound statement. Following this, an SQL variable can only be referenced within the compound statement in which it is declared, including any compound statements that are nested within that compound statement. When there is a reference to an unqualified SQL variable name, the variable can be declared within the compound-statement that contains the reference, or within a compound statement in which that compound statement is nested. In this case, DB2 uses the declaration of a variable by that name in the innermost compound statement.

If multiple variables with the same name exist within the stored procedure and there is an unqualified reference to the variable name, the name may not be interpreted as intended, because it resolves to the most local definition. For this reason we recommend that you always qualify the SQL variable with the name of the label from the compound statement in which it was declared.

In Figure 7-41 there are three declarations of the variable A. One instance is declared in the outer compound statement that is labeled by OUTER1. The other instances are declared in the inner compound statements that are labeled by INNER1 and INNER2. In the compound block labeled by INNER1, DB2 presumes that the unqualified references to A in the assignment statement and UPDATE statement (identified by (3) and (4)) refer to the instance declared in the local scope of the compound block, labeled by INNER1. However, if you intended to use the instance of the variable A that was declared in the compound statement block labeled with OUTER1, then you must qualify it with the label of innermost compound block.

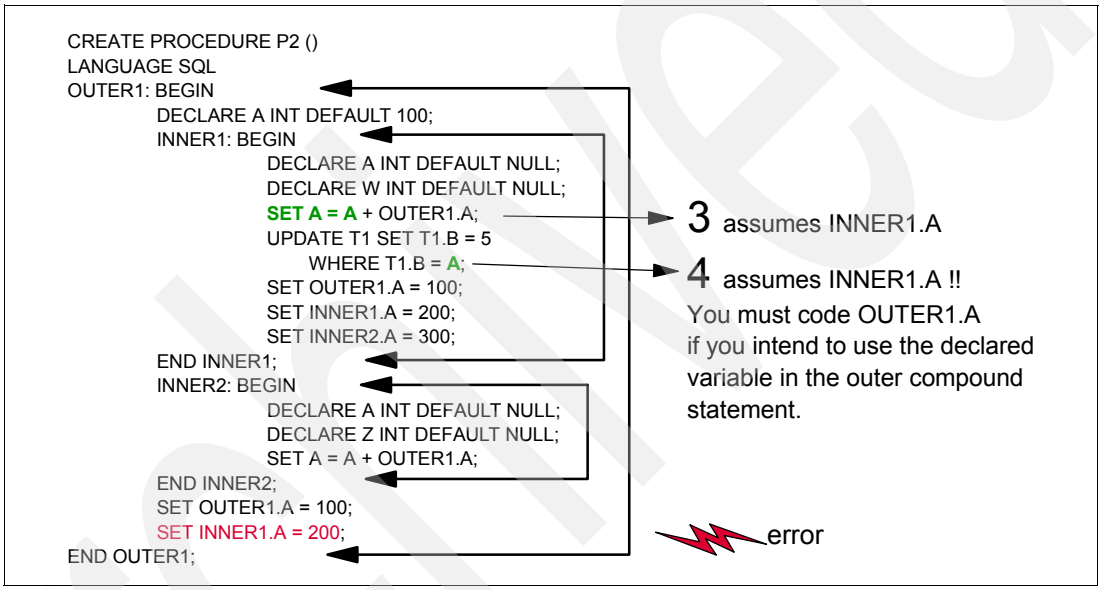


Figure 7-41 Scope of variable definitions

7.15.4 Scoping cursor definitions

Nested compound statements can be used within an SQL procedure to define the scope of cursor declaration. A cursor name must be unique within the stored procedure. You can only reference to a cursor name within the compound statement in which it is declared, including any compound statements that are nested within that compound statement.

A cursor name remains a single part name as in previous releases. You cannot reference a cursor that is defined at a lower level from a higher level within the SQL procedure.

However, any cursor that is declared in an SQL procedure as a result set cursor (that is, the WITH RETURN clause was specified as part of the cursor declaration) can be referenced by the calling application. This is true even for the case that the cursor is not declared in the outermost compound block of the procedure.

The stored procedure shown in Figure 7-42 contains a declaration of cursor X in the outer block. You can reference the cursor in the outer block where it was declared, as well as within any compound statements that are nested within the compound statement in which it was declared.

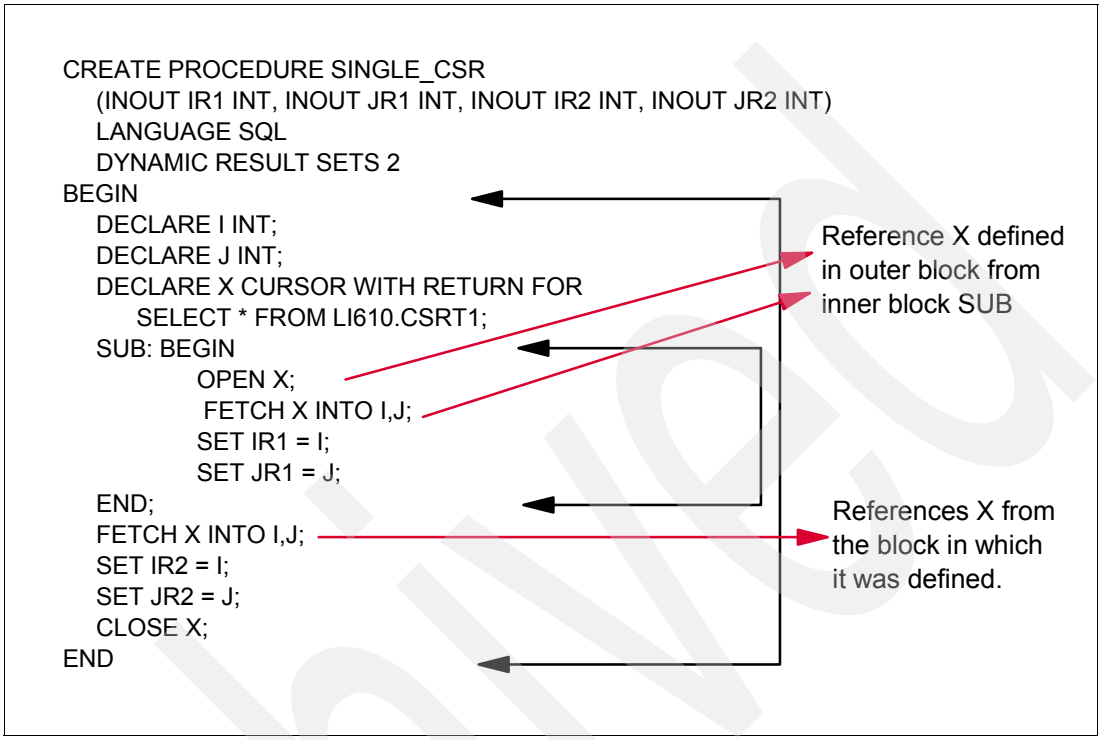


Figure 7-42 Scoping cursor declarations

7.15.5 Scoping condition names

Nested compound statements can be used within an SQL procedure to define the scope of SQL condition names. A condition is declared within a compound statement. With the use of nested compound statements, it is possible to define multiple conditions with the same name within an SQL procedure. When non-unique condition names exist in an SQL procedure, each cursor must be declared within a different scope. A condition name must be unique within the compound statement in which it is declared, excluding any declarations in compound statements that are nested within that compound statement. You can reference a condition name in the declaration of a condition handler, or in a RESIGNAL or SIGNAL statement, but note that a condition name can only be referenced within the compound statement in which it is declared, including any compound statements that are nested within that compound statement.

As for cursor names, a condition name remains a single part name as in previous releases. When multiple conditions are defined with the same name there is no way to explicitly refer to the condition that is not the most local in scope. Any reference to a non-unique condition name is an implicit reference to the innermost declaration of a condition of that name.

7.15.6 Scoping condition handler declarations

Nested compound statements can be used to localize condition handling by scoping the declarations of condition handlers. Each compound statement has its own scope for variable definitions as well as for its condition names and condition handlers. The declaration of a

condition handler associates the handler with an exception or completion condition in a compound statement.

The declaration specifies the condition that activates the condition handler, the type of the condition handler (CONTINUE or EXIT), and the handler action. The type of the condition handler determines where control is returned to after successful completion of the handler action.

The scope of the declaration of a condition handler is the compound statement in which it is declared, including any compound statements that are nested within that compound statement. A condition handler declared in a compound statement can handle a condition that is encountered in a compound statement that is enclosed within this compound statement if the condition is not handled at a lower level. However, a condition handler declared in an inner scope takes precedence over a condition handler defined to handle the same condition in an outer scope, even if the condition handler declared in an outer compound statement is more specific than a condition handler that is declared in an inner scope.

A condition handler is activated when it is the most appropriate condition handler for a condition that has been encountered. The most appropriate handler is the condition handler that most closely matches the SQLSTATE of the exception or completion condition.

Figure 7-43 explains that the code to check the SQLSTATE '22H11' error will only be generated for the statements within the INNER compound statement. Specific checking for this error will not be done for any statements in the routine outside of the INNER block. Code to check the SQLEXCEPTION error is generated for all statements in both the OUTER and INNER blocks.

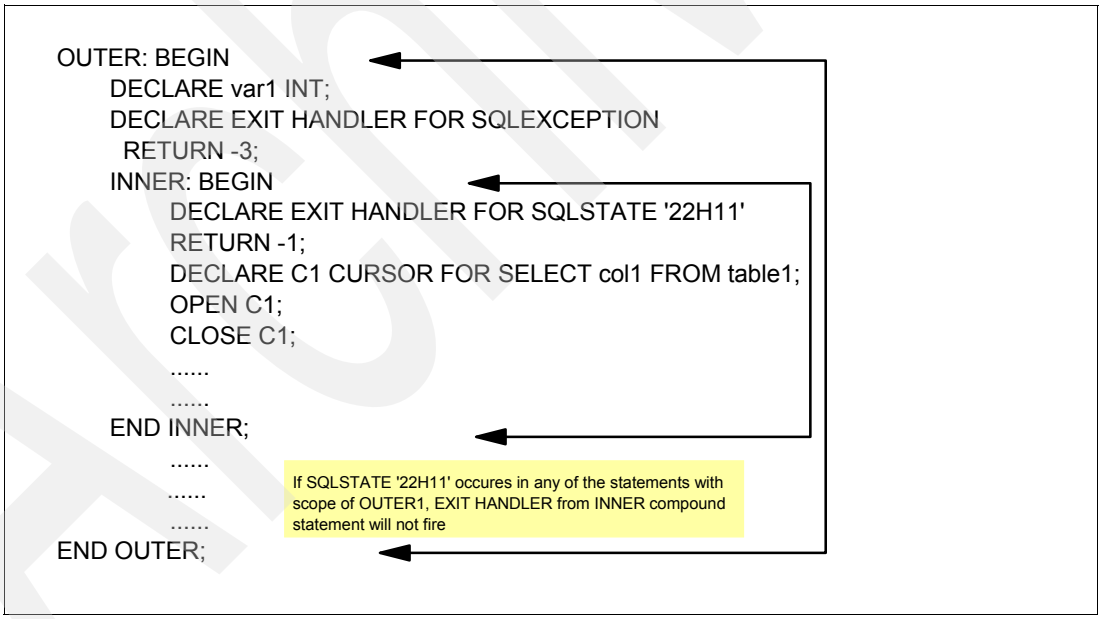


Figure 7-43 Scoping condition handler declaration

One additional example is shown in Figure 7-44.

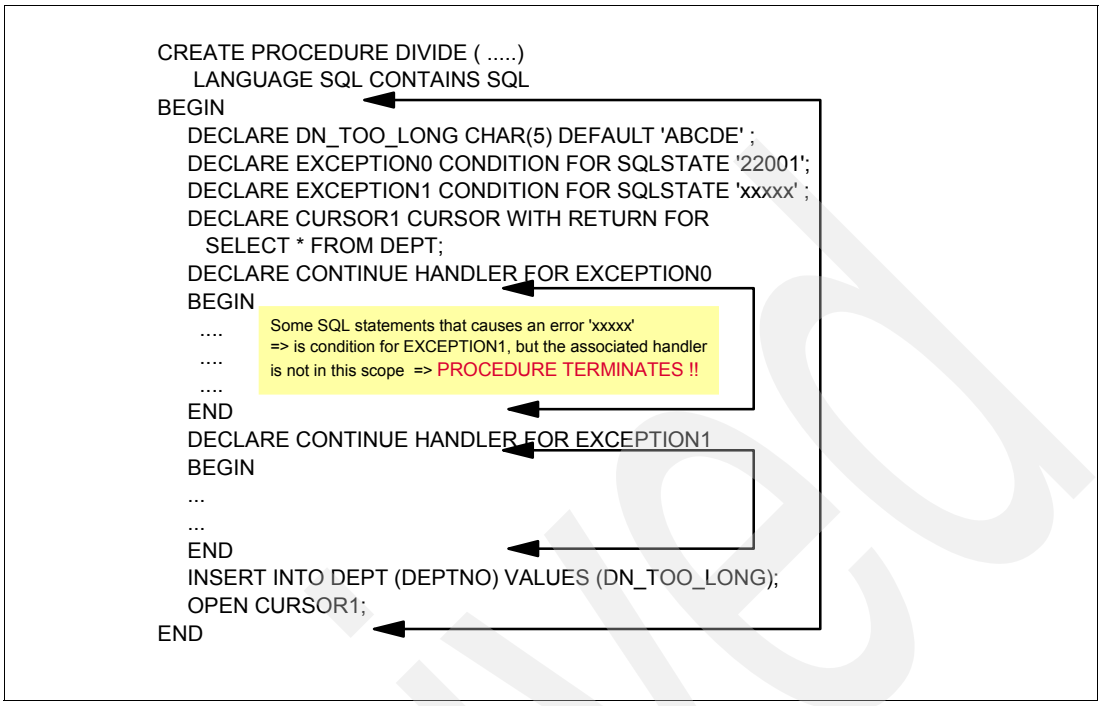


Figure 7-44 Scoping of condition handler declarations II

As shown in Figure 7-44, if exception condition exception1 is raised in the body of the condition handler for exception0, there is no appropriate handler. In this case the procedure is terminated because of the unhandled exception condition.

Note: Note that the condition handler defined for exception1 is not within the scope of the condition handler for exception0. Condition handlers that are declared in the same compound statement cannot handle conditions encountered in each other or in themselves. The scope of a condition handler includes the compound statement in which it is declared, but excludes the bodies of other condition handlers declared in the same compound statement.

7.15.7 Summary of name scoping in compound statements

We have described the name scoping in compound statements on the last few pages in detail. Use summarization Table 7-12 for future reference.

Table 7-12 Summary of name scoping

Type of name	Must be unique within...	Qualification allowed?	Can be referenced within...
SQL variable	The compound statement in which it is declared, excluding any declarations in compound statements that are nested within that compound statement.	Yes, it can be qualified with the label of the compound statement in which the variable was declared.	The compound statement in which it is declared, including any compound statements that are nested within that compound statement. When multiple SQL variables are defined with the same name you can use a label to explicitly refer to a specific variable that is not the most local in scope.

Type of name	Must be unique within...	Qualification allowed?	Can be referenced within...
condition	The compound statement in which it is declared, excluding any declarations in compound statements that are nested within that compound statement.	No	The compound statement in which it is declared, including any compound statements that are nested within that compound statement. Can be used in the declaration of a condition handler, or in a SIGNAL or RESIGNAL statement. Note: When multiple conditions are defined with the same name there is no way to explicitly refer to the condition that is not the most local in scope.
cursor	The compound statement in which it is declared, excluding any declarations in compound statements that are nested within that compound statement.	No	The compound statement in which it is declared, including any compound statements that are nested within that compound statement. Note: When multiple cursors are defined with the same name there is no way to explicitly refer to the cursor that is not the most local in scope. However, if the cursor is defined as a result set cursor (that is, the WITH RETURN clause was specified as part of the cursor declaration), the invoking application can access the result set.
label	The compound statement that declared the variable, including any declarations in compound statements that are nested within that compound statement.	No	The compound statement in which it is declared, including any compound statements that are nested within that compound statement. Use a label to qualify the name of an SQL variable or as the target of a GOTO, LEAVE, or ITERATE statement.

Allow an empty compound statement

As an aid to migration of applications from platforms and RDBMSs other than DB2 for z/OS, and for more compatibility within the DB2 family, DB2 V9 now allows you to have empty compound statements within your procedure body. The SQL Standard also allows the SQL procedure statement to be optional in the context of a compound statement in a routine.

Example 7-5 illustrates the use of an empty compound statement as a way of ignoring a condition. This might be useful when you are inserting a row into a table that has a unique column, and if the value to be inserted for the row already exists in the table, then the row does not need to be inserted. Although DB2 will detect the case that the value to be inserted is not unique, there is no need to tell the application about this. Instead, the INSERT statement will not be processed.

Example 7-5 Empty compound statement

```
DECLARE CONTINUE HANDLER FOR '23505'
BEGIN -- ignore error for duplicate value
END;
```

7.15.8 Name resolution differences between native and external SPs

There is one difference in the name resolution in native and external stored procedures that we need to explicitly mention. Assume that we have a procedure body that looks like that shown in Figure 7-45.

```
CREATE PROCEDURE ...
   BEGIN;
     DECLARE dept CHAR(3);
     DECLARE x CHAR(3);
     ...
     DECLARE c1 CURSOR FOR
       SELECT dept INTO x
       FROM emp;
     ..
  END;
```

Figure 7-45 Name resolution in external and native SQL stored procedures

As you can see, the dept is ambiguous. It is used twice:

- Declared as an SQL variable
- A column in the table emp

In the given situation, there is a difference between external and native SQL procedure in the way 'dept' is resolved:

- External SQL stored procedures match this dept to the SQL variable in the declare statement.
- Native SQL stored procedures and SQL stored procedures on LUW and iSeries® interpret this 'dept' as the column name of table emp in the SELECT statement of cursor c1.

Archived

Recommendation: Qualify names if there is a potential for ambiguity. Refer to Figure 7-46 to see how the changed CREATE PROCEDURE might look like when you follow this rule.

```
CREATE PROCEDURE ...
  STEP1: BEGIN;
    DECLARE dept CHAR(3);
    DECLARE x CHAR(3);
    DECLARE y CHAR(3);
    ...
    DECLARE c1 CURSOR FOR
      SELECT STEP1.dept, emp.dept INTO x,y
      FROM emp;
    ..
  END STEP1;
```

Figure 7-46 Better practice with name resolution in external and native SQL stored procedures

7.16 New stored procedure related special registers

Two new special registers are available with native SQL procedures: CURRENT DEBUG MODE and CURRENT ROUTINE VERSION.

CURRENT DEBUG MODE

The SET CURRENT DEBUG MODE statement shown in Figure 7-47 assigns a value to the CURRENT DEBUG MODE special register. The special register sets the default value for the DEBUG MODE option for CREATE PROCEDURE statements that define native SQL or Java procedures, or ALTER PROCEDURE statements that create or replace a version of a native SQL procedure.

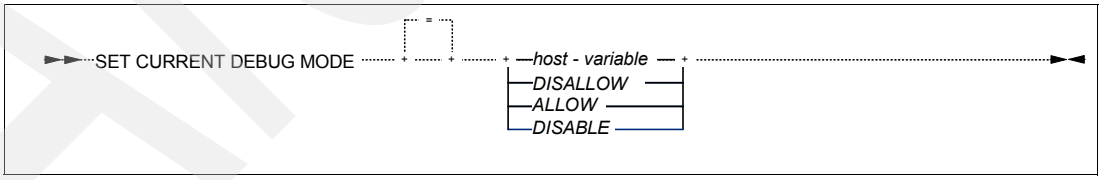


Figure 7-47 SET CURRENT DEBUG MODE syntax

Archived

The mode options are:

- DISALLOW

  This specifies that DISALLOW DEBUG MODE is the default option for CREATE PROCEDURE statements when defining an SQL or Java procedure, or ALTER PROCEDURE statements that create or replace a version of a native SQL procedure.

- ALLOW

  This specifies that ALLOW DEBUG MODE is the default option for CREATE PROCEDURE statements when defining an SQL or Java procedure, or ALTER PROCEDURE statements that create or replace a version of a native SQL procedure.

- DISABLE

  This specifies that DISABLE DEBUG MODE is the default option for CREATE PROCEDURE statements when defining an SQL or Java procedure, or ALTER PROCEDURE statements that create or replace a version of a native SQL procedure.

CURRENT ROUTINE VERSION

The SET CURRENT ROUTINE VERSION statement assigns a value to the CURRENT ROUTINE VERSION special register. The special register sets the override value for the version identifier of native SQL procedures when they are invoked. You can issue this statement interactively or embed it in an application program. It is an executable statement that can be dynamically prepared.

Figure 7-48 shows the syntax of the associated SET statement.

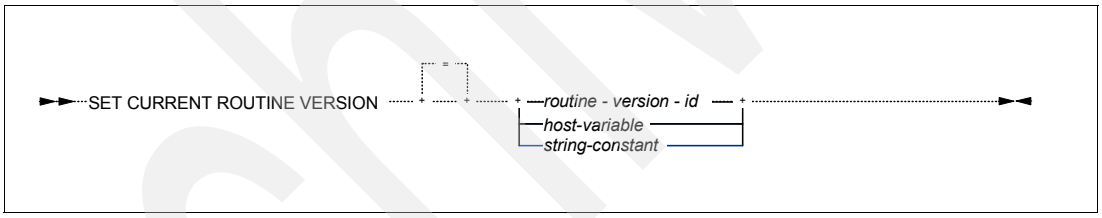


Figure 7-48 SET CURRENT ROUTINE VERSION syntax

If you want to reset the special register, specify an empty string constant, a string of blanks, or a host variable that is empty or contains only blanks. A routine version override is not in effect when the special register is reset.

If you set the CURRENT ROUTINE VERSION special register to a version identifier, it will affect all SQL procedures that are subsequently invoked using CALL statements that specify the name of the procedure using a host variable, until the value of CURRENT ROUTINE VERSION is changed. If a version of the procedure that is identified by the version identifier in the special register exists for an SQL procedure that is being invoked, that version of the procedure is used. Otherwise, the currently active version of the procedure (as noted in the catalog) is used.

Tip: When you use the CURRENT ROUTINE VERSION special register to test a version of one or more native SQL procedures, you should use a routine version identifier that is a value other than the default value (V1) on the CREATE PROCEDURE statement. This will avoid having the special register affect more procedures that you intend when testing a new version of a procedure. For example, assume that you want to run version VER2 of procedure P1, and procedure P1 invokes another procedure, P2. If a version exists for both procedures P1 and P2 with the routine version identifier VER2, that version will be used for both procedures.

7.17 DB2 and DSN commands for native SQL procedures

There are some changes when working with native versus external SQL procedure. They are explained here in order for you to understand the new command outputs.

7.17.1 -DISPLAY PROCEDURE

When you start working with native SQL procedures, you should be aware of slightly different behavior of the -DIS PROC DB2 command. Native SQL procedures are not displayed in the DISPLAY PROCEDURE output unless you run the procedure in DEBUG mode. If you do run the procedure in DEBUG mode the WLM environment column in the output contains the WLM ENVIRONMENT FOR DEBUG that you specified when you created the native SQL procedure.

The DISPLAY PROCEDURE output shows the statistics of native SQL procedures as '0' if the native SQL procedures are under the effect of a STOP PROCEDURE command. See Figure 7-49.

```
DSN SYSTEM(V91A)
DSN
-DISPLAY PROCEDURE(SCTG4100.SPTG4101)
DSNX940I  ) DSNX9DIS DISPLAY PROCEDURE REPORT FOLLOWS -
------- SCHEMA=SCTG4100
PROCEDURE     STATUS ACTIVE QUED MAXQ TIMEOUT FAIL WLM_ENV
SPTG4101
              STOPQUE     0    0    0       0    0
DSNX9DIS DISPLAY PROCEDURE REPORT COMPLETE
DSN9022I  ) DSNX9COM '-DISPLAY PROC' NORMAL COMPLETION
DSN
DSN
END
```

Figure 7-49 DISPLAY PROCEDURE output

7.17.2 -START/STOP PROCEDURE

When you issue a START or STOP PROCEDURE command for your native SQL stored procedure the command only applies to the current version of that procedure. There is no way to start or stop a specific version.

7.17.3 REBIND PACKAGE

When you issue a REBIND PACKAGE statement against a native SQL procedure, the only bind option that you can change with it is the EXPLAIN bind option.

REBIND PACKAGE only rebinds the SQL statements that are included in the procedure. The native representation of the logic part (that is, the control statements in the procedure definition) is not rebound.

Note: The behavior mentioned above is different from ALTER PROCEDURE REGENERATE, which is described in 7.14.2, "ALTER PROCEDURE syntax for native SQL procedures" on page 226. ALTER PROCEDURE REGENERATE does not only rebind all the SQL statements, but also regenerates the native representation of the logic part.

Archived

7.18 Changes to DB2-supplied dynamic SQL processing applications

Prior to V9, SQL procedures have been processed by either the precompiler or the SQL procedure processor DSNTPSMP stored procedure. With DB2 V9, you can easily use SPUFI, DSNTEP2, or DSNTEP4 to process your SQL procedures.

An SQL procedure is essentially a single SQL statement that consists of multiple SQL statements, and may also include comments and white space. For readability, it is usually formatted on multiple lines. SPUFI, DSNTEP2, and DSNTEP4 accept multi-line input that lies within columns 1 and 72 of one or more 80-column input records, the contents of which are copied seamlessly into an SQL buffer for PREPARE, effectively converting multi-line input to a single line. SQL comments are removed. Such preprocessing of SQL into a linear, comment-free format is the standard, long-established behavior of SPUFI, DSNTEP2, and DSNTEP4, but this behavior complicates diagnostics and debugging of SQL procedure language, and effectively modifies the source by discarding SQL comments and comment blocks.

The SELECT statement shown below:

```
SELECT * -- GET ALL COLUMNS
FROM DSN8910.EMP
WHERE EMPID = '000150';
```

is converted to:

```
SELECT * FROM DSN8910.EMP WHERE EMPID = '000150';
```

To enhance the usability of SPUFI, DSNTEP2, and DSNTEP4, three new processing options have been added to these applications that affect how they preprocess SQL input before handing it off for PREPARE:

- SQL specifies that SQL statements are to be preprocessed as in earlier versions of DB2. SPUFI, DSNTEP2, and DSNTEP4 accept multi-line SQL statements but copy them seamlessly into an SQL buffer for PREPARE, effectively converting the multi-line input to a single line. In addition, SQL comments are removed.
- SQLCOMNT specifies that SQL-style comments are to be passed in the statement buffer for PREPARE instead of stripped out and discarded (the default behavior of SPUFI, DSNTEP2, and DSNTEP4). A LF character is appended after the detected SQL comment if none are found in the remaining input buffer (the portion that would otherwise be discarded). The purpose is to prevent loss of SQL comments, to assist with diagnostics and debugging, and to retain the format of the SQL procedure.
- SQLPL, like SQLCOMNT, specifies that SQL-style comments are to be preserved. It also requests that a line formatting (LF) character is to be appended to each input line that ends without a token split. The purpose is to preserve the SQL comments and multi-line format of an SQL procedure in order to provide better diagnostics and debugging. It also helps retain the *format* of the SQL procedure when it is stored in the DB2 catalog.

To maintain multi-line granularity for debugging purposes and to preserve SQL comments, you can manually embed a line formatting character at the end of each line that ends with an undelimited token. Alternatively, you can use the new SQLPL option to have a line feed character x'25' added automatically.

With the option in effect, the above example is passed for PREPARE as:

```
SELECT * -- GET ALL COLUMNS<LF> FROM DSN8910.EMP<LF> WHERE EMPID = '000150';<LF>
```

Alternatively, you may wish to minimize modification of the input by preserving SQL comments but without adding line formatting characters. However, if SQL comments are to be preserved it is necessary to terminate them with line formatting characters because the parser otherwise cannot distinguish where they end, as the following example shows:

```
SELECT * -- GET ALL COLUMNS FROM DSN8910.EMP WHERE EMPID = '000150';
```

Use the SQLCOMNT option to specify that SQL comments are to be preserved, and terminated automatically by a line feed character if one is not provided in the source. Note that only SQL comments will be terminated. This is the difference between the SQLCOMNT option and the SQLPL option.

How to specify the SQL formatting

We examine the three different ways to specify the SQL format that you are using in your SQL statement.

SQL format on SPUFI defaults panel

The SPUFI defaults panel DSNESP02 has a new option 6 (that is, SQL format). Figure 7-50 shows the new appearance of the SPUFI defaults panel.

```
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
.
 DSNESP02                  CURRENT SPUFI DEFAULTS           SSID: DB9B
 ===>

  1  SQL TERMINATOR .. ===> #          (SQL Statement Terminator)
  2  ISOLATION LEVEL   ===> CS         (RR=Repeatable Read, CS=Cursor
Stability,
                                       UR=Uncommitted Read)
  3  MAX SELECT LINES  ===> 250        (Max lines to be return from SELECT)
  4  ALLOW SQL WARNINGS===> NO         (Continue fetching after sqlwarning)
  5  CHANGE PLAN NAMES ===> NO         (Change the plan names used by SPUFI)
  6  SQL FORMAT..... ===> SQL          (SQL, SQLCOMNT, or SQLPL)
 Output data set characteristics:
  7  SPACE UNIT ..... ===> TRK        (TRK or CYL)
  8  PRIMARY SPACE ... ===> 5          (Primary space allocation 1-999)
  9  SECONDARY SPACE . ===> 6          (Secondary space allocation 0-999)
 10  RECORD LENGTH ... ===> 4092       (LRECL=Logical record length)
 11  BLOCK SIZE ..... ===> 4096        (Size of one block)
 12  RECORD FORMAT ... ===> VB         (RECFM=F, FB, FBA, V, VB, or VBA)
 13  DEVICE TYPE ..... ===> SYSDA      (Must be DASD unit name)
 Output format characteristics:
 14  MAX NUMERIC FIELD ===> 33         (Maximum width for numeric fields)
 15  MAX CHAR FIELD .. ===> 20         (Maximum width for character fields)
 16  COLUMN HEADING .. ===> NAMES      (NAMES, LABELS, ANY or BOTH)
 PRESS:  ENTER to process    END to exit             HELP for more information
```

Figure 7-50 SPUFI defaults panel - DSNESP02

Functional comments on SQL statements

You can also select the SQL, SQLPL, and SQLCOMNT behavior by using a so-called functional comment directly in the SQL stream being processed by SPUFI, DSNTEP2, or DSNTEP4. The name of the control statement is SQLFORMAT.

The following example shows how to it to switch to SQLPL behavior and then to SQL behavior:

```
--#SET SQLFORMAT SQLPL
--#SET TERMINATOR #
SELECT * FROM /* THIS IS A COMMENT
SPANS ONTO THE NEXT LINE */
SYSIBM.SYSVOLUMES #
--#SET SQLFORMAT SQL
--#SET TERMINATOR ;
SELECT * FROM SYSIBM.SYSENVIRONMENT;
SELECT * FROM SYSIBM.SYSVOLUMES;
```

Note the use of the TERMINATOR control statement to change the SQL terminator character from the default semicolon to the pound sign, and then back to the semicolon.

New runtime option SQLFORMAT

When you execute DSNTEP2 or DSNTEP4, you can specify SQL, SQLCOMNT, or SQLPL behavior as the argument of a new runtime parameter called SQLFORMAT. Example 7-6 demonstrates the use of SQLFORMAT parameter to select SQLPL behavior when calling DSNTEP2.

Example 7-6 SQLFORMAT parameter

```
//DSNTEP2 EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(DB9B)
RUN PROGRAM(DSNTEP2) PLAN(DSNTEP91) +
LIB('DSN910.RUNLIB.LOAD') +
PARMS('/SQLFORMAT(SQLPL),SQLTERM(#)')
END
//*
//SYSIN DD *
CREATE PROCEDURE MY_SQLPROC
...
LANGUAGE SQL
...
P1: BEGIN NOT ATOMIC -- This is a comment
DECLARE MY_VAR CHAR(6) CCSID EBCDIC;
...
END P1!
```

Note also the use of the SQLTERM parameter to change the SQL terminator from a semicolon to a pound sign (#). Individual statements within the SQL procedure body are terminated with a semicolon, so a different character is required to mark where the SQL procedure itself terminates.

Archived

7.18.1 Change SPUFI plan names on SPUFI defaults panel

The following description does not really belong to native or external stored procedures, but as we are talking about enhancements regarding DB2 Interactive, we list it here. This functionality has been added for better usage of SPUFI. On the SPUFI defaults panel with panelid DSNESP02, you have an additional option 5, which lets you change the SPUFI plan names. Refer to Figure 7-50 on page 253 and find option 5 “Change the plan names used by SPUFI”. If you specify YES here and press Enter, a new panel DSNESP07 appears, as it is shown in Figure 7-51. This panel gives you the option to change plan names for your SPUFI execution based on isolation levels.

```
DSNESP07              CURRENT SPUFI DEFAULTS - PANEL 2       SSID: DB9B
===>

Enter the following to control your SPUFI session:
 1  CS ISOLATION PLAN   ===> DSNESPCS (Name of plan for CS isolation level)
 2  RR ISOLATION PLAN   ===> DSNESPRR (Name of plan for RR isolation level)
 3  UR ISOLATION PLAN   ===> DSNESPUR (Name of plan for UR isolation level)

 Indicate warning message status:
 4  BLANK CCSID WARNING ===> YES     (Show warning if terminal CCSID is blank)

PRESS:  ENTER to process    END to exit             HELP for more information
```

Figure 7-51 DSNESP07

With this option, you have much more flexibility when working with SPUFI.

Tip: One thing that you can very easily do with this option is to work with different application encoding schemes depending on what kind of data you want to access on your DB2 subsystems. Just bind additional SPUFI plans and packages using different application encoding schemes.

7.19 Differences from external SQL procedures

External SQL procedures have been around with DB2 for z/OS since Version 5. Starting with DB2 9, there are two different ways for you to work with SQL stored procedures. We now distinguish between native and external SQL procedures. This is the only type of SQL procedure that is available for versions of DB2 prior to Version 9. No changes have been made to the existing (external) SQL procedures functions. No new data types are supported. However, the new DML statements are included.

In this section we point out the changes to parameters that existed with external SQL procedures.

FENCED keyword on CREATE PROCEDURE statement

The FENCED keyword is not really new on CREATE PROCEDURE SQL statements. It has been used as the default for SQL and other stored procedures. The meaning of FENCED is that the procedure runs in an external address space, which typically used the assigned WLM stored procedure address space. In addition to that, FENCED also specifies that the SQL procedure program is an MVS load module with an external name.

Starting with DB2 V9, if you specify FENCED or EXTERNAL together with LANGUAGE SQL on a CREATE PROCEDURE statement, you are asking for the *creation of an external SQL procedure*.

WLM ENVIRONMENT

If WLM ENVIRONMENT is specified for native SQL procedures without the FOR DEBUG keywords, an error is issued. This means that if one of the forms of the WLM ENVIRONMENT clause supported for external SQL procedures is specified (WLM ENVIRONMENT name or WLM ENVIRONMENT (name,*)), then an error is issued. If WLM ENVIRONMENT is specified for a native SQL procedure, WLM ENVIRONMENT FOR DEBUG must be specified.

For compatibility with previous versions, the following alternatives can be used:

- For the DYNAMIC RESULT SETS clause, the keyword DYNAMIC is optional and the keyword SET can be specified instead of SETS.
- For DETERMINISTIC, NOT VARIANT can be used, and for NOT DETERMINISTIC, VARIANT can be used.
- For CALLED ON NULL INPUT, NULL CALL can be used.

Incompatible differences

We list the major differences in this section. For details see *DB2 Version 9.1 for z/OS Installation Guide,* GC18-9846.

Resolution of parameter names, variable names, and column names in SQL procedures

In V9 the rules used for name resolution within a native SQL procedure differ from the rules that were used for SQL procedures in prior releases. Because an SQL parameter or SQL variable can have the same name as a column name, you should explicitly qualify the names of any SQL parameters, SQL variables, or columns that have non-unique names.

For more information about how the names of these items are resolved, see the topic "References to SQL parameters and SQL variables" in the *DB2 Version 9.1 for z/OS SQL Reference,* SC18-9854. The rules used for name resolution within external SQL procedures remain lunchanged.

Changed behavior of CREATE PROCEDURE for an SQL procedure

With the introduction of native SQL procedures in Version 9.1, the semantics of the CREATE PROCEDURE statement for an SQL procedure have changed. Starting in Version 9.1, all SQL procedures that are created without the FENCED option lor the EXTERNAL option in the CREATE PROCEDURE statement are native SQL procedures. In previous releases of DB2, if you did not specify either of these options, the procedures were created as external SQL procedures.

If you do specify FENCED or EXTERNAL, the meanings are the same as in previous releases of DB2. Both of these keywords mean that an external SQL procedure is to be created.

Changed messages from SQL procedures

In Version 9.1, DB2 issues different messages for the new native SQL procedures than it does for external SQL procedures. For external SQL procedures, DB2 continues to issue DSNHxxxx messages. For native SQL procedures, DB2 issues SQL return codes.

7.20 Unified Debugger

Users investing in DB2 application development have a need for formal debugging facilities. This need is even intensified when working on DB2 stored procedure development since the code runs in isolation at a DB2 server.

For classic languages, such as COBOL and C on the one hand, the compiler products and their associated runtime facilities provide debug capabilities. On the other hand, we have interpreted languages such as SQL and JAVA where debugging sometimes has caused some problems. The Unified Debugger that has been introduced with DB2 9 focuses on these newer languages.

With the Unified Debugger, you can observe the execution of SQL procedure code, set breakpoints for lines, and view or modify variable values. The Unified Debugger supports external and native SQL procedures, including nested stored procedures. To use the Unified Debugger with SQL procedures, you must include breakpoints in your routines or executable files.

The Unified Debugger builds upon and includes the SQL Debugger technology from DB2 Version 9. The name has been changed to Unified Debugger to embrace the unified support for both LANGUAGE SQL and JAVA stored procedure debugging. Following this, one advanced capability that is now offered to you through the Unified Debugger is the ability to debug nested stored procedures of language sharing the same client application call stack. This means that users debugging a JAVA routine can step into and debug a called SQL procedure.

The Unified Debugger itself is *middleware*. It originates from, and is serviced by, DB2 LUW V9. This middle layer code is distributed to the DB2 server platforms as object code (for running on each of the various operating systems that host DB2). There are three basic elements:

- The Unified Debugger server library - APIs that provide the interface for the DB2 servers and the supported routine objects
- The session manager and the Unified Debugger routers stored procedures that implement the client interface to the Unified Debugger at DB2 servers. This layer is independent of any particular platform or DB2 server type.
- A debug client that is written to support the Unified Debugger middleware. This client is only based on LUW clients.

7.20.1 Setting up the Unified Debugger components

There are a couple of steps that you need to go through in order to being able to use the Unified Debugger functionality.

Install the Developer Workbench Client

To install the Developer Workbench, just execute the LAUCHPAD.EXE files that comes with the product. Unlike some other products, there is hardly anything you need to decide during your installation. See “DB2 Management Clients Package” on page 427 for packaging of this tool.

Run post - install job DSNTIJSD

You must run post-install job DSNTIJSD to create DB2 server objects that are required when using the Unified Debugger or the SQL Debugger.

When we say that this is a post-install job, it means that after stepping through the installation/migration panels, you will not find a customized version of this job on your hlq.NEW.SDSNSAMP library. Instead, you must copy this job from the DSN910.SDSNSAMP library and customize it manually. A description of what you must change in order to get it to work is provided within the job.

If you run this job, it will create the following DB2-supplied stored procedures for you:

- DB2DEBUG.DEBUGGERLEVEL
- DB2DEBUG.CREATE_SESSION
- DB2DEBUG.DESTROY_SESSION
- DB2DEBUG.QUERY_SESSION
- DB2DEBUG.LIST_SESSION
- DB2DEBUG.PUT_COMMAND
- DB2DEBUG.GET_REPORT
- SYSPROC.DBG_INITIALIZECLIENT
- SYSPROC.DBG_TERMINATECLIENT
- SYSPROC.DBG_SENDCLIENTREQUESTS
- SYSPROC.DBG_SENDCLIENTCOMMANDS
- SYSPROC.DBG_RECVCLIENTREPORTS
- SYSPROC.DBG_ENDSESSIONMANAGER
- SYSPROC.DBG_PINGSESSIONMANAGER
- SYSPROC.DBG_LOOKUPSESSIONMANAGER
- SYSPROC.DBG_RUNSESSIONMANAGER

Make sure that you specify an appropriate WLM environment for every single stored procedure. The recommendation is that you should use NUMTCB > 5 for the application environment that you are going to assign to them.

Attention: Stored procedure DBG_RUNSESSIONMANAGER must run as an authorized program in an authorized environment.

Run post-install job DSNTIJMS

This post-install job create objects required for the DB2 JDBC and ODBC metadata methods.

Again, you will not find a customized version of this job on your hlq.NEW.SDSNSAMP library. Instead, you must copy this job from the DSN910.SDSNSAMP library and customize it manually. A description of what you must change in order to get it to work is provided within the job.

As a result of executing this job, you will have some packages bound on your system and the following stored procedures will be created:

- SYSIBM.SQLCOLPRIVILEGES
- SYSIBM.SQLCOLUMNS
- SYSIBM.SQLFOREIGNKEYS
- SYSIBM.SQLPRIMARYKEYS
- SYSIBM.SQLPROCEDURECOLS
- SYSIBM.SQLPROCEDURES
- SYSIBM.SQLSPECIALCOLUMNS
- SYSIBM.SQLSTATISTICS
- SYSIBM.SQLTABLEPRIVILEGES
- SYSIBM.SQLTABLES
- SYSIBM.SQLGETTYPEINFO
- SYSIBM.SQLUDTS
- SYSIBM.SQLCAMESSAGE

Note: The JDBC metadata routine setup job (DSNTIJSM) has no direct relationship to debugging. Rather, that job is related to setup for JDBC (JCC connected) clients, such as DWB, merely to support operations such as run, meaning the calling of a stored procedure.

Set up the session manager - example on z/OS

You can have the session manager run on any platform that you prefer to. In our case, we have decided to set up the session manager on z/OS. You can think of the session manager as a daemon that in our case is running as a z/OS started task that is waiting for work to perform. Setting up the session manager on z/OS basically consists of three steps, which we describe next.

Step 1: RACF definitions

The RACF definitions shown in Figure 7-52 must be added to your z/OS security system.

```
//RACFDEF  EXEC TSOBATCH
//SYSTSIN DD *
  ALTUSER stcuserid OMVS( UID(5) HOME('/u/stcuserid') PROGRAM('/bin/sh') )
  RDEFINE STARTED DB2UDSMD.** STDATA(USER(stcuserid))
  SETROPTS RACLIST(STARTED) REFRESH
```

Figure 7-52 RACF definitions for Session Manager

The STARTED task is defined by a RACF profile named DB2UDSMD.**.

Important: It is mandatory to use DB2UDSMD as the started task name. The session manager is not tied to a specific DB2 subsystem, nor is it tied to any DB2 subsystem at all.

stcuserid was designated to be the ID associated with this started task. Since the task will run a Java program from OMVS, also assign an OMVS segment definition to the user (that is, UID, home dir, and so on). Finally, activate the STARTED task definition in current memory.

Archived

Step 2: Create a file in HFS to hold environment settings

The job shown in Figure 7-53 is used to create a file in the HFS to hold the environment settings used when the Unified Debugger Session Manager runs as a started task on z/OS.

```
//*-------------------------------------------------------------------
//* Create a file in the HFS to hold the Environment settings used when
//* the Unified Debugger Session Manager runs as a Started Task on z/OS
//*
//* stcuserid is the ID associated with the Started Task.
//* Place the file in that users home directory.
//* Name the file DB2UDSMDprofile
//*-------------------------------------------------------------------
//*-------------------------------------------------------------------
//* Create a file in the HFS from inline data using COPY
//*-------------------------------------------------------------------
//OCOPY    EXEC PGM=IKJEFT01,DYNAMNBR=30
//SYSTSPRT DD SYSOUT=*
//HFSOUT   DD  PATH='/u/stcuserid/DB2UDSMDprofile',
//             PATHOPTS=(OWRONLY,OCREAT,OAPPEND,OTRUNC),
//             PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIROTH)
//INLINE   DD *
#-------------------------------------------------------------------
# Environment settings for running the Unified Debugger Session Manager
# * _BPX_BATCH_SPAWN=NO
# * _BPX_SHAREAS=YES
#     Arrange for the JVM to run in the same address space. This avoids
#     launching 2 additional address spacss for the Started TasK.
# * CLASSPATH=
#     The location of the UDBG Session Manager jar file
# * JAVA_COMPILER=NONE
#     Disable the JIT. The Started Task runs the Session Manager only
#     one time, so disabling this saves space that will not be used.
#-------------------------------------------------------------------
_BPX_BATCH_SPAWN=NO
_BPX_SHAREAS=YES
CLASSPATH=/usr/lpp/db2910_base/DB9B/classes/db2dbgm.jar
JAVA_COMPILER=NONE
//SYSTSIN DD *
OCOPY INDD(INLINE) OUTDD(HFSOUT) TEXT
//
```

Figure 7-53 Job to create a file in HFS to hold the environment settings

Create a file in the HFS to hold the environment settings used when the Unified Debugger Session Manager runs as a started task on z/OS.

An ID must be designated to be associated with the started task. Suppose that stcuserid is that ID. Place a file in that user's home directory to serve as an execution environment profile. The file must point to the location of the SessionManager jar file, db2dbgm.jar. Name the file something distinctive, such as DB2UDSMDprofile.

In both Figure 7-53 and Figure 7-54 on page 261 we have used stcuserid as the place holder for the ID associated with started task DB2UDSMD that you must change to your specific situation.

The use of a named HFS application profile is suggested for simple setup and segregation of duties. At a minimum it needs to define the CLASSPATH to the Session Manager Java program. Other settings to tune the Java execution environment can be included. Note that

BPXBATCH reads the STDENV file, so no shell script symbol substitution can be utilized here. Symbol substitution processing is only available to the user profile (.profile) for the started task user ID and scripts executed from the shell command line.

The Session Manager is independent of DB2, so it can run anywhere in the network. But the server platform (that is, the operating system that the stored procedure that you want to debug runs) is often a better default choice than running at the client workstation. The session Manager JAR file is now distributed on all server platforms, so it does not have to be obtained, downloaded, sent, pulled, pushed, or transported by you.

Step 3 - Create a started task JCL

Create the Started Task JCL for DB2UDSMD and place it in the system proclib. This is used to launch the Unified Debugger Session Manager on z/OS. stcuserid is the ID associated with this started task, as defined in the RACF STARTED class profile DB2UDSMD.**.

```
//DB2UDSMD PROC PORT=4553,TIMEOUT=60
//*
//*    DB2 Unified Debugger Session Manager DAEMON FOR OPENEDITION
//*
//*    This JCL assumes no .profile exists for the OMVS user.
//*      * Environment settings (CLASSPATH) come from a named profile.
//*      * Explicitly identify the java edition to run. PATH cannot be
//*        controlled by the environment file.
//*
//DB2UDSMD EXEC PGM=BPXBATCH,DYNAMNBR=128,REGION=0M,
//             TIME=1440,PARM='SH date;/usr/lpp/java140/J1.4/bin/java
//             com.ibm.db2.psmd.mgr.Daemon -timeout &TIMEOUT -port
//             &PORT;date'
//STDOUT   DD  PATH='/tmp/DB2UDSMD.stdout',
//             PATHOPTS=(OWRONLY,OCREAT,OAPPEND,OTRUNC),
//             PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIROTH)
//STDERR   DD  PATH='/tmp/DB2UDSMD.stdout',
//             PATHOPTS=(OWRONLY,OCREAT,OAPPEND,OTRUNC),
//             PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIROTH)
//STDENV   DD  PATH='/u/stcuserid/DB2UDSMDprofile',
//             PATHOPTS=ORDONLY
```

Figure 7-54 Sample started task JCL for the Unified Debugger Session Manager on z/OS

Note: BPXBATCH does not derive PATH from STDENV. For the session manager, PATH only needs to point to the java runtime. One approach is to specify the PATH directly on the command line, as shown in Figure 7-54. Another method requires the use of a shell script profile (.profile) for the started task user, which we have not included in this documentation. Note that the job step options in the JCL as shown, which includes the OMVS shell command, were carefully arranged to efficiently utilize the limited space available. The space is limited to 100 characters in total. As shown in Figure 7-54, there are still about 18 characters left to adjust the path specification for Java.

Start the session manager started task on z/OS

After you have set up everything properly, you can verify that everything is okay using the /S DB2UDSMD operator command to start your Unified Debugger session manager started task on z/OS.

Grant DEBUGSESSION privilege

Grant the DEBUGSESSION privilege to the user that runs the debug client. The DEBUGSESSION privilege is a new system authorization. Refer to the new catalog column DEBUGSESSIONAUTH on table SYSIBM.SYSUSERAUTH to obtain information about who has already been granted this privilege.

Prepare your stored procedures for debugging

After you have successfully set up the environment for debugging your SQL stored procedures, you must now decide for every single stored procedure if you want to debug it.

For a native SQL procedure, define the procedure with the ALLOW DEBUG MODE option and the WLM ENVIRONMENT FOR DEBUG MODE option.

For an external SQL procedure, use DSNTPSMP or the Development Center to build the SQL procedure with the BUILD_DEBUG option.

For a Java procedure, define the procedure with the ALLOW DEBUG MODE option, select an appropriate WLM environment for Java debugging, and compile the Java code with the -G option.

Tip: For more information about the Unified Debugger, see information about the DB2 Developer Workbench in the DB2 Database for Linux, UNIX, and Windows information center at:

`http://publib.boulder.ibm.com/infocenter/db2help/index.jsp`

Archived

8

XML

eXtensible Markup Language (XML) is a World Wide Web Consortium (W3C) standard for the interchange of structured data.

XML, a revamping and Internet-driven document mark-up language derived from the Standard Generalized Markup Language (SGML), a key technology for data exchange. SGML is a descendant of IBM Generalized Markup Language (GML), which was used to script documents. XML is a specific subset of SGML, designed to be simpler and to allow better internationalization for data description and formatting of data exchange.

While XML has a look and feel similar to Hypertext Markup Language (HTML), the purpose of XML is dramatically different. HTML describes how information is to be presented (for example, HTML is used to create the display of the Web pages you look at), but XML describes its actual meaning.

XML is not a fixed format database (although it can contain data and database elements). Rather, it is a language that uses tags to describe data. Since XML is not fixed format, it can combine many diverse types of data into many diverse structures. This lack of fixed format makes it easy to exchange data between different companies and different hardware and software technologies

IBM, recognizing the need for XML support, has introduced, with DB2 9 for z/OS as well as with DB2 for Linux, UNI,X and Windows *pureXML*, a collection of XML capabilities that are built into the DB2 9 family. DB2 is now an hybrid database fully supporting relational and XML data.

This chapter provides a brief description of the XML functions introduced by DB2 9 for z/OS.

The XML Extender is still a feature of DB2, but we highly recommend investing in the new direct data type support for functionality and performance reasons.

This chapter discusses the following:

- Why use XML with DB2 for z/OS
- PureXML
- XML support
- Accessing XML data

© Copyright IBM Corp. 2007. All rights reserved.

- XPath
- XML indexing
- XML schema support
- XML schema validation
- XML decomposition
- Utilities support of XML data

Archived

8.1 Why use XML with DB2 for z/OS

Today, XML is predominant in most organizations and hosts an abundance of business information on public and private Web sites. This is because XML is vendor and platform independent, and is a very flexible data model for structured data, semi-structured data, and schema-less data. It is also self-describing and easy to extend. Furthermore, XML can be easily transformed into other XML documents or even into different formats such as HTML. Therefore, XML is the de facto standard for exchanging data across different systems, platforms, applications, and organizations.

Beyond XML for data exchange, enterprises are keeping large amounts of business-critical data permanently in XML format. This has various reasons, such as a need to keep it for auditing and regulatory compliance. Also, in life science applications, for example, the data is highly complex and hierarchical in nature and yet may contain significant amounts of unstructured information. Most of today's genomic data is still kept in proprietary flat file formats, but major efforts are under way to move to XML. These proprietary flat files can be accessed using WebSphere Federated Server technology.

Web services and SOA use XML as the base for most services and data. Almost every implementation of service-oriented architecture (SOA) includes XML at some point.

We can say that XML is the key technology for:

- Data exchange
- Data integration
- Data evolution and flexibility
- Service-oriented architectures
- Information on demand

DB2 for z/OS support for XML lets your client applications manage XML data in DB2 tables. You can store well-formed XML documents in their hierarchical form, and retrieve all or portions of those documents.

DB2 provides significant new capabilities for supporting XML, including a new XML data type and underlying engine-level components that automatically store and process XML data.

Because the stored XML data is fully integrated into the DB2 database system, you can access and manage the XML data by leveraging DB2 functionality.

To efficiently manage traditional SQL data types and XML data, DB2 uses two distinct storage mechanisms. However, the underlying storage mechanism that is used for a given data type is transparent to the application. The application does not need to explicitly specify which storage mechanism to use, or to manage the physical storage for XML and non-XML objects.

XML document storage

The XML column data type is provided for storage of XML data in DB2 tables. Most SQL statements support the XML data type. This enables you to perform many common database operations with XML data, such as creating tables with XML columns, adding XML columns to existing tables, creating indexes over XML columns, creating triggers on tables with XML columns, and inserting, updating, or deleting XML documents. Alternatively, a decomposition stored procedure is provided that lets you extract data items from an XML document and store those data items in columns of relational tables using an XML schema that has been annotated with instructions on how to store the data items.

Archived

XML document retrieval

You can use SQL to retrieve entire documents from XML columns, just as you retrieve data from any other type of column. When you need to retrieve portions of documents, you can specify XPath expressions, through SQL with XML extensions (SQL/XML).

Application development

Application development support of XML enables applications to combine XML and relational data access and storage. The following programming languages support the new XML data type:

- Assembler
- C and C++ (embedded SQL or DB2 ODBC)
- COBOL
- Java (JDBC and SQLJ)
- PL/I

Database administration

DB2 for z/OS database administration support for XML includes the following items:

- XML schema repository (XSR) - The XML schema repository (XSR) is a repository for all XML schemas that are required to validate and process XML documents that are stored in XML columns or decomposed into relational tables.
- Utility support - You can use DB2 for z/OS utilities on XML objects. The utilities handle XML objects in a similar way to the way that they handle LOB objects. For some utilities, you need to specify certain XML keywords.

Performance

Indexing support is available for data stored in XML columns. The use of indexes over XML data can improve the efficiency of queries that you issue against XML documents. An XML index differs from a relational index in that a relational index indexes an entire column, while an XML index indexes part of the data in a column. You indicate which parts of an XML column are indexed by specifying an XML pattern, which is a limited XPath expression.

Note that XQuery is not supported in DB2 9 for z/OS. An XPath expression is used to identify portions of an XML document and is used in XMLQUERY(), XMLEXISTS(), and XMLPATTERN of CREATE INDEX. DB2 9 for z/OS provides core XML query language features critical for application development, and will expand them into the full XQuery language in the follow-on releases.

8.1.1 DB2 XML Extender

Up to DB2 V8, when you needed to store XML data in a DB2, traditionally you stored the entire XML document as a CLOB or VARCHAR in a column, or decomposed the XML document into rows and columns, or shred the XML into an edge table. DB2 XML Extender is a fully integrated component of DB2 that provides data types to store XML documents in DB2 databases and functions to work with these structured documents. DB2 manages these documents and stores them as character data or external files. You can retrieve complete documents or individual elements via the functions provided by XML Extender. DB2 XML Extender provided rich functionality to store and retrieve XML documents. However, storing XML documents intact in a column or shredding them to rows in tables still creates a number of problems:

- The decomposition or shredding methods attempted to fit XML into DB2, which stored the data using the relational model. Usability and performance became an issue, as the shredded data is no longer in XML and becomes unmanageable. Storing the document as a CLOB or VARCHAR column prevents XML parsing at insert.

- Re-construction of the XML document is difficult and expensive. In addition, it does not make sense to decompose or shred the document when the document always needs to be re-composed before it can be used efficiently.

Therefore, XML Extenders work well, but they do not provide good performance and flexibility. In addition, it introduces administrative challenges and does not scale well for large, complex applications. You will now learn how the new native XML support in DB2 V9.1 will solve these problems.

Note: XML Extender is still supported in DB2 Version 9.1 for z/OS but it has been deprecated. We recommend that you to consider the XML native support for all new applications.

8.2 PureXML

DB2 9 for z/OS offers leading-edge technology for storing, managing, and searching XML data in a secure, highly scalable environment. You can now seamlessly integrate XML with their existing relational data, exploit both tabular and hierarchical data models, and enjoy the flexibility of using both SQL and XPath in your applications. To provide this first-class support for managing XML data, DB2 features new storage management, indexing, and optimization techniques. It also interfaces to a wide range of popular programming languages, allows users to optionally validate their XML data prior to storage, and extends popular database utilities important for importing data and administering the environment.

The new XML support in DB2 manages XML in a hierarchical format, representing the XML data model in a natural way. This provides key advantages over existing offerings that require XML data to be stored as large objects, which can be expensive to search and update, or decomposed across multiple relational tables, which requires complex schema mappings and query statements.

8.2.1 Platform integration and cost of ownership

In z/OS V1.8, IBM introduced a new system component of z/OS, z/OS XML System Services (z/OS XML), a system-level XML parser integrated with the base z/OS operating system and designed to deliver an optimized set of services for parsing XML documents (z/OS XML has also been made available on z/OS V1.7). The initial beneficiaries of this system component are middleware and applications requiring high-performance non-validating XML parsing. z/OS XML may currently be accessed by an Assembler programming interface, and one of the first exploiters, DB2 9 for z/OS, uses this Assembler interface for XML native support. IBM plans to add C/C++ support for z/OS XML with z/OS V1.9. DB2 9 for z/OS XML support requires z/OS XML System Services (XML z/OS). z/OS XML requires either z/OS V1.8 or z/OS V1.7 with APAR OA16303. In addition, use of XML schemas requires IBM 31-bit SDK for z/OS, Java 2 Technology Edition V5 (5655-N98) (SDK5).

With the April 18 Statement of Direction concerning z/OS XML System Services, IBM announced its intent to enable the z/OS XML component to take advantage of specialty engines:

- Middleware and applications requesting z/OS XML System Services (for example, DB2 processing via local connection) will have z/OS XML System Services processing execute on the zAAP. Specifically, all z/OS XML System Services parsing executing in TCB mode will be redirected to the zAAP.

- In addition, the z/OS XML component will fully take advantage of zIIPs, when present. With respect to DB2, z/OS XML processing may be partially directed to zIIPs when utilized as part of a distributed request (like DB2 DRDA). The future enhancement will further benefit DB2 by directing the full amount of the z/OS XML System Services processing to zIIPs when it is utilized as part of any zIIP eligible workload (like DRDA). Specifically, all z/OS XML System Services parsing that are executed in SRB mode from zIIP-eligible enclaves will be redirected to the zIIP.
- IBM intends to extend and expand on the use of z/OS XML System Services enabled for zAAP specialty processors as the basis for future enhancements:
  - IBM intends to enhance the XML Toolkit for z/OS so that eligible workloads may exploit the z/OS XML component. This extends zAAP exploitation to the XML Toolkit for z/OS.
  - IBM intends to add validating parsing to the z/OS XML component. This extends zAAP exploitation for XML validating parsing as well.

8.3 XML support

Figure 8-1 shows that, if we talk about DB2 now, we could also call it a hybrid database server. As you can see on the right, DB2 stores both relational and XML documents natively. Client applications can access both relational and XML data using SQL with XML extensions or XPath, which is parsed by the native XML interface. We discuss the XML publishing functions and XPath later in this chapter in more detail. XPath is a subset of XQuery. XQuery is a very powerful language that allows you to process XML data. With DB2 V9 for z/OS, you will not get full XQuery support, but you need to use XPath to get equivalent functions.

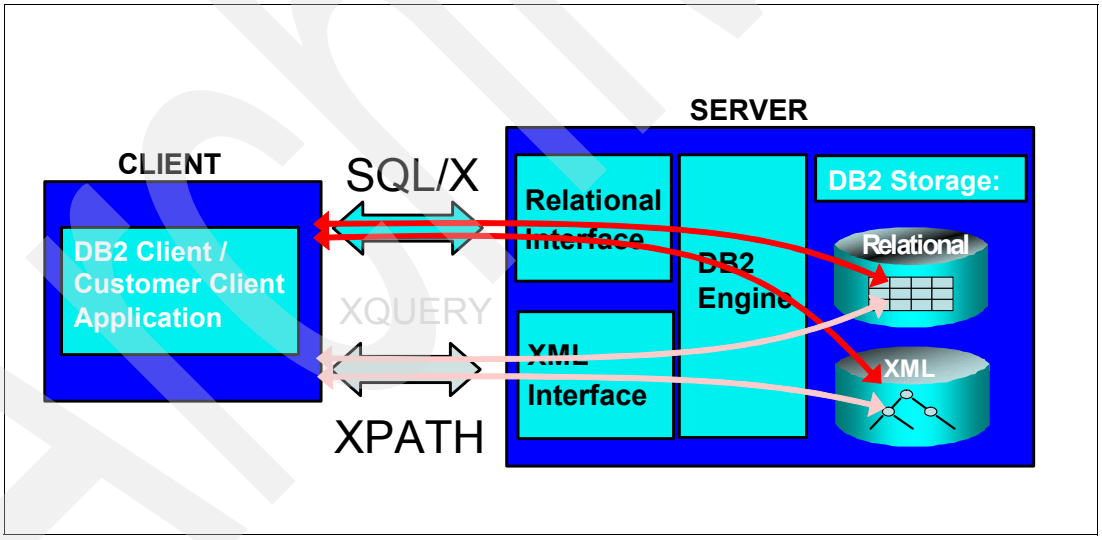


Figure 8-1 Hybrid data server

Note: XML Extender is still supported in DB2 Version 9.1, but we recommend that you consider the XML native support for all new applications.

Structure of an XML document

As you know, XML documents are typically represented as a hierarchical tree of nodes. The conceptual depiction of XML is important to understand, and we will refer back to it later in this chapter multiple times.

Figure 8-2 shows the types of XML nodes along with a sample XML document.

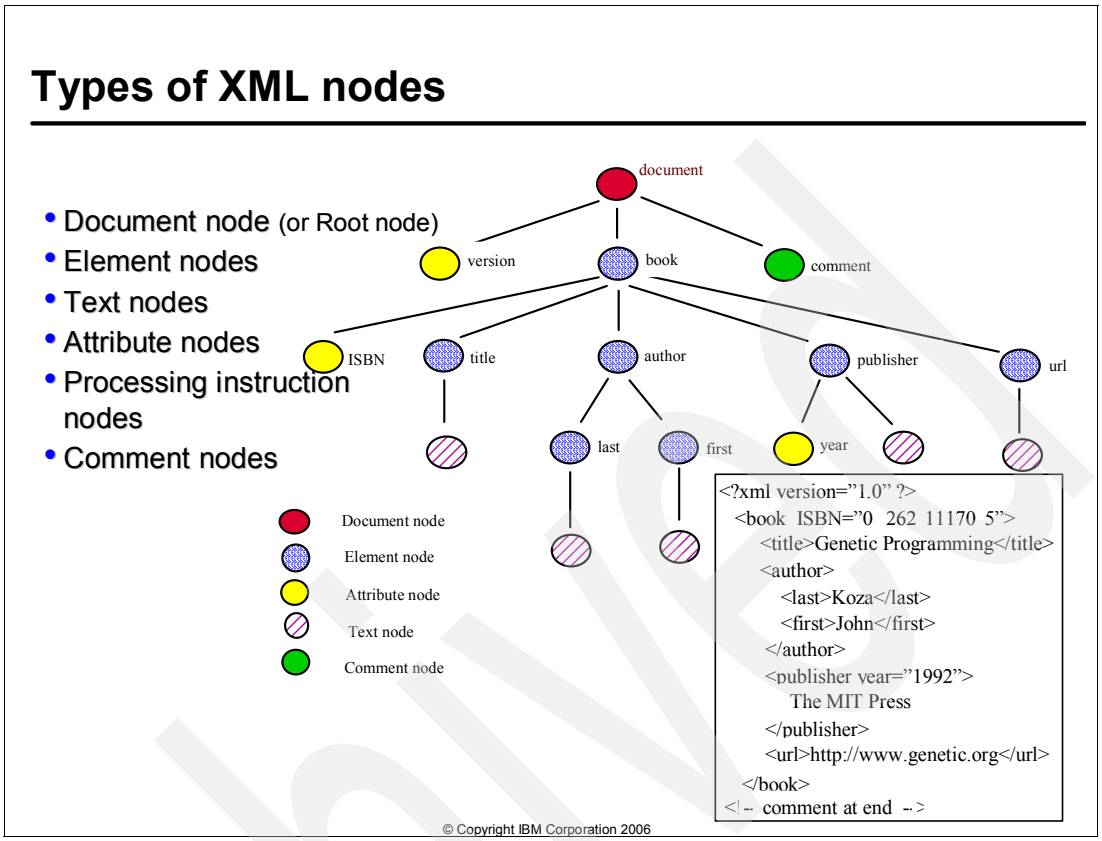


Figure 8-2 Types of XML nodes

The root node, sometimes also referred to as the document node, is the root of the tree. It does not occur anywhere else in the tree. The document element node for the document is a child of the root node. There are six nodes types in XML:

- Document nodes
- Element nodes
- Text nodes
- Attribute nodes
- Processing instruction nodes
- Comment nodes

We discuss these nodes later in this chapter when we look more closely at the XML data model in 8.5.1, "XML Data Model (XDM)" on page 283.

Comparison of various ways to store XML data

DB2 provides four ways to store XML. DB2 V8 supports all methods except the native XML storage. Only DB2 V9 provides native XML storage:

- CLOB/VARCHAR: stored as linear text in database as a CLOB column
- Shred: decomposed into rows and columns
- Edge: decomposed into a relational model based on graph theory
- Native: parsed, stored as interconnected nodes on fixed size database pages

Archive

Table 8-1 shows the performance attributes among these store types.

Table 8-1 Performance involving XML stores

Measure	CLOB	Shred	Edge	Native
Schema flexibility	Best	Bad	Best	Best
Search performance	Bad	Good	Bad	Best
Full document return performance 1)	Best	Bad	Bad	Good
Partial document return performance	Bad	Good	Good	Best
Insert performance	Best	Bad	Bad	Good
Partial document Update performance	Bad	Good	Bad	Good
Delete performance	Best	Bad	Bad	Good
1) Assuming the document is located from the base table index				

8.3.1 The XML data type

When we talk about native XML support, one important part is the new data type XML. It is a new SQL data type. This data type can be used in a CREATE TABLE statement. Refer to 8.3.2, “XML storage infrastructure” on page 271, to learn more about the usage of XML data type.

Any column of XML data type can hold one well-formed XML document for every row of the table.

Well-formed XML documents

DB2 only allows you to store so-called well-formed XML documents in columns with data type XML. An XML document is considered being well-formed if it conforms to the following requirements:

- An XML document must be contained in a single element, called the root element.
- End tags are required.
- Elements cannot overlap. In other words, they should be properly nested.
- Attributes must have values, and these values must be enclosed within quotation marks. You can use either single or double quotes, just as long as you are consistent. If the value of the attribute contains a single or double quote, you can use the other kind of quote to surround the value, or use the entities &quot: for a double quote and &apos: for a single quote.

Archive

- Symbols such as < cannot be used. Instead, use predefined entities. These are:
  - &lt: for less-than sign
  - &gt: for greater-than sign
  - &amp: for an ampersand
- Elements are case-sensitive. If you try to end an <a> element with a </A> tag, you get an error.
- Comments can appear anywhere in the document. They can even appear before or after the root element. A comment begins with <!-- and ends with --> . A comment cannot contain a double hyphen (--) except at the end. With that exception, a comment can contain anything.

  Most importantly, any markup inside a comment is ignored. If you want to remove a large section of an XML document, simply wrap that section in a comment. To restore the commented-out section, simply remove the comment tags. Refer also to Table 8-2, which shows sample cases for listed items 1–6.

Table 8-2 Well-formed XML document examples

Attribute	Not well formed	Well formed
Has exactly one root element	Canada <c>Germany</c>	<a> Canada <c>Germany</c>
Each opening tag is matched by a closing tag	<a>Germany	<a>Germany
All elements are properly nested	<a>Canada	<a>Canada
Attribute values must be quoted		
Does not use disallowed characters in tags or values	<a> 3<5 	<a> 3<5
Elements are case sensitive	<a id=”15”></A>	

All information stored in columns that exist in XML data type are encoded in Unicode UTF-8. XML data type values are processed in a representation that is not a string, and it is not directly comparable to string values.

You can insert XML documents using the ordinary INSERT SQL statement and specify the whole document in quotes. You can also select the contents of a specific XML column just using the SELECT statement. If, however, you would, for example, like to control preserve or strip whitespace, you would have to use the new XML publishing function XMLPARSE to populate an XML column value. Also, if you would like to transform the XML data type into any other string value such as CLOB, BLOB, or DBLOB, you can use the new publishing function XMLSERIALIZE to do so. Read more about XMLPARSE and XMLSERIALIZE in 8.4.1, “SQL/XML constructors” on page 275.

8.3.2 XML storage infrastructure

As mentioned above, starting with DB2 V9, a new data type, XML has been added that you can assign to a table column in order to store XML documents natively.

CREATE TABLE

A CREATE TABLE statement is shown in Figure 8-3. The result of this statement is a table consisting of two columns, character column C1 and column XMLCOL with data type XML.

```
CREATE TABLE SABIXML1 (C1 CHAR(10), XMLCOL XML)
```

Figure 8-3 CREATE TABLE with XML column

As a result of issuing the CREATE TABLE statement shown in Figure 8-3, the following objects have also been created implicitly by DB2 to support the XML column:

- A column called DB2_GENERATED_DOC_ID_FOR_XML. We refer to this column as DocID column from now on. DocID uniquely represents each row. This column is hidden. For each table, DB2 only needs one DocID column even if you would add additional rows with data type XML. This DocID is defined as *generated always*, meaning that you cannot update the docid column.
- A unique index on the DocID column that is defined as NOT NULL. This index is known as a document ID index.
- An XML table space. The implicitly created table space has characteristics depending on the base table space, as summarized in Table 8-3.

Table 8-3 XML - Base table and auxiliary table spaces

Base table space	Auxiliary XML table space	Notes
Simple	UTS partitioned by growth	No DPSI allowed.
Segmented	UTS partitioned by growth	No DPSI allowed.
Partitioned	UTS partitioned by range	If a row changes partition in the base table, the XML document moves as well.
UTS partitioned by range	UTS partitioned by range	If a row changes partition in the base table, the XML document moves as well.
UTS partitioned by growth	UTS partitioned by growth	An XML document can span partitions.

  Independent of wether a Segmented, Partitioned of Universal table space is created, the XML table space uses the Unicode UTF-8 encoding scheme.
- An XML table with columns docid, min_nodeid, and xmldata.
- A NodeID index on the XML table with key DocID and min_nodeid.

Archived

The above-mentioned list is also visualized in Figure 8-4.

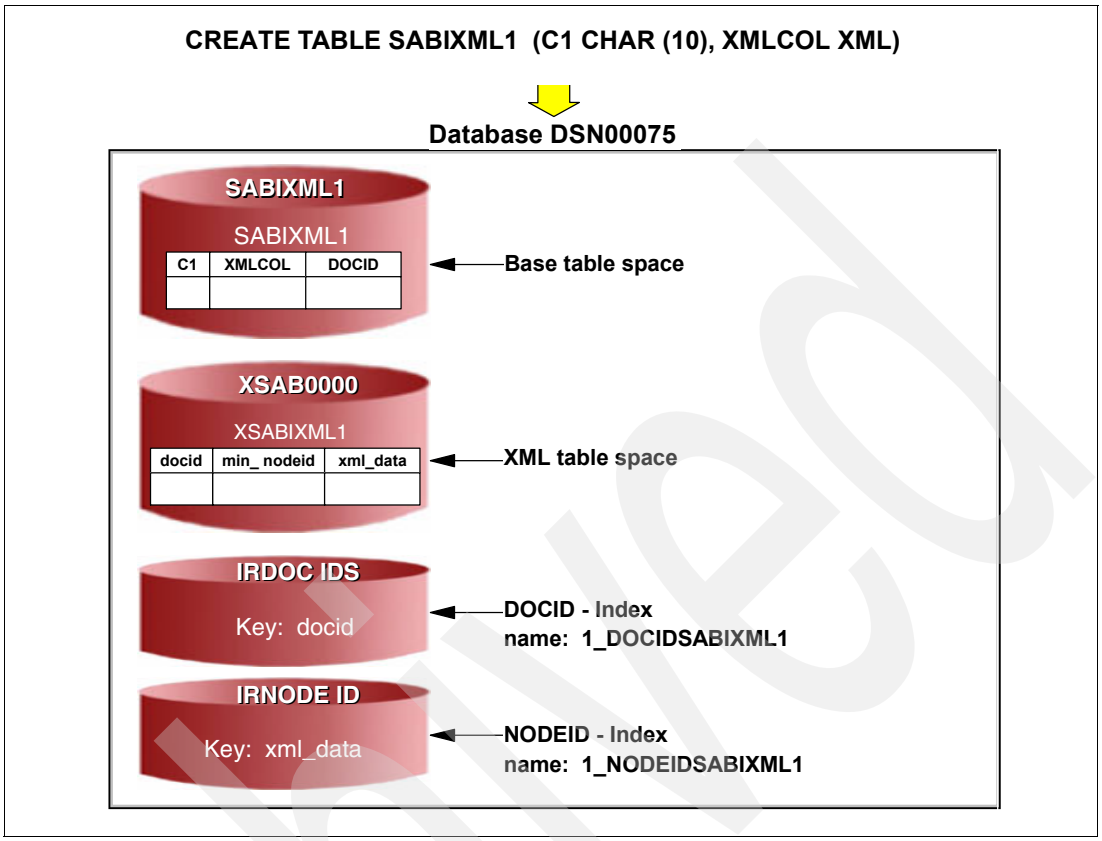


Figure 8-4 Implicitly and explicitly created objects for XML column

You can only perform limited SQL operations on the implicitly created objects. Refer to ALTER TABLE, ALTER TABLESPACE, and ALTER INDEX in the *DB2 Version 9.1 for z/OS SQL Reference*, SC18-9854 for a complete list of operations that you can perform.

Archived

Figure 8-5 shows the output of the -DIS DB command for the objects shown in Figure 8-4 on page 273. As you can see, the XML table space can easily be identified, because its type is not TS but XS.

```
DSNT360I  -DB9B *********************************
DSNT361I  -DB9B *  DISPLAY DATABASE SUMMARY
                *    GLOBAL
DSNT360I  -DB9B *********************************
DSNT362I  -DB9B     DATABASE = DSN00075  STATUS = RW
                  DBD LENGTH = 4028
DSNT397I  -DB9B
NAME     TYPE PART  STATUS            PHYERRLO PHYERRHI CATALOG  PIECE
-------- ---- ----- ----------------- -------- -------- -------- -----
SABIXML1 TS    0001 RW
XSAB0000 XS    0001 RW
IRDOCIDS IX    L*   RW
IRNODEID IX    L*   RW
******* DISPLAY OF DATABASE DSN00075 ENDED      **********************
DSN9022I  -DB9B DSNTDDIS 'DISPLAY DATABASE' NORMAL COMPLETION
***
```

Figure 8-5 -DIS DB command output for table holding XML column

Note: When you define an XML column, you do not specify a length for it. From an architectural point of view, there is no limit to the size of an XML value in a database, but serialized XML data that is exchanged within a DB2 database is limited to 2 GB. This means that the effective limit of an XML column is 2 GB.

ALTER TABLE

You can add XML columns to a regular relational table using the ALTER TABLE statement. For each XML column that you add to a table, DB2 implicitly creates an XML table and XML table space, which will store the XML data. If the XML column is the first XML column being added to that table, DB2 also implicitly adds a DocID column, as mentioned in “CREATE TABLE” on page 272. Also, the implicit indexes mentioned above will be created.

If you add additional XML columns to your table subsequently, only an XML table space and an XML table will be created.

Locking

DB2 gets XML documents locks to serialize update and fetch operations. UR transactions and lock avoiders also lock XML documents to ensure that the documents are completely inserted before being read and are not deleted while being read.

8.4 Accessing XML data

In 2003, the first edition of the SQL/XML standard was published by the International Organization of Standardization (ISO). SQLX.org is a Web site that makes the SQL/XML standard, and the technology that it includes, more highly visible to interested parties. The Web site makes it possible for the SQL and XML communities to follow the development of SQL/XML as it occurs and to readily ascertain the current status of the standard’s progress.

The DB2 V9 for z/OS engine processes SQL, SQL/XML, and XPath in an integrated manner since DB2 treats both SQL and a subset of XQuery as independent primary query languages. Applications can continue to use SQL, and, additionally, SQL/XML extensions, which allows publishing of relational data in XML format. XPath is a subset of XQuery and is typically used to access the native XML store, and optionally use SQL statements to combine XML data with SQL data. The full functionality of XQuery is currently only available on DB2 for Linux, UNIX, and Windows, but you can expect to see it some time later for z/OS as well.

Figure 8-6 shows the XML standards and how they interrelate to each other.

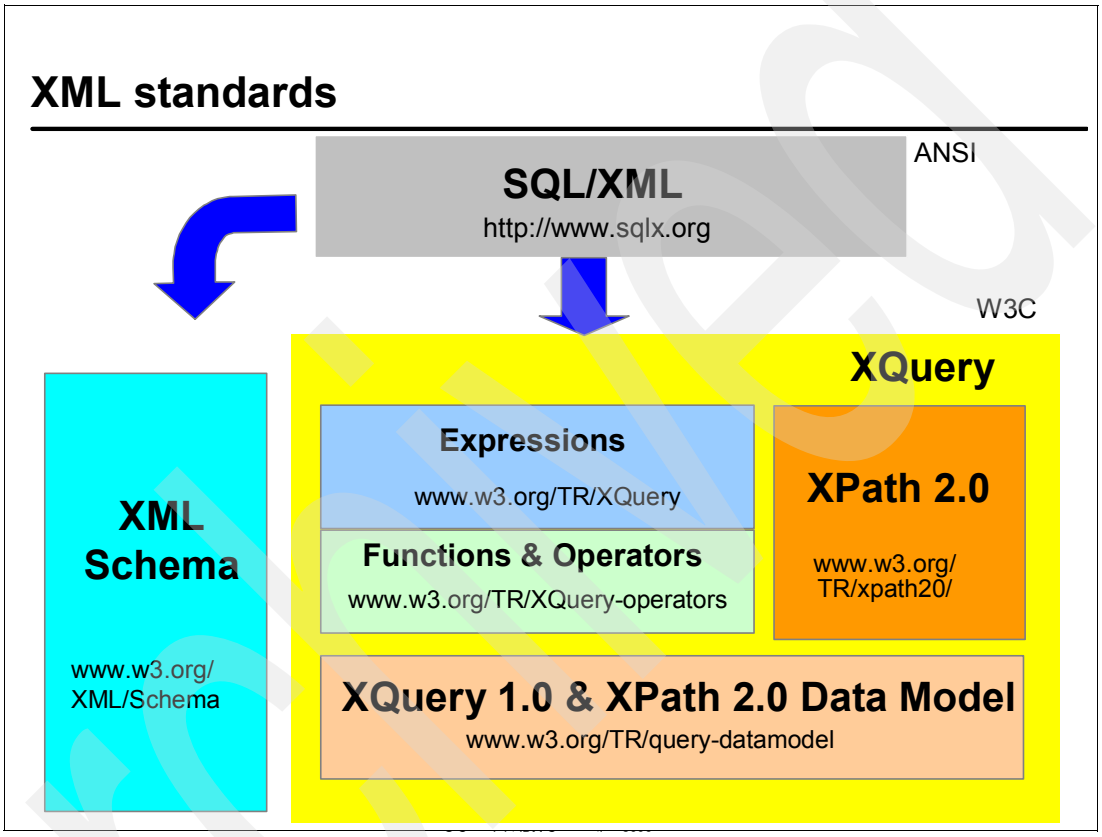


Figure 8-6 XML standards

8.4.1 SQL/XML constructors

Beginning with DB2 V8, DB2 has got some XML-related functions, known as XML publishing functions. The functions introduced with DB2 V8 were:

- XMLAGG
- XMLATTRIBUTES
- XMLCONCAT
- XMLELEMENT
- XMLFOREST
- XMLNAMESPACE

The V8 publishing functions are described in *DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ... and More*, SG24-6079.

DB2 V9 introduces:

- Four new SQL/XML constructors:
  - XMLCOMMENT
  - XMLDOCUMENT
  - XMLPI
  - XMLTEXT

  You can use these functions to construct XML processing instruction nodes, comment nodes, document nodes, and text nodes.
- XMLSERIALIZE as a function that converts XML values in to textual XML.
- XMLPARSE for parsing a given argument such as a string expressions and inserting it into an XML column.
- XMLQUERY, which is a scalar function that queries and extracts data from XML using XPath expressions. Refer to “XPath” on page 283 to learn more about XPath.
- XMLEXISTS, which you can use to test whether an XPath expression on XML values will return non-empty values.

We examine the V9 constructors in more detail.

XMLCOMMENT

The XMLCOMMENT functions return an XML value with a single comment node from a string expression. The content of the comment node is the value of the input string expression, converted to Unicode UTF-8.

The simple syntax of this command is:

```
XMLCOMMENT (string-expression)
```

The string-expression is an expression that returns a value of a built-in character or a graphic string that is neither a LOB nor bit data. The result of string-expression is converted to UTF-8 and then parsed to check for conformance to the content of the XML comment as specified by the following rules:

- A -- (double-hyphen) must not occur in the string expression
- The string expression must not end with a hyphen (’-’)
- Each character of the string can be any Unicode character, excluding the surrogate blocks, X'FFFE', and X'FFFF'.

If string-expression does not conform to the previous rules, an error is returned.

Look at the example shown below:

```
SELECT XMLCOMMENT('This is an XML comment')
FROM SYSIBM.SYSDUMMY1;
```

The result of that query is:

```
<?xml version="1.0" encoding="IBM037"?><!--This is an XML comment-->
```

If you refer back to item 7 in “Well-formed XML documents” on page 270, you can see that this generated comment matches this listed rule for well-formed XML documents.

Archived

XMLDOCUMENT

The XMLDOCUMENT function returns an XML value with a single document node and zero or more nodes as its children. The content of the generated XML document node is specified by a list of expressions.

The simple syntax of this command is shown in Figure 8-7.

Figure 8-7 XMLDOCUMENT syntax

An XML-expression is an expression that returns an XML value. A sequence item in the XML value must not be an attribute node. If XML-expression returns a null value, it is ignored for further processing. However, if all XML-expression values are null, the result of the function is the null value. The resulting XML value is built from the list of XML-expression arguments. The children of the resulting document node are constructed as follows:

1. All of the non-null XML values that are returned by XML-expression are concatenated together. The result is a sequence of nodes or atomic values, which is referred to in the following steps as the input sequence. Any document node in the input sequence is replaced by copies of its children.
2. For each node in the input sequence, a new deep copy of the node is constructed. A deep copy of a node is a copy of the whole subtree this is rooted at that node, including the node itself and its descendants and attributes. Each copied node has a new node identity. Copied element nodes are given the type annotation xdt:untyped, and copied attribute nodes are given the type annotation xdt:untypedAtomic. For each adjacent sequence of one or more atomic values that is returned in the input sequence, a new text node is constructed that contains the result of casting each atomic value to a string, with a single blank character inserted between adjacent values. The resulting sequence of nodes is called the content sequence.
3. The nodes in the content sequence become the children of the new document node. The result of the function is an XML value.

The result can be null. If all the arguments are null, the result is the null value.

Let us look at a very simple example:

```
INSERT INTO sabixml2 VALUES('123',
  (SELECT XMLDOCUMENT(XMLELEMENT(NAME "Emp",
   e.firstnme!!'    '!!e.lastname),
   XMLCOMMENT('This is just a simple example'))
FROM dsn8910.EMP e
WHERE e.empno = '000010'));
```

If you select from the XML column afterwards using either XMLSERIALIZE or the Control Center, for example, you can see the generated well-formed XML document shown below:

```
<Emp>CHRISTINE    HAAS</Emp><!--This is just a simple example-->
```

XMLPI

The XMLPI function returns an XML value with a single processing instruction node. The syntax for this XML function is as shown in Figure 8-8.

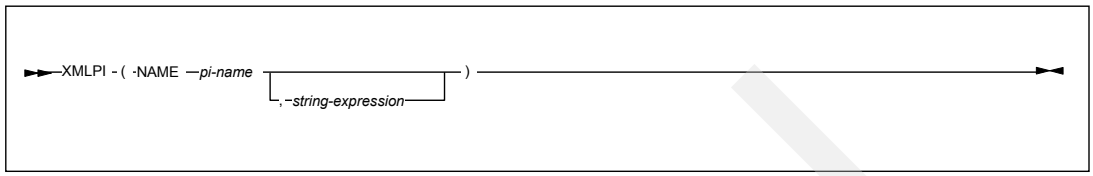


Figure 8-8 XMLPI syntax diagram

The NAME specifies the name of a processing instruction. The name is an SQL identifier that must be in the form of an XML NCName. NCName stands for non-colonized name (that is, it cannot include a colon). The name must not contain .xml. in any case combination.

The string-expression returns a value of a built-in character or graphic string that is not a LOB and is not bit data The resulting string will be converted to UTF-8 and parsed to check for conformance to the content of XML processing instruction as specified by the following rules:

- The string must not contain the substring '?>', as this terminates a processing instruction.
- Each character can be any Unicode character, excluding the surrogate blocks, X'FFFE', and X'FFFF'.

If the resulting string does not conform to the preceding rules, an error is returned. The resulting string becomes the contents of the constructed processing instruction node. If string-expression is not specified or is an empty string, the contents of the procession instruction node are empty.

The result of the function is an XML value. The result can be null. If the string-expression argument is null, the result is the null value.

In the following example, we used XMLPI to generate a PI for access control:

```
SELECT XMLPI(name "access-control"  ,
   'allow="w3.ibm.com" deny="www.*"')
FROM SYSIBM.SYSDUMMY1
```

The result would be:

```
<?xml version="1.0" encoding="IBM037"?><?access-control allow="w3.ibm.com"
deny="www.*"?>
```

Using something like that would allow hosts whose Web address is w3.ibm.com to access the content of the XML document. Those who are using any other www address cannot read the document.

XMLTEXT

The XMLTEXT function returns an XML value with a single text node that contains the value of the string-expression. The result type is XML.

The syntax is:

```
XMLTEXT(string-expression)
```

The result of the function is an XML value. The result can be null. If the argument is null, the result is the null value.

The following example shows you how you can make use of XMLTEXT. Let us assume that you have a table that contains the values shown in Table 8-4.

Table 8-4 Sample table SHOWXMLTEXT

seqno	plaintext	emphtext
1	This query show how to construct	mixed content
2	using XMLAGG and XMLTEXT. Without	XMLTEXT
3	XMLAGG cannot group text nodes with other nodes, therefore, cannot generate	mixed content

We now use the SELECT statement shown in Figure 8-9.

```
SELECT XMLELEMENT(NAME "para",
          XMLAGG(XMLCONCAT( XMLTEXT( plaintext),
            XMLELEMENT( NAME "emphasis", emphtext ))
ORDER BY seqno ),  .  ) as
"result" FROM SHOWXMLTEXT;
```

Figure 8-9 XMLTEXT example

We get the result shown in Figure 8-10.

```
result
--------------------------------------------------------------------------
<para>This query shows how to construct <emphasis>mixed content</emphasis>
using XMLAGG and XMLTEXT. Without <emphasis>XMLTEXT</emphasis>, XMLAGG cannot
group text nodes with other nodes, therefore, cannot generate <emphasis>mixed
content</emphasis>.</para>
```

Figure 8-10 XMLTEXT example result

Attention: If you try to run the statement above from SPUFI or DSNTEP2/DSNTEP4, you will be returned a maximum of 100 characters. This is the length assigned to the CLOB variable when retrieving via FETCH WITH CONTINUE.

XMLSERIALIZE

The XMLSERIALIZE function converts an XML value from its tree format into the corresponding textual XML. You can specify that the generated data type should be CLOB, DBCLOB, or BLOB.

Archived

The functionality of XMLSERIALIZE and XMLPARSE are opposite from each other. Figure 8-11 shows that you can use XMLSERIALIZE to retrieve an XML document from DB2 and XMLPARSE to store XML data in one of the XML columns within your DB2 tables.

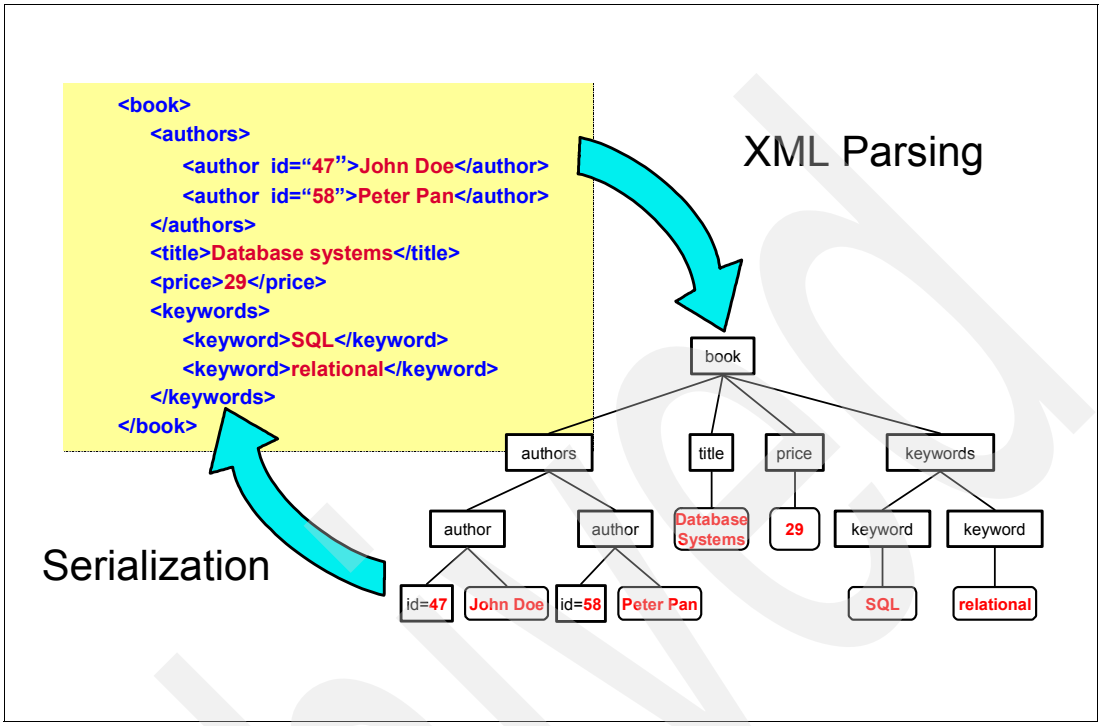


Figure 8-11 Serialization and parsing

Figure 8-12 shows the related syntax diagram.

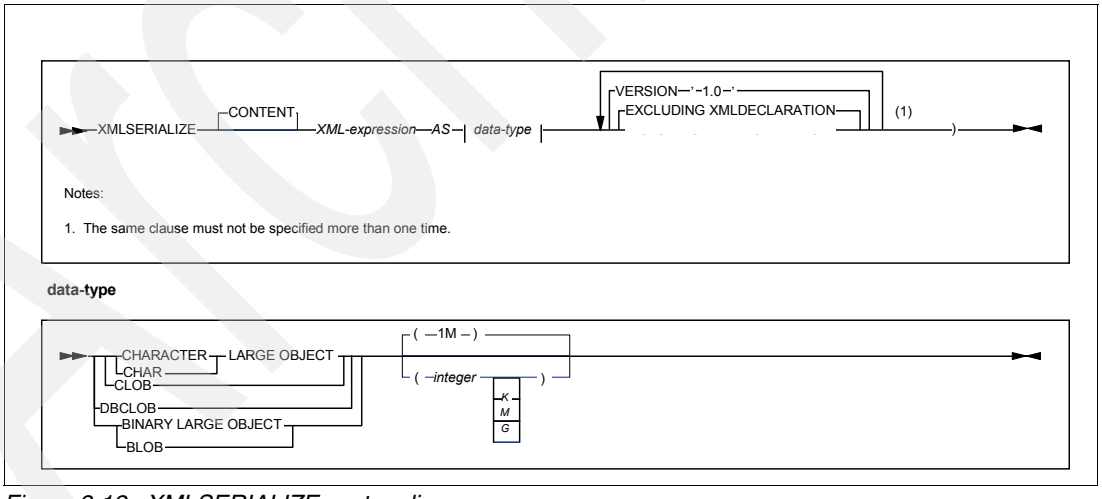


Figure 8-12 XMLSERIALIZE syntax diagram

The XML-expression returns an XML value that is not an attribute node. The atomic values in the input sequence must be castable to a string. The value that you specify here is the input to the serialization process.

As you can see from the syntax diagram shown in Figure 8-12, it is mandatory to specify the data type for your result.

We introduce two examples for the usage of XMLSERIALIZE.

Example 1: XMLSERIALIZE result of other XML function usage

The sample SELECT statement in Figure 8-13 serializes the XML value that is returned by the XMLELEMENT function into a CLOB of UTF-8. The resulting column *result* contains a simple XML element with Emp as the element name, and an employee name as the element content.

```
SELECT empno, XMLSERIALIZE(
                XMLELEMENT (NAME "Emp", firstnme !!' '!!lastname)
                AS CLOB(60)) AS "result"
FROM dsn8910.emp #
---------+---------+---------+---------+---------+---------+---------+--
EMPNO   result
---------+---------+---------+---------+---------+---------+---------+--
000010  <Emp>CHRISTINE HAAS</Emp>
000020  <Emp>MICHAEL THOMPSON</Emp>
000030  <Emp>SALLY KWAN</Emp>
000050  <Emp>JOHN GEYER</Emp>
000060  <Emp>IRVING STERN</Emp>
000070  <Emp>EVA PULASKI</Emp>
000090  <Emp>EILEEN HENDERSON</Emp>
000100  <Emp>THEODORE SPENSER</Emp>
000110  <Emp>VINCENZO LUCCHESI</Emp>
```

Figure 8-13 XMLSERIALIZE example 1

Also, if we look at the same example, but additionally ask for the result shown in hex, you can see that the resulting column is encoded in Unicode UTF-8. This is shown in Figure 8-14.

```
---------+---------+---------+---------+---------+---------+---------+---------
SELECT empno, XMLSERIALIZE(
                XMLELEMENT (NAME "Emp", firstnme !!' '!!lastname)
                AS CLOB(30)) AS "result"    ,
            hex(xMLSERIALIZE(
                XMLELEMENT (NAME "Emp", firstnme !!' '!!lastname)
                AS CLOB(60))) AS "hexresult"
FROM dsn8910.emp #
---------+---------+---------+---------+---------+---------+---------+---------
EMPNO   result                          hexresult
---------+---------+---------+---------+---------+---------+---------+---------
000010  <Emp>CHRISTINE HAAS</Emp>       3C456D703E434852495354494E4520484141533
000020  <Emp>MICHAEL THOMPSON</Emp>     3C456D703E4D49434841454C2054484F4D50534
000030  <Emp>SALLY KWAN</Emp>           3C456D703E53414C4C59204B57414E3C2F456D7
000050  <Emp>JOHN GEYER</Emp>           3C456D703E4A4F484E2047455945523C2F456D7
000060  <Emp>IRVING STERN</Emp>         3C456D703E495256494E4720535445524E3C2F4
000070  <Emp>EVA PULASKI</Emp>          3C456D703E4556412050554C41534B493C2F456
000090  <Emp>EILEEN HENDERSON</Emp>     3C456D703E45494C45454E2048454E444552534
000100  <Emp>THEODORE SPENSER</Emp>     3C456D703E5448454F444F5245205350454E534
000110  <Emp>VINCENZO LUCCHESI</Emp>    3C456D703E56494E43454E5A4F204C554343484
```

Figure 8-14 XMLSERIALIZE with hexadecimal column

Archived

Example 2: XMLSERIALIZE to serialize XML documents

In this example, we want to show you how you can use XMLSERIALIZE to extract XML documents from your DB2 table as is. Let us assume that we are looking at a table with two columns. Column C1 is defined as CHAR(10) and XMLCOL is of data type XML.

Although it is supposed to be possible to just specify the name of the XML column in your SELECT statement, SPUFI might run into SQL errors or at least warnings if it has to cut the result row. This error is shown in Figure 8-15.

```
select xmlcol from sabixml2
---------+---------+---------+---------+---------+---------+----
......
---------+---------+---------+---------+---------+---------+----
DSNE610I NUMBER OF ROWS DISPLAYED IS 0
DSNT408I SQLCODE = -433, ERROR:  VALUE <XML-Value> IS TOO LONG
DSNT418I SQLSTATE   = 22001 SQLSTATE RETURN CODE
```

Figure 8-15 SQLCODE -433

To circumvent this, you can use XMLSERIALIZE to select the entire XML document from your table. Following the syntax rules, the statement and the output might look like that shown in Figure 8-16.

```
select xmlserialize(xmlcol as clob(1k))
from sabixml2
---------+---------+---------+---------+---------+---------+----

---------+---------+---------+---------+---------+---------+----
<product xmlns="http://posample.org" pid="100-111-11"><description><name>hammer
</name><details>big hammer for tough workers</details><price>8.99
</price><weight>8 kg</weight></description></product>
```

Figure 8-16 XMLSERIALIZE to select whole XML document from XML column

XMLPARSE

You can use the XMLPARSE function to insert well-formed XML documents into an XML column. Refer to Figure 8-17 for the short syntax diagram of XMLPARSE.

Figure 8-17 XMLPARSE syntax diagram

As you can see, you can either specify a string expression or an XML host variable as input to XMLPARSE.

It is not necessary to use XMLPARSE to insert XML documents into an XML column. Instead, you can also just use a regular INSERT... VALUES statement. The advantage of using XMLPARSE over just INSERT.... VALUES is that you have the option to specify what you would like to do with whitespace (that is, the space that is between element nodes without any non-whitespace text nodes).

According to the XML standard, whitespace is space characters (U+0020), carriage returns (U+000D), line feeds (U+000A), or tabs (U+0009) that are in the document to improve readability. When any of these characters appear as part of the text string, they are not considered to be whitespace.

A boundary whitespace is whitespace characters that appear between elements. For example, in the following document, the spaces between <a> and and between and are boundary whitespace:

<a> and between

The WHITESPACE option on the XMLPARSE statement is meant for this boundary whitespace.

Tip: Whitespace is important in comparisons. The same value with and without whitespace is not the same in a comparison.

STRIP WHITESPACE is the default INSERT without XMLPARSE and also for XMLPARSE.

8.5 XPath

XPath is a language for navigating in XML documents and addressing parts of XML documents. As we discuss later in this chapter, XPath uses path expressions similar to file path notations in file systems.

You can use XPath in the context of XMLQUERY, XMLEXISTS, and in an XML index. We discuss those two SQL/XML constructors later in this section, and XML indexing in 8.6, “XML indexing” on page 312.

8.5.1 XML Data Model (XDM)

The XPath data model provides an abstract representation of one or more XML documents. The XML for DB2 data model is described in terms of sequences and items, atomic values, and nodes.

Sequences and items

A sequence is an ordered collection of zero or more items. An item is either an atomic value or a node. For example, each of the following values can be represented as a single sequence:

- “NATHAN”
- 36
- <dog/>
- (36,<dog/>,”NATHAN”)
- An XML document

Atomic values

An atomic value is an instance of one of the built-in atomic data types that are defined by XML schema. An atomic type is defined in the XML schema data types. Some types have literal values such as:

- 45
- “this is a string”

Other atomic types have a constructor function to build atomic values out of strings. For example, the following constructor function builds a value of type xs:decimal out of the string “12.34”:

```
xs:decimal(“12.34”)
```

Nodes

There are seven kinds of nodes that are defined for XPath. The nodes of a sequence form one or more trees that consist of a document node, and all of the nodes that are reachable directly or indirectly from the document node. Every node belongs to exactly one tree, and every tree has exactly one document node. The node types are:

- Document node

  A document node encapsulates an XML document. A document node cannot have parent nodes. To be a well-formed document, the document node must have exactly one child element and no child text nodes.

- Element node

  An element node encapsulates an XML element. An element can have zero or one parent, and zero or more children. The children can include all other sorts of nodes except document nodes.

- Attribute node

  An attribute node represents an XML attribute. An attribute node can have zero or one parents. The element node that owns an attribute is considered to be its parent, even though the attribute node is not a child of its parent elements.

- Text node

  A text node encapsulates XML character content.

- Processing Instruction node

  The PI node encapsulates an XML processing instruction. It can have zero or one parents.

- Comment node

  The Comment node encapsulates an XML processing instruction. It can have zero or one parents.

- Namespaces node

Note: Namespaces are considered to be nodes in XPath, but are not in XQuery.

Each node has a unique identity. This means that two nodes are distinguishable for DB2 even though their names and values might be the same. Among all of the nodes in a hierarchy, there is a total ordering called document order.

Archived

Figure 8-18 gives a simple example of how the XML documents are ordered. The order can be described as follows:

- Root node comes first.
- Siblings are ordered by the order that their start tags occur.
- Namespaces precede attributes, and attributes precede children.
- Elements precede their children, which precede their siblings.

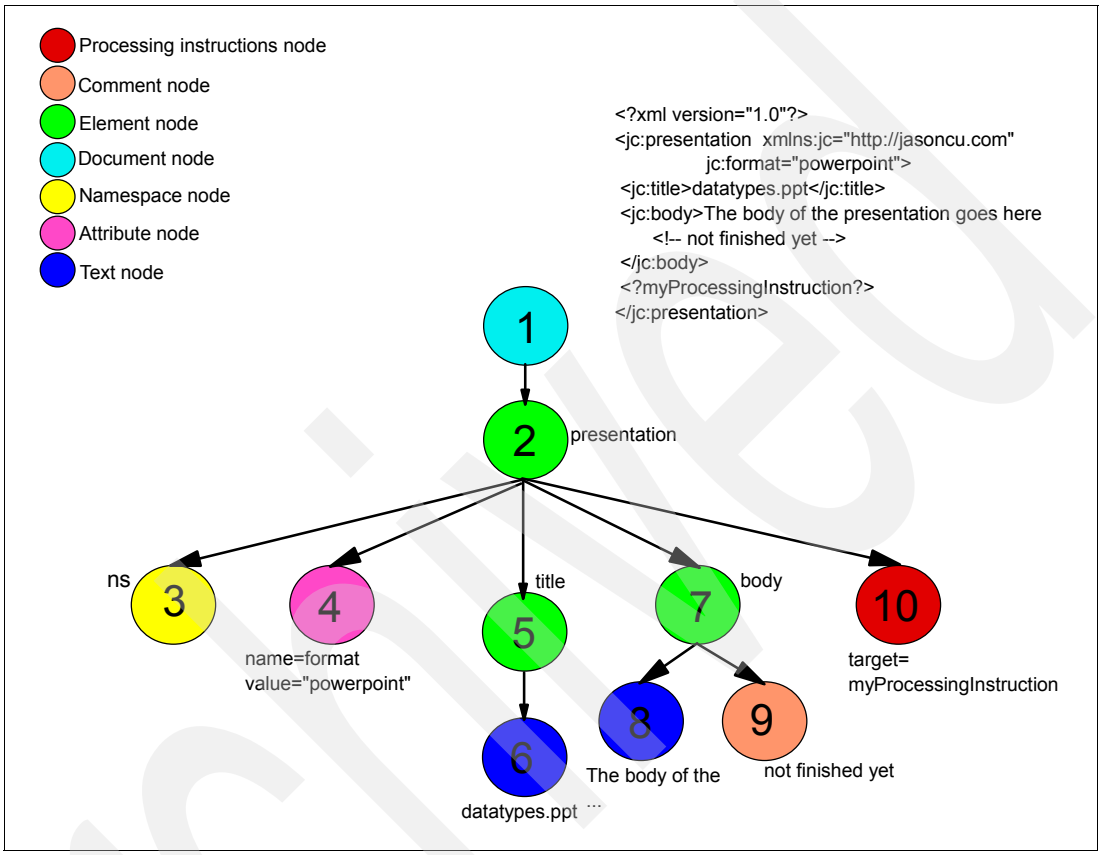


Figure 8-18 Document order

As mentioned before, each node in an XPath data model has a unique identity.

Atomization is applied to a value when the value is used in a context in which a sequence of atomic values is required. The result of atomization is either a sequence of atomic values or a type error. Atomization of a sequence is defined as the result of invoking the fn:data function on the sequence.

The atomized (typed) value of a document or element node is obtained by concatenating all of that node's descendant text nodes. That is, if $x is of type xml and contains value ('<a>ab<c>c</c>'), then fn:data($x) produces "abc".

8.5.2 XPath prologs and expressions

In DB2 XPath, an XPath expression consists of an optional prolog that is followed by an expression. The optional prolog establishes the processing environment and the expression generates a result.

Archived

Refer to Figure 8-19 for the general syntax of an XPath expression.

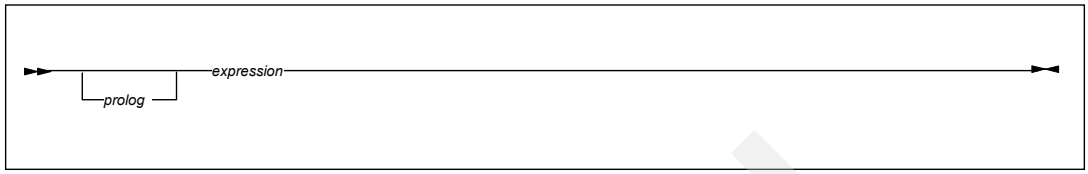


Figure 8-19 XPath expression syntax

Note: Next we occasionally show the related syntax diagrams slightly different from how they appear in the*DB2 Version 9.1 for z/OS XML Guide,* SC18-9858. If you combine the extra parts in the original manual, you will get what we show here.

A declaration in the *prolog*, which can contain zero or more namespace declarations and zero or one default namespace declarations, is always followed by a semicolon (;). Figure 8-20 shows the modified syntax diagram for the prolog.

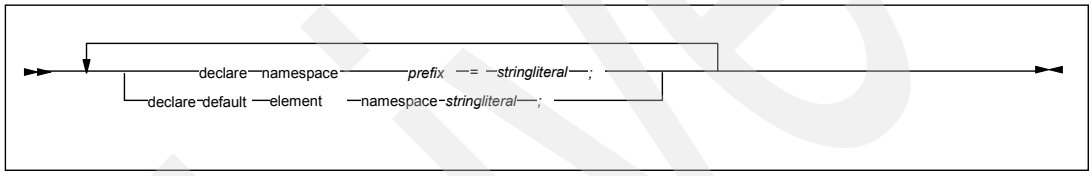


Figure 8-20 Prolog syntax diagram

There are two ways for you to define namespace prefixes:

- In the prolog using the syntax shown in Figure 8-20. These prefixes are valid throughout the XPath expression.

  Restriction: Prefixes smlns and sml are reserved. You cannot specify those as a prefix in a namespace declaration that is part of a prolog.

- In element constructors by namespace declaration attributes. Those namespaces are only valid locally within the element constructor defining it. Read more about it later in this chapter.

Namespace declarations versus default namespace declarations

A namespace declaration associates a name prefix with a namespace URI. In contrast to that, a default namespace declaration is used for unprefixed element names. Only one default namespace URI can exist per XPath expression.

The basic building block of XPath is the *expression*. DB2 XPath provides several kinds of expressions for working with XML data:

- Primary expressions, which include the basic primitives of the language, such as literals, variable references, and function calls
- Path expressions for locating nodes within a document tree
- Arithmetic expressions for addition, subtraction, multiplication, division, and modulus
- Comparison expressions for comparing two values
- Logical expressions for using boolean logic

Next we will describe the five different expression types.

Note: The examples that we use on the following few pages showing XPath expressions use the SQL/XML function XMLQUERY to send the XPath statement to and execute the statement on your DB2 9 subsystem. If you prefer to read more about XMLQUERY prior to proceeding with the XPath expressions, refer to 8.5.3, “XMLQUERY” on page 300.

Primary expressions

A primary expression contains one of the following types of items:

- Literals

  DB2 XPath supports numeric and string literals. A numeric literal is an atomic value of type xs:integer, xs:decimal, or xs:double.

  Figure 8-21 shows an example of an XPath expression with different numeric literals.

```
SELECT XMLQUERY ('1234') AS XSINTEGER,
XMLQUERY ('122.3322') AS XSDECIMAL,
XMLQUERY ('93948.87E+77') AS XSDOUBLE
FROM SYSIBM.SYSDUMMY1

The result is:
<?xml version="1.0" encoding="IBM037"?>1234 ......
<?xml version="1.0" encoding="IBM037"?>122.3322 ....
<?xml version="1.0" encoding="IBM037"?>9.394887E81
```

Figure 8-21 XPath expression with numeric literals example

Figure 8-22 shows an example of an XPath expression that contains a string literal with an embedded double quotation mark.

Tip: To include an apostrophe within a string literal that is delimited by apostrophes, specify two adjacent apostrophes. Similarly, to include a quotation mark within a string literal that is delimited by quotation marks, specify two adjacent quotation marks.

```
SELECT XMLQUERY
('"THIS IS A STRING LITERAL RETURNING DOUBLE QUOTE HEREAFTER: "" "')
FROM SYSIBM.SYSDUMMY1

The result is:

<?xml version="1.0" encoding="IBM037"?>THIS IS A STRING LITERAL RETURNING DOUBLE QUOTE
HEREAFTER: "
```

Figure 8-22 XPath expression with string literal containing double quotation mark

Archived

- Variable references

  A variable reference is an qualified name that is preceded by a dollar sign ($). When DB2 evaluates an XPath expression, each variable reference resolves to the value of the expression that is bound to the variable. See Figure 8-23.

```
SELECT XMLQUERY
('$x' PASSING '$x is bound as a string' AS "x") FROM SYSIBM.SYSDUMMYU

The result is an xml value of xs:string with contents:

<?xml version="1.0" encoding="IBM037"?>$x is bound as a string
```

Figure 8-23 XPath expression with variable reference to a string literal

- Context items

  Figure 8-24 shows a simple context item expression example.

```
SELECT XMLQUERY
('.' PASSING 'context item') FROM SYSIBM.SYSDUMMYU;

The result is:

<?xml version="1.0" encoding="IBM037"?>context item
```

Figure 8-24 XPath expression with context item

- Function call

  Figure 8-25 shows a simple function call expression example.

  <?xml version="1.0" encoding="IBM037"?>concatenate this string to this string.

```
SELECT XMLQUERY
('fn:concat($x,$y)' PASSING 'concatenate this string '
AS "x", 'to this string' AS "y") FROM SYSIBM.SYSDUMMYU;

The result is:

<?xml version="1.0" encoding="IBM037"?>concatenate this string to this string
```

Figure 8-25 XPath expression with function call

- Parenthesized expression

  Figure 8-26 shows a simple parenthesized expression example.

```
SELECT XMLQUERY
('fn:concat($x,$y)' PASSING 'concatenate this string '
AS "x", 'to this string' AS "y") FROM SYSIBM.SYSDUMMYU;

The result is:

<?xml version="1.0" encoding="IBM037"?>concatenate this string to this string
```

Figure 8-26 XPath expression with variable reference to a string literal

Archived

Path expressions

Path expressions locate nodes within an XML tree. This is designed to navigate the XML tree structure. The formal syntax of a path expression is shown in Figure 8-27.

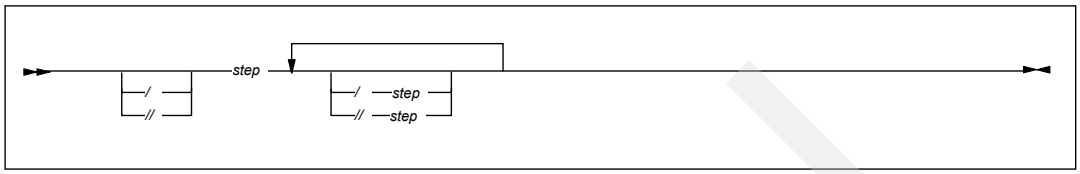


Figure 8-27 Path expression syntax

Before we can continue to explain to you how you can use those slashes and step keywords of the path expression syntax, we must look at the navigation axes that are used in XPath. Figure 8-28 gives you a good overview of the supported axes in DB2 9.

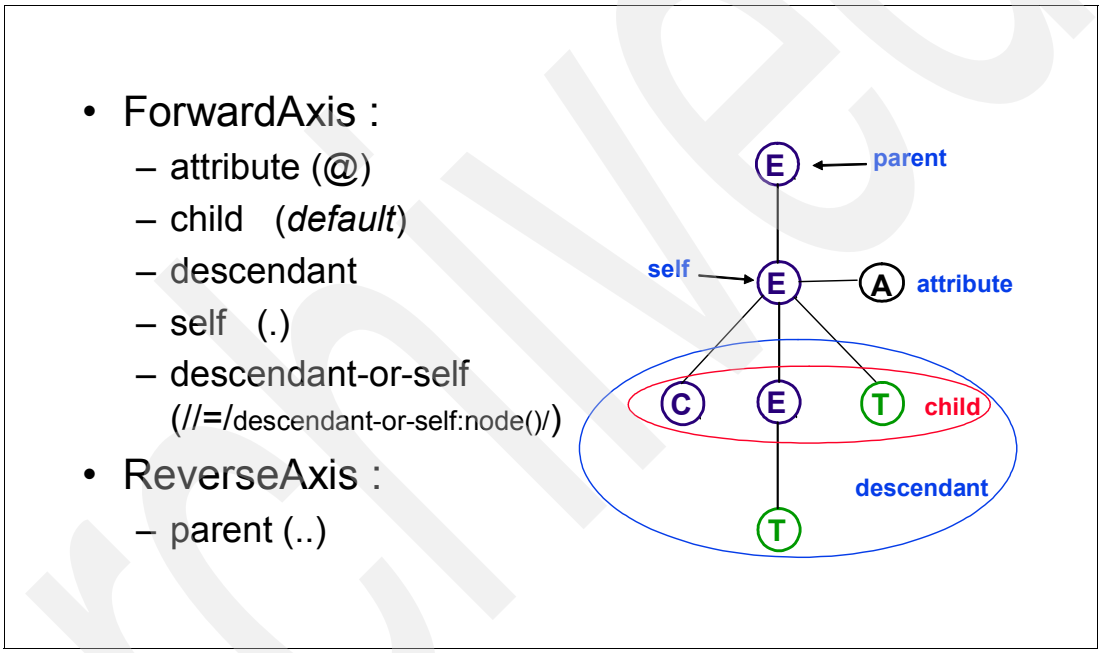


Figure 8-28 Supported Axes in DB2 9

A step that is part of the path expression syntax shown in Figure 8-27 consists of an axis step or a filter expression. An axis step can either be a forward step, which starts at the context node and moves down through the XML tree, or a reverse step, which starts at the context node and moves up through the XML tree.

An axis step consists of three parts:

- An optional axis, which specifies a direction of movement
- A node test, which specifies the criteria used to select nodes
- Zero or more predicates, which filter the sequence that is returned by the step.

The forward axes in DB2 XPath include the ones also listed in Figure 8-28 (that is, child, descendant, attribute, self, and descendant-or-self). The only supported reverse axis is the parent axis.

Archived

The context position in an axis step is assigned to the nodes in document order, starting with 1. Refer to Figure 8-18 on page 285 for an explanation of the document order.

Table 8-5 Supported axes in XPath

Axis	Description	Notes
child	Returns the children of the context node. This axis is the default.	In Figure 8-28, assuming that the node showing as *self* is the context node, circled nodes C, E, and T are its child nodes. Document and element nodes are the only nodes that have children. Attribute and document nodes can never appear as children.
descendant	Returns the descendants of the context node (that is, it returns the children, the children of the children, and so on).	In Figure 8-28, again assuming that the node showing as self is the context node, everything below self are the descendants.
attribute	Returns all attributes of the context node.	This axis is empty if the context node is not an element node. In Figure 8-28 the node showing as self is an element node. The attribute shown to the right of it is the result.
self	Returns the context node only.	
descendant-or-self	Returns the context node and the descendants of the context node.	
parent	Returns the parent of the context node, or an empty sequence if the context node has no parent.	

Syntax for path expressions

When working with DB2 XPath, you have the choice of working with abbreviated or unabbreviated syntax elements.

A path expression consists of a series of one or more steps that are separated by a slash character (/) or two slash characters (//). The path can begin with a slash character (/), two slash characters(//), or a step.

Two slash characters (//) in a path expression are expanded as /descendant-or-self::node()/, which leaves a sequence of steps separated by a slash character (/). A step generates a sequence of items. The steps in a path expression are evaluated from left to right. The sequence of items that a step generates is used as context nodes for the step that follows. For example, in the expression description/name, the first step generates a sequence of nodes that includes all description elements. The final step evaluates the name element once for each description item in the sequence. Each time a name element is evaluated, it is evaluated with a different focus, until all name elements have been evaluated. The sequences that result from each evaluation of the step are combined, and duplicate nodes are eliminated based on node identity. A slash character (/) at the beginning of a path expression means that the path begins at the root node of the tree that contains the context node. That root node must be a document node.

On the next few pages you will learn how to use SQL XPath on DB2 9 by looking at several examples. Unless otherwise stated, we refer to the XML document shown in Figure 8-29.

```
<docnode>
  <CUSTOMERINFO CID="1004">
     <NAME>MATT FOREMAN</NAME>
     <ADDR COUNTRY= "CANADA" >
        <STREET>1596 BASELINE</STREET>
        <CITY>TORONTO</CITY>
        <PROV-STATE>ONTARIO</PROV-STATE>
        <PCODE-ZIP>M3Z-5H9</PCODE-ZIP>
     </ADDR>
     <PHONE TYPE= "WORK" >905-555-4789</PHONE>
     <PHONE TYPE= "HOME" >416-555-3376</PHONE>
     <ASSISTANT>
          <NAME>GOPHER RUNNER</NAME>
          <PHONE TYPE= "HOME" >416-555-3426</PHONE>
     </ASSISTANT>
  </CUSTOMERINFO>
</docnode>
```

Figure 8-29 Sample XML document

When you code XPath expressions it is up to you to decide whether you prefer to use the long or abbreviated syntax in your axis steps. Table 8-6 gives you an overview of valid abbreviations.

Table 8-6 Abbreviated syntax for path expressions

Unabbreviated syntax	Abbreviated syntax	Description
child:: or attribute::	no axis specified	If the axis step specifies attribute() for the node test, omitting an axis is the shorthand for attribute:: . In all other cases it is child::.
attribute::	@	Shorthand for attribute::.
/descendant-or-self::node()/	//	When this abbreviation appears at the beginning of the path expression, the axis step selects an initial node sequence that contains the root of the tree in which the context node is found, plus all nodes that are descended from this root. If the root node is not a document node, you will receive an error.
parent::node()	..	

The column description in Table 8-6 contains things like attribute() and node(). Those are so-called kind tests. We cover those in “Node tests” on page 293. To keep it simple for now, we ignore this in the next few examples.

Important: You cannot use the syntax that we show to you in the following examples as is. You will learn later in this chapter how to correctly code a complete SQL XPath expression. For now, we only concentrate on parts of it.

Archived

Table 8-7 shows you some sample usages of path expressions.

Table 8-7 Path expression examples

Path expression	Description	Result
'/'	/ means that the path begins at the root node.	The whole XML document: <docnode> <CUSTOMERINFO CID="1004"> <NAME>MATT FOREMAN</NAME> <ADDR COUNTRY= "CANADA" > <STREET>1596 BASELINE</STREET> <CITY>TORONTO</CITY> <PROV-STATE>ONTARIO</PROV-STATE> <PCODE-ZIP>M3Z-5H9</PCODE-ZIP> </ADDR> <PHONE TYPE= "WORK" >905-555-4789</PHONE> <PHONE TYPE= "HOME" >416-555-3376</PHONE> <ASSISTANT> <NAME>GOPHER RUNNER</NAME> <PHONE TYPE= "HOME" >416-555-3426</PHONE> </ASSISTANT> </CUSTOMERINFO> </docnode>
'//ADDR'	// appears at the beginning of the path expression. The root node is therefore now ADDR.	All children of Element node ADDR and ADDR itself: <ADDR COUNTRY= "CANADA" > <STREET>1596 BASELINE</STREET> <CITY>TORONTO</CITY> <PROV-STATE>ONTARIO</PROV-STATE> <PCODE-ZIP>M3Z-5H9</PCODE-ZIP> </ADDR>
'//ADDR/@*'	Asks for all attributes that appear under the root, which is ADDR in our case.	The only attribute under ADDR is: CANADA
'//@*'	Asks for all attributes in the XML document.	All attributes: 1004 CANADA WORK HOME HOME
'/DOCNODE/CUSTOMERINFO/ASSISTANT/NAME/..'	Asks for all child elements of ASSISTANT.	Element ASSISTANT, which is the parent of NAME, plus all child elements of ASSISTANT: <ASSISTANT> <NAME>GOPHER RUNNER</NAME> <PHONE TYPE= "HOME" >416-555-3426</PHONE> </ASSISTANT>

Archived

Table 8-8 offers you a comparison of abbreviated and unabbreviated path expression syntax.

Important: You must always keep in mind that XML path expressions are case sensitive. Syntax elements such as child, attribute, and so on must be used as lower case characters. In contrast to that, if your XML document contains upper case node names, as shown in Figure 8-29 in our case, you must make sure that your path expression contains them exactly that way.

Table 8-8 Comparison of unabbreviated and abbreviated path expression syntax

Unabbreviated syntax	Abbreviated syntax	Result
'//child::ADDR'	'//ADDR'	Selects all child elements of element ADDR plus the ADDR itself.
'//child::*'	'//*'	Selects all child elements of the XML document including the root element. The result would be the whole XML document.
'//PHONE/attribute::TYPE'	'//PHONE/@TYPE'	Returns all values for the TYPE attribute. Referring to Figure 8-29, the result is: WORK HOME HOME
'//attribute::*'	'//@*'	Returns all attributes that appear in the XLM document.
'//child::PHONE[attribute::TYPE="HOME"]'	'//PHONE[@TYPE="HOME"]'	You ask for all phone numbers for which the attribute type "HOME" is. You can read the square brackets like WHERE in an SQL statement. TYPE must be used in upper case, because it is upper case in the XML document as well.
'//child::ADDR[child::CITY="TORONTO"]'	'//ADDR[CITY="TORONTO"]	Ask for all elements including ADDR, where element CITY has value TORONTO.

Tip: Square brackets [ ] used in a XML path expression are used for predicate evaluation. You can think of it as a WHERE clause in an SQL statement.

Node tests

As listed in on page 289, the second part of an axis step is a node test. A node test is a condition that must be true for each node that is selected by an axis step. Within node tests, we distinguish between name tests and kind tests.

A *name test* consists of a qualified name or a wildcard. When you specify a name test in an axis step, the step selects the nodes on the specified axis that mach the qualified name or wildcard. If you specify a name test on the attribute axis, the step selects any attributes that match the name test. Otherwise, on all other axes, the step selects any elements that match the name test.

Archive

If you decided to specify a *kind test* in an axis step, the step selects only those nodes on the specified axis that match the kind test. Table 8-9 gives you an overview of the supported kind tests in DB2 XPath.

Table 8-9 Supported kind tests in DB2 XPath

Test	Description	Examples
node()	Matches any node on the specified axis.	In the expression child::node(), the kind test node() selects any nodes on the child axes. The difference between child::node() and child::* is that child::node() returns every node, while child:* only returns element nodes.
text()	Matches any text node on the specified axis.	In the expression child::text(), the kind test text() selects any text nodes on the child axis. As an example referring to Figure 8-29 on page 291, path expression '//child::text()' returns the following values: MATT FOREMAN1596 BASELINETORONTOONTARIOM 3Z-5H9905-555-4789416-555-337 6GOPHER RUNNER416-555-3426 As you can see, it is all the text information that is part of the XML document.
comment()	Matches any comment node on the specified axis.	In the expression child::comment(), the kind test comment() selects any comment nodes on the child axis. Since there are no comment nodes in our sample XML document, '//child::comment()' does not return any result.
processing-instruction(NCName)	Matches any processing-instruction node on the specified axes whose name matches the NCName that is specified in the name test.	In the expression child::processing-instruction(xml-stylesheet), the kind test processing-instruction(xml-stylesheet) selects any processing instruction nodes on the child axis whose PITarget is xml-stylesheet.
processing-instruction(StringLiteral)	Matches any processing-instruction node on the specified axis whose name matches the string literal that is specified in this test. This node test provides backwards compatibility with XPath 1.0.	In the expression child::processing-instruction("xml-stylesheet"), the kind test processing-instruction("xml-stylesheet") selects any processing instruction nodes on the child axis whose PITarget is xml-stylesheet.

Archive

Test	Description	Examples
element()	Matches any element node on the specified axis.	In the expression child::element(), the kind test element() selects any element nodes on the child axis.
element(QName)	Matches any element node on the specified axis whose name matches the qualified name that is specified in this test.	In the expression child::element("price"), the kind test element("price") selects any element nodes on the child axis whose name is price. Path expression '//child::element(NAME)' returns the following result: <NAME>MATT FOREMAN</NAME><NAME>GOPHER RUNNER</NAME>
element(*)	Matches any element node on the specified axis.	In the expression child::element(*), the kind test element(*) selects any element nodes on the child axis.
attribute()	Matches any attribute node on the specified axis.	In the expression child::attribute(), the kind test attribute() selects any attribute nodes on the child axis.
attribute(QName)	Matches any attribute node (on the specified axis) whose name matches the qualified name that is specified in this test.	In the expression child::attribute("price"), the kind test attribute("price") selects any attribute nodes on the child axis whose name is price.
attribute(*)	Matches any attribute node on the specified axis.	In the expression child::attribute(*), the kind test attribute(*) selects any attribute nodes on the child axis.
document-node()	Matches any document node on the specified axis.	In the expression child::document-node(), the kind test document-node() selects any document nodes on the child axis.
document-node(element(QName))	Matches any document node on the specified axis that has only one element node (on the specified axis), and the name of the node matches the qualified name that is specified in this test.	In the expression child::document-node(element("price")), the kind test document-node(element("price")) selects any document nodes on the child axis that have a single element whose name is price.

Archive

Test	Description	Examples
document-node(element(*))	Matches any document node on the specified axis that has element nodes.	In the expression child::document-node(element(*)), the kind test document-node(element(*)) selects any document nodes on the child axis that have element nodes.

Filter expressions

A filter expression consists of a primary expression that is followed by zero or more predicates. The predicates filter the result of the primary expression.

We have already implicitly introduced filter expressions to you in “Path expressions” on page 289. You code filter expressions using the square brackets. Wenow show you some sample usages of filter expressions. In Example 8-1 we ask DB2 for all address information of those customers who live in TORONTO.

Example 8-1 List all addresses where element CITY contains text TORONTO

```
'//child::*[CITY="TORONTO"]'

Returns:

<ADDR COUNTRY= "CANADA" >
      <STREET>1596 BASELINE</STREET>
       <CITY>TORONTO</CITY>
      <PROV-STATE>ONTARIO</PROV-STATE>
       <PCODE-ZIP>M3Z-5H9</PCODE-ZIP>
   </ADDR>
```

Example 8-2 is similar to the previous example, but we first set the context node to ASSISTANT, and second we use the filter on attribute TYPE, rather than an element, as we did in Example 8-1.

Example 8-2 List all ASSISTANT phone numbers for TYPE HOME

```
//ASSISTANT/PHONE[@TYPE="HOME"]

Returns:

<PHONE TYPE="HOME">416-555-3426</PHONE>
```

Archived

Example 8-3 shows you an extension to Example 8-2 on page 296. The beginning of the path expression is exactly the same, but then at the end we added '/..' . Remember, '..' asks for the parent. Due to this, the result is not just the PHONE element, but also the parent element ASSISTANT, plus its other child element NAME.

Example 8-3 List all ASSISTANT information for those who have phone TYPE HOME

```
//ASSISTANT/PHONE[@TYPE="HOME"]/..

Returns:

<ASSISTANT>
      <NAME>GOPHER RUNNER</NAME>
       <PHONE TYPE= "HOME" >416-555-3426</PHONE>
   </ASSISTANT>
```

Example 8-4 is also very interesting. In the first part, we ask DB2 to give us all ADDR information if it contains an element called CITY. The result is the whole address information. In the second part, we replace CITY with COUNTRYCODE and we do not get any information back.

Example 8-4 List ADDR information

```
'//ADDR[CITY]'

Return:

<ADDR COUNTRY= "CANADA" >
      <STREET>1596 BASELINE</STREET>
       <CITY>TORONTO</CITY>
      <PROV-STATE>ONTARIO</PROV-STATE>
       <PCODE-ZIP>M3Z-5H9</PCODE-ZIP>
</ADDR>

But:

//ADDR[COUNTRYCODE]

returns nothing.
```

Arithmetic expressions

Arithmetic expressions perform operations that involve addition, subtraction, multiplication, division, and modulus.

The arithmetic operators in XPath are:

*	multiplication
div	division
idiv	integer division
mod	modulus
+	addition
-	subtraction

Tip: A subtraction operator must be preceded by whitespace, because the operator could otherwise be interpreted as part of a previous token. a-b, for example, is interpreted as a name, but a - b and a -b are interpreted as an arithmetic operation.

Archived

The result of an arithmetic expression is an empty sequence, an error, or a numeric value.

For the purpose of giving you examples of arithmetic expressions, we must use other XML documents that contain numbers. Below we refer to the documents shown in Figure 8-30.

```
<product pid="100-100-01">
    <description>
        <name>Snow Shovel, Basic 22 inch</name>
        <details>Basic Snow Shovel, 22 inches wide, straight handle
            with D-Grip
        </details>
        <price>9.99</price>
        <weight>1 kg</weight>
    </description>
</product>


<product pid="100-201-01">
    <description>
        <name>Ice Scraper, Windshield 4 inch</name>
        <details>Basic Ice Scraper 4 inches wide, foam handle
        </details>
        <price>3.99</price>
    </description>
</product>
```

Figure 8-30 Sample XML Document II

In Example 8-5 we use an arithmetic expression to calculate the prices, including taxes for the products that are stored in our table. Assuming that the VAT is 19 %, we let DB2 calculate the new prices and display them as a result.

Example 8-5 Arithmetic expression example

```
//product/description/price*1.19'

results in:

11.8881
4.7481
```

Comparison expressions

Comparison expressions allow you to compare two values. Comparison operators are:

= True if any value in the first sequence is equal to any value in the second sequence.

!= True if any value in the first sequence is not equal to any value in the second sequence.

< True if the first sequence is less than any value in the second sequence.

<= True if any value in the first sequence is less than or equal to any value in the second sequence.

Archived

>	True if the first sequence is greater than any value in the second sequence.
>=	True if any value in the first sequence is greater than or equal to any value in the second sequence.

Example 8-6 shows the usage of a comparison expression. We ask for all items for which the price is less than 5. If you refer to Figure 8-30 on page 298, you can see that the snow shovel costs 9.99 and the Ice Scraper's price is only 3.99. Due to that, we receive false for the XML document describing the snow shovel and *true* for the ice scraper.

Example 8-6 Comparison expression example - true/false

```
'//product/description/price < 5'

The result is:

false
true
```

In Example 8-7, we also test for all products for which the price is less than 5, but this time we do not just want to see whether the comparison is true or false, but we want to get the product's description for those who cost less than 5.

Example 8-7 Example of comparison expression

```
'//product/description[price < 5]'

The result is:

<description>
   <name>Ice Scraper, Windshield 4 inch</name>
   <details>Basic Ice Scraper 4 inches wide, foam handle</details>
   <price>3.99</price>
</description>
```

The described kind of comparison is the so called *general comparison*. Other comparisons are *logical comparisons*. Logical expressions return the boolean value true if both of two expressions are true, or if one or both of two expressions are true. Available operands for logical comparisons are:

- AND
- OR

Example 8-8 shows you how to use logical comparisons. We ask DB2 for the description of those products that either have the pid 100.201-01 or 100-100-01. The result is a list of two products.

Example 8-8 Logical comparison

```
'//product[@pid="100-201-01" or @pid="100-100-01"]'

Result is:

<description>
      <name>Snow Shovel, Basic 22 inch</name>
      <details>Basic Snow Shovel, 22 inches wide, straight handle with
D-Grip</details>
```

```
      <price>9.99</price>
      <weight>1 kg</weight>
</description>
<description>
      <name>Ice Scraper, Windshield 4 inch</name>
      <details>Basic Ice Scraper 4 inches wide, foam handle</details>
      <price>3.99</price>
</description>
```

8.5.3 XMLQUERY

You can retrieve entire XML documents from XML columns using an SQL SELECT statement, as shown in “Example 2: XMLSERIALIZE to serialize XML documents” on page 282. If, however, you do not want to retrieve entire XML documents but portions of it, XMLQUERY is the SQL/XML function that supports it. XMLQUERY is an SQL scalar function that lets you execute an XPath expression from within an SQL context.

When you execute XPath expressions from within an XMLQUERY function, you can:

- Retrieve parts of stored XML documents, instead of entire XML documents.
- Enable XML data to participate in SQL queries.
- Operate on both relational and XML data in the same SQL statement.
- Apply further SQL processing on the returned XML values such as ordering results with the ORDER BY clause of a subselect. If you plan to do something like this, you must use XMLCAST to cast the results to a non-XML type.

The syntax of XMLQUERY is shown in Figure 8-31.

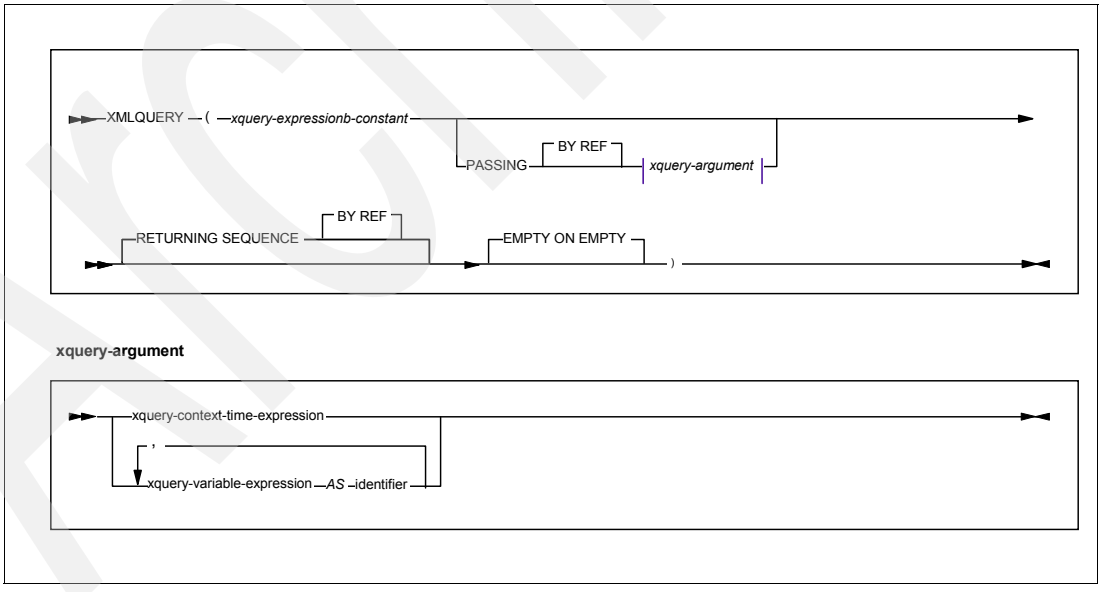


Figure 8-31 XMLQUERY syntax

XMLQUERY returns an XML value, which is an XML sequence. This sequence can be empty or can contain one or more items.

Archive

Let us assume that our XML column contains the two XML documents that are shown in Figure 8-32.

```
<customerinfo Cid="1004">
   <name>Matt Foreman</name>
   <addr country= "Germany" >
        <street>Karl-Arnold-Platz 1a</street>
     <city>Duesseldorf</city>
     <pcode-zip>40474</pcode-zip>
      </addr>
      <phone type= "work" >0211-477-1247</phone>
</customerinfo>
```

```
<customerinfo Cid="1003">
      <name>Robert Shoemaker</name>
      <addr country="Canada">
            <street>1596 Baseline</street>
            <city>Aurora</city>
            <prov-state>Ontario</prov-state>
            <pcode-zip>N8X 7F8</pcode-zip>
      </addr>
      <phone type="work">905-555-7258</phone>
      <phone type="home">416-555-2937</phone>
      <phone type="cell">905-555-8743</phone>
      <phone type="cottage">613-555-3278</phone>
</customerinfo>
```

Figure 8-32 Sample XML documents III

First of all, we show you how you imbed your XMLQUERY statement within a SELECT statement. Example 8-9 shows you a simple XMLQUERY statement that asks DB2 for all XML documents that are stored in column INFO in table SABIXML5.

Example 8-9 Sample XMLQUERY statement I

```
SELECT XMLQUERY('//*' passing INFO)
     FROM SABIXML5 ;

The result:

****************************** Top of Data ********************************************************************
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+------
  SELECT XMLQUERY('//*' passing INFO)                                    00020099
       FROM SABIXML5  ;                                                  00030099
---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+-------

---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+---------+-------
<?xml version="1.0" encoding="IBM037"?><customerinfo Cid="1004"><name>Matt Foreman</name><addr country="
DSNT400I SQLCODE = 000,  SUCCESSFUL EXECUTION
DSNT418I SQLSTATE   = 01004 SQLSTATE RETURN CODE
DSNT415I SQLERRP    = DSN SQL PROCEDURE DETECTING ERROR
DSNT416I SQLERRD    = 0 0  0  -1  0  0 SQL DIAGNOSTIC INFORMATION
DSNT416I SQLERRD    = X'00000000'  X'00000000'  X'00000000'  X'FFFFFFFF'
         X'00000000'  X'00000000' SQL DIAGNOSTIC INFORMATION
DSNT417I SQLWARN0-5 = W,W,,,, SQL WARNINGS
DSNT417I SQLWARN6-A = ,,,,   SQL WARNINGS
<?xml version="1.0" encoding="IBM037"?><customerinfo Cid="1003"><name>Robert Shoemaker</name><addr count
DSNT400I SQLCODE = 000,  SUCCESSFUL EXECUTION
DSNT418I SQLSTATE   = 01004 SQLSTATE RETURN CODE
DSNT415I SQLERRP    = DSN SQL PROCEDURE DETECTING ERROR
DSNT416I SQLERRD    = 0 0  0  -1  0  0 SQL DIAGNOSTIC INFORMATION
DSNT416I SQLERRD    = X'00000000'  X'00000000'  X'00000000'  X'FFFFFFFF'
```

```
        X'00000000'  X'00000000' SQL DIAGNOSTIC INFORMATION
DSNT417I SQLWARN0-5 = W,W,,,, SQL WARNINGS
DSNT417I SQLWARN6-A = ,,,,   SQL WARNINGS
```

Important: You always have to surround the XPath statement with apostrophes (').

'passing INFO' in the SQL statement shown in Example 8-9 on page 301 means that the XML data for which you have coded the XMLQUERY statement is stored in column INFO.

If you look at the result, you can see that apparently you have not obtained all of the XML documents. The result is truncated after 100 bytes of output. This is the case for both DSNTEP2 and SPUFI. Since XML does not have a specific length, the application always truncates the output after 100 bytes.

You can work around this problem using function XMLSERIALIZE around the XMLQUERY statement. Example 8-10 shows you the correct statement and the result.

Example 8-10 XMLQUERY statement sample with XMLSERIALIZE

```
******************************** Top of Data ********************************
---------+---------+---------+---------+---------+---------+---------+---------+
  SELECT XMLSERIALIZE(XMLQUERY('//*' passing INFO) AS CLOB(2K))         00020099
       FROM SABIXML5  ;                                                00030099
---------+---------+---------+---------+---------+---------+---------+---------+

---------+---------+---------+---------+---------+---------+---------+---------+
<customerinfo Cid="1004"><name>Matt Foreman</name><addr country="Germany"><stree
t>Karl-Arnold-Platz 1a</street><city>Duesseldorf</city><pcode-zip>40474</pcode-z
ip></addr><phone type="work">0211-477-1247</phone></customerinfo>
<customerinfo Cid="1003"><name>Robert Shoemaker</name><addr country="Canada"><st
reet>1596 Baseline</street><city>Aurora</city><prov-state>Ontario</prov-state><p
code-zip>N8X 7F8</pcode-zip></addr><phone type="work">905-555-7258</phone><phone
type="home">416-555-2937</phone><phone type="cell">905-555-8743</phone><phone t
ype="cottage">613-555-3278</phone></customerinfo>
```

In reality, the information belonging to each XML document is displayed as one long row. We have just made it multiple rows per XML document so that you can see the complete information.

Non-empty sequences returned by XMLQUERY

Let us assume that you execute the statement shown in Example 8-11.

Example 8-11 Non-empty sequences returned by XMLQUERY

```
SELECT XMLSERIALIZE(XMLQUERY('//customerinfo/phone'
       passing INFO) AS CLOB(2K))
       FROM SABIXML5  ;

The result is:

<phone type="work">0211-477-1247</phone>
<phone type="work">905-555-7258</phone><phone type="home">416-555-2937</phone><p
hone type="cell">905-555-8743</phone><phone type="cottage">613-555-3278</phone>
```

Again, in reality you would see two rows — one containing just one phone number and one containing four different phone numbers for the second XML document.

Note: This result demonstrates that when XMLQUERY returns a sequence that contains more than one element, the serialization process concatenates the elements into a single string. The result in the second row is not a well-formed document. Ensure that any application that receives this result can properly handle this behavior.

Empty sequences returned by XMLQUERY

If the XPath expression returns an empty sequence, XMLQUERY returns an empty sequence to you also. The SQL statement in Example 8-12 asks for all XML documents that have a prov-state element.

Example 8-12 Empty sequence returned by XMLQUERY

```
---------+---------+---------+---------+---------+---------+---------+---------+
  SELECT XMLSERIALIZE(XMLQUERY('//customerinfo/addr[prov-state]/..'     00020099
       passing INFO) AS CLOB(2K))                                      00021099
       FROM SABIXML5  ;                                                00030099
---------+---------+---------+---------+---------+---------+---------+---------+

---------+---------+---------+---------+---------+---------+---------+---------+

<customerinfo Cid="1003"><name>Robert Shoemaker</name><addr country="Canada"><st
reet>1596 Baseline</street><city>Aurora</city><prov-state>Ontario</prov-state><p
code-zip>N8X 7F8</pcode-zip></addr><phone type="work">905-555-7258</phone><phone
type="home">416-555-2937</phone><phone type="cell">905-555-8743</phone><phone t
ype="cottage">613-555-3278</phone></customerinfo>
```

As you can see, the result is one empty and one non-empty sequence. If you do not want those empty sequences in your result, you can filter those out using the XMLEXISTS predicate that we describe in 8.5.4, “XMLEXISTS” on page 305.

Archive

XMLQUERY on XML documents containing namespaces

Up to now, we have not yet dealt with XML documents that contained namespace definitions. This, however, is not the way you would usually work with XML documents. If your XML documents contain namespace definitions, your XMLQUERY statement must refer to it too. We have inserted two new XML documents into our table. The documents that we have inserted are shown in Figure 8-33 for your reference.

Figure 8-33 XML documents containing namespace definitions

If you now execute the exact same statement that we have shown in Example 8-11 on page 302, the results are two empty sequences.

You must now write your SELECT statement, as shown in Example 8-13.

Example 8-13 XMLQUERY statement for XML docs with namespace

```
SELECT XMLSERIALIZE(XMLQUERY('
     declare default element namespace "http://posample.org";
     //customerinfo'
     passing INFO) AS CLOB(2K))
     FROM SABIXML5  ;
```

Archived

8.5.4 XMLEXISTS

As mentioned above, you can use the XMLEXISTS predicate to restrict the set of rows that a query returns. You can code this restriction based on the values in XML columns. Figure 8-34 shows the XMLEXISTS syntax.

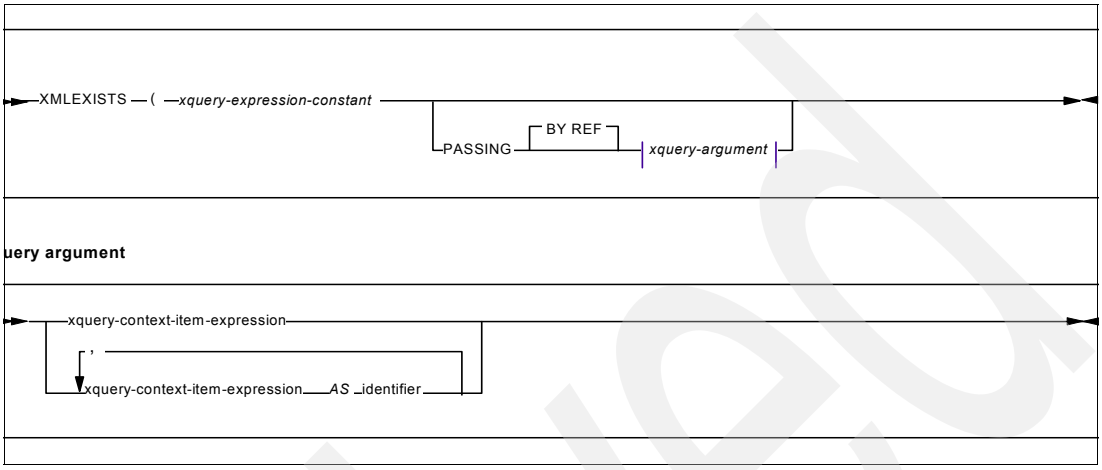


Figure 8-34 XMLEXISTS syntax diagram

As you can see, the XMLEXISTS predicate specifies an XPath expression. If the XPath expression returns an empty sequence, the XMLEXISTS predicate is false. Otherwise, XMLEXISTS returns true. Rows that correspond to an XMLEXISTS value of true are returned.

Referring back to the empty sequence that we have got in Example 8-12 on page 303, Example 8-14 now shows you how you can avoid this empty sequence using XMLEXISTS.

Example 8-14 XMLEXISTS sample

```
SELECT XMLSERIALIZE(XMLQUERY('
     declare default element namespace "http://posample.org";
     //customerinfo/addr[prov-state]/..'
     passing INFO) AS CLOB(2K))
     FROM SABIXML5
       WHERE XMLEXISTS('
     declare default element namespace "http://posample.org";
     //customerinfo/addr[prov-state]/..'
     passing INFO)   ;
```

8.5.5 XPath functions

In addition to what we saw so far, there are several XPath functions available that provide you with additional flexibility when selecting XML data. Next we give you an overview over these functions with some examples.

Archived

fn:abs

The fn:abs function returns the absolute value of a numeric value. Example 8-15 shows how this function can be used in an XPath statement. We have selected from the two XML documents shown in Figure 8-30 on page 298.

Example 8-15 vn:abs example

```
SELECT XMLSERIALIZE(
   XMLQUERY('fn:abs(//price)'
   PASSING INFO)
   as clob(1k)) from sabixml6 ;
---------+---------+---------+----

---------+---------+---------+----
9.99
3.99
```

If the result was an empty sequence, fn:abs of this empty sequence is also an empty sequence.

fn:boolean

The fn:boolean function returns the effective boolean value of a sequence.

The returned value depends on the value of sequence-expression:

- If sequence-expression is the empty sequence, false is returned.
- If sequence-expression is not the empty sequence, and sequence-expression contains one value:
  - If the value is the xs:boolean value false, false is returned.
  - If the value is a string of length 0, and the type is xs:string or xdt:untypedAtomic, false is returned.
  - If the value is 0, and the type is a numeric type, false is returned.
  - If the value is NaN, and the type is xs:double, false is returned.
  - Otherwise, true is returned.
- If sequence-expression is not the empty sequence, and sequence-expression contains more than one value, true is returned.

fn:compare

The fn:compare function compares two strings. DB2 compares the numeric Unicode UTF-8 code value of each character from string-1 and string-2 and returns:

-1	If string-1 is less than string-2
0	If string-1 is equal to string-2
1	If string-1 is greater than string-2.

Archived

Example 8-16 is based on the XML documents shown in Figure 8-33 on page 304. We compare the values of attribute country to literal “Germany”. The result for the first row is 0, because the value in the attribute country is Germany. We receive a -1 for the second row, because Canada is encoded as x’43616E616461’, while Germany is x’4765726D616E79’. The hexadecimal encoding of Canada is therefore less than the one for Germany.

Example 8-16 fn:compare values of attribute country to literal ’Germany’

```
SELECT XMLSERIALIZE(
   XMLQUERY('
   declare default element namespace "http://posample.org";
   fn:compare(//addr/@country, "Germany")'
   PASSING INFO)
   as clob(1k)) from sabixml5 ;
---------+---------+---------+---------+---------+---------+

---------+---------+---------+---------+---------+---------+
0
-1
```

Tip: The namespace declaration, which is part of the prolog, must come before the fn:compare function.

fn:concat

The fn:concat function concatenates two or more strings into a single string. In Example 8-17 we concatenate the value of attribute country with a blank and literal Germany.

Example 8-17 fn:concat function example

```
SELECT XMLSERIALIZE(
   XMLQUERY('
   declare default element namespace "http://posample.org";
   fn:concat(//addr/@country," ","Germany")'
   PASSING INFO)
   as clob(1k)) from sabixml5 ;
---------+---------+---------+---------+---------+---------+--

---------+---------+---------+---------+---------+---------+--
Germany Germany
Canada Germany
```

fn:contains

The fn:contains function determines whether a string contains a given substring. Working with the XML documents shown in Figure 8-30 on page 298, we check which product description talks about foam.

Example 8-18 fn:contains function

```
SELECT XMLSERIALIZE(
   XMLQUERY('
(: declare default element namespace "http://posample.org";  :)
   fn:contains(//description,"foam")'
   PASSING INFO)
   as clob(1k)) from sabixml6 ;
---------+---------+---------+---------+--
```

Archived

```
---------+---------+---------+---------+--
false
true
```

The ice scraper has a foam handle. *true* is therefore returned for this XML document.

Attention: We must now remove the namespace declaration, since the XML documents do not contain namespace declarations as well.

Do not use the common '--' commenting characters. Since this appears within a string literal, it is not acknowledged.

Use XPath commenting characters instead. If you want to add XPath comments, enclose the text that you want to have as comment in '(:' and ':)'.

fn:count

The fn:count function returns the number of values in a sequence. Again we use the XML document shown in Figure 8-33 on page 304. In the Example 8-19 we count the number of phone numbers that are stored in each of the XML documents.

Example 8-19 fn:count function

```
SELECT XMLSERIALIZE(
   XMLQUERY('
   declare default element namespace "http://posample.org";
   fn:count(//phone)'
   PASSING INFO)
   as clob(1k)) from sabixml5 ;
---------+---------+---------+---------+---------+---------+

---------+---------+---------+---------+---------+---------+
1
4
```

fn:data

The fn:data function converts a sequence of items to a sequence of atomic values.

Archive

For each item in the sequence:

- If the item is an atomic value, the returned value is that value. Refer to Example 8-20 for a sample.

Example 8-20 fn:data example for an atomic value

```
SELECT XMLSERIALIZE(
   XMLQUERY('
   declare default element namespace "http://posample.org";
   fn:data(//addr/@country)'
   PASSING INFO)
   as clob(1k)) from sabixml5 ;
---------+---------+---------+---------+---------+---------+-

---------+---------+---------+---------+---------+---------+-
Germany
Canada
```

- If the item is a node, the returned value is the typed value of the node. Refer to Example 8-21 for a sample.

Example 8-21 fn:data function example for a node

```
SELECT XMLSERIALIZE(
   XMLQUERY('
   declare default element namespace "http://posample.org";
   fn:data(//customerinfo/name)'
   PASSING INFO)
   as clob(1k)) from sabixml5 ;
---------+---------+---------+---------+---------+---------

---------+---------+---------+---------+---------+---------
Matt Foreman
Robert Shoemaker
```

Example 8-22 helps to see the difference between using fn:data and omitting it.

Example 8-22 XPath on a node as comparison to fn:data

```
SELECT XMLSERIALIZE(
   XMLQUERY('
   declare default element namespace "http://posample.org";
   //customerinfo/name'
   PASSING INFO)
   as clob(1k)) from sabixml5 ;
---------+---------+---------+---------+---------+---------+--

---------+---------+---------+---------+---------+---------+--
<name xmlns="http://posample.org">Matt Foreman</name>
<name xmlns="http://posample.org">Robert Shoemaker</name>
```

fn:not

The fn:not function returns false if the effective boolean value of an item is true, and true if the effective boolean value of an item is false.

Archived

fn:round

The fn:round function returns the integer that is closest to a numeric value.

Since we need a numeric value for this function, we used this on the XML documents shown in Figure 8-30 on page 298. The original prices of the two products stored in this table are 3.99 and 9.99. Example 8-23 shows the rounded decimal values for it.

Example 8-23 fn:round function

```
SELECT XMLSERIALIZE(
   XMLQUERY('
   fn:round(//price)'
   PASSING INFO)
   as clob(1k)) from sabixml6 ;
---------+---------+---------+---

---------+---------+---------+---
10
4
```

fn:string

The fn:string function returns the string representation of a value.

If value is not the empty sequence:

- If value is a node, the returned value is the string value property of the value node.
- If value is an atomic value, the returned value is the result of casting value to the xs:string type.

Example 8-24 shows you an example.

Example 8-24 fn:string function

```
SELECT XMLSERIALIZE(
   XMLQUERY('
   fn:string(//price)'
   PASSING INFO)
   as clob(1k)) from sabixml6 ;
---------+---------+---------+----

---------+---------+---------+----
9.99
3.99
```

fn:substring

The fn:substring function returns a substring of a string.

If the source-string is not the empty sequence, the returned value is a substring of source-string that starts at position start and is length bytes. If source-string is the empty sequence, the result is a string of length 0.

Archived

In Example 8-25 we use the substring function to just extract all different area phone codes from our XML documents shown in Figure 8-33 on page 304.

Example 8-25 vn:substring function with cardinality problem

```
SELECT XMLSERIALIZE(                                                    00014099
   XMLQUERY('                                                           00015099
   declare default element namespace "http://posample.org";             00015199
   fn:substring(//phone ,1 ,3)'                                         00015299
   PASSING INFO)                                                        00016099
   as clob(1k)) from sabixml5 ;                                         00017099
---------+---------+---------+---------+---------+---------+---------+---------+

---------+---------+---------+---------+---------+---------+---------+---------+
021
DSNE610I NUMBER OF ROWS DISPLAYED IS 1
DSNT408I SQLCODE = -16003, ERROR:  AN EXPRESSION OF DATA TYPE ( item(),
         item()+ ) CANNOT BE USED WHEN THE DATA TYPE xs:string IS EXPECTED IN
         THE CONTEXT. ERROR QNAME=err:XPTY0004
```

As you can see from Example 8-25, our query returned an error message. The problem here is that the second XML document contains multiple phone numbers. To avoid this cardinality problem, we must modify our query, as shown in Example 8-26.

Example 8-26 fn:substring function with solved cardinality problem

```
---------+---------+---------+---------+---------+---------+----
   SELECT XMLSERIALIZE(
   XMLQUERY('
   declare default element namespace "http://posample.org";
   //phone/fn:substring(.,1,3)'
   PASSING INFO)
   as clob(1k)) from sabixml5 ;
---------+---------+---------+---------+---------+---------+----

---------+---------+---------+---------+---------+---------+----
021
905 416 905 613
```

fn:sum

The fn:sum function returns the sum of the values in a sequence.

Archived

In order to illustrate the usage of this function, we introduce another XML sample document in Figure 8-35.

```
<PurchaseOrder xmlns="http://posample.org"
PoNum="5002" OrderDate="2004-02-29"
Status="Shipped">
    <item>
        <partid>100-100-01</partid>
        <name>Snow Shovel, Basic 22 inch</name>
        <quantity>3</quantity>
        <price>9.99</price>
    </item>
    <item>
        <partid>100-101-01</partid>
        <name>Snow Shovel, Deluxe 24 inch</name>
        <quantity>5</quantity>
        <price>19.99</price>
    </item>
    <item>
        <partid>100-201-01</partid>
        <name>Ice Scraper, Windshield 4
inch</name>
        <quantity>5</quantity>
        <price>3.99</price>
    </item>
</PurchaseOrder>
```

Figure 8-35 XML Document sample with multiple numeric values

In Example 8-27 we show you how to use the fn:sum function together with arithmetic expressions to calculate the value of the purchase order shown in Figure 8-35.

Example 8-27 fn:sum function

```
---------+---------+---------+---------+---------+---------+---
   SELECT XMLSERIALIZE(
   XMLQUERY('
   declare default element namespace "http://posample.org";
fn:sum(//item/(price * quantity))'
   PASSING INFO)
   as clob(1k)) from sabixml5 ;
---------+---------+---------+---------+---------+---------+---

---------+---------+---------+---------+---------+---------+---
149.87
```

Note: Apply PTF UK25075 for APAR PK41550. It solves issues when using the XPath function fn:sum in XMLQUERY or XMLEXISTS.

8.6 XML indexing

An XML index can be used to improve the efficiency of queries on XML documents that are stored in an XML column.

In contrast to traditional relational indexes, where index keys are composed of one or more table columns that you specify, an XML index uses a particular XML pattern expression to index paths and values in XML documents stored within a single column. The data type of that column must be XML.

Instead of providing access to the beginning of a document, index entries in an XML index provide access to nodes within the document by creating index keys based on XML pattern expressions. Because multiple parts of an XML document can satisfy an XML pattern, DB2 might generate multiple index keys when it inserts values for a single document into the index.

You create an XML index using the CREATE INDEX statement, and drop an XML index using the DROP INDEX statement. The GENERATE KEY USING XMLPATTERN clause you include with the CREATE INDEX statement specifies what you want to index. Figure 8-36 shows the XML-related first extensions to the CREATE INDEX statement.

Some of the keywords used with the CREATE INDEX statement for indexes on non-XML columns do not apply to indexes over XML data. The UNIQUE keyword also has a different meaning for indexes over XML data.

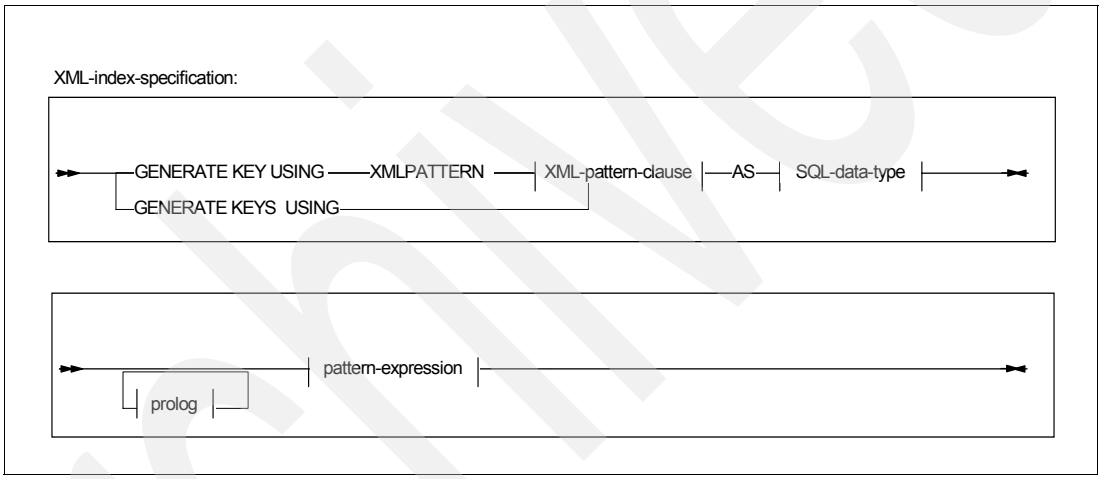


Figure 8-36 XML related parts of CREATE INDEX statement - I

Archived

Figure 8-37 shows the XML-related second extensions to the CREATE INDEX statement.

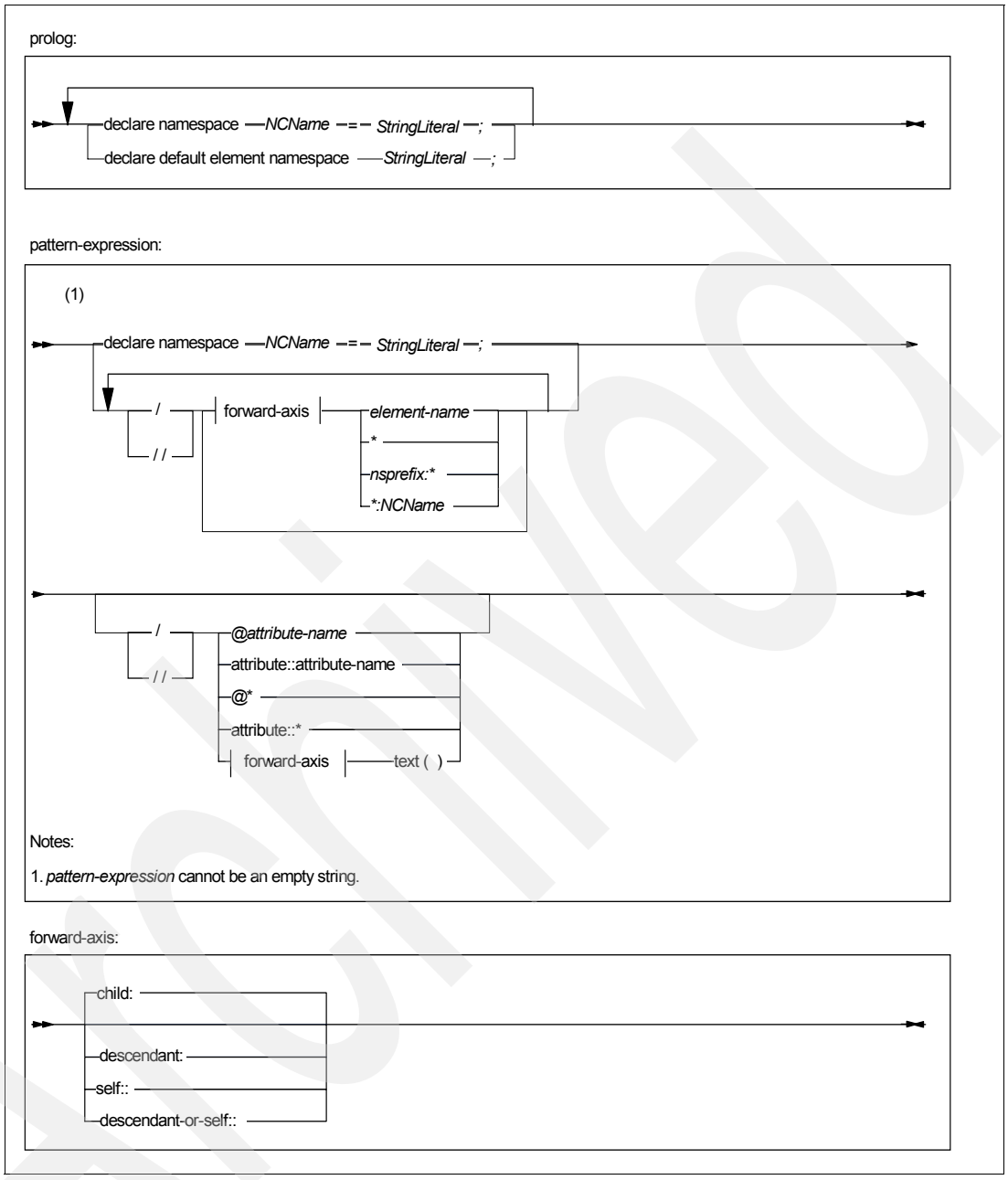


Figure 8-37 XML-related parts of CREATE INDEX statement - I I

Note: The fact that you can index XML data is a big advantage of native XML support over just storing your XML data in LOB columns. The content of LOB columns cannot be indexed.

Archived

To get started with XML indexes, we show you an example of how to create an XML index in Figure 8-38. Later in this chapter we cover different aspects of XML indexing in more detail.

In our example we refer to the XML documents shown in Figure 8-32 on page 301. Those XML documents contain customer information. Many application programs would probably use the customer ID (Cid) to retrieve the related information about a specific customer. Due to this fact, the creation of an index that points directly to the customer ID might make sense.

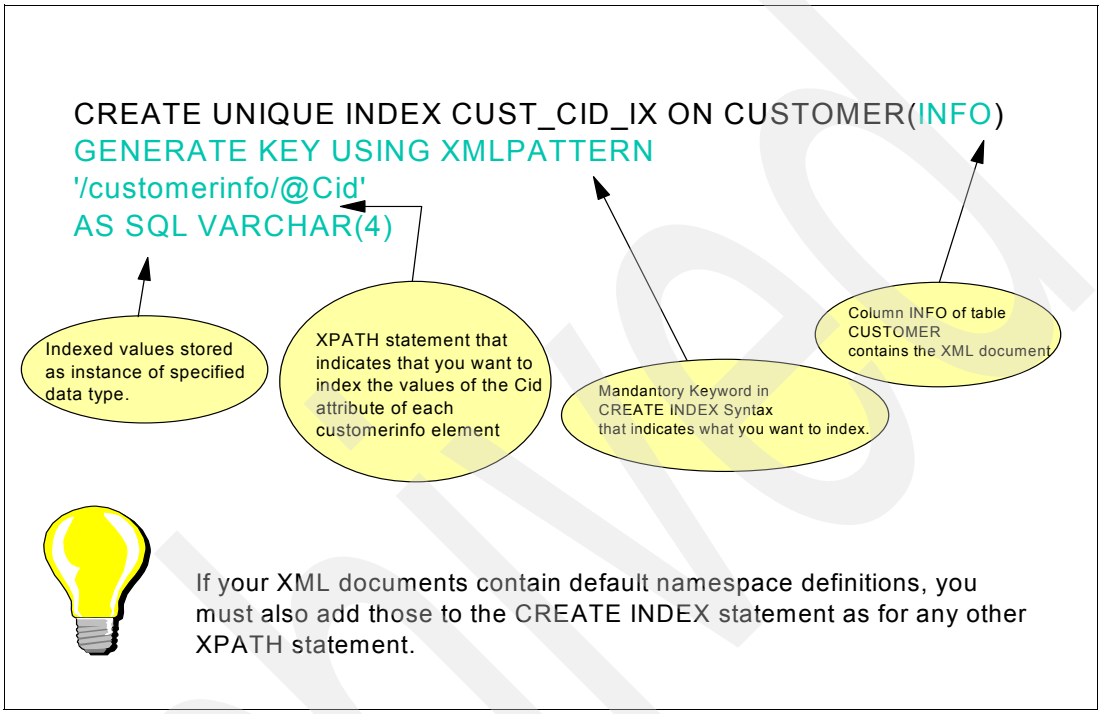


Figure 8-38 XML index example

8.6.1 Data types associated with pattern expressions

Every XML pattern expression that you specify in a CREATE INDEX statement must be associated with a data type. Valid data types are VARCHAR(n), where n <= 1000 or DECFLOAT.

You can interpret the result of pattern expression as multiple data types. For example, the value 123 has a character representation, but it can also be interpreted as the number 123. You can create different indexes on the same pattern expression with different data types so that the data can be indexed, regardless of its data type.

Let us assume that you are working with the very simple XML document shown in Example 8-28.

Example 8-28 Numeric index

```
<a>
   <b>X</b>
   <b>5</b>
</a>
```

If you created an index using XML pattern expression '/a/b' AS SQL DECFLOAT, the index would only contain one entry with value 5.

Let us now look at more complex example. In Example 8-29 we show you the creation of a simple table containing just one XML column, the insertion of one XML document, and the creation of an XML index with XML pattern expression '/CUSTOMERINFO/@CID' AS SQL VARCHAR(4).

Example 8-29 XML index example

```
CREATE TABLE SABIXML5 (CUSTID CHAR(4), INFO XML) ;

COMMIT;

INSERT INTO SABIXML5 VALUES('1006',
'<CUSTOMERINFO
CID="1006"><NAME>MATT FOREMAN</NAME>
<ADDR COUNTRY= "CANADA" ><STREET>1596 BASELINE</STREET>
<CITY>TORONTO</CITY><PROV-STATE>ONTARIO</PROV-STATE>
<PCODE-ZIP>M3Z-5H9</PCODE-ZIP></ADDR> <PHONE
TYPE= "WORK" >905-555-4789</PHONE> <PHONE
TYPE= "HOME" >416-555-3376</PHONE> <ASSISTANT><NAME>GOPHER
RUNNER</NAME>
<PHONE TYPE= "HOME" >416-555-3426</PHONE> </ASSISTANT>
</CUSTOMERINFO>')

COMMIT;

CREATE INDEX CUST_CID_IX3 ON SABIXML5(INFO)
 GENERATE KEY USING XMLPATTERN
 '/CUSTOMERINFO/@CID'
 AS SQL VARCHAR(4)   ;
COMMIT;
```

As you can see, we have associated VARCHAR(4) to the XML pattern expression. As a consequence, if you now try to insert an XML document into table SABIXML5 that contains a customerid that is more than 4 bytes, the index does not allow you to do so. In Example 8-30 we specified CID="10300", which is a 5-byte data string. The resulting SQLCODE -20305 is also shown at the bottom of Example 8-30.

Example 8-30 XML index not allowing insert

```
INSERT INTO SABIXML5 VALUES('1030',
'<CUSTOMERINFO
CID="10300"><NAME>GIALLO BIRDY</NAME>
<ADDR COUNTRY= "GERMANY" ><STREET>7 HOMEWAY</STREET>
<CITY>DUISBURG</CITY><PROV-STATE></PROV-STATE>
<PCODE-ZIP>47111</PCODE-ZIP></ADDR> <PHONE
TYPE= "WORK" >211-471-2345</PHONE> <PHONE
TYPE= "HOME" >203-555-3376</PHONE>
</CUSTOMERINFO> ')

DSNT408I SQLCODE = -20305, ERROR:  AN XML VALUE CANNOT BE INSERTED OR UPDATED
         BECAUSE OF AN ERROR DETECTED WHEN INSERTING OR UPDATING THE INDEX
         IDENTIFIED BY 'DBID~268 OBID~12' ON TABLE *N. REASON CODE = 1.
```

Archived

To remind you that all XML data is stored as Unicode UTF-8, Example 8-31 shows you DB2's behavior when trying to insert value '§300' into CID. Looking at the value, one would think that §300 is a 4-byte character and should therefore be accepted by XML index pattern '/CUSTOMERINFO/@CID' as SQL VARCHAR(4)', but character '§' has Unicode codepoint U+00A7. Codepoints that are beyond U+007F need at least two bytes for storing it as Unicode UTF-8. As a consequence, '§300' needs 5 bytes to store this value in the XML column.

Example 8-31 Insert character that is larger than Unicode codepoint U+007F

```
INSERT INTO SABIXML5 VALUES('1030',
'<CUSTOMERINFO
CID="§300"><NAME>GIALLO BIRDY</NAME>
<ADDR COUNTRY= "GERMANY" ><STREET>7 HOMEWAY</STREET>
<CITY>DUISBURG</CITY><PROV-STATE></PROV-STATE>
<PCODE-ZIP>47111</PCODE-ZIP></ADDR> <PHONE
TYPE= "WORK" >211-471-2345</PHONE> <PHONE
TYPE= "HOME" >203-555-3376</PHONE>
</CUSTOMERINFO> ')

DSNT408I SQLCODE = -20305, ERROR:  AN XML VALUE CANNOT BE INSERTED OR UPDATED
         BECAUSE OF AN ERROR DETECTED WHEN INSERTING OR UPDATING THE INDEX
         IDENTIFIED BY 'DBID~268 OBID~12' ON TABLE *N. REASON CODE = 1.
```

Now let us look at it the other way around. Assuming that your table already contains the two XML documents that we had inserted and tried to insert (shown in Example 8-29 on page 316 and Example 8-31) before we created the index. Now we issue the CREATE INDEX statement shown in Example 8-32. As you can see, the creation fails with SQLCODE -20306, because CID="§300" is a 5-byte value.

Example 8-32 CREATE INDEX on column that already contains XML documents

```
CREATE INDEX CUST_CID_IX3 ON sabixml5(INFO)
GENERATE KEY USING XMLPATTERN
'/CUSTOMERINFO/@CID'
AS SQL VARCHAR(4)   ;

DSNT408I SQLCODE = -20306, ERROR:  AN INDEX ON AN XML COLUMN CANNOT BE CREATED
         BECAUSE OF AN ERROR DETECTED WHEN INSERTING THE XML VALUES INTO THE
         INDEX. REASON CODE = 1
```

For indexes storing any data type other than VARCHAR, DB2 tries to cast the value to the data type as specified in the CREATE INDEX statement. If cast fails, because of incompatible data types, the value is ignored without inserting into the index. Although the index is not updated as a result of the insertion, the XML document will nevertheless be inserted into the table without any error or warning.

An example of the above explanation could be the attempt to cast 'A5B' to DECFLOAT.

8.6.2 The UNIQUE keyword in an XML index definition

The UNIQUE keyword in XML index definitions has a different meaning than it does for relational index definitions. For relational indexes, the UNIQUE keyword in the CREATE INDEX statement enforces uniqueness across all rows in a table.

For indexes over XML data, the UNIQUE keyword enforces uniqueness across all documents in an XML column. For an XML index, DB2 enforces uniqueness for:

- The data type of the index.
- The XML path to a node.
- The value of the node after the XML value has been cast to the SQL data type that is specified for the index.

Because rounding can occur during conversion of an index key value to the specified data type for the index, multiple values that appear to be unique in the XML document might result in duplicate key errors.

8.6.3 Miscellaneous facts and limits on XML indexes

Looking at XML indexes, there are some things that we would like to bring to your attention. Please refer to the following list to learn more about some interesting facts and limits on XML indexes:

- Composite indexes are not supported.
- You can define multiple indexes per XML column.
- The number of keys for each XML document depends on the document itself and the XMLPattern that you use in your index definition.
- Valid XMLPatterns are path expressions without predicates.
- Indexes are only used when using XMLEXIST. XMLSERIALIZE does not use indexes.
- You can use a maximum of 50 steps in your index XMLPatterns.
- XML indexes can only index values from element nodes, attribute nodes, or text nodes in a document. Comment nodes or Processing Instruction nodes are not supported.
- The maximum length of the XMLPattern text is 4000 bytes.

  Attention: The XMLPattern text is encoded in Unicode UTF-8. Be aware that characters might need more than one byte for its hexadecimal representation.

- Index key values cannot span multiple rows.

  Tip: It is always safe for you to index leaf nodes with short values.

- DB2 does not support partitioned indexes for XML columns.

8.6.4 XML index exploitation

XML indexing is only supported for the XMLEXISTS predicate. It does not work for XMLQUERY. XPath index exploitation is much like summary table matching. DB2 must use matching logic for the exploitation.

Some general rules for index exploitation are:

Containment The index contains the data that XMLEXISTS XPath needs (for example, //c can be used for /a/b/c).

This means that the index pattern is equal to or less restrictive than the query predicate.

Index: //product/regprice versus

Query: /catalog//product[regprice > 10]

Data type The data type of the index and the XML document value must match except for existential match (that is, a numeric can use a string index).

Now we introduce some heuristic rules:

- Matching without // step is better than with // step.
- Matching without a wildcard (*) is better than with a wildcard.
- Matching with more steps is better than matching with less steps.

In conjunction with XML indexing, a couple of new access methods have been introduced. Refer to Table 8-10.

Table 8-10 New access methods

Access method	Description
DocScan “R” (QuickXScan)	Base algorithm: given document, scan and evaluate XPath
DocID list access “DX” unique DocID list from an XML index, then access the base table and XML table.	’/Catalog/Categories/Product[RegPrice > 100]’ with index on ’/Catalog/Categories/Product/RegPrice’ as SQL DECFLOAT
DocID ANDing/ORing “DX/DI/DU” intersect or union (unique) DocID lists from XML indexes, then access the base table and XML table.	’/Catalog/Categories/Product[RegPrice > 100 and Discount > 0.1]’ With index on: ’//RegPrice’ as SQL DECFLOAT and ’//DISCOUNT’ as SQL DECFLOAT

After this theoretical explanation, next we show you some examples of index definitions and queries that use them.

Example 1

The query shown in Example 8-33 includes an equal predicate on a numeric type. It retrieves documents from the Description column of the sample Product table for items with a price equal to 9.99.

Example 8-33 XPath query example for indexing 1

```
SELECT Description FROM Product WHERE XMLEXISTS('$x/product/description[price =
9.99]' PASSING BY REF Description AS "x")
```

To be compatible, the XML index needs to include price nodes among the indexed nodes and to store values as the DECFLOAT type. The query can use the XML index shown in Example 8-34.

Example 8-34 XML index usage example 1

```
CREATE INDEX prodIndex on Product(Description) GENERATE KEY USING XMLPATTERN
'//price' AS SQL DECFLOAT
```

Example 2

The query shown in Example 8-35 includes an equal predicate on a string type. It retrieves all documents from the Info column of the sample Customer table for customers who have cell phone numbers.

Example 8-35 XPath query example for indexing 2

```
SELECT Info FROM Customer WHERE XMLEXISTS('$x/customerinfo/phone[@type="cell"]'
PASSING BY REF Info AS "x")
```

To be compatible with this query, the XML index needs to include the type attribute node among the indexed nodes and to store values in the index as a VARCHAR type. The query can use the index shown in Example 8-36.

Example 8-36 XML index usage example 2

```
CREATE INDEX custIndex on Customer(Info) GENERATE KEY USING XMLPATTERN '//@type'
AS SQL VARCHAR(20)
```

Example 3

The query shown in Example 8-37 includes an equal predicate on a text node. It retrieves all documents from the Info column of the sample Customer table for which the assistant name is Gopher Runner.

Example 8-37 XPath query example for indexing 3

```
SELECT Info FROM Customer WHERE XMLEXISTS('$x/customerinfo/assistant[name="Gopher
Runner"]' PASSING BY REF Info AS "x")
```

To be compatible with this query, the XML index needs to include the text node within the name element that is under the assistant element, and needs to store values in the index as a VARCHAR type. The query can use the XML index shown in Example 8-38.

Example 8-38 XML index usage example 3

```
CREATE INDEX custIndex on Customer(Info) GENERATE KEY USING XMLPATTERN
'/customerinfo/assistant/name/text()' AS SQL VARCHAR(20)
```

Example 4

The query shown in Example 8-39 includes a test for the existence of a node. The XMLEXISTS predicate evaluates to true if at least one product node has a batteries node.

Example 8-39 XPath query example for indexing 4

```
SELECT Description FROM Product WHERE XMLEXISTS('/product//batteries' PASSING
Description)
```

To be compatible with this query, the XML index needs to include the product//batteries path, and needs to store values in the index as a VARCHAR type. The query can use the XML index shown in Example 8-40.

Example 8-40 XML index usage example 4

```
CREATE INDEX prodIndex on Product(Description) GENERATE KEY USING XMLPATTERN
'/product//batteries' AS SQL VARCHAR(20)
```

Archived

8.6.5 Catalog support for XML indexes

You can find the qualified names of the index and the associated table in SYSIBM.SYSINDEXES with the following values:

- UNIQUERULE = 'U'

  If it is a unique index, the key value will have to be unique. If it is a non-unique index, the triplet (keyvalue, Docid, nodeID) is unique.
- KEYTARGET_COUNT = 3
- UNIQUE_COUNT = 3 for a non-unique index, 1 for a non-unique index
- IX_EXTENSION_TYPE = 'V'

The name of the XML columns on which the index is created is shown in the SYSIBM.SYSKEYS catalog table. Column ORDERING is set to blank.

8.7 XML schema support

XML schema support allows you to request that the XML documents that you are inserting into your DB2 tables or those that you update be validated and augmented against an XML schema. In order for you to do this, the XML schema must be available to DB2.

A DB2 for z/OS XML schema repository (XSR) is a set of DB2 tables where you can store XML schemas. The DB2 database system creates the XSR tables during installation or migration. After you add XML schemas to the DB2 XSR, you can use them to validate XML documents before you store them in XML columns.

8.7.1 XML schema information in the DB2 catalog

The XML schema repository is what is stored in database DSNXSR. The creation of DSNXSR is part of installation job DSNTIJSG. DSNTIJSG creates database DSNXSR and all dependent objects.

Archived

Database DSNXSR contains one base table space SYSXSR holding four base tables that build the external interface for you regarding XML schema support. Figure 8-39 shows all of the objects and their interrelations.

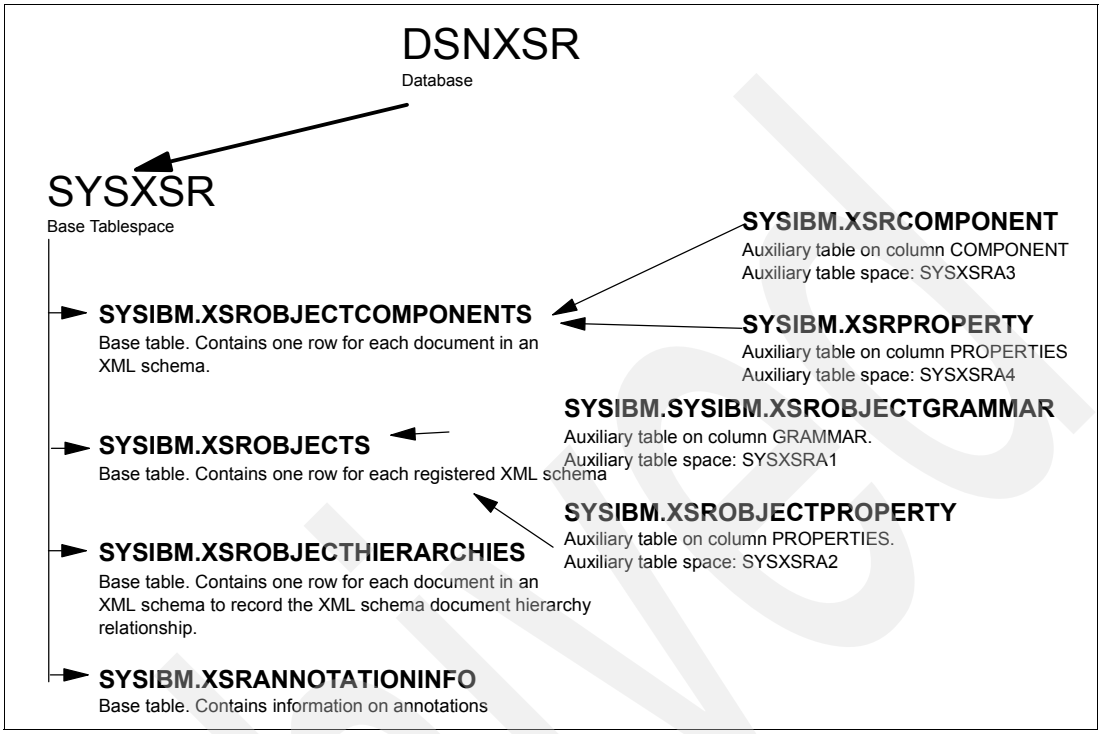


Figure 8-39 DSNXSR schema repository database

Attention: The auxiliary tables shown in Figure 8-39 are also listed in the SQL reference. If you try to select from those tables, you run into an SQL error telling you that you cannot directly select from an auxiliary table.

8.8 XML schema validation

Because DSN_XMLValidate is a UDF, it is not as efficient as XMLParse.

To insert data into an XML column, you use the SQL INSERT statement. The data that you insert into the XML column must be a well-formed XML document, as defined in the XML 1.0 specification.

XML data in an application is in its serialized string format. When you insert the data into an XML column, it must be converted to its XML hierarchical format. If the application data type is an XML data type, the DB2 database server performs this operation implicitly. If the application data type is not an XML type, you can invoke the XMLPARSE function explicitly when you perform the insert operation, to convert the data to the XML hierarchical format. During document insertion, you might also want to validate the XML document against a registered XML schema. You can do that with the DSN_XMLVALIDATE function.

XML validation is the process of determining whether the structure, content, and data types of an XML document are valid. In addition, XML validation strips ignorable whitespace from the input document. Validation is optional.

Before you can invoke DSN_XMLVALIDATE, all schema documents that make up an XML schema must be registered in the built-in XML schema repository (XSR). An XML schema provides the rules for a valid XML document.

You can invoke DSN_XMLVALIDATE only in the string-expression argument of the XMLPARSE function. When you invoke DSN_XMLVALIDATE within XMLPARSE, you must specify the STRIP WHITESPACE option for XMLPARSE.

In your INSERT statement, the DSN_XMLVALIDATE function specifies the XML schema to use for validation, as shown in Example 8-41.

Example 8-41 Schema validation

```
INSERT into PurchaseOrdersVALUES( '200300001', CURRENT DATE, 'A',
DSN_XMLValidate(:xmlPo,SYSXSR.ORDERSchema));
```

8.9 XML decomposition

Decomposition, also often called shredding, is the process of storing content from an XML document in columns of relational tables. After it is decomposed, the data has the SQL type of the column into which it is inserted.

An XML schema consists of one or more XML schema documents. In annotated XML schema decomposition, you control decomposition by including decomposition annotations in the XML schemas. The annotations specify:

- The name of the target table and column
- The default SQL schema for the target table
- Instructions for transformation of the data before storage

DB2 for z/OS always validates the data from XML documents during decomposition. If information in an XML document does not comply with its specification in an XML schema document, DB2 for z/OS does not insert the information into a DB2 table.

Annotated XML schema decomposition breaks an XML document down for storage in tables, based on the annotations. The procedure is:

1. Annotate XML schema documents with XML decomposition annotations.
2. Register the XML schema documents, and enable the XML schemas for decomposition.
   - If you use stored procedures, set the *issuedfordecomposition* parameter to 1 in the SYSPROC.XSR_COMPLETE stored procedure.
   - If you use JDBC methods, set the isUsedForShredding parameter to true in the DB2Connection.registerDB2XmlSchema method call.
3. Decompose the XML document by calling the SYSPROC.XDBDECOMPXML stored procedure.

For details see *DB2 Version 9.1 for z/OS XML Guide,* SC18-9858.

8.10 Utilities support of XML data

You can use DB2 9 for z/OS utilities on XML objects. The utilities handle XML objects in a similar way how they handle LOB objects, but for some utilities you need to specify certain XML keywords.

In this section we discuss the different utilities that are affected by XML.

8.10.1 REPORT

When you specify REPORT TABLESPACESET, the output report includes XML objects in the list of members in the table space set. Since most utilities run largely on SPACE names, we can rely on the REPORT utility showing SPACE names for all the implicitly created objects and associating those names with the proper base table column.

Example 8-42 shows the table space set for our sample table SABIXML5.

Example 8-42 REPORT TABLESPACESET sample output

```
TABLESPACE SET REPORT:

TABLESPACE          : DSN00003.SABIXML5
  TABLE             : PAOLOR7.SABIXML5
  INDEXSPACE        : DSN00003.IRDOCIDS
  INDEX             : PAOLOR7.I_DOCIDSABIXML5


XML TABLESPACE SET REPORT:

TABLESPACE                 : DSN00003.SABIXML5

  BASE TABLE               : PAOLOR7.SABIXML5
  COLUMN                   : INFO
    XML TABLESPACE         : DSN00003.XSAB0000
      XML TABLE            : PAOLOR7.XSABIXML5
      XML NODEID INDEXSPACE: DSN00003.IRNODEID
      XML NODEID INDEX     : PAOLOR7.I_NODEIDXSABIXML5
      XML INDEXSPACE       : DSN00003.CUSTRCID
      XML INDEX            : PAOLOR7.CUST_CID_IX3
      XML INDEXSPACE       : DSN00003.CUST16J2
      XML INDEX            : PAOLOR7.CUST_CID_IX6
```

As you can see, this output is a good reference to all DB2 objects that are related to your XML table space.

This is the easiest way to relate the generated space names to the table and column names. The table and column names are new in this release and are useful working with LOB page sets also.

8.10.2 CHECK DATA

CHECK DATA checks XML relationships. If the base table space is not consistent with any related XML table spaces, CHECK DATA reports the error.

The utility does not check informational referential constraints for XML objects.

Archived

SCOPE REFONLY and AUXONLY

If you do not specify anything else, the default behavior of CHECK DATA is to check all objects in your utility list that are in CHKP restrictive state. As for LOB table spaces, you can also limit the scope of which objects you want CHECK DATA to check. The two SCOPE keywords are:

SCOPE REFONLY Checks base table only

SCOPE AUXONLY Checks LOB and XML objects only

CHECK DATA error keywords

You can also specify the action that DB2 performs when it finds an error in one of these columns. Specify one of the appropriate keywords shown in Table 8-11.

Table 8-11 CHECK DATA error keywords

Column in error	Action that CHECK DATA takes	Keyword
XML	Report the error only.	XMLERROR REPORT
	Report the error and set the column in error to an invalid status.	XMLERROR INVALIDATE
LOB	Report the error only.	LOBERROR REPORT
	Report the error and set the column in error to an invalid status.	LOBERROR INVALIDATE
XML or LOB	Report the error only.	AUXERROR REPORT
	Report the error and set the column in error to an invalid status.	AUXERROR INVALIDATE

8.10.3 CHECK INDEX

You can use the CHECK INDEX utility to check DocID and NodeID indexes.

Example 8-43 shows the sample output of a CHECK INDEX utility run.

Example 8-43 CHECK INDEX sample utility output

```
DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = TEMP
DSNUGTIS - PROCESSING SYSIN AS EBCDIC
DSNUGUTC -  CHECK INDEX(ALL) TABLESPACE DSN00003.SABIXML5
DSNUKPIK - INDEXES WILL BE CHECKED IN PARALLEL, NUMBER OF TASKS = 3
.44 DSNUKIUL - 3 INDEX ENTRIES UNLOADED FROM 'DSN00003.SABIXML5'
DSNUKPIK - UNLOAD PHASE COMPLETE - ELAPSED TIME=00:00:00
DSNUKPIK - 3 ENTRIES CHECKED FOR INDEX 'PAOLOR7.I_DOCIDSABIXML5'
DSNUKPIK - SORTCHK PHASE COMPLETE, ELAPSED TIME=00:00:00
.65 DSNUKTER - 3 ENTRIES CHECKED FOR INDEX 'PAOLOR7.I_DOCIDSABIXML5'
DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

Archived

If you compare the number of indexes that are checked by the CHECK INDEX utility with the number of indexes identified by -DIS DB(DSN00003), which is shown in Example 8-44, you can see that not all indexes are checked when you run the CHECK INDEX utility.

Example 8-44 -DIS DB(DSN00003) output

```
DSNT360I  -DB9A **********************************
DSNT361I  -DB9A *  DISPLAY DATABASE SUMMARY
                *    GLOBAL
DSNT360I  -DB9A **********************************
DSNT362I  -DB9A     DATABASE = DSN00003  STATUS = RW
                    DBD LENGTH = 8066
DSNT397I  -DB9A
NAME     TYPE PART  STATUS            PHYERRLO PHYERRHI CATALOG  PIECE
-------- ---- ----- ----------------- -------- -------- -------- -----
SABIXML5 TS    0001 RW
XSAB0000 XS    0001 RW
CUST16J2 IX    L*   RW
CUSTRCID IX    L*   RW
IRDOCIDS IX    L*   RW
IRNODEID IX    L*   RW
******* DISPLAY OF DATABASE DSN00003 ENDED     **********************
DSN9022I  -DB9A DSNTDDIS 'DISPLAY DATABASE' NORMAL COMPLETION
***
```

Restriction: CHECK INDEX does not check the user-defined XML indexes such as CUST16J2 and CUSTRCID in Example 8-44. Instead, only the DocID index and the NodeID index are checked.

8.10.4 COPY

You can use the COPY utility to copy XML objects. No additional keyword is needed, but when you specify that you want DB2 to copy a table space with XML columns, DB2 does not automatically copy any related XML table spaces or indexes.

To be able to recover your XML documents, you must explicitly copy the base and XML table spaces, as shown in Example 8-45. Table space DSN00003.SABIXML5 contains the base table and DSN00003.XSAB0000 holds the XML data.

Example 8-45 COPY base and XML table

```
//UTIL EXEC DSNUPROC,SYSTEM=DB9A,UID='TEMP',UTPROC=''
//DSNUPROC.SYSCOPY DD DSN=PAOLOR7.SABIXML.BASE.IC1,
//     DISP=(MOD,CATLG),
//     SPACE=(16384,(20,20),,,ROUND),
//     UNIT=SYSDA
//DSNUPROC.SYSCOPY1 DD DSN=PAOLOR7.SABIXML.XML.IC1,
//     DISP=(MOD,CATLG),
//     SPACE=(16384,(20,20),,,ROUND),
//     UNIT=SYSDA
//DSNUPROC.SYSIN    DD  *
COPY TABLESPACE DSN00003.SABIXML5 COPYDDN(SYSCOPY)
COPY TABLESPACE DSN00003.XSAB0000 COPYDDN(SYSCOPY1)
```

8.10.5 REPORT RECOVERY

This can be used on XML objects with no additional keywords to show recovery information for XML objects.

8.10.6 RECOVER table space, RECOVER index, REBUILD index

Talking about the COPY UTILITY, we must also look at the RECOVER utility. You can use the RECOVER utility to recover XML objects. You do not have to specify any additional keywords in the RECOVER statement.

Nevertheless, when recovering page sets that contain XML columns, analogously to the COPY utility, it is not enough to just recover the base table space. Let us assume that you have lost all page sets that existed for database DSN00003 containing table space SABIXML5.

To reestablish the environment you first run a RECOVER TABLESPACE DSN00003.SABIXML5 utility. If you now check the existing page sets in ISPF 3.4, the result is as shown in Example 8-46.

Example 8-46 Page sets after RECOVER base table

```
DSLIST - Data Sets Matching DB9AU.*.DSN00003                    Row 1 of 2
Command ===>                                                 Scroll ===> CSR

Command - Enter "/" to select action              Message           Volume
-------------------------------------------------------------------------------
         DB9AU.DSNDBC.DSN00003.SABIXML5.I0001.A001                  *VSAM*
         DB9AU.DSNDBD.DSN00003.SABIXML5.I0001.A001                  SBOX0C
```

Trying to select from your XML column, the result is shown in Example 8-47.

Example 8-47 SELECT from XML column after RECOVER base table space

```
select info from sabixml5
  where xmlexists('//CUSTOMERINFO[@CID="§300"]'
  passing info)
---------+---------+---------+---------+---------+---------+---------+---------+
...
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE610I NUMBER OF ROWS DISPLAYED IS 0
DSNT408I SQLCODE = -904, ERROR:  UNSUCCESSFUL EXECUTION CAUSED BY AN
         UNAVAILABLE RESOURCE. REASON 00C200E2, TYPE OF RESOURCE 00000220, AND
         RESOURCE NAME DB9AU.DSNDBC.DSN00003.XSAB0000.I0001.A001
```

The error message clearly shows that your XML table space page set marked in bold in Example 8-47 is not available. To make it available, you must recover your XML table space or table spaces if your table contains more than one column with data type XML. This works just as you would need to recover related LOB table spaces or related RI table spaces. You can use REPORT TABLESPACESET to identify related table spaces and their index spaces.

Attention: In this situation it is absolutely necessary that you have taken image copies of every single XML table space that is associated with your base table space.

Archive

After recovering DSN00003.XSAB0000, ISPF 3.4 displays the VSAM clusters shown in Example 8-48.

Example 8-48 ISPF 3.4 after recovering base and XML table space

```
DSLIST - Data Sets Matching DB9AU.*.DSN00003                        Row 1 of 4
Command ===>                                                     Scroll ===> CSR

Command - Enter "/" to select action                 Message          Volume
-------------------------------------------------------------------------------
         DB9AU.DSNDBC.DSN00003.SABIXML5.I0001.A001                    *VSAM*
         DB9AU.DSNDBC.DSN00003.XSAB0000.I0001.A001                    *VSAM*
         DB9AU.DSNDBD.DSN00003.SABIXML5.I0001.A001                    SBOX0C
         DB9AU.DSNDBD.DSN00003.XSAB0000.I0001.A001                    SBOX0C
```

Next we tried to select from XML column INFO of table SABIXML5. The result is shown in Example 8-49.

Example 8-49 Attempt to select from SABIXML5 column INFO

```
select info from sabixml5
  where xmlexists('//CUSTOMERINFO[@CID="§300"]'
  passing info)
---------+---------+---------+---------+---------+---------+---------+---------+
....
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE610I NUMBER OF ROWS DISPLAYED IS 0
DSNT408I SQLCODE = -904, ERROR:  UNSUCCESSFUL EXECUTION CAUSED BY AN
         UNAVAILABLE RESOURCE. REASON 00D70024, TYPE OF RESOURCE 00000220, AND
         RESOURCE NAME DB9AU.DSNDBC.DSN00003.IRNODEID.I0001.A001
```

You get another -904 SQLCODE. The reason for it is that your NodeID index is still not available. If you check the status of this index using the -DIS DB(DSN00003) command, you receive the result shown in Example 8-50.

Example 8-50 -DIS DB(..) after recovering base and XML table space

```
DSNT360I  -DB9A ************************************
DSNT361I  -DB9A *  DISPLAY DATABASE SUMMARY
                *    GLOBAL
DSNT360I  -DB9A ************************************
DSNT362I  -DB9A     DATABASE = DSN00003  STATUS = RW
                   DBD LENGTH = 8066
DSNT397I  -DB9A
NAME     TYPE PART  STATUS            PHYERRLO PHYERRHI CATALOG  PIECE
-------- ---- ----- ----------------- -------- -------- -------- -----
SABIXML5 TS    0001 RW
XSAB0000 XS    0001 RW
CUST16J2 IX    L*   RW
CUSTRCID IX    L*   RW
IRDOCIDS IX    L*   RW
IRNODEID IX    L*   RW
******* DISPLAY OF DATABASE DSN00003 ENDED      **********************
DSN9022I  -DB9A DSNTDDIS 'DISPLAY DATABASE' NORMAL COMPLETION
***
```

Important: As you can see from Example 8-50, none of the remaining objects show restrictive states. So you must remember on your own that you must either recover or rebuild all indexes that are defined on your XML table. Talking about *all* means that those are not only the indexes that you have created for performance or unique reasons, but also the NodeID and DocID index.

Since in our sample scenario we have not copied any index, we must now rebuild all existing indexes.

We use utility control statement REBUILD INDEX(ALL) TABLESPACE DSN00003.SABIXML5. After successful completion of the utility we check ISPF 3.4 again and find what is shown in Example 8-51.

Example 8-51 ISPF 3.4 after REBUILD INDEX(ALL) TABLESPACE DSN00003.SABIXML5

```
Menu  Options  View  Utilities  Compilers  Help
 sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
 DSLIST - Data Sets Matching DB9AU.*.DSN00003                    Row 1 of 6
 Command ===>                                                 Scroll ===> CSR

 Command - Enter "/" to select action                Message          Volume
 -------------------------------------------------------------------------------
          DB9AU.DSNDBC.DSN00003.IRDOCIDS.I0001.A001                     *VSAM*
          DB9AU.DSNDBC.DSN00003.SABIXML5.I0001.A001                     *VSAM*
          DB9AU.DSNDBC.DSN00003.XSAB0000.I0001.A001                     *VSAM*
          DB9AU.DSNDBD.DSN00003.IRDOCIDS.I0001.A001                     SBOX0C
          DB9AU.DSNDBD.DSN00003.SABIXML5.I0001.A001                     SBOX0C
          DB9AU.DSNDBD.DSN00003.XSAB0000.I0001.A001                     SBOX0C
```

Only one index has been rebuilt, because the other missing indexes are not associated to table space DSN00003.SABIXML5, but to the XML table space DSN00003.XSAB0000.

As a consequence, we now run another REBUILD INDEX utility using control statement REBUILD INDEX(ALL TABLESPACE DSN00003.XSAB0000. After successful completion we have finally reestablished all necessary page sets that are listed in Example 8-52.

Example 8-52 Complete list of page sets associated to SABIXML5

```
Menu  Options  View  Utilities  Compilers  Help
sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
DSLIST - Data Sets Matching DB9AU.*.DSN00003                    Row 1 of 12
Command ===>                                                 Scroll ===> CSR

Command - Enter "/" to select action                Message          Volume
-------------------------------------------------------------------------------
         DB9AU.DSNDBC.DSN00003.CUSTRCID.I0001.A001                     *VSAM*
         DB9AU.DSNDBC.DSN00003.CUST16J2.I0001.A001                     *VSAM*
         DB9AU.DSNDBC.DSN00003.IRDOCIDS.I0001.A001                     *VSAM*
         DB9AU.DSNDBC.DSN00003.IRNODEID.I0001.A001                     *VSAM*
         DB9AU.DSNDBC.DSN00003.SABIXML5.I0001.A001                     *VSAM*
         DB9AU.DSNDBC.DSN00003.XSAB0000.I0001.A001                     *VSAM*
         DB9AU.DSNDBD.DSN00003.CUSTRCID.I0001.A001                     SBOX0C
         DB9AU.DSNDBD.DSN00003.CUST16J2.I0001.A001                     SBOX0C
         DB9AU.DSNDBD.DSN00003.IRDOCIDS.I0001.A001                     SBOX0C
         DB9AU.DSNDBD.DSN00003.IRNODEID.I0001.A001                     SBOX0C
```

```
DB9AU.DSNDBD.DSN00003.SABIXML5.I0001.A001                    SBOX0C
DB9AU.DSNDBD.DSN00003.XSAB0000.I0001.A001                    SBOX0C
```

Restriction: You cannot use REBUILD INDEX SHRLEVEL CHANGE for XML indexes.

8.10.7 COPYTOCOPY

You can use the COPYTOCOPY utility to copy existing copies of the XML objects. You do not have to specify any additional keywords.

8.10.8 EXEC SQL

You cannot declare a cursor that includes XML data. Thus, you cannot use the DB2 family cross-loader function to transfer data from XML columns.

8.10.9 LISTDEF

When you create object lists with the LISTDEF utility, you can specify whether you want related XML objects to be included or excluded. Use the following keywords to indicate the objects to include or exclude:

ALL Base, LOB, and XML objects. This keyword is the default.

BASE Base objects only

LOB LOB objects only

XML XML objects only.

Referring to the rows of examples that we used in 8.10.4, “COPY” on page 326, and 8.10.6, “RECOVER table space, RECOVER index, REBUILD index” on page 327, we could have used the statement shown in Example 8-53 to copy all objects belonging to SABIXML5, and the one shown in Example 8-54 to recover everything that needs to be recovered.

Example 8-53 Copy all table spaces related to SABIXML5

```
//UTIL EXEC DSNUPROC,SYSTEM=DB9A,UID='TEMP',UTPROC=''
//DSNUPROC.SYSIN    DD  *
LISTDEF XMLTS INCLUDE TABLESPACES DATABASE DSN00003
TEMPLATE CP2 DSN(PAOLOR7.&TS..IC6) UNIT SYSDA
COPY LIST XMLTS  COPYDDN(CP2)
```

Example 8-54 Recover all objects related to SABIXML5

```
//UTIL EXEC DSNUPROC,SYSTEM=DB9A,UID='TEMP',UTPROC=''
//DSNUPROC.SYSIN    DD  *
LISTDEF XMLTS INCLUDE TABLESPACES DATABASE DSN00003
LISTDEF XMLIX INCLUDE INDEXSPACES DATABASE DSN00003
RECOVER LIST XMLTS
REBUILD INDEXSPACE LIST XMLIX
```

Archived

The resulting job output of Example 8-54 on page 330 is shown in Example 8-55.

Example 8-55 Job output for LISTDEF RECOVER

```
06:54:02.12 DSNUGUTC -  LISTDEF XMLTS INCLUDE TABLESPACES DATABASE DSN00003
06:54:02.12 DSNUILDR - LISTDEF STATEMENT PROCESSED SUCCESSFULLY
06:54:02.12 DSNUGUTC -  LISTDEF XMLIX INCLUDE INDEXSPACES DATABASE DSN00003
06:54:02.12 DSNUILDR - LISTDEF STATEMENT PROCESSED SUCCESSFULLY
06:54:02.12 DSNUGUTC -  RECOVER LIST XMLTS
06:54:02.13 DSNUGULM - PROCESSING LIST ITEM: TABLESPACE DSN00003.SABIXML5
06:54:02.13 DSNUGULM - PROCESSING LIST ITEM: TABLESPACE DSN00003.XSAB0000
06:54:02.15 DSNUCBMD - RECOVER TABLESPACE DSN00003.SABIXML5   START
06:54:02.15 DSNUCBAL - THE IMAGE COPY DATA SET PAOLOR7.SABIXML5.IC6 WITH DATE=20
       IS PARTICIPATING IN RECOVERY OF TABLESPACE DSN00003.SABIXML5
06:54:02.57 DSNUCBMD - MERGE STATISTICS FOR TABLESPACE DSN00003.SABIXML5   -
       NUMBER OF COPIES=1
       NUMBER OF PAGES MERGED=3
       ELAPSED TIME=00:00:00
06:54:02.58 DSNUCBMD - RECOVER TABLESPACE DSN00003.XSAB0000   START
06:54:02.58 DSNUCBAL - THE IMAGE COPY DATA SET PAOLOR7.XSAB0000.IC6 WITH DATE=20
       IS PARTICIPATING IN RECOVERY OF TABLESPACE DSN00003.XSAB0000
06:54:02.95 DSNUCBMD - MERGE STATISTICS FOR TABLESPACE DSN00003.XSAB0000   -
       NUMBER OF COPIES=1
       NUMBER OF PAGES MERGED=3
       ELAPSED TIME=00:00:00
084 06:54:02.96 DSNUCALA - FAST LOG APPLY WAS NOT USED FOR RECOVERY
06:54:02.96 DSNUCBLA - LOG APPLY PHASE COMPLETE, ELAPSED TIME = 00:00:00
06:54:03.01 DSNUCBDR - RECOVERY COMPLETE, ELAPSED TIME=00:00:00
06:54:03.04 DSNUGUTC -  REBUILD INDEXSPACE LIST XMLIX
06:54:03.05 DSNUGULM - PROCESSING LIST ITEM: INDEXSPACE DSN00003.CUST16J2
06:54:03.05 DSNUGULM - PROCESSING LIST ITEM: INDEXSPACE DSN00003.CUSTRCID
06:54:03.05 DSNUGULM - PROCESSING LIST ITEM: INDEXSPACE DSN00003.IRNODEID
06:54:03.12 DSNUCRIB - INDEXES WILL BE BUILT IN PARALLEL, NUMBER OF TASKS = 7
084 06:54:03.14 DSNUCRUL - UNLOAD PHASE STATISTICS - NUMBER OF RECORDS PROCESSED
06:54:03.15 DSNUCRIB - UNLOAD PHASE COMPLETE - ELAPSED TIME=00:00:00
084 06:54:03.87 DSNURBXC - SORTBLD PHASE STATISTICS - NUMBER OF KEYS=3 FOR INDEX
XSABIXML5
084 06:54:03.89 DSNURBXC - SORTBLD PHASE STATISTICS - NUMBER OF KEYS=3 FOR INDEX
084 06:54:03.91 DSNURBXC - SORTBLD PHASE STATISTICS - NUMBER OF KEYS=3 FOR INDEX
06:54:04.05 DSNUCRIB - SORTBLD PHASE STATISTICS. NUMBER OF INDEXES = 3
06:54:04.05 DSNUCRIB - SORTBLD PHASE COMPLETE, ELAPSED TIME = 00:00:00
06:54:04.06 DSNUGULM - PROCESSING LIST ITEM: INDEXSPACE DSN00003.IRDOCIDS
06:54:04.12 DSNUCRIB - INDEXES WILL BE BUILT IN PARALLEL, NUMBER OF TASKS = 3
084 06:54:04.13 DSNUCRUL - UNLOAD PHASE STATISTICS - NUMBER OF RECORDS PROCESSED
06:54:04.13 DSNUCRIB - UNLOAD PHASE COMPLETE - ELAPSED TIME=00:00:00
084 06:54:04.53 DSNURBXC - SORTBLD PHASE STATISTICS - NUMBER OF KEYS=3 FOR INDEX
ABIXML5
06:54:04.53 DSNUCRIB - SORTBLD PHASE STATISTICS. NUMBER OF INDEXES = 1
06:54:04.53 DSNUCRIB - SORTBLD PHASE COMPLETE, ELAPSED TIME = 00:00:00
06:54:04.55 DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

For your reference, we have marked all objects that are either recovered or rebuilt as bold.

8.10.10 LOAD

You can use the LOAD utility to load XML data. The steps for loading XML data are similar to the steps for loading other types of data, except that you need to also perform the following actions:

- In the input data set:
  - If the data set is in delimited format, ensure that the XML input fields follow the standard LOAD utility delimited format.
  - If the data set is not in delimited format, specify the XML input fields similar to the way that you specify VARCHAR input. Specify the length of the field in a 2-byte binary field that precedes the data.
- In the LOAD statement:
  - Specify the keyword XML for all input fields of type XML.
  - If you want the whitespace to be preserved in the XML data, also specify the keywords PRESERVE WHITESPACE. By default, LOAD strips the whitespace.

Loading XML data with the LOAD utility has the following restrictions:

- You cannot specify that XML input fields be loaded into non-XML columns, such as CHAR or VARCHAR columns.
- DB2 does not perform any specified compression until the next time that you run the REORG utility on this data.
- DB2 ignores any specified FREEPAGE and PCTFREE values until the next time that you run the REORG utility on this data.
- If you specify PREFORMAT, DB2 preformats the base table space, but not the XML table space.
- You cannot directly load the DocID column of the base table space.
- You cannot specify a default value for an XML column.
- You cannot load XML values that are greater than 32 KB. To load such values, use file reference variables in the LOAD utility, or use applications with SQL XML AS file reference variables.
- LOAD REPLACE deletes an existing compression dictionary. LOAD RESUME keeps it.

Note: Apply PTF UK25744 for APAR PK43315, when available, for LOAD/UNLOAD XML file references to support PRESERVE WHITESPACE keyword and UNICODE CCSID.

8.10.11 QUIESCE

When you specify QUIESCE TABLESPACESET, the table space set includes related XML objects. You do not have to specify any additional keywords in the QUIESCE statement.

The Quiesce utility job output listed in Example 8-56 shows that you can specify either the XML or base table space to quiesce both of them, and related index spaces if they are copy enabled.

Example 8-56 QUIESCE TABLESPACESET job output

```
DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = TEMP
DSNUGTIS - PROCESSING SYSIN AS EBCDIC
DSNUGUTC -  QUIESCE TABLESPACESET DSN00003.XSAB0000
```

```
.40 DSNUQUIA - QUIESCE SUCCESSFUL FOR TABLESPACESET DSN00003.XSAB0000
.40 DSNUQUIA -    QUIESCE SUCCESSFUL FOR TABLESPACE DSN00003.SABIXML5
.40 DSNUQUIA -    QUIESCE SUCCESSFUL FOR TABLESPACE DSN00003.XSAB0000
.40 DSNUQUIA - QUIESCE AT RBA 00000429C527 AND AT LRSN 00000429C527
DSNUQUIB - QUIESCE UTILITY COMPLETE, ELAPSED TIME= 00:00:00
DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

8.10.12 REORG INDEX and REORG TABLESPACE

You can use the REORG utility to reorganize XML objects. You do not need to specify any additional keywords in the REORG statement.

When you specify the name of the base table space in the REORG statement, DB2 reorganizes only that table space and not any related XML objects. If you want DB2 to reorganize the XML objects, you must specify those object names. When you specify that you want XML table spaces to be reorganized, you must also specify the WORKDDN keyword and provide the specified temporary work file. The default is SYSUT1.

For XML table spaces and base table spaces with XML columns, you cannot specify the following options in the REORG statement:

- DISCARD
- REBALANCE
- UNLOAD EXTERNAL

8.10.13 REPAIR

You can use the REPAIR utility on XML objects. You do not have to specify any additional keywords in the REPAIR statement.

8.10.14 RUNSTATS

You can use the RUNSTATS utility to gather statistics for XML objects.

RUNSTATS TABLESPACE ignores the following keywords for XML table spaces:

- COLGROUP
- FREQVAL MOST/LEAST/BOTH
- HISTOGRAM

RUNSTATS TABLESPACE collects COLCARD for the DOCID column only for an XML table. COLGROUP, FREQVAL MOST/LEAST/BOTH, and HISTOGRAM do not apply, since DB2 is only interested in the number of XML documents stored in the table, not the specific docid values.

RUNSTATS INDEX ignores the following keywords for XML indexes or NodeID indexes:

- KEYCARD
- FREQVAL MOST/LEAST/BOTH
- HISTOGRAM/UNLOAD

RUNSTATS INDEX collects, for an XML index, HIGH2KEY, LOW2KEY, AVGKEYLEN, FIRSTKEYCARD, FULLKEYCARD, and so on. DB2 is interested in the key value extracted from the XML column by the XPath specified in the XML indexes.

Archived

8.10.15 UNLOAD

You can use the UNLOAD utility to unload XML data.

As stated many times already, your XML data is stored in Unicode UTF-8. If you unload your table data without specifying that you want to unload the data using the Unicode encoding scheme, your XML documents are converted to your system CCSID or to whatever you specify in your UNLOAD DATA statement.

Example 8-57 is the leftmost part of the SYSREC data set that results from unloading the data stored in table SABIXML5. As you can see from the information in bold, the XML document has been converted to the EBCDIC 37 encoding scheme.

Example 8-57 UNLOAD DATA without specifying UNICODE - SYSREC

```
EDIT       PAOLOR7.SYSREC                                  Columns 00001 00072
Command ===>                                                 Scroll ===> CSR
****** ***************************** Top of Data ******************************
000001    1006  z<?xml version="1.0" encoding="IBM037"?><CUSTOMERINFO CID="1006
000002    1030   <?xml version="1.0" encoding="IBM037"?><CUSTOMERINFO CID="$300
000003    1030   <?xml version="1.0" encoding="IBM037"?><CUSTOMERINFO CID="§300
```

For maximum portability, you should specify UNICODE in the UNLOAD statement and use Unicode delimiter characters. The conversion from Unicode UTF-8 to EBCDIC 37 that was used in the unload resulted in the output of Example 8-57 and is a so-called enforced subset conversion. Enforced subset means that those characters stored in your XML documents that are not part of the EBCDIC 37 codepage will be converted to x'1A' and cannot be converted back to the original values.

Example 8-58 shows the same data as above encoded in Unicode UTF-8.

Example 8-58 UNLOAD DATA with specifying Unicode - SYSREC

```
UNLOAD DATA FROM TABLE SABIXML5 UNICODE
```

Archive

9

e-business and connectivity

Technology has not stood still for mainframe systems in the last few decades. They have become adept at accommodating software that has been around for the last several decades such as IMS, DB2, and CICS. They have also become a host for running Web applications built in Java and accommodate the latest business requirements.

Java Web applications need the services of a *driver* to access the database server. DB2 Version 8 provided a universal driver that uses the DRDA protocol to access data on any local or remote server. The universal driver has enhanced usability and portability.

DB2 9 for z/OS continues to add functions related to better alignment with the customer's strategic requirements and general connectivity improvements.

In this chapter we provide a description of the following topics:

- Internet protocol Version 6 (IPv6) support
- Positioning DB2 for removal of private protocol
- Run DDF without SNA when using TCP/IP only
- IBM DB2 Driver for JDBC and SQLJ

© Copyright IBM Corp. 2007. All rights reserved.

9.1 Internet Protocol Version 6 support

Figure 9-1 highlights the restrictions with the current Internet protocol IPv4 and the need to support the new Internet protocol IPv6.

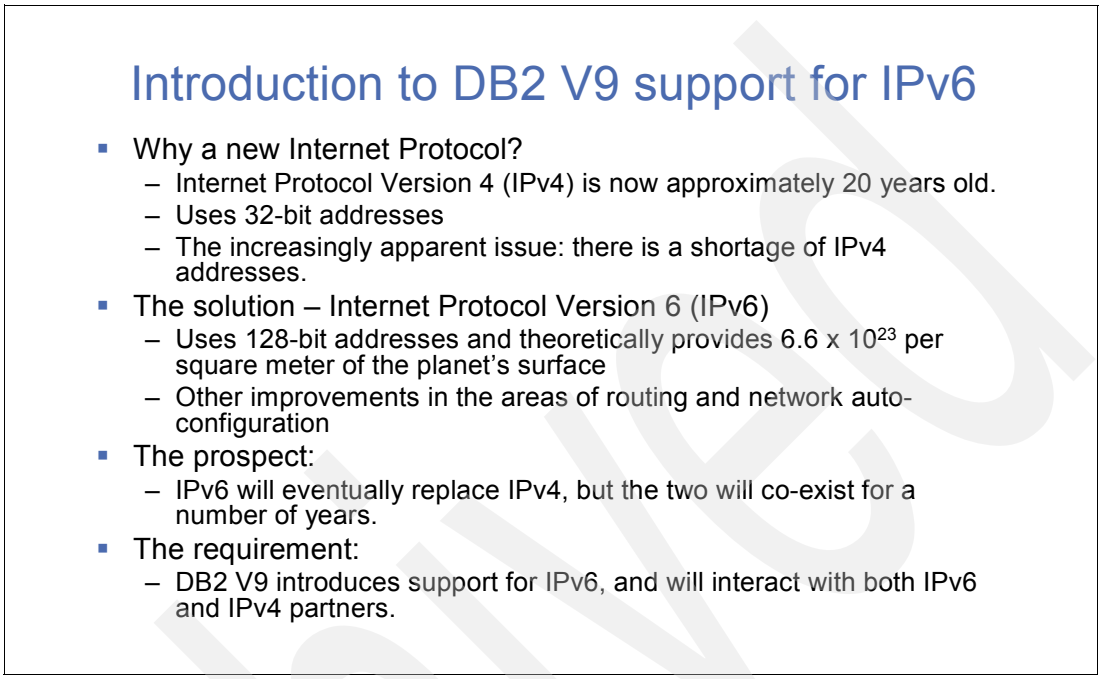


Figure 9-1 Introduction to DB2 V9 support for IPv6

Internet Protocol Version 6 (IPv6) is the next generation of the Internet protocol designed to replace the current version, Internet Protocol Version 4 (IPv4). Most of today's Internets use IPv4, which is approximately 20 years old and is approaching the end of its physical limits. The most significant issue surrounding IPv4 is the growing shortage of IPv4 addresses. In theory, the 32-bit IPv4 address allows over 4 billion nodes, each with a globally unique address. In practice, the interaction between routing and addressing makes it impossible to exploit more than a small fraction of that number of nodes. To utilize IPv6 addresses the requirement is z/OS Communications Server Version 1 Release 5 (z/OS V1R5).

IPv6 uses a 128-bit address space, which has no practical limit on global addressability and provides 3.4 x 10^{50} unique addresses. There are enough addresses so that every single person on the planet could have a single IPv6 network with as many as 1.8 x 10^{19} unique nodes on it, and still the address space would be almost completely unused. To put it another way, this theoretically provides 6.6 x 10^{23} addresses per square meter of the planet's surface (assuming that the earth's surface is 5.1 x 10^{14} square meters).

9.1.1 Textual representation of IPv6 addresses

IPv4 addresses are represented in the familiar dotted-decimal format. The 32-bit address is divided along 8-bit boundaries. Each of the four sets of 8 bits is converted to its decimal equivalent and separated by periods. An IPv4 dotted-decimal address has a maximum of 15 characters.

In contrast, IPv6 addresses are 128 bits divided along 16-bit boundaries. Each of the eight sets of 16-bit blocks is converted to a 4-digit hexadecimal number that is separated by colons.

The resulting representation is called colon-hexadecimal, shown in Figure 9-2. An IPv6 colon-hexadecimal address has a maximum of 39 characters.

It is not necessary to write the leading zeros in an individual field, but there must be at least one numeral in every field. However, it is common for addresses to contain long strings of zero bits. A special syntax is used to compress the zeros. The use of a double colon (::) indicates multiple groups of 16 bits of zeros. The :: can also be used to compress both the leading and trailing zeros in an address. The double colon can only be used once.

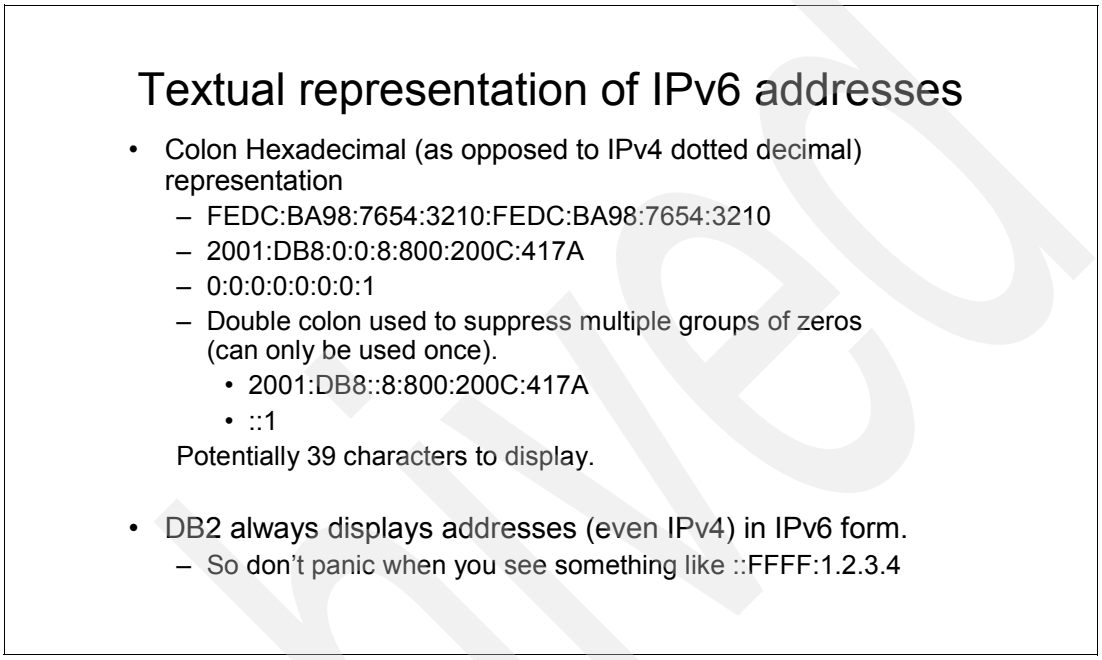


Figure 9-2 Textual representation of IPv6 addresses

Archived

9.1.2 Dual-mode stack support

z/OS Communications Server can be an IPv4-only stack or a dual-mode stack. Dual-mode stack (see Figure 9-2 on page 337) refers to a single TCP/IP stack supporting both IPv4 and IPv6 protocols at the same time. There is no support for an IPv6-only stack.

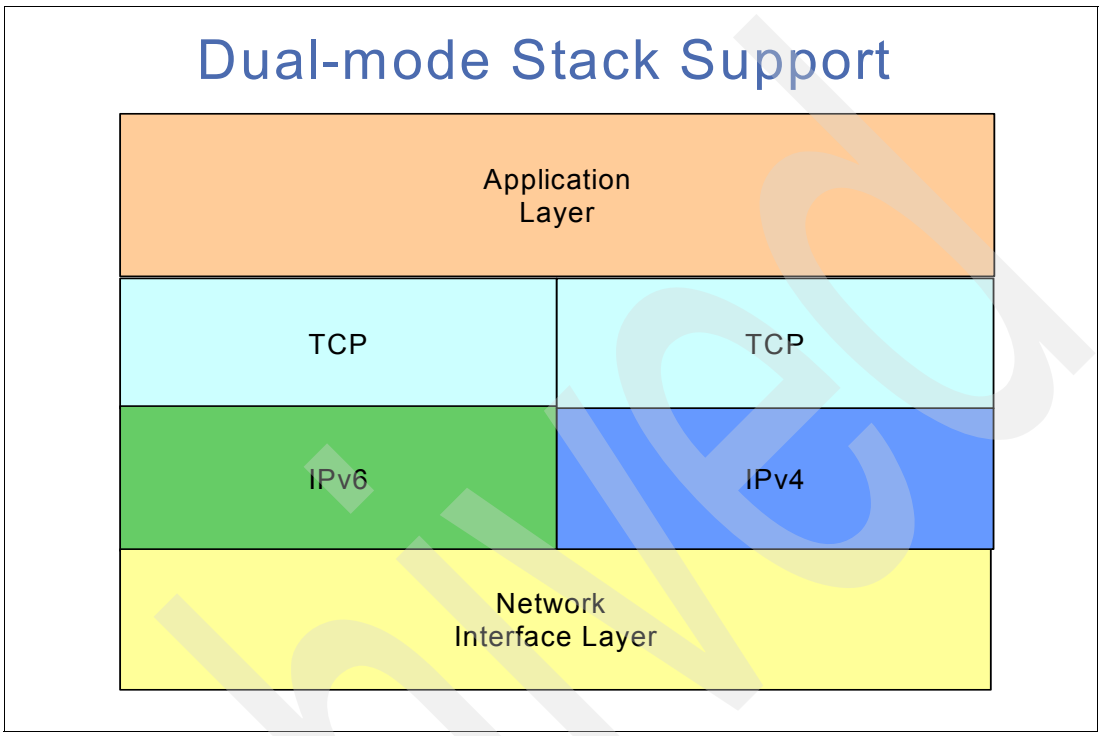


Figure 9-3 Dual-mode stack support

There are several advantages of running a dual-mode stack configuration:

- IPv4 and IPv6 applications can coexist on a single dual-mode stack.
- Unmodified applications can continue to send data over an IPv4 network.
- A single IPv6-enabled application (such as DB2 9 for z/OS) can communicate using IPv4 and IPv6.
- IPv4 and IPv6 can coexist on the same devices and networks.

By default, IPv6-enabled applications can communicate with both IPv4 and IPv6 peers. An IPv6 application on a dual-mode stack can communicate with IPv4 and IPv6 partners as long as it does not bind to a native IPv6 address. If it binds to a native IPv6 address, then it cannot communicate with IPv4 partners since the native IPv6 address cannot be converted to an IPv4 address. If both partners are IPv6, then all communication will use IPv6 packets.

To enable TCP/IP for IPv4/IPv6 dual-mode stack, the PARMLIB member BPXPRMxx must be modified to include an additional NETWORK statement that includes DOMAINNAME(AF_INET6).

No DB2 configuration changes are required. Refer to the *z/OS Communication server - IPv6 Network and Design Guide* for more information.

Support for SQL over IPv6 is only allowed in New Function Mode and when a dual-mode stack exists. If the Communications Database configuration indicates the use of an IPv6 address, either directly or indirectly via the DNS, an SQLCODE -4700 is returned if DB2 is in

Compatibility Mode, or a -904 00D31204 is returned if TCP/IP is not configured with a dual-mode stack.

Attention: Once in New Function Mode with a dual-mode stack, and IPv6 is used, it is possible, albeit unlikely, to get distributed related indoubt threads associated to an IPv6 partner. It is then possible, albeit even more unlikely, that the user could go back to Compatibility Mode while these IPv6-related indoubt threads still exist. It is best to not go back to Compatibility Mode with outstanding IPv6-related indoubt threads, but if this happens then DB2 still requires the dual-mode stack so its *automatic resolution* (resynchronization/resync) processing can resolve the indoubt work with the remote IPv6 partner. Only DB2 resync processing uses IPv6 in Compatibility Mode. New SQL induced IPv6 use is restricted with SQLCODE -4700.

9.1.3 DB2 commands

Figure 9-4 lists the reasons for changes to DB2 commands because of the support for IPv6.

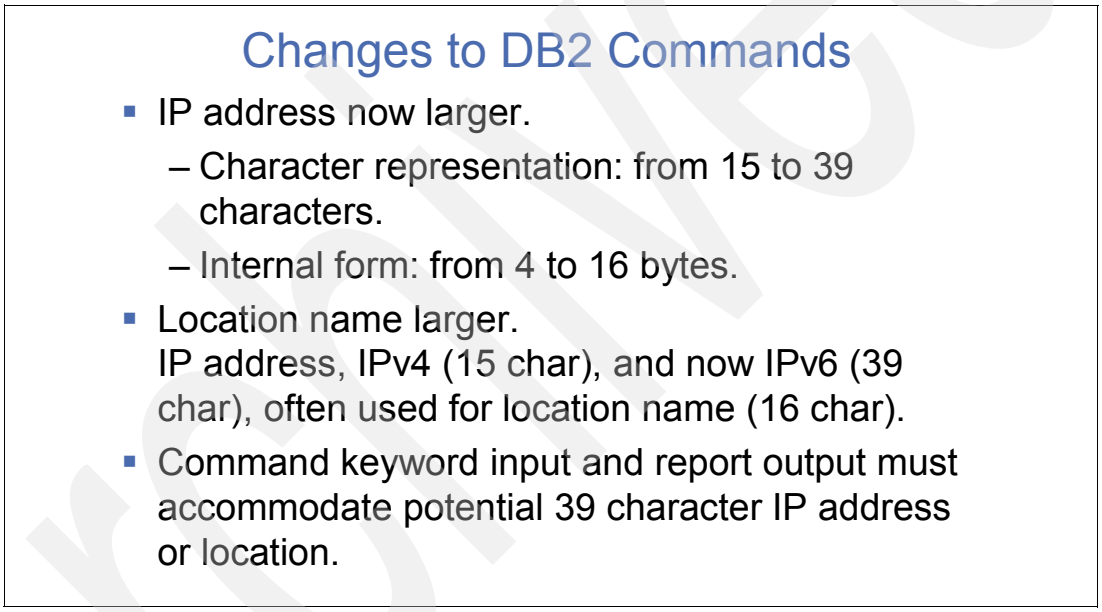


Figure 9-4 Changes to DB2 commands

Some command reports contain the IP address (which now potentially requires up to 39 characters to display) or a location name (which can now also potentially require up to 39 characters since a partner's IP address is sometimes used for a location name). As a result, some command report output must be reconfigured to accommodate the longer IP address or location name. This becomes an issue for programs that parse DB2 command reports, as the report format is different in DB2 9 for z/OS, regardless of whether IPv6 support is exploited. Also, some commands require an IP address as input to the command and changes are therefore necessary to support the new IPv6 address form.

Archive

Figure 9-5 lists the DB2 command reports that have changed because of the support for IPv6.

Figure 9-5 DB2 command reports changed

We now examine a complex example with the intent of highlighting the differences in the output for major commands.

Archived

Figure 9-6 shows the output of the -DISPLAY THREAD TYPE ACTIVE command prior to V9, and Figure 9-7 on page 342 shows the changed output in V9.

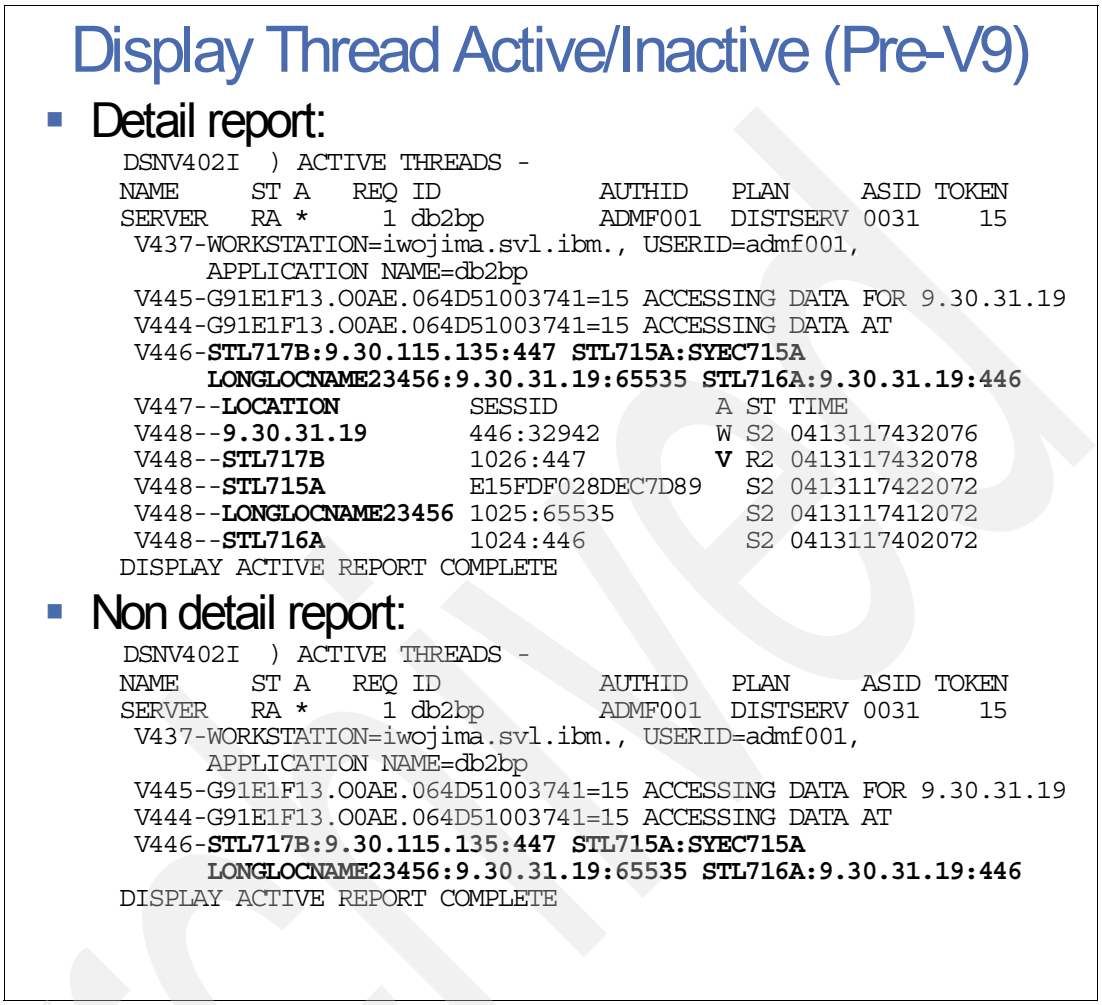


Figure 9-6 Display Thread Active/Inactive (Pre-V9)

Previously, the V446 message used to delimit the location name, the IP address, and the SQL Port number of the downstream server by colons. In the case of an SNA address, there are only two components, a location name and LUNAME, also delimited by a colon. As the IP address can be either an IPv4 dotted-decimal address or an IPv6 colon-hexadecimal address, a colon delimiter is no longer suitable. If DETAIL is specified, a DSNV447I message and one or more DSNV448I messages are issued that also contain a location name. These may contain an IPv4 dotted-decimal address and may now also contain an IPv6 colon-hexadecimal address. Therefore, there is not enough room in the command report for the DSNV447I and DSNV448I messages to directly include a potential for a 39-character location name.

Since the TCP/IP NETSTAT command reports use a double period (..) to delimit an IP address from a port, DB2 now also adopts this convention. To delimit the location name from the IP address or LUNAME, a single dash (-) is now used instead of a colon. Since a dash is now being used as a delimiter of the two components of a DSNV446I SNA location, for consistency, a dash is now also used as a delimiter of the two components of a DSNV445I SNA location. If DETAIL is specified, the DSNV445I and DSNV446I messages now provide an index number, in parenthesis, which is associated to the conversation detail information described by the DSNV447I /DSNV448I messages.

Archived

This index reference is appended to the front of the location name information. To provide the maximum possible space for the worst-case DSNV446I three-part entry, the entry information on DSNV446I lines is shifted to the left by four characters, and the V446 message identifier is removed. The DSNV444I and DSNV450I messages already indicate that a DSNV446 message follows. For the sake of readability, only one location now appears on a DSNV446I line, whether or not the detail report is requested. See Figure 9-7.

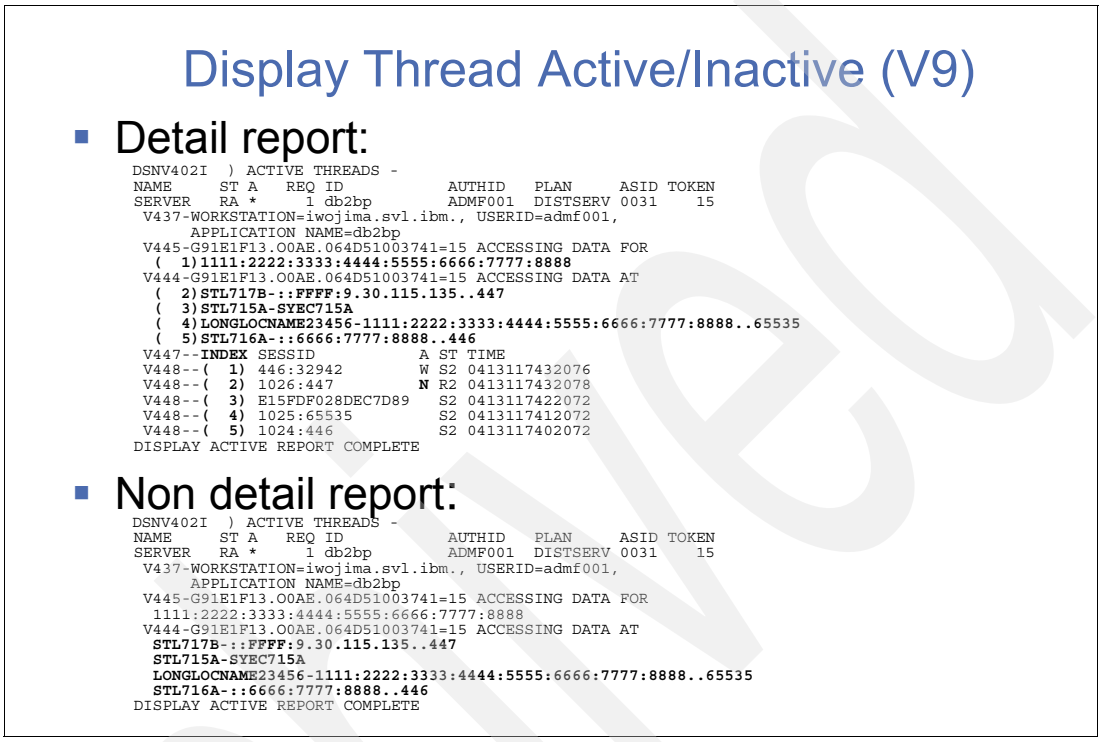


Figure 9-7 Display Thread Active/Inactive (V9)

The structure of the conversation detail information (messages DSNV447I and DSNV448I) is modified so that it no longer contains location information (already provided in the DSNV445I/DSNV446I information anyway), but shows an index column that relates to the DSNV445I or DSNV446I messages. Even if only one location is being reported, the detail report still contains the index reference. Finally, the status information in the DSNV448I detail message indicates 'V' when the conversation is active in the network (SNA or TCP/IP). The 'V' was used when VTAM® (SNA) was the only network interface, but continued to be used for TCP/IP connections. This is the network status indicator is modified to something suitable for both SNA and TCP/IP networks. As a result, the 'V' indicator is changed to 'N', for Network.

9.1.4 Utilities

The Change Log Inventory (DSNJU003) and Print Log Map (DSNJU004) utilities are changed so that the IPv4 and IPv6 values in the BSDS can be configured and displayed.

Figure 9-8 shows the changes to the Change Log Inventory (DSNJU003) utility.

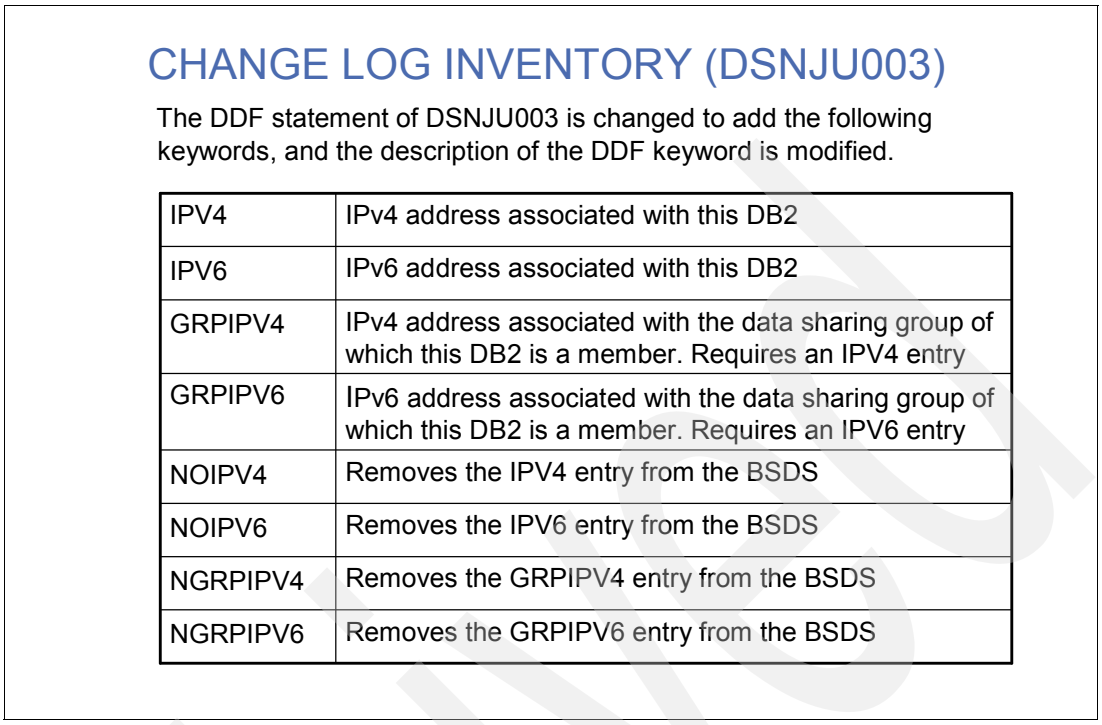


Figure 9-8 CHANGE LOG INVENTORY (DSNJU003)

IPV4 identifies a constant IPv4 (dotted-decimal) address to be associated with DDF for the purposes of accepting incoming connection requests to this specific subsystem only. If an IP address is not specified, DB2 automatically determines the IP address from TCP/IP.

IPV6 identifies a constant IPv6 (colon-hexadecimal) address to be associated with DDF for the purposes of accepting incoming connection requests to this specific subsystem only.

GRPIPV4 identifies a constant IPv4 (dotted-decimal) address to be associated with the data sharing group for which this DDF is a member. It is used for the purposes of accepting incoming connection requests that can be serviced by any member of the data sharing group. An associated IPv4 subsystem/member address (see IPV4) must also be specified in order to identify the IP address associated to this specific member of the group.

GRPIPV6 identifies a constant IPv6 (colon-hexadecimal) address to be associated with the data sharing group for which this DDF is a member. It is used for the purposes of accepting incoming connection requests that can be serviced by any member of the data sharing group. An associated IPv6 subsystem/member address (see IPV6) must also be specified in order to identify the IP address associated to this specific member of the group.

The IP address moves with DB2 to different LPARs.

The last four all remove the corresponding entry from the BSDS.

The Print Log Map (DSNJU004) output is changed. Previously, the DDF information was provided at the end of the output, after all the log information. Since the log information can be very lengthy, and the DDF information can be very difficult to find, it is now provided at the front of the output, before the log information.

The Print Log Map (DSNJU004) displays IPv4 and IPv6 addresses, and IPNAME.

9.1.5 Miscellaneous

Instrumentation trace records for IFCIDs 0165, 0184, 0189, and 0319 have been changed to accommodate IPv6 addresses. IFCID 0319 is implemented for the Kerberos-based authentication and replaces IFCID0312, which was used prior to DB2 V7 for DCE security. IFCID0312 has been removed. Changes are also in place for various messages and codes.

9.2 Positioning DB2 for removal of private protocol

SQL applications in DB2 for z/OS can access remote data using two protocols or methods: private protocol or DRDA protocol. In almost all cases, you should use DRDA access instead of DB2 private protocol access because DRDA access has the following advantages over DB2 private protocol access:

- Integration

  DRDA access is available to all DBMSs that implement Distributed Relational Database Architecture (DRDA). Those include supported releases of DB2 for z/OS, other members of the DB2 family of IBM products, and many products of other companies.

  DB2 private protocol access is available only to supported releases of DB2 for z/OS.

- SQL compatibility

  DRDA access allows any statement that the server can execute. DB2 private protocol access supports only data manipulation statements: INSERT, UPDATE, DELETE, SELECT, OPEN, FETCH, and CLOSE. In addition, you cannot use any syntax of an SQL statement that was introduced after DB2 Version 5. You cannot invoke user-defined functions and stored procedures or use LOBs or distinct types in applications that use DB2 private protocol access.

- Reduced network load

  DRDA access uses a more compact format for sending data over the network, which improves the performance on slow network links.

- Reduced bind processing

  A DBRM for statements executed by DRDA access is bound to a package at the server only once. Those statements can include PREPARE and EXECUTE, so your application can accept dynamic statements that are to be executed at the server. Binding the package is an extra step in program preparation. Queries that are sent by DB2 private protocol access are bound at the server whenever they are first executed in a unit of work. Repeated binds can reduce the performance of a query that is executed often.

- Stored procedures

  You can use stored procedures with DRDA access. While a stored procedure is running, it requires no message traffic over the network. This reduces the biggest obstacle to high performance for distributed data.

- Scrollable cursors

  You can use scrollable cursors if you use DRDA access.

- Savepoints

  You can set savepoints only if you use DRDA access with explicit CONNECT statements. If you set a savepoint and then execute an SQL statement with a three-part name, an SQL error occurs.

The site at which a savepoint is recognized depends on whether the CONNECT statement is executed before or after the savepoint is set. For example, if an application executes the

statement SET SAVEPOINT C1 at the local site before it executes a CONNECT TO S1 statement, savepoint C1 is known only at the local site. If the application executes CONNECT to S1 before SET SAVEPOINT C1, the savepoint is known only at site S1.

Figure 9-9 summarizes the reasons why support for private protocol is planned to be removed from DB2 after Version 9.

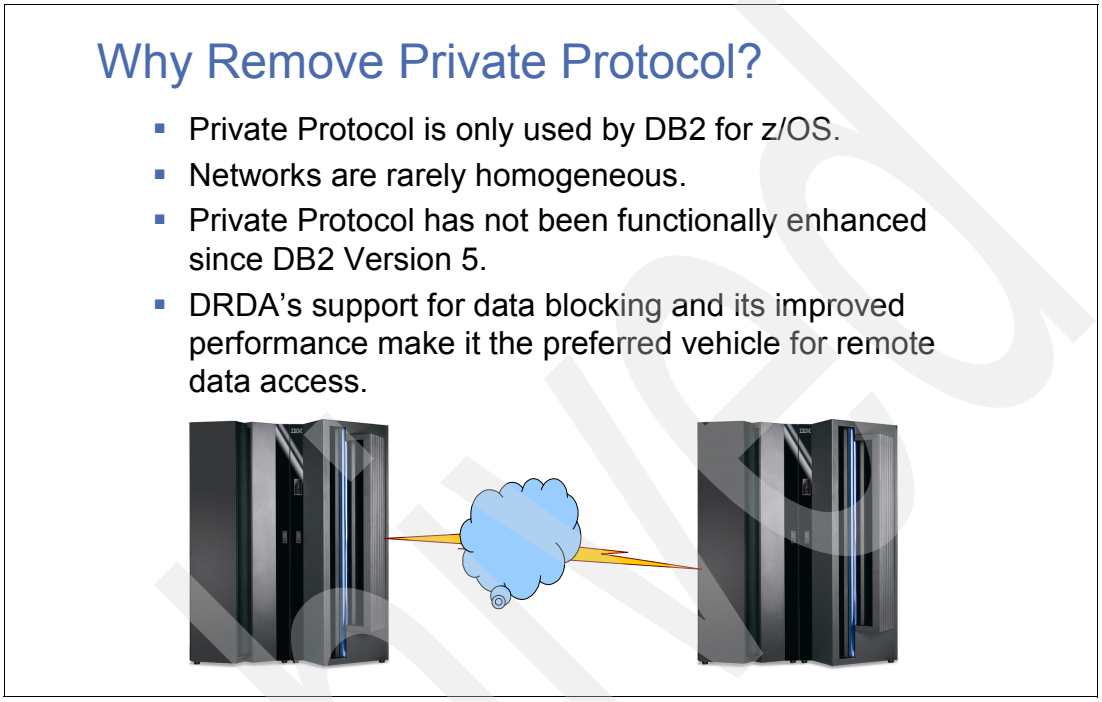


Figure 9-9 Why Remove Private Protocol?

Since you should be prepared for the eventual removal of private protocol support, V9 provides a three-pronged approach to help you start the move towards total DRDA remote processing. Figure 9-10 shows the tools V9 provides to enable this.

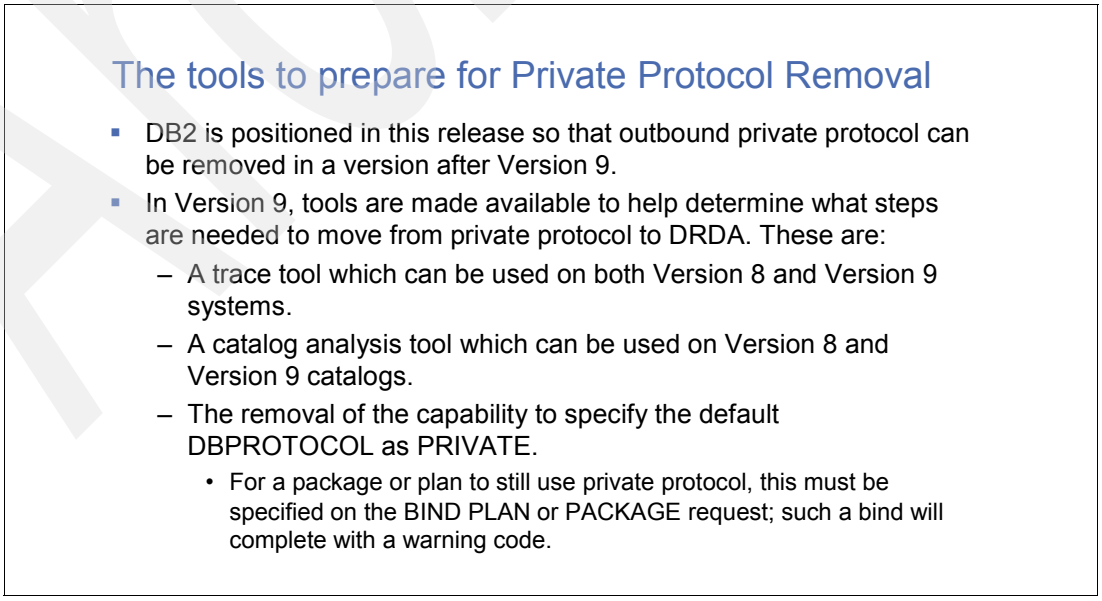


Figure 9-10 The Tools to Prepare for Private Protocol Removal

V9 provides:

- A trace tool to be used on V8 and V9 systems

  Specific traces and records are suggested for your use. By running these traces during execution windows of opportunity, you get a record of the packages and DBRMs that are executing private protocol statements. The trace records produced would have information such as package/ DBRM name, section number, remote location name, statement type, and SQL statement before and after alias resolution. With this information, customers can then create their own lists of packages that should be bound in preparation for DB2's elimination of Private Protocol. This is not a new trace, but rather is a performance trace of specific IFCIDs. This trace can be run on any DB2 for z/OS system.

- A catalog analysis tool to be used on V8 and V9 catalogs

  The catalog analysis tool is provided in the form of a REXX program. This new program searches the system's DB2 catalog tables to determine all private protocol dependencies known to DB2 in existing bound applications. From this search a series of bind jobs that specify DBPROTOCOL(DRDA) are automatically created.

- The removal of the capability to alter the default DBPROTOCOL value from DRDA to PRIVATE

  Since V6 when the DRDA protocol was enhanced to support three-part name object references, either through fully named objects or aliases, there has been the capability to set the default of the DBPROTOCOL BIND option to PRIVATE protocol when binding packages or plans of applications utilizing three-part name references. Starting with V9, it is no longer possible to change the default to private protocol. However, it is still possible to specify DBPROTOCOL(PRIVATE) when performing a BIND, and the request completes with a warning message.

9.2.1 DB2 performance trace

Since DRDA requires packages at remote sites, all programs currently using private protocol must have packages bound to the remote sites referenced in those applications. One way to determine the remote sites that are referenced in currently running programs is to run the performance trace for IFCIDs 157 and 168 to obtain this information:

```
START TRACE(PERFM) CLASS(30) IFCID(157,168) DEST(GTF)
```

Archive

Figure 9-11 shows the information gathered by these trace records.

The Trace Tool

- A Performance Trace is documented which provides information on the programs or packages using private protocol.
- This is started with the command:

```
-START TRACE(PERFM) CLASS(30) IFCID(157,168) DEST(GTF)
```

- IFCID(168) is now designated for general use
  - OMEGAMON XE for DB2 Performance Expert on z/OS is enhanced to format these trace records for Version 8 as well as for Version 9.
- The trace records obtained from this command show information such as:
  - The call type
  - The section number
  - The name of program
  - The responding location
  - The SQL statement before any alias resolution
  - The SQL statement after alias resolution
- With this information, you can see which statements are using private protocol, and which packages need to be bound to specific remote sites.

Figure 9-11 The Trace Tool

Figure 9-12 shows an edited DB2 Performance Expert Record Trace report of the IFCID 168 trace. The report shows an SQL statement that references aliases before they have been resolved. The package or plan that the statement is running under was bound with DBPROTOCOL(PRIVATE). The aliases that are referenced in the statement are EC2BTABLES and EC2BCOLUMNS.

```
PRIMAUTH INSTANCE     IFC DATA
PLANNAME              ID
-------- ------------ --- ------------------------------------------------
SYSADM   BDF2D3FF0645 168 NETWORKID:  USIBMSY   LUNAME:  SYEC2DB2  LUWS
DSNESPCS                  REQUESTING LOCATION:  'BLANK'
                          REQUESTING TIMESTAMP: N/P
                          AR NAME: 'BLANK'              PRDID: N/P
                          QW0168LL:   224
                          QW0168ST:   SELECT T.CREATOR AS TBCREATOR, T.
                                    TBNAME, C.COLNO AS COLNO, C.NAME AS
                                    C.COLTYPE AS COLTYPE, C.LENGTH AS C
                                    C.SCALE AS COLSCALE, C.NULLS AS COL
                                    FROM EC2BTABLES T, EC2BCOLUMNS C WH
                                    T.CREATOR=C.TBCREATOR AND T.NAME=C.
                                    ORDER BY 1,2,3
```

Figure 9-12 DB2 Performance Expert Record Trace Report before Alias Resolution

Archive

Figure 9-13 shows the resolved aliases in the SQL statement. The aliases EC2BTABLES and EC2BCOLUMNS have been replaced by "SYSIBM.SYSTABLES" and "SYSIBM.SYSCOLUMNS", respectively.

```
PRIMAUTH INSTANCE     IFC DATA
PLANNAME              ID
-------- ------------ --- ----------------------------------------------
SYSADM   BDF2D3FF0645 168 NETWORKID:  USIBMSY   LUNAME:  SYEC2DB2  LUWS
DSNESPCS                  REQUESTING LOCATION:  'BLANK'
                          REQUESTING TIMESTAMP: N/P
                          AR NAME: 'BLANK'              PRDID: N/P
                          QW0168LL:   244
                          QW0168ST:   SELECT T.CREATOR AS TBCREATOR, T.
                                    TBNAME, C.COLNO AS COLNO, C.NAME AS
                                    C.COLTYPE AS COLTYPE, C.LENGTH AS C
                                    C.SCALE AS COLSCALE, C.NULLS AS COL
                                    FROM "SYSIBM"."SYSTABLES" T, "SYSIB
                                    "SYSCOLUMNS" C WHERE T.CREATOR=C.TB
                                    AND T.NAME=C.TBNAME ORDER BY 1,2,3
```

Figure 9-13 DB2 Performance Expert Record Trace Report after Alias Resolution

9.2.2 The private to DRDA protocol REXX tool (DSNTP2DP)

To help you convert your plans and packages from using private protocol to DRDA protocol, DB2 provides the private to DRDA protocol REXX tool, DSNTP2DP, shown in Figure 9-14.

The Catalog Analysis Tool

- To convert plans and packages from private protocol to DRDA, a REXX tool, DSNTP2DP is provided, along with sample JCL.
- It scans the catalog and generates the necessary commands, etc., to convert all objects that have a private protocol dependency to DRDA.
- The generated output can then be tailored and run at a convenient time.
- It expects three parameters:
  1. The subsystem id of the DB2 to be scanned, specified as SSID=ssid
  2. A default collection name, specified as DEFCOLLID=collectionid|DSNCOLLID
  3. Run options, "PLANS=Y|N PACKAGES=Y|N ALIASES=Y|N"
- The first parameter is mandatory so that DSNTP2DP can connect to the DB2 subsystem.
- The second parameter represents the default collection name to be used in the generated BIND commands, where a collection name cannot be determined. If not specified, then DSNTP2DP assumes a default collection name of DSNCOLLID.
- The Run parameters trigger what processing the tool performs.

Figure 9-14 The Catalog Analysis Tool

Use job DSNTIJPD, which is customized during migration, to invoke DSNTP2DP.

A package or plan has a remote location private protocol dependency only when the tool can extract from the catalog remote location dependency information that is related to a plan or package. Just having the DRPROTOCOL column of the catalog tables that manage plans and

packages set to a value of ‘P’ (Private) does not mean that the plan or package has a remote location private protocol dependency. The syntax and options for the tool DSNTP2DP are shown in Figure 9-15.

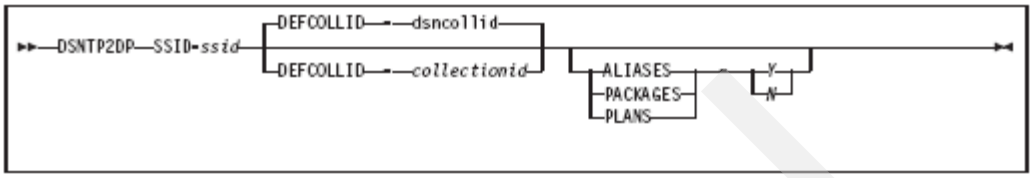


Figure 9-15 Options for tool DSNTP2DP

The tool has three parameters:

- Subsystem ID of DB2 to be examined (mandatory parameter) specified as SSID=ssid
- Default collection name (optional parameter) specified as DEFCOLLID=collectionid
- Run options (optional parameters) "PLANS=Y|N PACKAGES=Y|N ALIASES=Y|N"

The first parameter is self-explanatory in that it is used to connect the tool to the DB2 subsystem.

The second parameter is the default collection name to be used in the generated BIND commands where a collection name cannot be determined. If this parameter is not specified, then the tool assumes a default collection name of DSNCOLLID.

The run parameters trigger what processing the tool performs. If none of the run option parameters are specified, then all catalog objects are examined by the tool (that is, a value of Y is assumed for processing the PLANs, PACKAGEs, and ALIASes in the catalog). If any of the run option parameters are specified and the value assigned to a particular run option is an N, then the processing for those catalog objects controlled by that run option is not performed.

There are two catalog tables that have a database protocol (DBPROTOCOL) flag, SYSIBM.SYSPLAN and SYSIBM.SYSPACKAGE. The catalog analysis tool uses this flag to extract the potential packages, DBRMs, or plans to be converted to DRDA from private protocol. Commands are only generated for those packages, DBRMs, or plans that have a remote location dependency.

If the ALIASES run option is set to Y, the tool first examines all the aliases in the catalog to generate the needed two-part name CREATE ALIAS statements that would have to be executed on the target remote locations because the referenced two-part name object in the PACKAGE that triggered the alias reference does not match the two-part name object at the target remote location as specified in the local alias.

If the PACKAGES run option is Y, then all the packages in the catalog are examined next. For each local package that has a DBPROTOCOL set to 'P' and a dependency on a remote location, a command is generated to REBIND the local PACKAGE with the DBPROTOCOL(DRDA) option. This PACKAGE is then used as the source for the next generated BIND PACKAGE COPY commands against each server location where the PACKAGE has a remote dependency. When binding any of the packages to remote location servers, SQLERROR CONTINUE is also specified to ensure that the package is bound to the remote locations since not all of the statements within the package may actually reference objects in the remote servers.

Next, if the PLANS run option is set to Y, then the tool examines all the plans in the catalog. Regardless of the setting of the PACKAGES run option, no further packages in the catalog are examined as part of this phase of the tool's processing. Thus, this phase of the tool only

generates actions to convert all the DBRMs that are bound directly into plans that have a remote dependency.

For each DBRM that is bound directly into a plan that has a remote dependency, a BIND PACKAGE command is generated that binds the DBRM as a PACKAGE locally within a specified collection (or default collection DSNCOLLID) using BIND PACKAGE parameters that can be extrapolated from the current PLAN parameters and any parameter values from the corresponding SYSDBRM row, but ensure that DBPROTOCOL(DRDA) is specified. The source for the DBRM is obtained from the PDSNAME column of the corresponding SYSDBRM row. The next set of generated commands is BIND PACKAGE COPY with specified collection (or default collection DSNCOLLID) specified against each server location to be accessed by the DBRM package, and the source is the package just created. In addition, as already mentioned, the binding of these packages to remote location servers also has the SQLERROR CONTINUE option specified.

As a final step of this phase, a BIND PLAN is generated to replace the existing PLAN with a new PKLIST if none was previously present, or an updated PKLIST if one was previously present and DBPROTOCOL(DRDA) is specified. PLAN BIND PKLIST parm must now include the local and remote collections.

As mentioned, the tool generates the commands that should be performed to make a PLAN or PACKAGE utilize DRDA protocols when accessing remote locations. These commands with the appropriate JCL are stored in a file that can then be tailored for the environment.

9.2.3 DBPROTOCOL Bind option

Figure 9-16 lists the bind considerations that you should be aware of.

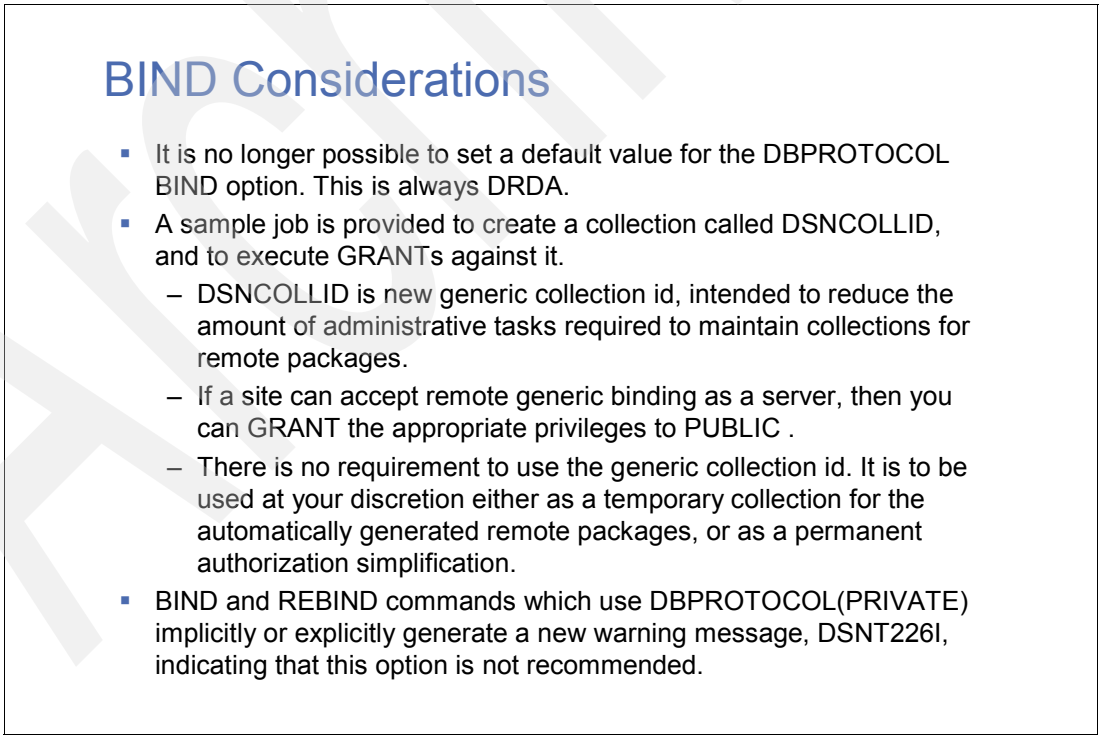


Figure 9-16 BIND considerations

To reduce the amount of administrative tasks required to maintain collections for remote packages, there is a new generic collection ID, DSNCOLLID. If a system administrator

determines that a site can accept remote generic binding as a server, then the administrator would GRANT the appropriate privileges to PUBLIC on the DSNCOLLID collection.

There is no requirement to use the generic collection ID. It is to be used at the discretion of the system administrator as either a temporary collection for these automatically generated remote packages or as a permanent authorization simplification. If a system administrator does not want a particular site to allow a generic collection ID, then this collection ID should not be created at that site.

Job DSNTIJSG creates the generic collection ID, DSNCOLLID. By default, the step that creates DSNCOLLID is commented out in the JCL. To have DB2 create DSNCOLLID, uncomment the step.

9.3 Run DDF without SNA when using TCP/IP only

DB2 used to require that VTAM is installed and the SNA/APPC support be configured and started even when the only communication that is used to and from DDF is TCP/IP. Starting with V9, DDF can be set up to no longer activate its SNA/APPC support, and thus no longer requires VTAM be installed (VTAM may still need to be installed for other reasons).

Setting up DDF to not activate its SNA/APPC support is not supported by the DB2 installation option panels. DB2 TCP/IP communications uses the IPNAME value and a character representation of the TCP/IP resync port (RESPORT) hexadecimal value to identify units of work.

All you need to do to implement this is to update the BSDS DDF record to give the DB2 system an IPNAME value. However, the value you choose for IPNAME is important. For a single DB2 standalone (no data sharing) subsystem, that name must be unique within your enterprise, much like LUNAME, so that a good choice for IPNAME is the value given to LUNAME. For a data sharing group, ALL members that are to start with TCP/IP only must share the same IPNAME. Yes, you can have some members do both SNA and TPC/IP and some do just TCP/IP. However, the group's IPNAME must again be unique within the enterprise, and for certain purposes, the value you choose for IPNAME should match the group's GENERIC LU name. This is due to trying to connect to any member of the group, whether both SNA and TCP/IP or just TCP/IP, and you want to use RACF passtickets from the requesting DB2.

In the case of both the SNA and TCP/IP members, the GENERIC LU name is used as the RACF applname for the authentication of the passticket, and on the TCP/IP only members, the IPNAME is used as the RACF applname for passticket authentication. Thus, if the IPNAME and GENERIC LU name of the group are the same, then the requester only has to worry about using a LINKNAME that matches the similar IPNAME/GENERIC LU name of the group.

9.4 IBM DB2 Driver for JDBC and SQLJ

The e-business world is very much a Java world, and DB2 V8 makes great strides in this area with the introduction of a new JDBC (and SQLJ) driver for the DB2 Family (the so-called IBM DB2 Driver for JDBC and SQLJ, also known as Java Common Client (JCC)). On the same note, SQLJ is very important to DB2 for z/OS because it brings the static SQL model, which has brought us so much over the years, in terms of performance and security, to the Java world. With the tooling and runtime support in place, SQLJ becomes a very attractive alternative to JDBC on the mainframe platform.

9.4.1 Overview

The DB2 database system provides driver support for client applications and applets that are written in Java using JDBC, and for embedded SQL for Java (SQLJ). JDBC is an application programming interface (API) that Java applications use to access relational databases. DB2 support for JDBC lets you write Java applications that access local DB2 data or remote relational data on a server that supports DRDA. SQLJ provides support for embedded static SQL in Java applications. In general, Java applications use JDBC for dynamic SQL and SQLJ for static SQL. However, because SQLJ can inter-operate with JDBC, an application program can use JDBC and SQLJ within the same unit of work.

The DB2 product includes support for two types of JDBC driver architecture.

According to the JDBC specification, there are four types of JDBC driver architectures:

- Type 1

  Drivers that implement the JDBC API as a mapping to another data access API, such as Open Database Connectivity (ODBC). Drivers of this type are generally dependent on a native library, which limits their portability. The DB2 database system does not support a type 1 driver.

- Type 2

  Drivers that are written partly in the Java programming language and partly in native code. The drivers use a native client library specific to the data source to which they connect. Because of the native code, their portability is limited.

- Type 3

  Drivers that use a pure Java client and communicate with a server using a database-independent protocol. The server then communicates the client's requests to the data source. The DB2 database system does not support a type 3 driver.

- Type 4

  Drivers that are pure Java and implement the network protocol for a specific data source. The client connects directly to the data source. DB2 for z/OS supports a driver that combines type 2 and type 4 JDBC implementations. The driver that is supported in DB2 9 for z/OS is IBM DB2 Driver for JDBC and SQLJ (type 2 and type 4).

Archived

Figure 9-17 describes the different options available for you to connect your Java application to a DB2 for z/OS, or DB2 for Linux, UNIX, and Windows server, using the IBM DB2 Driver for JDBC and SQLJ.

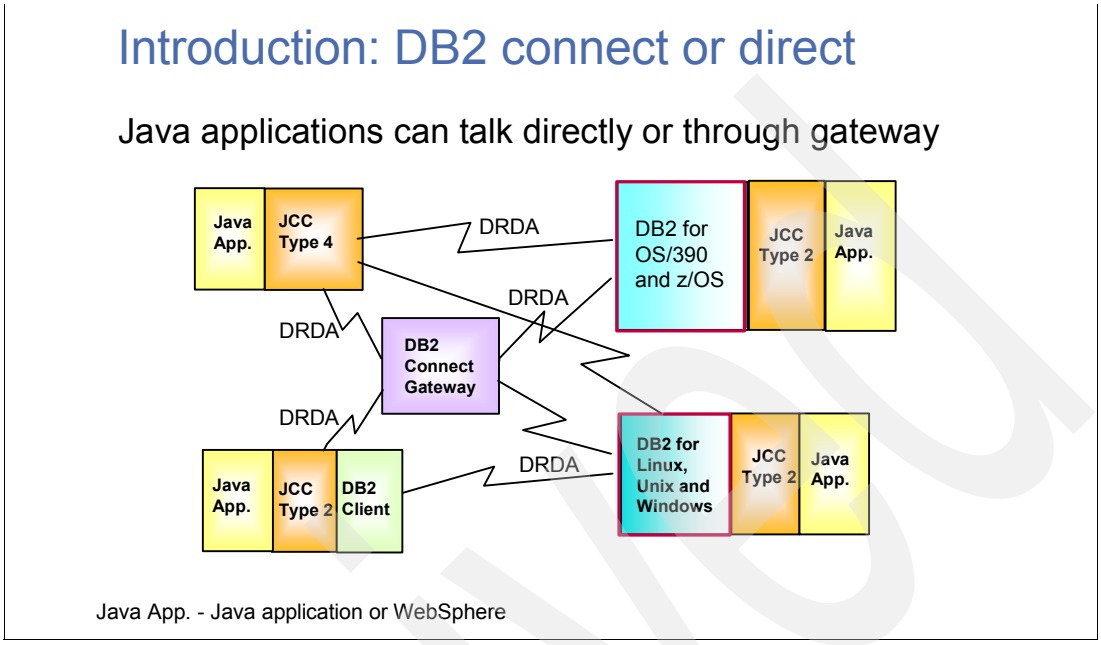


Figure 9-17 DB2 Connect 0r Direct

The IBM DB2 Driver for JDBC and SQLJ is a single driver that includes JDBC type 2 and JDBC type 4 behavior, as well as SQLJ support. When an application loads the IBM DB2 Driver for JDBC and SQLJ, a single driver instance is loaded for type 2 and type 4 implementations. The application can make type 2 and type 4 connections using this single driver instance. The type 2 and type 4 connections can be made concurrently. IBM DB2 Driver for JDBC and SQLJ type 2 driver behavior is referred to as IBM DB2 Driver for JDBC and SQLJ type 2 connectivity. IBM DB2 Driver for JDBC and SQLJ type 4 driver behavior is referred to as IBM DB2 Driver for JDBC and SQLJ type 4 connectivity.

Recommendation: In general, you should use IBM DB2 Driver for JDBC and SQLJ type 2 connectivity for Java programs that run on the same z/OS system or zSeries logical partition (LPAR) as the target DB2 subsystem. Use IBM DB2 Driver for JDBC and SQLJ type 4 connectivity for Java programs that run on a different z/OS system or LPAR from the target DB2 subsystem.

For z/OS systems or LPARs that do not have DB2 for z/OS, the DB2 JDBC Type 4 and SQLJ Driver for z/OS optional feature can be installed to provide IBM DB2 Driver for JDBC and SQLJ type 4 connectivity to a DB2 Database for Linux, UNIX, and Windows server.

To use the IBM DB2 Driver for JDBC and SQLJ, you need Java 2 Technology Edition, SDK 1.4.2 or later.

Important: The JDBC/SQLJ Driver for OS/390® and z/OS is no longer supported. To migrate from the JDBC/SQLJ Driver for OS/390 and z/OS to the IBM DB2 Driver for JDBC and SQLJ, follow the steps documented in Chapter 10 in *DB2 Version 9.1 for z/OS Application Programming Guide and Reference FOR JAVA*, SC18-9842.

9.4.2 Security enhancement

When you use the IBM DB2 Driver for JDBC and SQLJ, you choose a security mechanism by specifying a value for the securityMechanism property. You can set this property in one of the following ways:

- If you use the DriverManager interface, set securityMechanism in a java.util.Properties object before you invoke the form of the getConnection method that includes the java.util.Properties parameter.
- If you use the DataSource interface, and you are creating and deploying your own DataSource objects, invoke the DataSource.setSecurityMechanism method after you create a DataSource object.

Figure 9-18 lists the security enhancements.

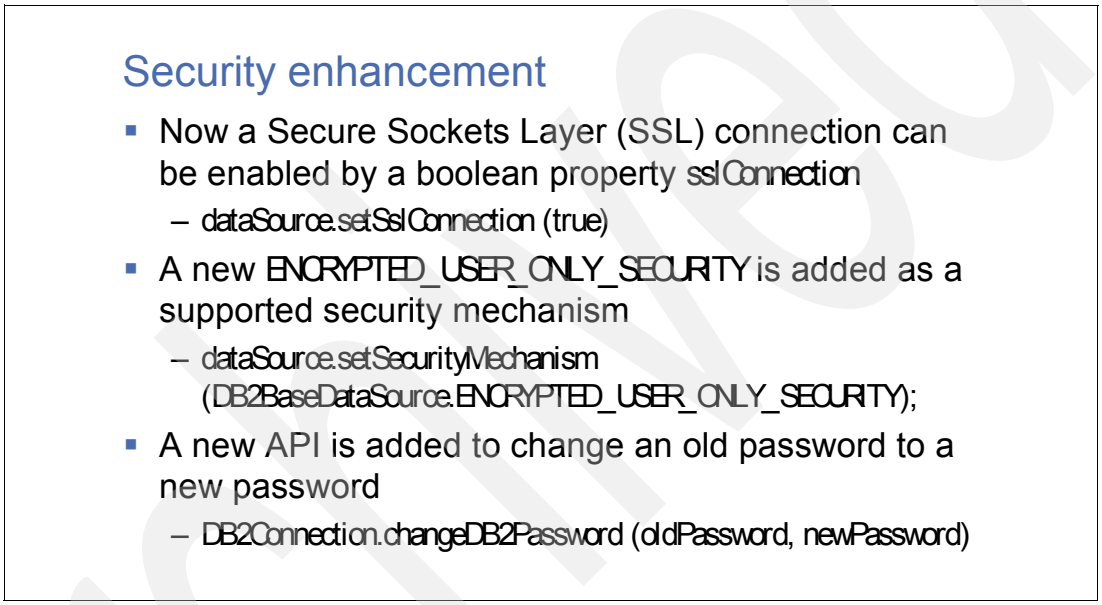


Figure 9-18 Security enhancement

IBM DB2 Driver for JDBC and SQLJ properties define how the connection to a particular data source should be made. Most properties can be set for a DataSource object or for a Connection object.

Property sslConnection specifies whether the IBM DB2 Driver for JDBC and SQLJ uses an SSL socket to connect to the DB2 server. If sslConnection is set to true, the connection uses an SSL socket. If sslConnection is set to false, the connection uses a plain socket.

Property securityMechanism specifies the DRDA security mechanism. The data type of this property is int. If this property is specified, the specified security mechanism is the only mechanism that is used. ENCRYPTED_USER_ONLY_SECURITY (16) introduced in DB2 9 means encrypted user ID.

The DB2Connection method changeDB2Password changes the password for accessing the DB2 database server for the user of the Connection object. The parameter oldPassword specifies the original password for the Connection, and the parameter newPassword specifies the new password for the Connection.

Archived

9.4.3 Package utility

DB2Binder provides a command-line utility for the JDBC packages used by the IBM DB2 Driver for JDBC and SQLJ either for adding new packages or for replacing existing packages. The JDBC packages consist of one static package and multiple dynamic packages. The static package is used for special operation, such as stored procedure CALL and large object manipulation. The dynamic packages are used for normal operation, such as query and update.

Three dynamic packages are created per holdability (WITHOUT HOLD and WITH HOLD) and per isolation (UR, CS, RS, and RR) by default, which can be adjusted by option *size.*

Until now there is no way to drop a package bound on the server by using the DB2Binder. A new function has been added that allows you to drop a package. Also, a run time API on DB2Binder allows you to add, replace, and drop the JDBC packages.

9.4.4 New data types

Java supports the new data types BIGINT, BINARY, VARBINARY, DECFLOAT, and XML.

BIGINIT is mapped to long or Long in Java.

BINARY and VARBINARY are mapped to byte[] in Java.

Figure 9-19 and Figure 9-20 on page 356 show the mapping and considerations for DECFLOAT in Java.

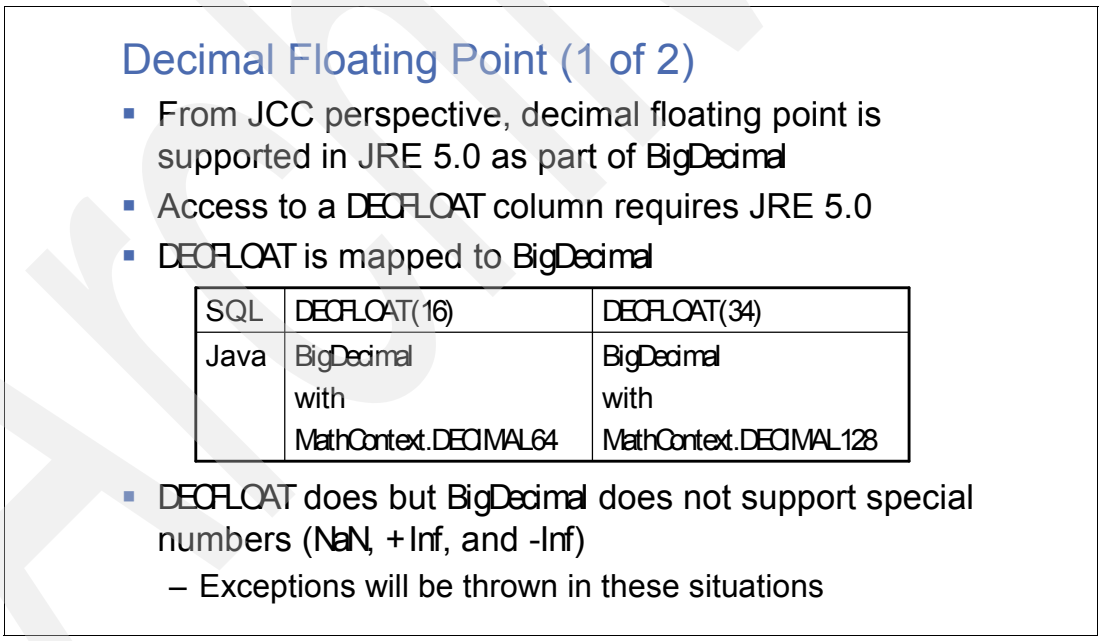


Figure 9-19 Decimal Floating Point (1 of 2)

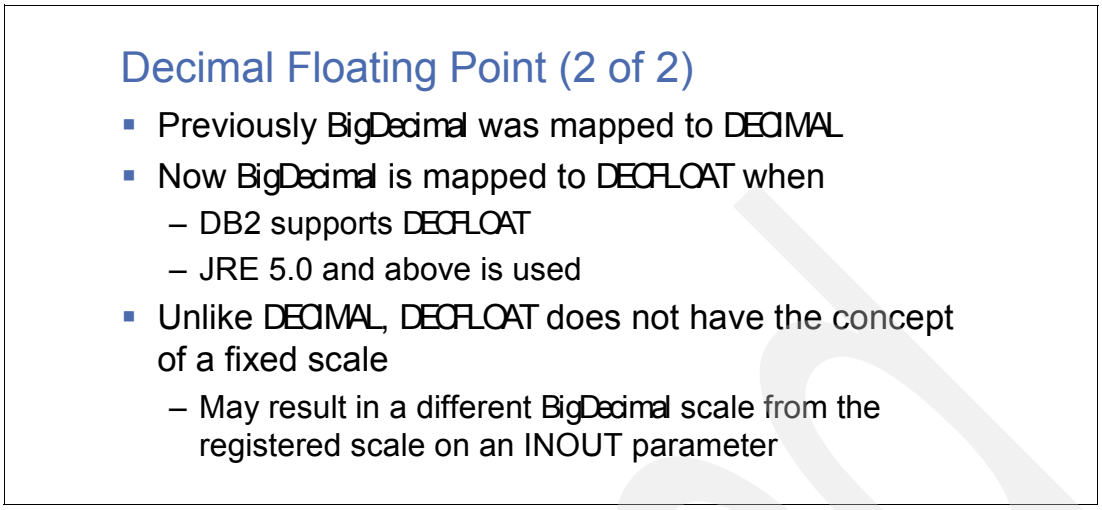


Figure 9-20 Decimal Floating Point (2 of 2)

Use of DECFLOAT requires the SDK for Java Version 5 (1.5) or later.

9.4.5 Support for LOB and XML in Java

For LOB support refer to 13.12, “LOBs performance improvements” on page 528, and for XML support refer to Chapter 8, “XML” on page 263.

9.4.6 Multi-row SQL operations

IBM DB2 Driver for JDBC and SQLJ supports multi-row insert and is enabled if and only if batch API is used. There is no need to make any changes in the application. The SQL string does not need to and should not contain the “For *n* ROWS” clause. Support for multi-row insert means reduced elapsed time and CPU time. In the current implementation the driver splits the batch at a predefined value and the value cannot be specified by the user.

There is now enhanced support for multi-row fetch. This is enabled by default when the scrollable cursor is used. Prior to V9, multi-row fetch was supported only for sensitive dynamic cursors, and now it is supported for ALL scrollable cursors. Multi-row fetch may be disabled by setting a connection property useRowsetCursor = false.

Applications that are using JDBC 1 styled-positioned update/delete:

```
UPDATE .. SET .. WHERE CURRENT OF CURSOR
DELETE .. SET .. WHERE CURRENT OF CURSOR
```

need to be aware that this syntax on a multi-row fetch cursor not only updates/deletes the current row, but also updates/deletes the entire rowset. In order to just update/delete the current row, they need to use

```
UPDATE ... SET... WHERE CURRENT OF CURSOR FOR ROW n OF ROWSET
DELETE ... SET... WHERE CURRENT OF CURSOR FOR ROW n OF ROWSET
```

However, in many cases it may be difficult for the application to figure out the correct *n* value to use. For example, if an implicit fetchSize is used it may lead to undesirable effects if the wrong n value is used. Therefore, we discourage the use of JDBC 1 styled-positioned update/delete, and recommend using the JDBC 2 styled-positioned update/delete by using

the updateRow()/deleteRow() APIs on the ResultSet, where the driver internally builds the positioned update/delete statement on behalf of the application.

There is also support for extended diagnostics, which is useful when multi-row SQL operations are performed. This is automatically enabled whenever the function requires it.

9.4.7 deferPrepares and sendDataAsIs

IBM DB2 Driver for JDBC and SQLJ properties define how the connection to a particular data source should be made. Most properties can be set for a DataSource object or for a Connection object. Two existing properties, deferPrepares and sendDataAsIs, now have enhanced semantics. Figure 9-21 shows the interaction of these two properties.

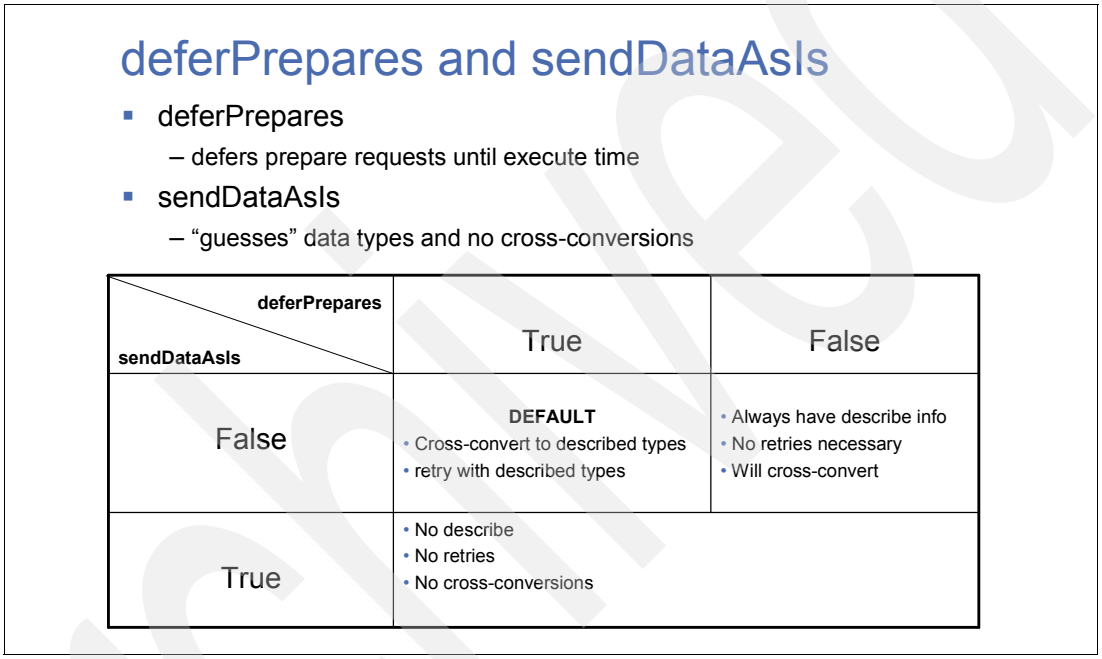


Figure 9-21 deferPrepares and sendDataAsIs

Property deferPrepares specifies whether to defer prepare operations until statement execution. The data type of this property is boolean. The default is true for IBM DB2 Driver for JDBC and SQLJ type 4 connectivity. This property is not applicable to IBM DB2 Driver for JDBC and SQLJ type 2 connectivity. Deferring prepare operations can reduce network delays. However, if you defer prepare operations, you need to ensure that input data types match DB2 table column types.

Property sendDataAsIs specifies that the IBM DB2 Driver for JDBC and SQLJ does not convert input parameter values to the target column data types. The data type of this property is boolean. The default is false. You should use this property only for applications that always ensure that the data types in the application match the data types in the corresponding DB2 tables. This property is not applicable to IBM DB2 Driver for JDBC and SQLJ type 2 connectivity.

Archive

9.4.8 SQLJ

SQLJ provides a lot of benefits, and these are listed in Figure 9-22.

SQLJ is the static model for Java applications and provides the performance benefits that are similar to static SQL. Like JDBC, which is the dynamic model, SQLJ supports the new data types BINARY, VARBINARY, DECFLOAT, LOB, and XML. Multi-row operations with SQLJ offer significant performance benefits.

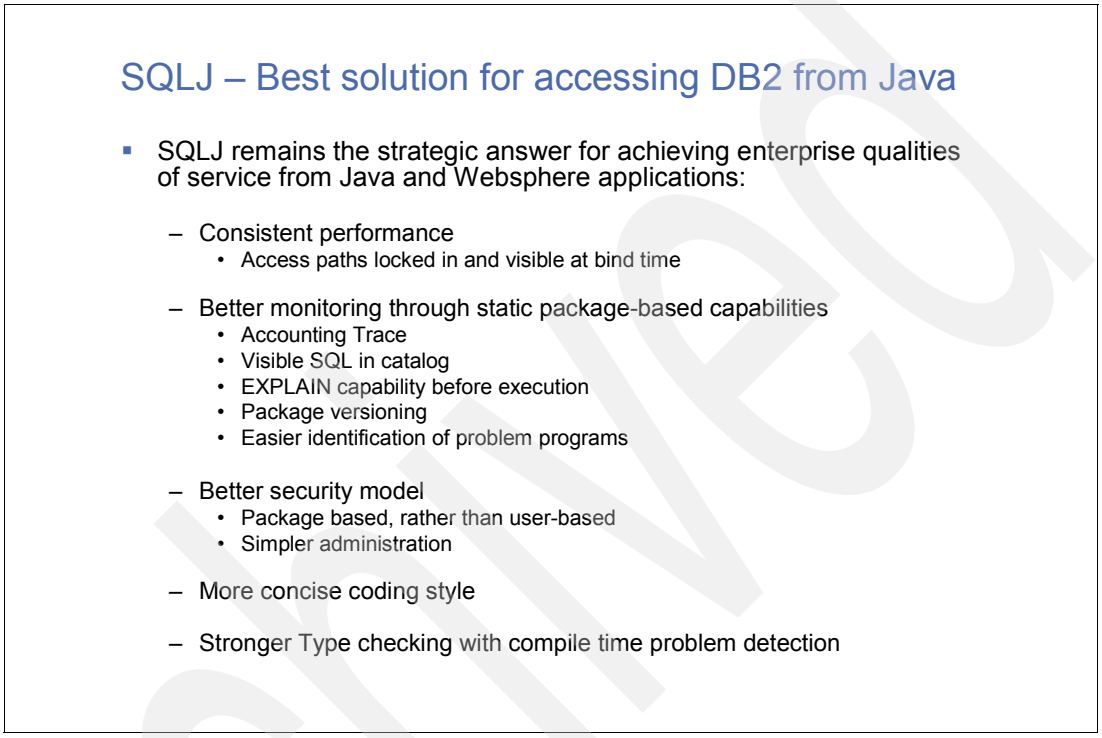


Figure 9-22 SQLJ

9.4.9 Client reroute support

The DB2 automatic client reroute feature allows client applications to recover from a loss of communication with the server so that they can continue to work with minimal interruption. JDBC and SQLJ client applications can take advantage of that support.

IBM DB2 Driver for JDBC and SQLJ client reroute support is available for IBM DB2 Driver for JDBC and SQLJ type 4 connectivity to DB2 Database for Linux, UNIX, and Windows, or DB2 for z/OS servers. It works only for connections that are obtained using the javax.sql.DataSource, javax.sql.ConnectionPoolDataSource, or javax.sql.XADataSource interface. The DriverManager interface is not supported.

Configuration of DB2 for z/OS servers for client reroute

If Sysplex routing is enabled for a DB2 data sharing group, work is distributed among all members of a data sharing group by the DB2 for z/OS server and Workload Manager for z/OS (WLM). If one of the members of a DB2 data sharing group fails, work is automatically transferred to other members. In this case, IBM DB2 Driver for JDBC and SQLJ client reroute support is not needed.

If Sysplex routing is disabled, and a data sharing group is set up for member-specific access, you can use IBM DB2 Driver for JDBC and SQLJ client reroute support. Setting up

member-specific access includes setting up a location alias that represents one or more members of the data sharing group. Before a JDBC or SQLJ client application can use the client reroute facility, a location alias needs to exist that represents at least two members of the data sharing group. The client application connects to those members using the location alias, instead of using the location name to connect to the entire group. If one member fails, connections are attempted to other members of the data sharing group, based on their priority.

When a communication failure occurs, IBM DB2 Driver for JDBC and SQLJ first attempts recovery to the original server. Reconnection to the original server is called failback. If failback fails, the driver attempts failover to the alternate location. If failover is successful during the initial connection, the driver generates an SQLWarning. If a successful failover occurs after the initial connection, the driver throws an SQLException to the application with SQLCODE -4498 to indicate to the application that the connection to the alternate server was automatically reestablished and the transaction was implicitly rolled back. The application can then retry its transaction without doing an explicit rollback first.

For details on this topic refer to *DB2 Version 9.1 for z/OS Application Programming Guide and Reference for JAVA,* SC18-9842.

Archived

Archived

Part 3

Operations and performance

Technical innovations in operational compliance (both regulatory and internal governance) help teams work more efficiently, within guidelines, and with enhanced auditing capabilities.

DB2 Utilities Suite for z/OS Version 9.1 delivers full support for the significant enhancements in DB2 9 for z/OS such as universal table spaces, not-logged table spaces, clone tables, new data types (DECFLOAT, BIGINT, and VARBINARY), new rounding modes, and pureXML© storage. In addition, the DB2 Utilities Suite includes enablement for IBM System z Integrated Information Processors (zIIP).

The key DB2 9 performance improvements are reduced CPU time in many utilities, deep synergy with System z hardware and z/OS software, improved performance and scalability for inserts and LOBs, improved SQL optimization, zIIP processing for remote native SQL procedures, index compression, and reduced CPU time for data with varying lengths and better sequential access. This version also continues to improve virtual storage use below the 2 GB bar.

Installation and migration are similar to V8 by utilizing the compatibility mode and new function mode statuses. New additional functions are made available. We point out enhancements and differences.

Performance is a major area with reduced regression and several opportunities for reduction. Virtual storage relief sees continued progress by more usage moved above the bar.

© Copyright IBM Corp. 2007. All rights reserved.

This part contains the following chapters:

- Chapter 10, “Security” on page 363
- Chapter 11, “Utilities” on page 385
- Chapter 12, “Installation and migration” on page 417
- Chapter 13, “Performance” on page 493

Archived

10

Security

There are a variety of ways that a user can access data held within DB2 both locally and remotely. Locally, users access DB2 via TSO, CICS, or IMS. Remotely, users connect to DB2 from another DB2 or via DB2 Connect. Increasingly, DRDA access involving ODBC or JDBC is being used with the user going through an intermediate software layer, for example, WebSphere.

If you are developing an application accessing DB2 using an Application Server, with DB2 V8 the Application Server uses a single or limited number of DB2 auth IDs for all access control and object resolution. However, for regulatory compliance reasons (for example, Sarbanes-Oxley), and other reasons such as accountability and auditability, many organizations are having to focus on more advanced security functions when accessing their IT systems.

DB2 9 for z/OS provides a set of options to improve and further secure access to data held within DB2.

The security topics highlighted in this chapter are about how to:

- Change the Application Server to use Secure Socket Layer (SSL) client authentication instead of DRDA encrypt user ID/password encryption.
- Change the Application Server to flow Windows user IDs (not RACF IDs) to DB2. DB2 needs to be configured to use the new z/OS identity mapping plug-in to map distributed IDs to RACF ID using the z/OS Enterprise Identity Mapping (EIM) service.
- Show how the distributed user IDs are included in the DB2 and RACF audit records.
- Utilize SCHEMA and PATH special registers for object resolution instead of using CURRENT SQLID to qualify objects.
- Define a trust relationship between an application instance and DB2. The Trusted Context enforces which users by which role can access DB2 adhering to company policies while using a client station. If all access is granted only to the role, then there is no way any user can access any data outside a trusted context. Any attempt to bypass the client station by any user can then be blocked as well as traced for later audit purposes.
- Show how DBAs can manage objects owned by ROLES.

© Copyright IBM Corp. 2007. All rights reserved.

- Show how a trusted context can identify trusted connections and from where privileges can be exercised and allow users of the context to acquire additional privileges.
- Describe how dynamic statement cache works with ROLES.

We also show scenarios where a DBA holding a DBADM privilege can perform actions on objects (views, and so on) owned by others (or on behalf of others) by using the switch-user capability of a trusted context.

An area that could be confusing in V9 is the subject of object ownership. It was difficult to begin with because of the various permutations and combinations that come into play (objects created using static versus dynamic SQL, a table object versus a user-defined function, qualified versus unqualified objects, and so on). DB2 9 has added another layer of flexibility with CURRENT SCHEMA support, and then added role ownership in a trusted context on top of that.

Archived

10.1 Security functions in general

Regulatory compliance, security, and auditing are important for companies of any size. More progressive companies are taking a proactive role and have begun implementing compliance-oriented policies and procedures while leveraging automation opportunities to streamline their operations. The requirements on IT have been growing.

IBM mainframe is the only computing system to earn the highest level of Common Criteria security classification, EAL5. This is achieved through industry-leading virtualization technology, a hardware environment that provides end-to-end data protection, a centralized security server that integrates security management and enforcement across all resources in the infrastructure, and integrated crypto facilities. System z is the platform that businesses trust to secure their most sensitive data.

DB2 integrates with these System z hardware and operating system features to provide unmatched database security capabilities. System z9 now provides end-to-end encryption (IPSec) across heterogeneous platforms and devices, enabled for zIIP specialty engine support. DB2 encryption options have also been expanded with DB2 9 for z/OS database.

z/OS V1.7 with the RACF optional feature has achieved EAL4+ for Controlled Access Protection Profile (CAPP) and Labeled Security Protection Profile (LSPP). z/OS and DB2 are evaluated for the Common Criteria EAL4 evaluation. DB2 for z/OS Version 8 is in evaluation under the Common Criteria with a conformance claim of EAL3. See:

```
http://www.ibm.com/systems/z/security/
http://www.ibm.com/systems/z/security/mls.html
http://www.ibm.com/systems/z/security/ccs_certification.html
http://www.ibm.com/security/standards/st_evaluations.shtml
```

DB2 9 for z/OS provides security enhancements to enable companies to further streamline their regulatory compliance efforts and to enact even more effective security policies

DB2 9 helps you respond to increasing needs to protect security, to assure integrity, and to comply with regulations such as the Sarbanes-Oxley Act, the Health Insurance Portability and Accountability Act (HIPAA), and Payment Card Industry (PCI) Security Standards. Improved access control with network trusted context and roles allows more precise control of security. Improved filtering makes auditing more usable. Secure Sockets Layer (SSL) data encryption on networks is more secure. In this section we briefly introduce the main security topics. For details, refer to the *DB2 Version 9.1 for z/OS Administration Guide,* SC18-9840, and *DB2 Version 9.1 for z/OS RACF Access Control Module Guide,* SC18-9852, as *Securing DB2 and Implementing MLS on z/OS,* SG24-6480-01.

10.2 Enterprise identity mapping

Today's network and multiplatform environments are made up of a complex group of systems and applications, resulting in the need to manage multiple user registries. These user registries are intended to be used by applications to achieve user identification and authentication. The authenticated user ID is eventually used for access control as performed by the application itself, or the middleware it runs on, or by the local operating system. Since systems-specific local user IDs and registries are not going to unify for many years in a common format across all vendors, the solution seems to be for a user to authenticate to an installation using a network identity and then the involved applications will map this network identity to the local user ID this user.

Enterprise Identity Mapping (EIM) was introduced as part of the IBM Autonomic Computing Initiative that allows administrators and application developers to address this problem more easily and inexpensively than previously possible. By providing centralized and protected identification facilities, EIM also contributes to enabling products and applications to run in a more secure way in today's extremely open environments, as expected for the On Demand operating infrastructure.

In the EIM terminology, the unique name given at an enterprise level for a user or an entity is the EIM identifier. EIM actually defines associations between an EIM identifier and user IDs in registries that are part of OS platforms, applications, and middlewares. Applications, typically servers, can then use an EIM API to find a mapping that transforms the installation-level user ID initially used for authentication to a local user ID, which can in turn be used to access local resources.

EIM administration tools

The EIM Domain Controller administration can be performed by programs using the administration API, or can be done from the z/OS UNIX console with the eimadmin line command.

The EIM client

EIM applications on z/OS must be APF authorized. Requiring APF authorization prevents inadvertent or malicious use of EIM APIs to change information in an EIM domain or to extract unauthorized information, which means that you must set the APF authorization extended attribute for each EIM application program residing in HFS files. This attribute is set by using the extattr command.

Using RACF profiles for EIM defaults

Several profiles can be used to store in RACF an LDAP URL, the bind distinguished name, and the bind password, that can then be retrieved by the EIM services. The choice of the profile is dictated by the intent to either establish a system default that is used for EIM only or that can be used by other functions (such as z/OS PDAS or PKI Services), or to establish a default setup for a specific application user ID.

An example of definition of this profile is:

```
RDEFINE LDAPBIND BUCKSDOMAIN +
EIM(DOMAINDN('ibm-eimDomain=Bucks Domain.o=ibm.c=us')) +
OPTIONS(ENABLE)) +
PROXY(LDAPHOST(ldap://another.big.host) +
BINDDN('cn=EIM Application Lookups') BINDPW('secret'))
```

To establish this profile as a default for a specific user SERVERID, the EIM segment in the USER profile can be updated as follows:

```
ALTUSER SERVERID EIM(LDAPPROF(BUCKSDOMAIN))
```

Having established the connection, it then provides the possibility of switching to another user ID, giving the opportunity of taking on the identity of this other user ID only within the trusted context. In addition, it is possible to assign a role to a user of a trusted context.

Support for trusted context also provides the ability within a specific trusted context for a DB2 authorization ID to acquire a special set of privileges that are not available outside of that trusted context by defining roles. A role is a database entity that groups together one or more privileges and can be assigned to users. A role provides privileges, in addition to the current set of privileges that are granted to the primary and secondary authorization identifiers. A role can own objects if the objects are created in a trusted context with the role defined as the

owner. If a role is defined as an owner, then only the privileges that are granted to the role are considered for object ownership.

The role can be granted privileges and can as such represent a role within the organization. Depending on your organization, you can associate privileges to a certain job or role and then associate a user to the role, implicitly granting the sum of those privileges.

These two constructs together supply security enhancements for a variety of different scenarios, ranging from any three-tier layered application like SAP or WebSphere to the daily duties of a DBA maintaining the DB2 subsystem.

The major benefits are:

- A role can be used as a single database authid that can be used to simplify administration of dynamic SQL privileges.
- The user's authid can be used to run database transactions so that the DB2 audit is able to identify the end users individually (important capability for meeting some regulatory compliance requirements).
- The Trusted Context retains many of the performance benefits of connection pooling.
- The Trusted Context and role support can be used to implement DBA privileges that can easily be disconnected and reconnected to individual employees. This provides function similar to shared SYSADM or DBADM user IDs, but it avoids the audit compliance problems associated with shared user IDs.

DB2 9 allows the Application Server to propagate end-user IDs to DB2, utilizes Enterprise Identity Mapping for improved DB2 and RACF auditing, utilizes ROLEs for access control and has the objects owned by ROLES, and finally utilizes special registers to manage object resolution instead of SQLIDs.

10.3 More security options with INSTEAD OF triggers

INSTEAD OF triggers, which are defined on views, provide another way of ensuring security within the database. INSTEAD OF triggers are used to process insert, update, and delete operations (through trigger logic) instead of the INSERT, UPDATE, or DELETE statement that activates the trigger. In addition to other capabilities, you can use INSTEAD OF triggers to encode and decode data from the database within a view. The view might encapsulate decryption functions, while the INSTEAD OF triggers use the encryption functions to ensure security within the database.

For more information about INSTEAD OF triggers see 6.2, "INSTEAD OF triggers" on page 136.

10.4 Improved tracing in DB2

Audit-related tracing offers enhancements in properly scoping the trace and support for the additional functions.

New filtering options on the START TRACE command give the ability to select your trace records more carefully, thus minimizing the amount of data collected and monitoring more closely access from specific users or locations. The new possibilities include:

- Roles (used via a trusted context) can be filtered via the new keyword ROLE.
- Other trace filter capabilities for package name, collection name, and so on.

- New keywords are introduced to provide exclude trace filtering capabilities.
- The ability to use wild cards, rather than the full names.

These traces are described in 3.10.3, “Trace filtering” on page 70.

Headers were also extended, so you get the ROLE in all the records.

New IFCIDs and instrumentation have been introduced to control trusted connections.

The following foils show how IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS V4.1.0 can format the new IFC records.

Figure 10-1 show the IFCID 269 record, written when a trusted connection is established.

Audit Trace – Trusted Security Context and Database Roles

- New IFCID 269 – written when a trusted connection was established or when it is reused at a DB2 server.

```
LOCATION: STM4D61Y                        OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V4)          PAGE: 1-1
      GROUP: N/P                                    AUDIT TRACE                      REQUESTED FROM: NOT SPECIFIED
     MEMBER: N/P                                                                             TO: NOT SPECIFIED
  SUBSYSTEM: Y61Y                                                                  ACTUAL FROM: 01/29/98 23:48:01.86
DB2 VERSION: V9                                    SCOPE: MEMBER                     PAGE DATE: 01/29/98
PRIMAUTH CORRNAME CONNTYPE
 ORIGAUTH CORRNMBR INSTANCE
 PLANNAME CONNECT               TIMESTAMP   TYPE                                  DETAIL
 -------- -------- ------------ ----------- -------- ------------------------------------------------------------------------------
 SOF      ZSQLASA1 TSO          23:48:01.86 AUTHCHG  TYPE:              ESTABLISH TRUSTED CONTEXT       STATUS:            SUCCESS
 ORSOF    'BLANK'  B1BAA2A382C5                       OBJECT OWNER:      AUTHID                          SQLCODE:           XXXXXXXX
 POCDRIV6 BATCH                                       SECURITY LABEL:    XXXXXXXX
                                                      CONTEXT NAME:      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                                                         XXXXXXXXXXXXXXXXXX
                                                      CONTEXT ROLE:      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                                                         XXXXXXXXXXXXXXXXXX
                                                      USER ROLE:         XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                                                         XXXXXXXXXXXXXXXXXX
                                                      SYSTEM AUTHID:     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                                                         XXXXXXXXXXXXXXXXXX
                                                      REUSE AUTHID:      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                                                         XXXXXXXXXXXXXXXXXX
                                                      SERVAUTH NAME:     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                                                         XXXXXXXXXXXXXXXXXX
                                                      JOB NAME:          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                                                         XXXXXXXXXXXXXXXXXXXX
                                                      REUSE AUTHID:      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                                                         XXXXXX
                                                      ENCRYPTION:        XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                                                         XXXXXXXXXXXXXXXXXXXX
                                                      TCP/IP USED:       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                                                         XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

  AUDIT TRACE COMPLETE
```

Figure 10-1 IFCID 269

Figure 10-2 shows IFCID 270, written at creation or update of a trusted context.

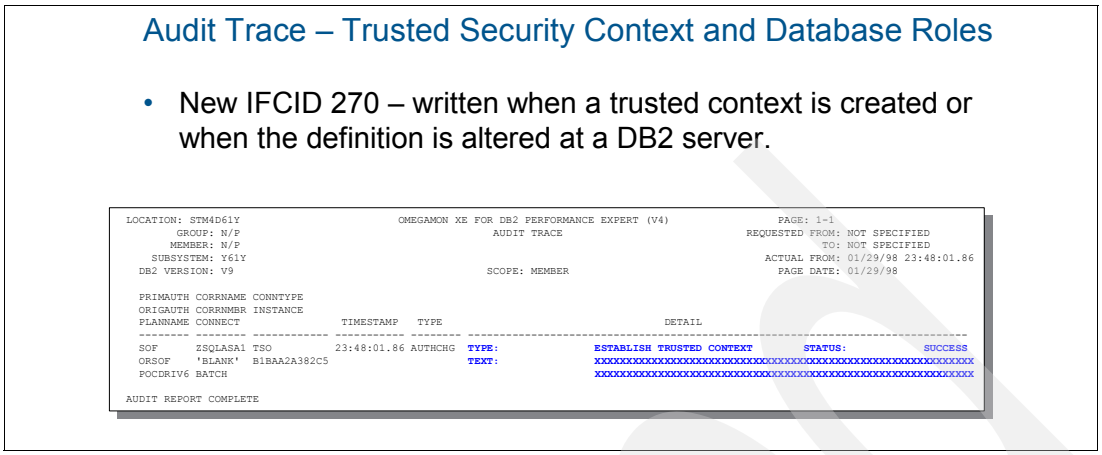


Figure 10-2 IFCID 270

Figure 10-3 shows IFCID 141 and 142, which have been extended to include authid and table owner type.

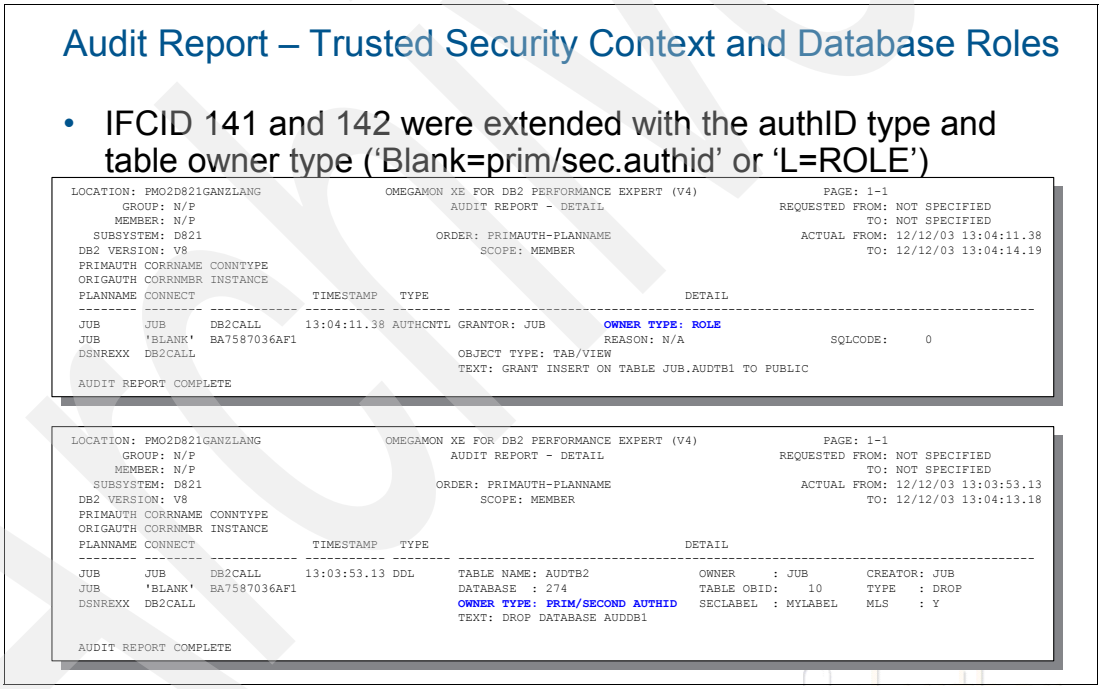


Figure 10-3 IFCID 141 and 142

Figure 10-4 shows the IFCIDs introduced or updated to support IPv6.

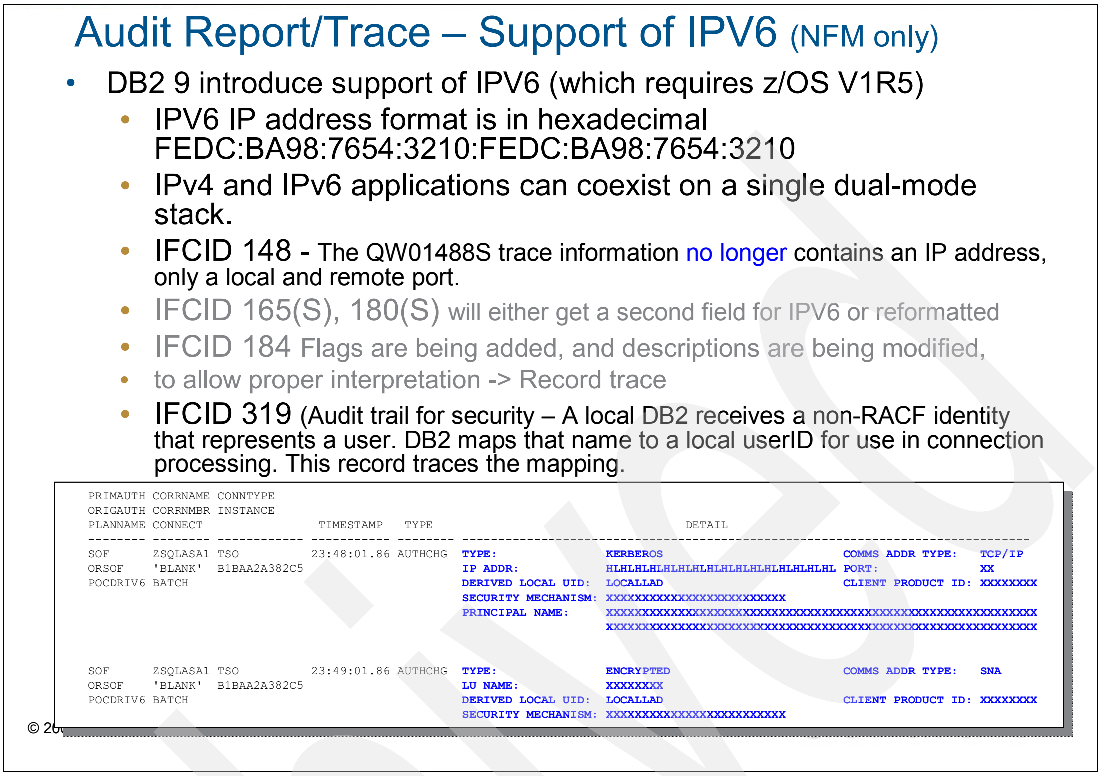


Figure 10-4 IPv6 IFCIDs

10.5 Audit Management Expert - a new tool for DB2

Many auditors have little familiarity with databases and even less with determining which objects to audit, starting appropriate traces, and collecting and filtering and producing reports. So, the task usually requires a lot of the DBA's time to interpret what an auditor is requesting and to gather that information.

Responding to these auditors' requests can distract DBAs from more strategic responsibilities.

To address this situation, IBM recently released the newest member of the IBM DB2 regulatory compliance suite, IBM DB2 Audit Management Expert. This tool can be used by DBAs and auditors to provide all the information needed for an audit while also maintaining database security.

With an easy to use graphical interface, IBM DB2 Audit Management Expert requires no DB2 expertise – allowing auditors to gather and correlate a coherent view of DB2 activity without involving a DBA. Auditors are not required to log into DB2 nor are they able to directly manipulate any DB2 resource, thus securing a further layer of protection for your DB2 data. Auditors are able to collect log and trace data in an audit repository, and then view, analyze and generate comprehensive reports on the data. They can selectively filter SELECT, INSERT, UPDATE, and DELETE activity by user or by object and export these filters for use on another DB2 system.

Archived

The DB2 Audit Management Expert for z/OS V1.1 focuses on aiding compliance with a plethora of regulations. See:

http://www.ibm.com/software/data/db2imstools/

10.6 The Consul zSecure and InSight Suites

Consul's security management offerings can enhance the administration of mainframe security management with innovative audit, alert, and monitoring capabilities for z/OS Resource Access Control Facility (RACF). Consul also extends z/OS security through security policy enforcement technology. The Consul InSight and zSecure suites provide a solution to help meet your mainframe and distributed security administration, audit, and compliance challenges. They offer a full, integrated solution to enable mainframe administration managers, security officers, and auditors to:

- Comply with regulations.
- Prepare for and pass security audits.
- Improve operational efficiency.
- Enable networked business opportunities.
- Lower chances of down time.

Many customers worldwide rely on Consul to enable easier mainframe management. The Consul zSecure Suite adds a user-friendly layer onto the mainframe that allows for easier administration and analysis of the mainframe.

```
http://www.ibm.com/systems/z/security/solutions/tivoli.html#consul
http://www.consul.com/Content.asp?id=70&PageLoc=Home
```

10.7 Encryption hardware advances

The performance advantages of hardware-assisted cryptography are readily available to applications, such as banking and finance, via the cryptography interfaces of z/OS.

Machines prior to the z990 have a Cryptographic Coprocessor Feature to improve the performance of encryption and decryption. However, only CPU 0 and 1 could perform encryption. To encrypt and decrypt data, tasks running on other processors need to be redispatched to run on CPU 0 or 1. Performance is therefore a problem if there is contention among tasks (for example, parallel query). In addition, dedicated LPARs might not be able to use the encryption hardware feature.

The z990 introduced a new hardware instruction, CP Assist for Cryptographic Function (CPACF), which can run on all CPUs and is a feature available only on the z990 hardware and later, not the older z900. The z990 also introduces a PCIXCC card, which is needed for the IBM Data Encryption Tool, but not for the DB2 encryption function.

Each CP on the z990 has an assist processor on the chip in support of cryptography. This feature provides for hardware encryption and decryption support. Peripheral Component Interconnect Extended Cryptographic Coprocessor (PCIXCC) provides a cryptographic environment with added function. The PCIXCC consolidates the functions previously offered on the z900 by the Cryptographic Coprocessor feature (CCF) and the PCI Cryptographic Coprocessor (PCICC) feature. For a more detailed discussion of CPACF and PCIXCC, refer to *IBM eServer zSeries 990 (z990) Cryptography Implementation*, SG24-70700.

The cost of encrypting DB2 data is reduced on the z990 hardware, compared with the older z900 hardware. CPU time for the Encryption Tool is 10 times faster on a z990 versus the

z900, and the elapsed time for multi=threaded applications using encryption is much faster on z990 versus z900 depending on the number of concurrent tasks. Similar observations are applicable to the z890 processor, which has the same level of relative performance as the z990. See *DB2 UDB for z/OS Version 8 Performance Topics*, SG24-6465.

The z9 BC supports the following cryptographic hardware features:

- CP Assist for Cryptographic Function (CPACF)
- Crypto Express2 (CEX2) feature
- Crypto Express2-1P (CEX2-1P) feature

The CP Assist for Cryptographic Function delivers cryptographic support on every Central Processor (CP) with Data Encryption Standard (DES), Triple DES (TDES), and Advanced Encryption Standard (AES)-128 bit data encryption/decryption, as well as Secure Hash Algorithm (SHA-1) and SHA-256 hashing.

The Crypto Express2 (CEX2) feature combines the functions of Coprocessor (for secure key encrypted transactions) and Accelerator (for Secure Sockets Layer [SSL] acceleration) modes in a single optional feature with two PCI-X adapters. New on System z9, using the HMC console, the PCI-X adapters can be customized as having either two Coprocessors, two Accelerators, or one of each. The Crypto Express2 is a follow-on to the PCIXCC and PCICA features. All of the analogous PCIXCC and PCICA functions are implemented in the Crypto Express2 with equivalent or greater performance.

Introduced in May 2007, the Crypto Express2-1P (CEX2-1P) feature provides the same functions as the CEX2, but contains only one PCI-X adapter. The PCI-X adapter can be configured as a Coprocessor or Accelerator.

See the white paper IBM System z9 Business Class Performance of Cryptographic Operations (Cryptographic Hardware: CPACF, CEX2C, CEX2A) available at:

`http://www.ibm.com/systems/z/security/pdf/ZSO03009-USEN-00.pdf`

10.8 Encryption in the controllers

Security is increased through encryption. In the payment card industry, IT organizations are required to encrypt DB2 data that is in tables, indexes, image copies, and even logs, which can create a performance drag. With DB2 9, you are able to use System z disk and tape controllers to encrypt the data at rest on these devices, and System z has advanced capabilities to centrally manage all of the encryption keys. By offloading the encryption work to the storage devices, you can save a lot of server processing power.

Protecting that data, controlling access to it, and verifying its authenticity while maintaining its availability are priorities in our security-conscious world.

IBM System Storage™ TS1120 Tape Drive and now IBM Ultrium 4 tape drives, the latest generation of IBM LTO technology, include data encryption capabilities within the drives, helping to avoid the need for host-based encryption of data—and the concurrent drain on host performance—or the use of specialized encryption appliances. In addition, data encryption is performed without significantly impacting the tape drives native performance, meaning that batch processing windows are likely not affected. This capability is intended to provide customers with greater ability to protect information if tape cartridges are lost or stolen by supporting the storage of the data in an encrypted form.

The IBM Encryption Key Manager component for the Java platform, which is supported on a wide variety of environments including z/OS, i5/OS®, AIX, HP, Sun, Windows, and Linux, can

help generate and manage encryption keys for TS1120 and IBM Ultrium 4 tape drives across the enterprise. This feature uses standard key repositories on supported platforms and supports three different encryption management methods: application managed, system managed, or library managed.

The TS1120 and IBM Ultrium 4 tape drives support transparent encryption, minimizing application changes in the system and library managed implementations.

IBM has announced that the encryption technology will eventually be added to IBM disk-based systems as well.

10.9 Network trusted context

Trusted context addresses the problem of establishing a trusted relationship between DB2 and an external entity, such as a middleware server. A series of trust attributes are evaluated at connect time to determine whether a specific connection is to be trusted. The relationship between a connection and a trusted context is established when a connection to the server is first created.

The users must be allowed to use a trusted context, and trusted connections (for local or remote applications) can be restricted to a specific server.

We examine a three-tiered architecture, which typically presents complex challenges for security.

10.9.1 Challenges with three-tiered architectures

The use of three-tier application architectures presents challenges for security. Figure 10-5 shows a typical three-tiered architecture.

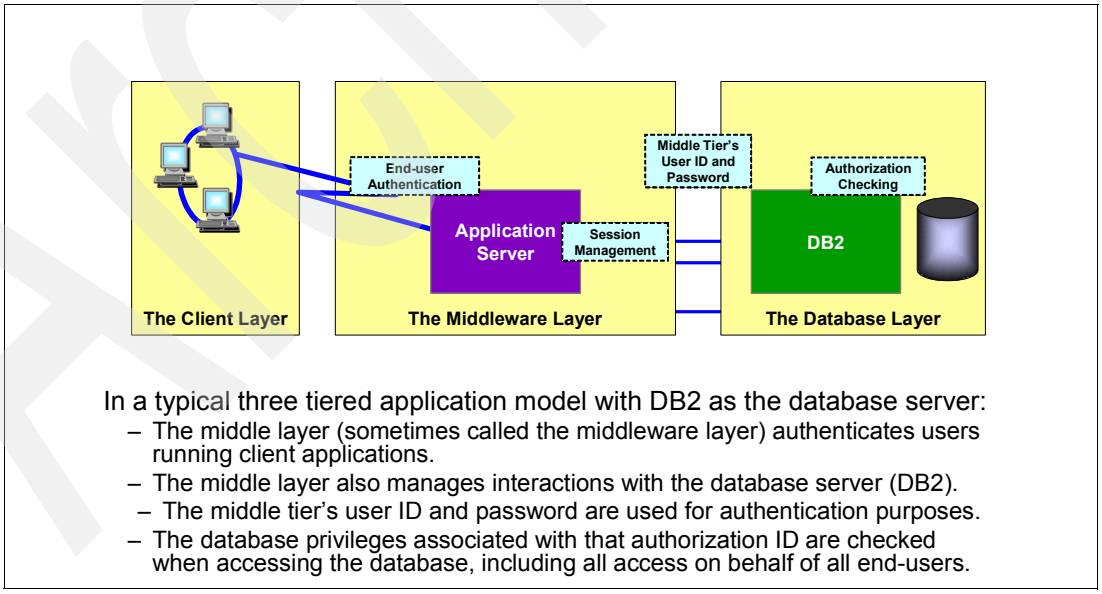


Figure 10-5 Three-tier application architecture

There are two types of issues associated with this type of architecture. The first involves the user ID of the middle layer being used to control the database access. The second is the

implications of the middle layer managing the access to DB2 (that is, controlling the usage or *pooling* of threads into DB2).

The user ID of the middle layer being used has the following problems:

- The real user's identity is lost, and control over that user accessing the database is subsequently lost.
- The middle layer may access DB2 for its own needs. There is no way of telling whether accesses are being done directly for a user.
- The user ID of middle layer must be given the aggregate authority for its own needs and all of the users connecting through it.
- Should the security of the middle layer or its user ID be compromised, then a significant exposure arises.

If the middle layer used the user ID of the user, then the problems include:

- Firstly, it may not be able to authenticate the end-user credentials.
- Secondly, there is a potential performance degradation in the repeated creation of threads into DB2.
- Other security challenges are:
  - Should DB2 trust all connection requests?
  - Lack of already verified DRDA.
  - Shared SYSADM ID or DBADM user ID.
  - DBA with eual responsibilities (production and development).
  - Full-time access to sensitive/private data by production DBAs.
  - DBADM can create view for another ID, but cannot drop/alter.
  - Without reserving a RACF group, a table can be dropped.
  - Privileges granted can be exercised from anywhere.

DB2 V9 introduces the two constructs, trusted contexts and roles, that will be used to assist in solving this issues.

10.9.2 Trusted contexts

A powerful security enhancement in DB2 V9 for z/OS is the introduction of the network trusted context.

Support for trusted context addresses the problem of establishing a trusted relationship between DB2 and an external entity such as a database administrator or a middleware server. With trusted context support, a series of trusted attributes is evaluated to determine whether a specific context can be trusted. After a trusted context is established, you can define a unique set of interactions between DB2 and the external entity, such as a middleware server, so that the existing database connection can be used by a different user without requiring authentication of the new connection user.

Briefly, it can establish a connection as trusted when connecting to DB2 for z/OS from a specific location.

Having established the connection, it then provides the possibility of switching to another user ID, giving the opportunity of taking on the identity of this other user ID only within the trusted context. In addition, it is possible to assign a role to a user of a trusted context.

Support for trusted context also provides the ability within a specific trusted context for a DB2 authorization ID to acquire a special set of privileges that are not available outside that trusted context by defining roles. A role is a database entity that groups together one or more

privileges and can be assigned to users. A role provides privileges, in addition to the current set of privileges that are granted to the primary and secondary authorization identifiers. A role can own objects if the objects are created in a trusted context with the role defined as the owner. If a role is defined as an owner, then only the privileges that are granted to the role are considered for object ownership.

The role can be granted privileges and can as such represent a role within the organization. Depending on your organization, you can associate privileges to a certain job or role and then associate a user to the role, implicitly granting the sum of those privileges.

These two constructs together supply security enhancements for a variety of different scenarios, ranging from any three-tier layered application like SAP or WebSphere to the daily duties of a DBA maintaining the DB2 subsystem.

The major benefits are:

- A role can be used as a single database authid that can be used to simplify administration of dynamic SQL privileges.
- The user's authid can be used to run database transactions so that the DB2 audit is able to identify the users individually (important capability for meeting some regulatory compliance requirements).
- The trusted context retains many of the performance benefits of connection pooling.
- The trusted context and role support can be used to implement DBA privileges that can easily be disconnected and reconnected to individual employees. This provides function similar to shared SYSADM or DBADM user IDs, but it avoids the audit compliance problems associated with shared user IDs.
- The role object ownership in a trusted context eliminates the object dependency on authorization IDs.
- You now have the ability to control where, when, and how local and remote requesters communicate with DB2.

DB2 9 allows the application Server to propagate user IDs to DB2, utilizes Enterprise Identity Mapping for improved DB2 and RACF auditing, utilizes ROLEs for access control and has the objects owned by ROLES, and finally utilizes special registers to manage object resolution instead of SQLIDs.

A trusted context is used to establish a trusted relationship for a DB2 connection and lasts for the duration of the connection. Once established, a trusted context allows:

- The ability for an already established database connection to be used under a different user ID without the need to authenticate that user with DB2.
- The ability for a DB2 authorization ID to acquire one or more privileges within a trusted context that are not otherwise available to it. This requires the use of the roles.

A series of attributes will be verified when the connection is established to determine whether the context can be trusted.

- The context is established for a specified authorization name.
  - For local connections this could be the TSO logon ID that specified by the USER= parm in JCL or as supplied by RRSAF.
  - For remote connections it could be the system user ID supplied by the middle layer software.

- For remote connections the actual TCP/IP address that is used by the connection to communicate with DB2. The only communication protocol supported is TCP/IP. For a given trusted context, you can specify more than one address.
- For remote connections the minimum level of encryption of the data stream (network encryption).
- For remote connections the name of a resource in the RACF SERVAUTH class. This resource is the network access security zone name that contains the IP address of the connection that is used to communicate with DB2. The SERVAUTH attribute can be specified multiple times.
- For local connections, the job name or started task name.

DB2 allows an established trusted connection to be used under a different user. To allow this, the trusted context must be defined with use for the specific user. If PUBLIC is specified for the user, it allows the trusted connection to be used by any authorization ID.

In Figure 10-6 we show a new option, AS USER, in the DB2I Defaults Panel 1 (DSNEOP01) used to switch authids.

```
DB2I DEFAULTS PANEL 1
 COMMAND ===>

 Change defaults as desired:

  1  DB2 NAME ............. ===> DB9B      (Subsystem identifier)
  2  DB2 CONNECTION RETRIES ===> 0         (How many retries for DB2 connection)
  3  APPLICATION LANGUAGE   ===> IBMCOB    (ASM, C, CPP, IBMCOB, FORTRAN, PLI)
  4  LINES/PAGE OF LISTING  ===> 60        (A number from 5 to 999)
  5  MESSAGE LEVEL ........ ===> I         (Information, Warning, Error, Severe)
  6  SQL STRING DELIMITER   ===> DEFAULT   (DEFAULT, ' or ")
  7  DECIMAL POINT ........ ===> .         (. or ,)
  8  STOP IF RETURN CODE >= ===> 8         (Lowest terminating return code)
  9  NUMBER OF ROWS ....... ===> 20        (For ISPF Tables)
 10  AS USER               ===>            (Userid to associate with the
                                            trusted connection)


 PRESS:  ENTER to process    END to cancel           HELP for more information
```

Figure 10-6 New option for switching IDs

10.9.3 Secure Socket Layer

DB2 Version 9.1 for z/OS supports the Secure Socket Layer (SSL) protocol by implementing the z/OS Communications Server IP Application Transparent Transport Layer Security (AT-TLS) function. The z/OS V1R7 Communications Server for TCP/IP introduces the AT-TLS function in the TCP/IP stack for applications that require secure TCP/IP connections. AT-TLS performs transport layer security on behalf of the application, such as DB2, by invoking the z/OS system SSL in the TCP layer of the TCP/IP stack. The z/OS system SSL provides support for TLS V1.0, SSL V3.0, and SSL V2.0 protocols.

Encryption of data over the wire is made easier in z/OS 1.7. The Communications Server now offers Application Transparent Transport Layer Support (AT-TLS), which facilitates the use of SSL encryption of data on behalf of the application.

Archived

SSL encryption has been available on z/OS for a long time, but with this new facility more applications will be able to offer this level of encryption. DB2 9 for z/OS makes use of this new facility and now offers SSL encryption using a new secure port.

When acting as a requester, a DB2 for z/OS can request a connection using the secure port of another DB2 subsystem. When acting as a server, and from within a trusted context, SSL encryption can be required for the connection.

10.9.4 Roles

DB2 extends the trusted context concept to optionally assign a default role to a trusted context and optionally assign a role to a user of the context. A role is a database entity that groups together one or more privileges. A role is available only in a trusted context.

Roles provide a more flexible technique than groups or users in assigning and controlling authorization, while improving consistency with the industry and improving security.

A database role is a virtual authorization ID that is assigned to an authid through an established trusted connection.

A role is a database entity to which one or more DB2 privileges can be granted to or revoked from. Roles provide a means to acquire context-specific privileges.

Role privileges are in addition to other sets of privileges granted to the primary and secondary authids, except for object ownership. Roles can create and own objects.

The role can be assigned and removed from individuals through the trusted context as needed. This allows a DBA to perform object maintenance during a change control window and then lose the role privileges when the window is shut.

Roles are a way to allow multiple DBA authids have ownership of an object at the same time or at different times, rather than only one DBA authid having the ownership all the time.

To illustrate some of the new capabilities we examine two scenarios.

10.10 Illustrative scenario 1

This scenario involves an application — let us say it is called XYZ. We assume that there is some middleware software that will connect to DB2. XYZ uses this middleware. The middleware is capable of managing the threads and will pass the actual user Id of the user. In addition to the regular users, there is an administrator who may need to perform inserts, updates, selects and deletes. The administrator is only allowed to run the SQL when using this interface.

Archived

First, a trusted context needs to be established, to allow the middleware to switch user IDs. The safeguard shown in Figure 10-7 will be used to make sure that the middleware is the genuine version: the static IP address is specified. The ID of the middleware is XYZAS. This newly established context is called XYZAPP.

```
CREATE TRUSTED CONTEXT XYZAPP BASED UPON CONNECTION USING SYSTEM AUTHID XYZAS
NO DEFAULT ROLE
ENABLE
ATTRIBUTES (ADDRESS '9.120.45.87')
WITH USE FOR XYZADM ROLE XYADMIN WITH AUTHENTICATION,
WITH USE FOR XYZ001 WITHOUT AUTHENTICATION,
WITH USE FOR XYZ002 WITHOUT AUTHENTICATION,
.
.
.
WITH USE FOR XYZ500 WITHOUT AUTHENTICATION
```

Figure 10-7 SQL to establish a trusted context

The creation of the role for the administrator is shown in Figure 10-8.

```
CREATE ROLE XYADMIN
```

Figure 10-8 SQL to create a role

Using this example to illustrate some points, we can note that:

- The trusted context is just for one specific system user ID, in this case XYZAS, the ID associated with the middleware.
- There is the ability to assign a default role for the context. In our example, no default is assigned, as we want the general end user to have just their own authority used.
- The context can exist in an enabled or disabled state (use the ALTER TRUSTED CONTEXT SQL to change).
- One or more attributes are needed to establish the context (in this case a specific TCP/IP address).
- Every user that can be switched to needs to be defined in the context. Our example shows 500 general uses and the one administration (ALTER TRUSTED CONTEXT SQL can be used to maintain the list).
- For our administrator we are asking that DB2 authenticates the user. The administrator then receives the authorities granted to the role XYADMIN, in addition to any others he may have.
- The select, insert, update, and delete authority will need to be granted to the role XYADMIN for the tables concerned.
- Our general users, like XYZ001, do not go through any authentication. It is accepted that this has been performed by the middleware. XYZ001, along with the other general users, will not be assigned a role, so the authority to the application must be granted via packages or underlying data. Note that the regular security exits are driven, so secondary authorizations are available.

Archived

The information related to the trusted contexts and roles is stored in the following four Catalog tables:

- SYSIBM.SYSCONTEXT
- SYSIBM.SYSROLES
- SYSIBM.SYSCONTEXTAUTHIDS
- SYSIBM.SYSCTXTTRUSTATTRS
- SYSIBM.SYSOBJROLEDEP

10.11 Illustrative scenario 2

Suppose that we have a production DBA that routinely needs to run utilities like COPY, RUNSTATS, and REORG, but in rare circumstances needs to run other utilities and even gain full access to the data. Trusted contexts and roles could be set up to allow the additional authorities to be obtained in certain circumstance (for example, only when run with a certain jobname). See Figure 10-9.

```
CREATE TRUSTED CONTEXT XYZDBA1 BASED UPON CONNECTION USING SYSTEM AUTHID TSG001
DEFAULT ROLE PRODDBA WITHOUT ROLE AS OWNER
ENABLE
ATTRIBUTES (JOBNAME 'PRODXYZ');
CREATE ROLE PRODDBA;
GRANT DBADM ON DATABASE XYZDB TO ROLE PRODDBA;
```

Figure 10-9 SQL for trusted context and role

The IMAGCOPY, STATS, or REORG authority would need to be granted to the user ID TSG001, or ideally a group he is connected to.

The role PRODDBA can only be used by TSG001 (our DBA) from a job called 'PRODXYZ'. The installation could capture the output of such jobs for historical/auditing reasons.

Unless the DBA submits utilities, command, or SQL in a job called 'PRODXYZ', he will be restricted to IMAGCOPY, STATS, or REORG authorities. Note that we have stated that the role cannot be the owner, so even in a job called 'PRODXYZ' he will not be able to add a new table space, for example.

10.12 Other examples of trusted context and role usage

We have listed here shortened examples of role usage. For details, see *Securing DB2 and Implementing MLS on z/OS*, SG24-6480-01.

Archived

10.12.1 View maintenance on behalf of another user

A user ID (USRT001) holding the DBADM privilege can create a view for someone else. However, this same user ID is not able to drop or alter the view, or grant privileges on the view. See Figure 10-10.

```
CREATE TRUSTED CONTEXT CTXT_VIEW_MAINT
  BASED UPON CONNECTION USING SYSTEM AUTHID USRT001
  ATTRIBUTES (JOBNAME 'DROPVW',JOBNAME 'GRANTVW',JOBNAME 'ALTERVW')
  ENABLE
  WITH USE FOR USRT005 ;

USRT001 submits the following BATCH job:
//DROPVW JOB 'USER=$$USER','<USERNAME:JOBNAME>',CLASS=A
//      ,MSGCLASS=A,MSGLEVEL=(1,1),USER=USRT001,REGION=0M,
//         PASSWORD=LOSGATOS
/*ROUTE PRINT STLVM14.YUKI
//*************************************************
//* DROPVW  JOB: - Drop view USRT005.VCUST to show
//*       USRT001 can do this in a trusted context.
//* - Must use DROPVW as the job name and run job with
//*       USER=USRT001 and ASUSER(USRT005).
//*************************************************
//* DSNTEP3 STEP
//*************************************************
//CREATE1  EXEC TSOBATCH,DB2LEV=DB2A
//SYSTSIN  DD *
DSN SYSTEM(SSTR) ASUSER(USRT005)
RUN PROGRAM(DSNTEP3)
//SYSIN    DD *
 DROP VIEW  USRT005.VCUST ;
/*
//
```

Figure 10-10 Job to define a view maintenance role

In this way, a database administrator can assume the identity of other users and perform actions on their behalf.

10.12.2 Backing up a DBA, assuming the identity of another user ID

In this example, there are three application databases in a production environment. Each database is supported by an independent DBA who holds the DBADM privileges. Work needs to be done on a particular database and the DBA is not available.

The three databases are:

- DLOAN
- DCUST
- DCARD

1. A user ID with SYSADM executes the following statements:

```
GRANT DBADM ON DATABASE DLOAN TO USRT020;
GRANT DBADM ON DATABASE DMTG  TO USRT030;
GRANT DBADM ON DATABASE DCARD TO USRT040;
```

2. A new table needs to be created in the DCARD database.

   DBA USRT040 is not available.

3. USRT030 is connected to RACF group CARD.

4. Create trusted context:

```
CREATE TRUSTED CONTEXT CTXT_BACKUP_PROD_DBA
  BASED UPON CONNECTION USING SYSTEM AUTHID USRT030
  ATTRIBUTES (JOBNAME 'USRT030')
  WITH USE FOR USRT040
  ENABLE
  ;
```

5. USRT030 logs on and goes to the DB2I defaults panel and sets ASUSER=USRT040. See Figure 10-11.

```
DB2I DEFAULTS PANEL 1
COMMAND ===>

Change defaults as desired:

 1  DB2 NAME ............. ===> V91A      (Subsystem identifier)
 2  DB2 CONNECTION RETRIES ===> 0         (How many retries for DB2
                                          connection)
 3  APPLICATION LANGUAGE   ===> IBMCOB    (ASM, C, CPP, IBMCOB,
                                          FORTRAN, PLI)
 4  LINES/PAGE OF LISTING  ===> 60        (A number from 5 to 999)
 5  MESSAGE LEVEL ........ ===> I         (Information, Warning,Error,
                                          Severe)
 6  SQL STRING DELIMITER   ===> DEFAULT   (DEFAULT, ' or ")
 7  DECIMAL POINT ........ ===> .         (. or ,)
 8  STOP IF RETURN CODE >= ===> 8         (Lowest terminating return
                                          code)
 9  NUMBER OF ROWS ....... ===> 20        (For ISPF Tables)
10  AS USER               ===> USRT040    (Userid to associate with the
                                          trusted connection)
```

Figure 10-11 Setting AS USER ID

USRT030 as user USRT040 now holds the DBADM privilege on the DCARD database and can create the table.

10.12.3 Securing DBA activities

Many customers are concerned about DBA access to sensitive customer data. Another common problem is the use of a shared SYSADM user ID or a shared DBADM user ID.

We provide an example of two DBAs implementing changes to an application under the same ROLE. The DBAs only have access to the database during the implementation weekend. Afterward, they will have no access at all to the DCUST database.

The following steps illustrate this scenario.

A user ID with SYSADM authority executes the following commands:

```
CREATE ROLE PROD_CUST_DBA_ROLE ;
```

```
GRANT DBADM ON DATABASE DCUST TO ROLE PROD_CUST_DBA_ROLE ;

 CREATE TRUSTED CONTEXT CTXT_PROD_CUST_DBA_ROLE_USRT029
   BASED UPON CONNECTION USING SYSTEM AUTHID USRT029
   DEFAULT ROLE PROD_CUST_DBA_ROLE WITH ROLE AS OBJECT OWNER
   ATTRIBUTES (JOBNAME 'USRT029')
   DISABLE
   ;

 CREATE TRUSTED CONTEXT CTXT_PROD_CUST_DBA_ROLE_USRT039
   BASED UPON CONNECTION USING SYSTEM AUTHID USRT039
   DEFAULT ROLE PROD_CUST_DBA_ROLE WITH ROLE AS OBJECT OWNER
   ATTRIBUTES (JOBNAME 'USRT039')
   DISABLE
   ;
```

Both contexts are created in a disabled state and enabled at the beginning of the implementation window. Now the two DBAs can work in parallel on the same application by exercising the same ROLE.

Outside of an established trusted connection, USRT029 and USRT039 do not have access to the DCUST database or the data in its tables.

Note also that the PROD_CUST_DBA_ROLE role is the owner of all the objects created during the established trusted connections because both trusted contexts have the same default role and both context definitions have the WITH ROLE AS OBJECT OWNER clause.

We could have also turned on a trace and produced an audit report to review all the activities done by each DBA.

You can use trusted context and role support to implement DBA privileges that can easily be disconnected and reconnected to individual employees.

With these capabilities, customers are able to create DBA procedures that can be audited and protected so that one individual cannot violate the established rules without being detected during the audit review.

10.12.4 Reducing risk of a table being dropped by another person

This statement implements a role for database create and drop:

```
CREATE ROLE DEV_XYZ_DBA_ROLE;
GRANT DBADM ON DATABASE DXYZ TO ROLE DEV_XYZ_DBA_ROLE;

 CREATE TRUSTED CONTEXT CTXT_DEV_XYZ_DBA_ROLE_USRT011
   BASED UPON CONNECTION USING SYSTEM AUTHID USRT011
   DEFAULT ROLE DEV_XYZ_DBA_ROLE WITH ROLE AS OBJECT OWNER
   ATTRIBUTES (JOBNAME 'USRT011')
   ENABLE
   ;
```

Now all tables created in this database will be created through the role, and even though another user (for example, USRT044 can can create RACF group XYZ and connect USERT044 to it, and that person cannot drop tables created in database DXYZ).

10.12.5 Limiting salary updates from a single source

To allow salary updates to be done only from a specific user and a specific IP address within a trusted connection, execute:

```
CREATE ROLE USRROLE1;
CREATE TRUSTED CONTEXT CTX1
BASED UPON CONNECTION USING SYSTEM AUTHID PAUL
ATTRIBUTES (ADDRESS '2.65.102.200')
ENABLE;
ALTER TRUSTED CONTEXT CTX1
ADD USE FOR SAM ROLE USRROLE1;

GRANT UPDATE (EMPNO, EMPSALARY) ON TABLE EMPPAYROLL TO ROLE USRROLE1;
```

SAM can use the privileges associated with role USRROLE1 to update the EMPNO and EMPSALARY columns in the EMPPAYROLL table, which are not available to SAM outside the context.

Archived

Archived

11

Utilities

There have been a variety of improvements to utilities in DB2 9. First of all, utilities have all been enhanced to support all new functions in DB2 9, such as universal table spaces, XML data type, CLONE table spaces, compressed indexes, NOT LOGGED table spaces, and so on.

In this chapter we describe functional improvements specific to individual utilities. It should be noted that major performance enhancements were also delivered. They are introduced in 13.6, “Utilities CPU reduction” on page 508.

This chapter includes the description of the following utility-related topics:

- REORG enhancements
- CLONE TABLE support
- BACKUP and RESTORE SYSTEM
- RECOVER enhancements
- LOAD enhancements
- Histogram statistics
- COPY improvements
- TEMPLATE switching
- MODIFY RECOVERY
- DSN1LOGP message enhancement
- Online CHECK DATA
- Online CHECK LOB
- REPAIR LOCATE SHRLEVEL CHANGE
- DB2I and DSNU CLIST enhancement
- Online REBUILD INDEX
- Time stamps in messages
- Large format data sets
- Support of large block interface
- DSN1COPY RESET enhancement
- DSNJU004
- Enhancements to V8

© Copyright IBM Corp. 2007. All rights reserved.

11.1 REORG enhancements

There are a series of enhancements to the REORG utility, including a significant improvement to availability when reorganizing a subset of partitions within a partitioned table space. The topics covered here are:

- REORG elapsed time reduction
- Removal of the BUILD2 phase of Online REORG
- Change to REORG SHRLEVEL REFERENCE by part
- Online REORG usability and keyword changes
- LOB REORG enhancements

11.1.1 REORG elapsed time reduction

In order to reduce the elapsed time of the REORG utility for partitioned table spaces, REORG is now able to do the unload/reload of partitions in parallel. Note that this function applies to all types of SHRLEVEL. It is not confined to Online REORG (OLR). To get this functionality, the NOSYSREC keyword needs to be specified or defaulted to, or the UNLDDN keyword is used with a template that operates at the part level. For the template associated with UNLDDN, to be eligible it must include a partition number (that is, &PA or &PART). Note that NOSYSREC is always used for SHRLEVEL change.

Additionally, note that parallelism is not enabled if any of the following is true:

- DATAWKnn ddnames are specified in the JCL.
- The SORTDEVT keyword is not specified.
- UTPRINT is allocated to anything other than SYSOUT.
- The REBALANCE keyword is used.

The SHRLEVEL CHANGE (OLR) has a further parallelism enhancement. REORG now attaches one or more subtasks during the LOG phase to speed up the processing of log records. These new subtasks have phase names of 'LOGAPPLY' in the DSNU111I message. This occurs when unload and reload have been done by part and there are records to apply for the parts concerned.

11.1.2 Removal of the BUILD2 phase of Online REORG

Prior to DB2 V5, REORG would unload the data contained in the table space, redefine the table space, and reload of the data, then rebuild of the indexes.

What we call Online REORG (OLR) was introduced in V5 and greatly improves the availability of data. It reloads the table space and rebuilds the indexes into shadow data sets. It then switches the original data sets to the new ones at the end of the execution.

The parameter SHRLEVEL controls the execution of REORG with the options:

- SHRLEVEL NONE

  No OLR, same as before, default
- SHRLEVEL REFERENCE

  OLR to allow only readers
- SHRLEVEL CHANGE

  OLR to allow readers and writers, this is what is properly called OLR

When reorganizing a single partition of a table space with one or more NPIs, the phases of REORG SHRLEVEL CHANGE are as shown in Figure 11-1. Data is still available during the the reload and rebuild phases. However, reorganization by partition suffers from a relatively long outage when NPIs are defined on the table.

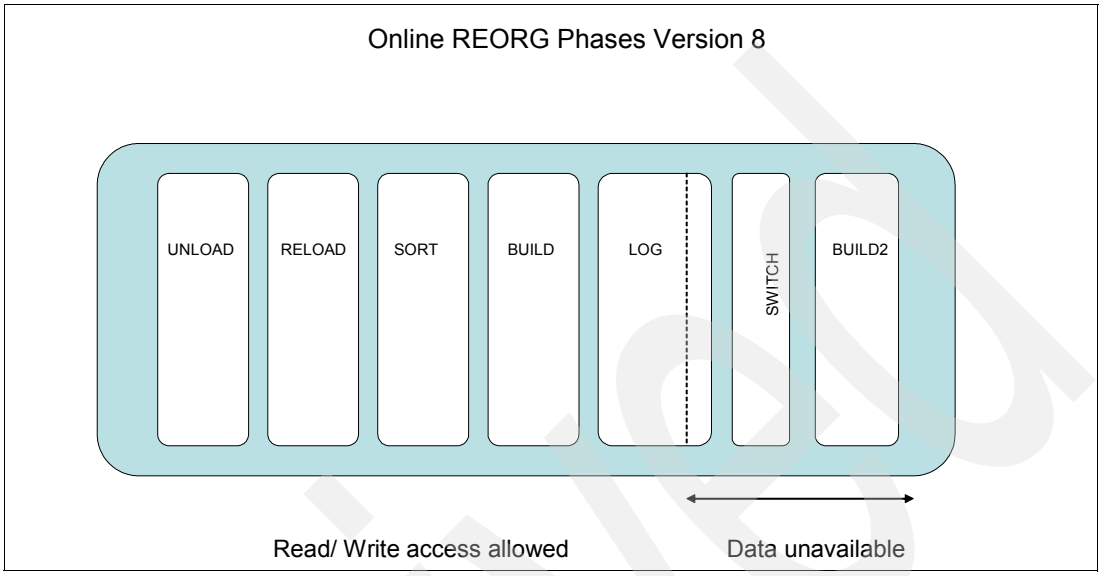


Figure 11-1 Online REORG phases in V8 for a partition and NPIs

During OLR prior to V9, the NPIs had to be updated with the new RIDS for the data of the partition being reorganized during the BUILD2 phase. The availability outage to the index could be significant.

DB2 V9 removes the need for a BUILD2 phase, by reorganizing the whole of the NPI. The reorganized NPIs are then switched in the SWITCH phase along with the other data sets. The only outage is during the relatively short SWITCH phase, as shown in Figure 11-2.

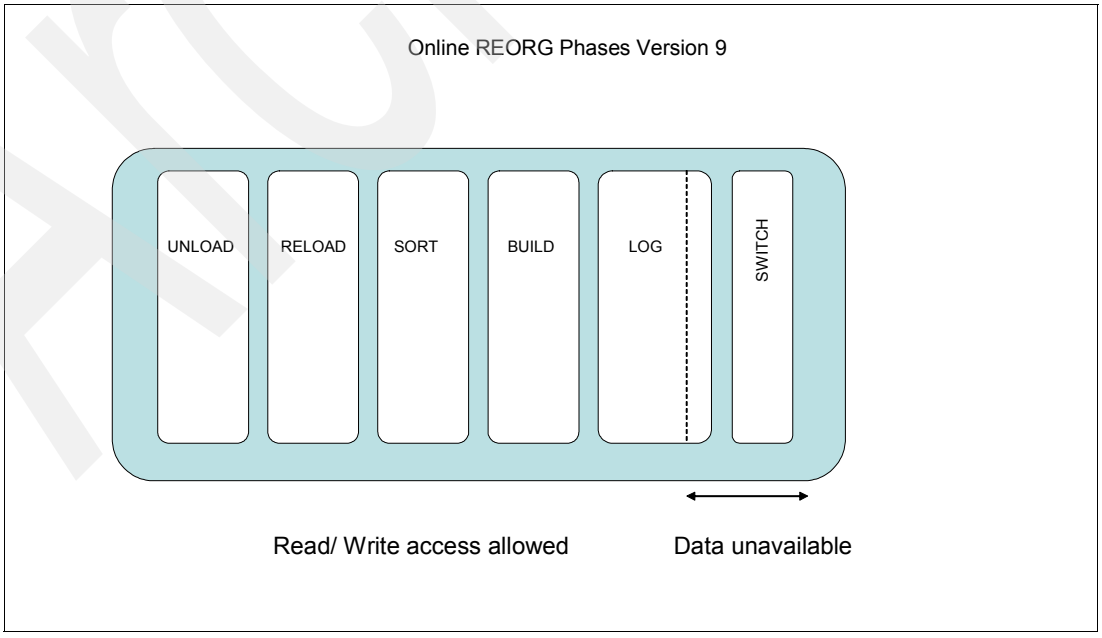


Figure 11-2 Online REORG phases in V9 for a partition and NPIs

Archive

Figure 11-3 shows the output of a REORG by partition range. The BUILD2 is no longer needed.

```
DSNU000I    242 18:13:14.14 DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = RTSA
 DSNU1044I   242 18:13:14.17 DSNUGTIS - PROCESSING SYSIN AS EBCDIC
 DSNU050I    242 18:13:14.18 DSNUGUTC -  REORG TABLESPACE ROY.ROYTSP PART 2:5 SHRLEVEL CHANGE MAPPINGTABLE ROYMAP
 SORTDEVT SYSDA SORTNUM 3 DRAIN_WAIT 120
 DSNU1160I   242 18:13:17.92 DSNURPRD - PARTITIONS WILL BE UNLOADED/RELOADED IN PARALLEL, NUMBER OF TASKS = 2
 DSNU395I    242 18:13:17.92 DSNURPRD - INDEXES WILL BE BUILT IN PARALLEL, NUMBER OF TASKS = 2
 DSNU397I    242 18:13:17.92 DSNURPRD - NUMBER OF TASKS CONSTRAINED BY CPUS
 DSNU251I  -DB9B 242 18:13:18.23 DSNURPUT - UNLOAD PHASE STATISTICS - NUMBER OF RECORDS UNLOADED=8935 FOR TABLESPACE ROY.ROYTSP PART 2
 DSNU251I  -DB9B 242 18:13:18.23 DSNURPUT - UNLOAD PHASE STATISTICS - NUMBER OF RECORDS UNLOADED=162 FOR TABLESPACE ROY.ROYTSP PART 3
 DSNU251I  -DB9B 242 18:13:18.23 DSNURPUT - UNLOAD PHASE STATISTICS - NUMBER OF RECORDS UNLOADED=0 FOR TABLESPACE ROY.ROYTSP PART 4
 DSNU251I  -DB9B 242 18:13:18.23 DSNURPUT - UNLOAD PHASE STATISTICS - NUMBER OF RECORDS UNLOADED=81 FOR TABLESPACE ROY.ROYTSP PART 5
 DSNU250I    242 18:13:18.23 DSNURPRD - UNLOAD PHASE COMPLETE, ELAPSED TIME=00:00:00
 DSNU303I  -DB9B 242 18:13:19.38 DSNURWT - (RE)LOAD PHASE STATISTICS - NUMBER OF RECORDS=8935 FOR TABLE PAOLOR9.ROYTABP PART=2
 DSNU303I  -DB9B 242 18:13:19.38 DSNURWT - (RE)LOAD PHASE STATISTICS - NUMBER OF RECORDS=162 FOR TABLE PAOLOR9.ROYTABP PART=3
 DSNU303I  -DB9B 242 18:13:19.38 DSNURWT - (RE)LOAD PHASE STATISTICS - NUMBER OF RECORDS=0 FOR TABLE PAOLOR9.ROYTABP PART=4
 DSNU303I  -DB9B 242 18:13:19.38 DSNURWT - (RE)LOAD PHASE STATISTICS - NUMBER OF RECORDS=81 FOR TABLE PAOLOR9.ROYTABP PART=5
 DSNU302I    242 18:13:19.39 DSNURILD - (RE)LOAD PHASE STATISTICS - NUMBER OF INPUT RECORDS PROCESSED=9178
 DSNU300I    242 18:13:19.39 DSNURILD - (RE)LOAD PHASE COMPLETE, ELAPSED TIME=00:00:01
 DSNU394I  -DB9B 242 18:13:19.48 DSNURBXA - SORTBLD PHASE STATISTICS - NUMBER OF KEYS=9178 FOR INDEX PAOLOR9.XMAP_TBL
 DSNU393I  -DB9B 242 18:13:19.49 DSNURBXA - SORTBLD PHASE STATISTICS - NUMBER OF KEYS=8935 FOR INDEX PAOLOR9.IX1 PART 2
 DSNU393I  -DB9B 242 18:13:19.55 DSNURBXA - SORTBLD PHASE STATISTICS - NUMBER OF KEYS=162 FOR INDEX PAOLOR9.IX1 PART 3
 DSNU393I  -DB9B 242 18:13:19.63 DSNURBXA - SORTBLD PHASE STATISTICS - NUMBER OF KEYS=81 FOR INDEX PAOLOR9.IX1 PART 5
 DSNU391I    242 18:13:19.64 DSNURPTB - SORTBLD PHASE STATISTICS. NUMBER OF INDEXES = 2
 DSNU392I    242 18:13:19.64 DSNURPTB - SORTBLD PHASE COMPLETE, ELAPSED TIME = 00:00:00
 DSNU1162I   242 18:13:19.64 DSNURLGD - LOG APPLIES WILL BE PERFORMED IN PARALLEL, NUMBER OF TASKS = 2
 DSNU386I    242 18:13:34.98 DSNURLGD - LOG PHASE STATISTICS. NUMBER OF ITERATIONS = 1, NUMBER OF LOG RECORDS = 891
 DSNU385I    242 18:13:34.98 DSNURLGD - LOG PHASE COMPLETE, ELAPSED TIME = 00:00:15
 DSNU400I    242 18:13:35.00 DSNURBID - COPY PROCESSED FOR TABLESPACE ROY.ROYTSP PART 2:5
                       NUMBER OF PAGES=1001
                       AVERAGE PERCENT FREE SPACE PER PAGE =  3.65
                       PERCENT OF CHANGED PAGES =100.00
                       ELAPSED TIME=00:00:17
 DSNU387I    242 18:13:35.89 DSNURSWT - SWITCH PHASE COMPLETE, ELAPSED TIME = 00:00:00
 DSNU428I    242 18:13:35.89 DSNURSWT - DB2 IMAGE COPY SUCCESSFUL FOR TABLESPACE ROY.ROYTSP PARTITION 2:5
 DSNU010I    242 18:13:37.30 DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

Figure 11-3 REORG by partition range - Output

The new processing without the BUILD2 phase is not optional. Users should be aware of the following implications:

- DB2 will need additional temporary disk storage for the shadow data sets for each NPI.
- The cost of the REORG will increase, as the build of the entire NPI will require more CPU than the updates that occurred in the BUILD2 phase.
- As the entire NPI will be built and then switched, it is no longer possible to run reorganizations of different parts in separate jobs in parallel. Jobs that ran in parallel will need to be changed.

  If contiguous partitions are being reorganized and specified in a range (for example, PART 4:8), then parallelism will occur within the utility. Similarly, SCOPE PENDING, could have parallelism within the utility.

It should also be noted that in V8, NPIs remained potentially disorganized, following a REORG by part. In V9 the NPIs are in a reorganized state, following a REORG.

11.1.3 Change to REORG SHRLEVEL REFERENCE by part

In a similar fashion to OLR SHRLEVEL CHANGE by part, SHRLEVEL REFERENCE by part now rebuilds the whole of any NPI in full shadows. The NPIs is switched along with the other data sets in the SWITCH phase. Here are the implications:

- There is a LOGAPPLY phase, during which index updates to the logical parts of the shadow NPIs for which the partition is not being updated will be applied.

- DB2 needs additional temporary disk storage for the shadow data sets for each NPI.
- The cost of the REORG increases, as the build of the entire NPI is more expensive than the updates that occurred in the BUILD2 phase
- As the entire NPI is built and then switched, it is no longer possible to run reorganizations of different parts in separate jobs in parallel.

11.1.4 Online REORG usability and keyword changes

The following are a series of usability changes that make improvements introduced in earlier versions, the standard way of processing, the new default, or have a more useful setting.

Changes to draining and retry options

OLR now always uses the DRAIN_WAIT functionality for more consistent waits on drains. If the keyword is omitted or specified as 0, then a value equal to IRLMRWT multiplied by UTIMOUT is used.

When OLR cannot obtain a drain that it needs, then retry logic is used by default. If the RETRY keyword is not specified then a value equal to the UTIMOUT is used. RETRY 0 can be specified if the user does not require any retries.

The RETRY_DELAY keyword has the default reduced from 5 minutes to 10 seconds or the DRAIN_WAIT value multiplied by the RETRY value if the resultant value is less. So, for example, if RETRY 3 and DRAIN_WAIT 2 are in effect, the default for RETRY_WAIT would be 6 seconds. If RETRY 5 and DRAIN_WAIT 20 are being used, then 100 seconds would be the RETRY_WAIT default. Users can still specify their own value in seconds in the range of 1 to 1800.

Should OLR not be able to drain and the retry limit is exceed, the default action has changed from ABENDing to terminating with a return code 8 (that is, the TIMEOUT default has changed to TERM from ABEND). The TIMEOUT keyword is still available, so TIMEOUT ABEND could be used if desired.

The MAXRO keyword that controls the last log iteration has the default changed from five minutes to the same value as is being used for RETRY_DELAY.

Switch method rationalization

The fast switch method to be used in the Switch phase is now the only option for use with user table spaces. With the DB2 9 General Availability code level, the FASTSWITCH YES/NO keyword is still permitted, but is ignored and a value of YES is always used, unless the object being reorganized is in the Catalog or Directory, when the fast switch method is not used.

However, since this can affect regularly run DSN1COPY jobs or replication jobs that rely on a consolidated data set naming convention (ending with I000n.A00x), a planned APAR is re-instating the option YES/NO.

11.1.5 Compressed parts handled by Reorg

In V8, for virtual storage reasons, REORG was limited to operating on 254 compressed partitions. This limit is now lifted in V9.

11.1.6 LOB REORG enhancements

DB2 introduced large object support (LOBs) in V6. The REORG utility, however, handled the LOB table spaces in a significantly different fashion than regular table spaces. The REORG of LOBs on V8 had the following disadvantages, all of which had been resolved for the regular table spaces:

- Concurrency: The only SHRLEVEL supported for LOB table spaces is NONE, meaning that there was no access to the LOB during the execution of the utility.
- LOBs are manipulated within the existing table space. The LOB table space data sets are not reset. As a consequence, physical disk space is not reclaimed
- The REORG has the aim of “re-chunking” the LOB, this results in a optimal reorganization versus physical space consumption trade-off.
- The *in place* nature of the Reorg meant the activity was logged. With LOG YES the only option, significant logging could occur.
- Recovering from failures and problem determination is harder than with the Reorg of conventional table spaces.

The DB2 9 enhancements to REORG mean LOBs can be handled in a similar way to regular table spaces, in that the data is unloaded and reloaded into a new data set.

The processing of LOBs is controlled by the SHRLEVEL option. The options are now NONE and REFERENCE. If NONE, which is the default, is chosen, then REORG of LOBs works in an identical way to V8. SHRLEVEL REFERENCE results in the new style processing.

REORG SHRLEVEL REFERENCE for LOBs has the following characteristics:

- Shadow data sets are required in the same way as any other SHRLEVEL REFERENCE REORG. There are ‘J’ data sets associated with LOB table spaces as well as ‘I’ data sets. Additional disk space is consequently required during the duration of the utility. The shadow data sets can be sized to facilitate reclaiming of space.
- The LOBs are *copied* from the original to the shadow table space. This results in an implicit reorganization. There is no sorting as with regular table spaces. The existing LOB allocation algorithms are used.
- The loading of data into the shadow table spaces is not logged. LOG YES is not a valid option for REORG SHRLEVEL REFERENCE when executed against LOBs.
- An in-line image copy must be produced.
- There is a switch phase, as with other REORG SHRLEVEL REFERENCE. During the switch phase all SQL activity against the LOB table space is drained. At other times writing of LOBs is drained. Note that as with V8 or SHRLEVEL NONE, the base table is not restricted at all during the Reorg.
- Two SYSCOPY records are inserted to record the REORG (non-recoverable event) and the image copy.
- The utility can be restarted or terminated in the same way as other REORG SHRLEVEL REFERENCE.

11.2 CLONE TABLE support

All utilities have been enhanced to fully supported cloned tabled. See 4.2.6, “Utilities and clone tables” on page 92, for details.

11.3 BACKUP and RESTORE SYSTEM

BACKUP and RESTORE SYSTEM utilities were added in DB2 V8 and use disk volume FlashCopy backups and copypool z/OS DFSMShsm™ V1R5 constructs. In DB2 V9 these utilities are enhanced to use new functions available with z/OS V1R8 DFSMShsm.

One enhancement allows individual table spaces or index spaces to be recovered (the V8 support was for the entire system).

Maintaining multiple copies of all the data on disk can be expensive, DB2 V9 also allows for the backup to be implemented directly to tape.

The physical copying to disk in the background can be improved by the use of incremental FlashCopy.

Even if incremental FlashCopy is used, the dump to tape is always a full dump. Incremental FlashCopy has no benefit when dumping to tape except that the dumping to tape might begin earlier because less data needs to be copied to disk before writing to tape can be started.

11.3.1 Object-level recovery

The backups taken by the BACKUP SYSTEM utility (referred to here as system level backups) in V8 are used for recovery of the entire DB2 system. DB2 V9 has an enhancement to allow a subset of the data to be recovered from the system-level backups. Recovery is via the RECOVER utility, which is now capable of using system-level backups for the restore phase in addition to image copies. In order to allow RECOVER to consider using system level backups there is a DSNZPARM option which can be set from panel DSNTIP6. This is the first option on the panel, as shown in Figure 11-4 on page 392. Having enabled the use of system-level backups, they will be automatically considered in object level recoveries..

Note: You need to alter your indexes to the COPY YES attribute to enable RECOVER from system-level backups.

Unless the installation is also planning to offload the system-level volume backups to tape see 11.3.2, “Tape support for BACKUP SYSTEM” on page 393, there are no syntax changes to RECOVER. The user will specify the table spaces or indexes in a list, individually, or via a LISTDEF, and the RECOVER utility will determine the most recent recovery base. In addition to the image copies recorded in the SYSCOPY (or the log for some catalog or directory spaces), the system-level volume backups are also examined. If the last copy before the recovery point is a system-level volume backup, then the specific data sets will be extracted and restored via DFSMShsm.

Note that if RECOVER cannot successfully restore from this system-level backup, then the utility will terminate, assuming that the OPTION EVENT(ITEMERROR,SKIP) was not specified. This is in contrast to RECOVER having a problem locating an image copy on tape or disk, when an alternative such as the *local backup* or an earlier image copy is tried. If the system level volume backup is not usable for any reason, then the user can bypass it by using the RESTOREBEFORE option on RECOVER. See 11.4.2, “RECOVER using earlier copy” on page 398.

Restriction: If a data set has moved to a different volume or was deleted between the time of the BACKUP SYSTEM utility and the time of the RECOVER utility, then object level recovery is not possible unless an image copy is available.

Allowing system-level volume backups to be used by RECOVER means that conventional image copies need be taken on a potentially much-reduced frequency. For example, if there is a system backup daily, then running image copies daily may not be needed, unless for special reasons like DR. Due to the restriction noted above, we do not recommend that image copies are dispensed with entirely. Image copies will still be required following LOAD REPLACE and REORG LOG NO, as enforced by DB2.

Note: Running BACKUP SYSTEM will now result in Real Time Statistics columns for COPY to be updated because system-level backups may now be used in object-level recovery.

The installation panel DSNTIP6 is shown in Figure 11-4. The first four options are discussed in this section.

```
DSNTIP6          INSTALL DB2 - DB2 UTILITIES PARAMETERS
===>

Enter system-level backup options for RESTORE SYSTEM and RECOVER below:
 1 SYSTEM-LEVEL BACKUPS ===> NO         As a recovery base: NO or YES
 2 RESTORE/RECOVER      ===> NO         From dump: NO or YES
 3 DUMP CLASS NAME      ===>            For RESTORE/RECOVER from dump
 4 MAXIMUM TAPE UNITS   ===> NOLIMIT    For RESTORE SYSTEM: NOLIMIT or 1-255

Enter other DB2 Utilities options below:
 5 TEMP DS UNIT NAME    ===> SYSDA      Device for temporary utility data sets
 6 UTILITY CACHE OPTION ===> NO         3990 storage for DB2 utility IO
 7 STATISTICS HISTORY   ===> NONE       Default for collection of stats history
 8 STATISTICS ROLLUP    ===> NO         Allow statistics aggregation: NO or YES
 9 STATISTICS CLUSTERING===> ENHANCED   For RUNSTATS (ENHANCED or STANDARD)
10 UTILITY TIMEOUT      ===> 6          Utility wait time multiplier



PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 11-4 Panel: DSNTIP6

Option 9, Statistics Clustering, is a new option related to the new cluster ratio formula Implemented in RUNSTATS utility with DB2 9 for z/OS. This new formula calculates cluster ratio values which are more accurate for the purpose of Access Path selection, and takes into consideration both the forward and backward order scan. With V9, when the RUNSTATS utility is executed on indexes with highly duplicated keys or reverse order data clustering significant clusterratio value changes from the previous version are possible.

You can use option 9 or change the DSNZPARM STATCLUS to control the collection of clusterratio value. Enhanced, the default, is the recommended value and it will activate the V9 new formula. Standard reverts to the old calculation and it can be used to maintain the same cluster ratio values.

11.3.2 Tape support for BACKUP SYSTEM

The BACKUP SYSTEM utility has been enhanced to allow a FlashCopy to go directly to tape. See Figure 11-5 for the new options.

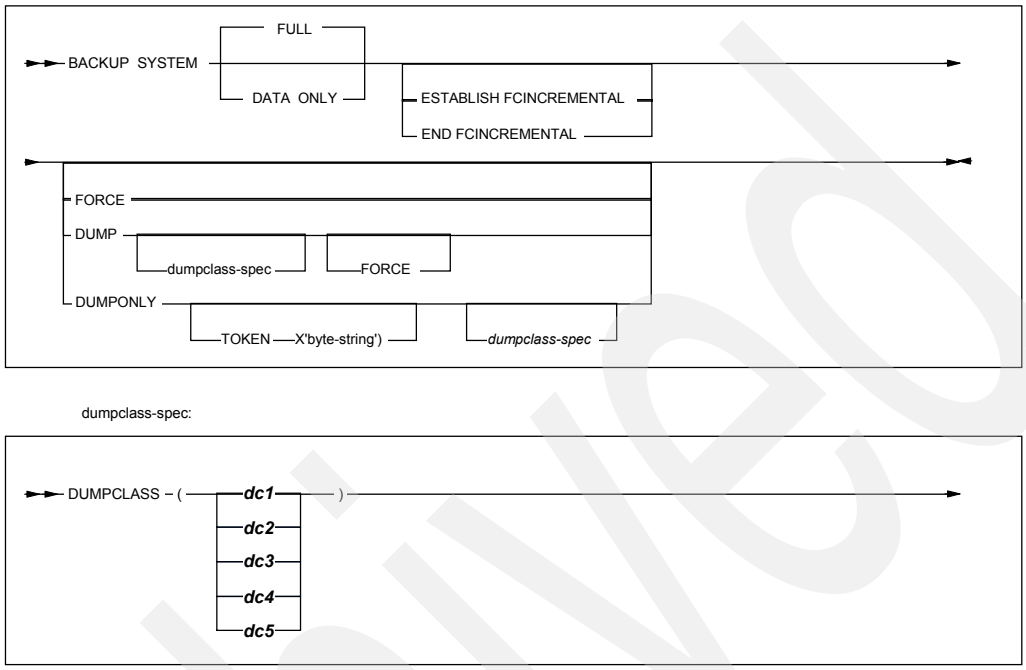


Figure 11-5 BACKUP SYSTEM syntax

They are the DUMP or DUMPONLY options on the BACKUP SYSTEM utility control cards. The output of the DUMP or DUMPONLY is directed to what is called DFSMShsm *dump class*. Each dump class will specify the unit type the data will be directed to.

This DB2 enhancement was designed to allow for the data to go to tape. However, an SMS dump class is not restricted to a tape medium.

When the data class is tape, you can specify in SMS to what degree volumes will be stacked on tapes. There will be a trade-off in tape capacity usage and performance to be considered since the more volumes are stacked on a tape, the longer it will take to search.

DB2 allows up to five dump classes to be specified. Each specification results in a complete copy of the copypools being backed up being directed to each dump class. Any specification in the utility control cards of dump classes will override the dump classes defined in the COPYPOOL definitions.

By having multiple dump classes, it could be possible, for example, to send one copy to remote devices for DR purposes, while keeping another copy for local operational reasons.

Important: Having data on tape may be advantageous for moving the data off site for DR situations. It may also allow for longer retention of data locally. It does, however, not begin to compare with the speed of a FlashCopy during restore. Full system restores from tape will be lengthy processes.

Although the logical FlashCopy is extremely fast, the physical copying of data does take time and can lead to a significant load on the I/O subsystem. See 11.3.3, “Incremental FlashCopy”

Archived

on page 395, for one way of reducing I/O load during SYSTEM BACKUP. The choice of DUMP or DUMPONLY is another way to reduce impact when off loading to tape. When the DUMP option is selected, the copying to tape is started ahead of the physical completion of the FlashCopy. This means that for a period of time there is additional load on the I/O subsystem, that is, the disk-to-disk copy associated with the FlashCopy, plus the disk-to-tape copy. The DUMPONLY option can be used to copy a previous backup to tape at a later time, thereby not causing such a big peak in I/O activity.

Tip: To minimize I/O activity peaks, perform a BACKUP SYSTEM without DUMP and follow it with a BACKUP SYSTEM with DUMPONLY.

Since the copying to tape is relatively slow, a new FORCE keyword has been added to BACKUP SYSTEM. This allows a new backup to be started even though a previous DUMP has not finished. FORCE should only be used if it is more important to take a new backup than to let the offloading to tape of the previous dump to tape complete.

Note: The LIST COPYPOOL with DUMPVOLS option can be used to verify the status of DUMP and DUMPONLY.

There is a *no copy* option set in SMS to allow for copies to go to a dump class, without the FlashCopy versions going to disk. COPY NO is set on the SMS COPYPOOL PANEL. It is the *number of recoverable disk versions* that is set to zero for COPY NO. Note that this option could never provide for quick restores, and the backup could be very lenghty as well.

The RESTORE SYSTEM utility has been improved to use the system-level backups that have been dumped to tape. Note that RESTORE only handles the *database* COPYPOOL, since DB2 must be started to run the utility.

Archive

The panel in Figure 11-4 on page 392 shows three parameters to control the RESTORE. All of these can be overridden by the utility control statements.

- Option 2 determines whether the FlashCopy version from a dump class is to be used rather than one on disk. This parameter applies also to RECOVER. The default NO may be preferred by customers who do not want to use the potentially slower recovery method.
- Option 3 determines the default dump class to restore from. This also applies to RECOVER.
- Option 4 determines the maximum number of tape units to be used by RESTORE. The default is not to limit the number of drives, but this can be capped at anything from 1 to 255.

Note: The reason for option 2 is to allow the tape version to be specified, rather than the significantly faster disk FlashCopy version, to support disaster recovery. The tapes could be sent to the disaster recovery location and registered in HSM.

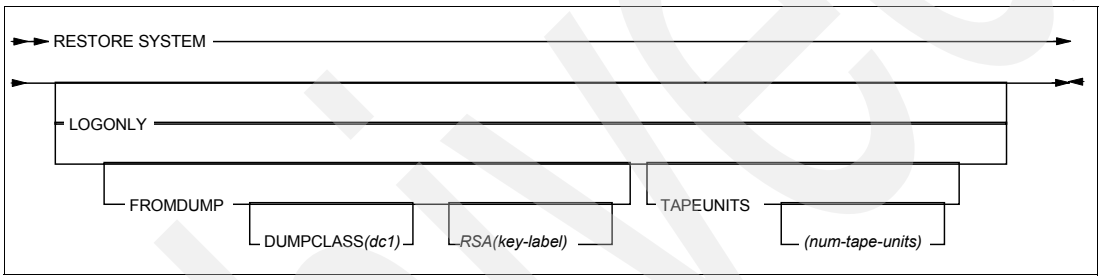


Figure 11-6 RESTORE SYSTEM syntax

In summary, while tape backups can be useful, the time to restore from such backups should never be underestimated. It is expected that installations with high-availability requirements will restore via disk FlashCopy in all but catastrophic situations.

11.3.3 Incremental FlashCopy

In order to reduce I/O activity, support is being added to allow incremental FlashCopy. In addition to z/OS Version 1 Release 8 DFSMShsm, this function requires APAR OA17314 for z/OS. Note that incremental FlashCopy does not reduce the need for disk volumes, unlike an incremental image copy potentially does.

In order for FlashCopy incrementals to be used, the relationship must first be established. This can be done either by HSM commands or by using the new options on the BACKUP SYSTEM utility. See Figure 11-5 on page 393, ESTABLISH FCINCREMENTAL and END FCINCREMENTAL are added to specify a persistent or last incremental FlashCopy relationship is to be established. Similarly, they can be ended via an HSM command or the utility. All tracks on the source volume are considered to be changed when the relationship is established, so all tracks are copied. Subsequent incremental FlashCopies will copy only the tracks that have changed on the source volume since the last copy was taken. The previous content on the volumes will be replaced by the current contents.

Do not confuse the adjective *incremental* as used with image copies with the one used with FlashCopy. With COPY you can merge incremental and previous copies. With FlashCopy the incremental changes are immediately applied to the previous full copy.

Note: A disk volume can have only one incremental relationship. This means if a copypool has more than one disk generation, then the other generations will be full backups.

DFSMShsm Fast Replication associates a defined number of versions to a copypool. This specifies the maximum number of versions that can be maintained in parallel. Each invocation of the FRBACKUP command creates a new version in the Copy Pool (with version numbers assigned automatically in the range from 1 to 999). If more versions are created than the copypool can support in parallel, then the oldest version will be deleted and replaced by the newest version. This is sometimes referred to as *roll-off* processing.

Since version numbers are always increased (with the maximum being 999 and then restarting at 1), a relative addressing of the versions in the Copy Pool is more appropriately provided by generations. They follow the same concept as Generation Data Groups (GDGs). Generation 0 always refers to the most current version in the Copy Pool. The second most current version is referred to as generation 1, and so forth.

Figure 11-7 shows the different time to take the copies. V1 is a full backup. V2 is the first incremental, and therefore all pages are copied. This takes as long as the full backup. V3 is an incremental, and this is much quicker. Similarly, V4 and V5 are incrementals with a shorter duration than a full backup. The backup that withdraws the relationship is still an incremental. T6 and T7 are full backups.

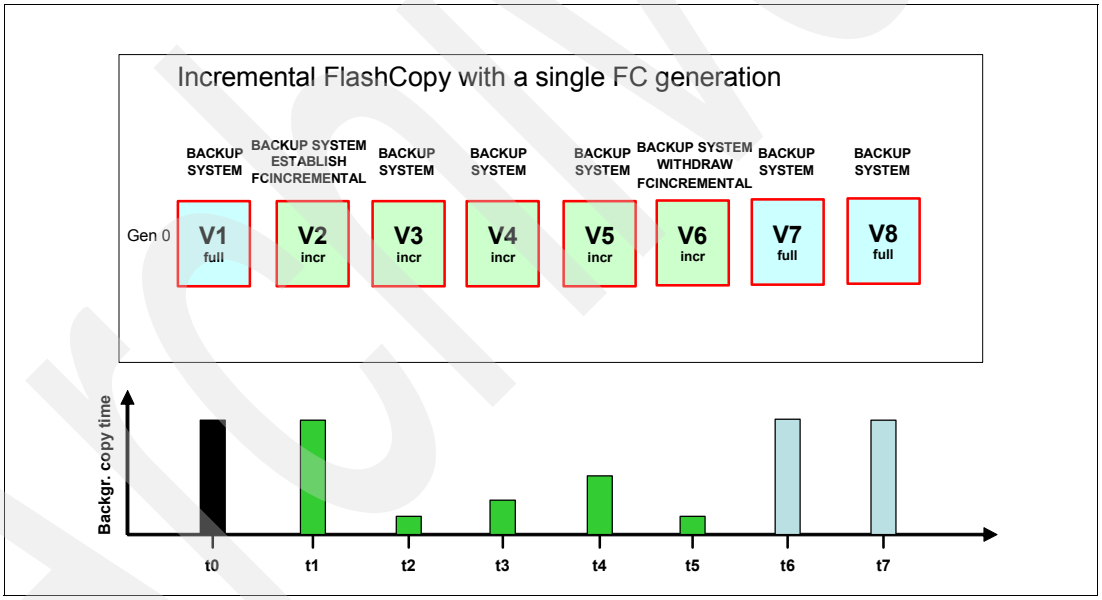


Figure 11-7 Incremental FlashCopy with single version

Archive

Figure 11-8 shows what happens when the copypool is set up to have two versions. So for each source volume we have two sets of target volumes we can copy to. However, we cannot be using incremental FlashCopy to both of them, as we are restricted to a single incremental relationship per volume.

V1 is the backup that establishes the incremental, so it copies all the data. V2 cannot be an incremental, as it is going to the other set of volumes and is a full backup. V3 will be an incremental, and hence shows a shorter time. The next backup V4 will be a full backup. Every other backup is a full in this setup, alternating with the incremental. So V5 will be a incremental. However, it withdraws the relationship and all subsequent backups are full.

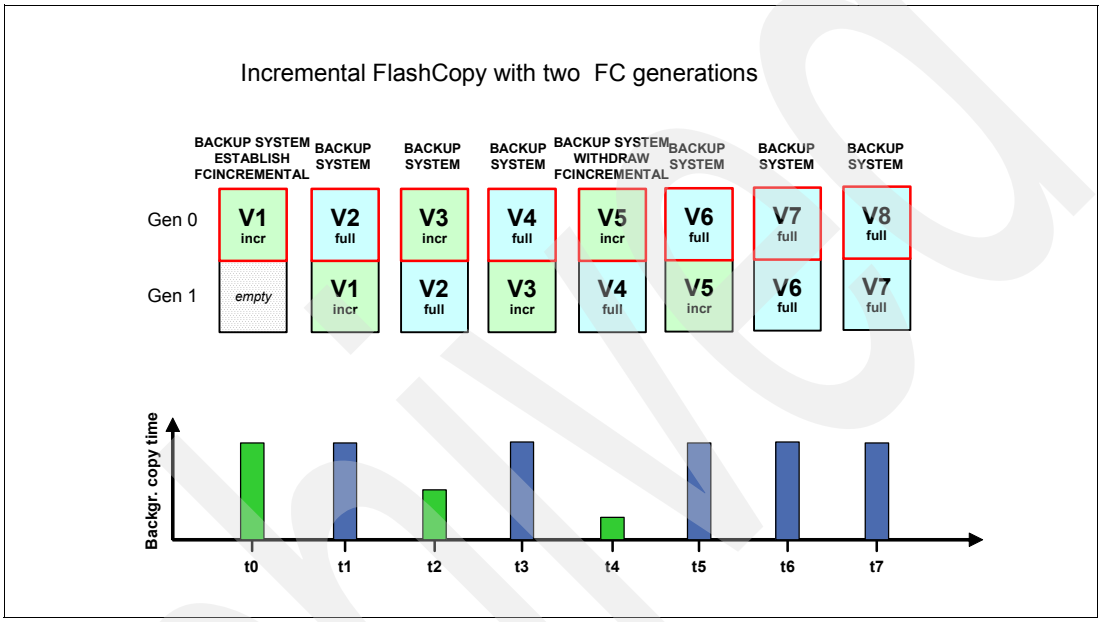


Figure 11-8 Incremental FlashCopy with two versions

Clearly, the incremental reduces the copy time. It also reduces the workload on the I/O subsystem. When there are multiple versions allowed for a copypool, the benefit diminishes. If we had five volumes, then only 20% of backups could be incremental. In a two generation example it maybe possible to take two backups a day. One in quiet time of the day (the full) and one at a busier time of the day (incremental).

11.4 RECOVER enhancements

The RECOVER utility has been enhanced in three areas:

- Displaying progress of RECOVER
- Recovery to point in time with consistency
- RECOVER using earlier copy

In this section we examine the first and the third areas. For the second one refer to 4.3, “Recovery to point in time with consistency” on page 100.

11.4.1 Displaying progress of RECOVER

The -DISPLAY UTILITY command has been enhanced so that during the LOGAPPLY phase of RECOVER, a new message DSNU116I will be issued. This new message indicates the

range of the log that needs to be applied for the list of objects being recovered. It also shows the progress that has been made up to the last commit and the elapsed time since the start of the log apply phase of the recovery.

From these values an estimate can be made as to how much longer there is to perform the recovery. See Example 11-1.

Example 11-1 DSNU116I message

```
DSNU116I csect-name RECOVER LOGAPPLY PHASE DETAILS:
                  STARTING TIME = timestamp
                  START RBA = ss START LRSN =rr
                  END RBA = ee END LRSN = nn
                  LAST COMMITTED RBA = cc LAST COMMITTED LRSN = ll
                  ELAPSED TIME = hh:mm:ss
```

11.4.2 RECOVER using earlier copy

This enhancement to the RECOVER utility allows the user to specify the latest point in the log that a copy will be selected during recover. So, for example, if the most recent copy is known to have a problem, then the use of an earlier copy could be forced, by specifying an RBA/LRSN just before the most recent copy. The new keyword is RESTOREBEFORE.

This option could be of particular value when system-level backups are being used. Unlike regular image copies, there is no fallback support to a local backup (ICTYPE 'LB') or prior backups. It would also be useful when image copies taken to tape are known in advance not to be available (for example, in a DR situation).

11.5 LOAD enhancements

The LOAD enhancements we describe in this section are associated to:

- LOAD restarting restrictions when the LOBs involved are lifted
- LOAD and UNLOAD to support rounding with DECFLOAT new data type
- UNLOAD to skip locked rows

For LOAD-reduced CPU utilization, this performance enhancement is mostly due to changes in the BUILD indexes phase, which has optimized index management. See 13.6, "Utilities CPU reduction" on page 508.

11.5.1 Allowing LOAD with SORTKEYS NO

When a LOAD SHRLEVEL NONE of a table with a LOB column fails, it can only be restarted with RESTART(CURRENT). This restriction was introduced with LOB support.

However, when the SORTKEYS keyword was introduced in V5, a restriction with its use was that a LOAD that abended in the RELOAD, SORT, or BUILD phases could only be restarted with RESTART(PHASE).

Restart will convert any RESTART option to RESTART(PHASE) when SORTKEYS is specified.

In V8 there is an issue in that SORTKEYS became the default. SORTKEYS and LOB restart processing became incompatible. LOB restriction demands RESTART(CURRENT), but the SORTKEYS requires RESTART(PHASE).

In V9, DB2 allows SORTKEYS to be turned off, by specifying SORTKEYS NO. DB2 will also allow RESTART(PHASE) for LOAD REPLACE of a table with a LOB. See Figure 11-9.

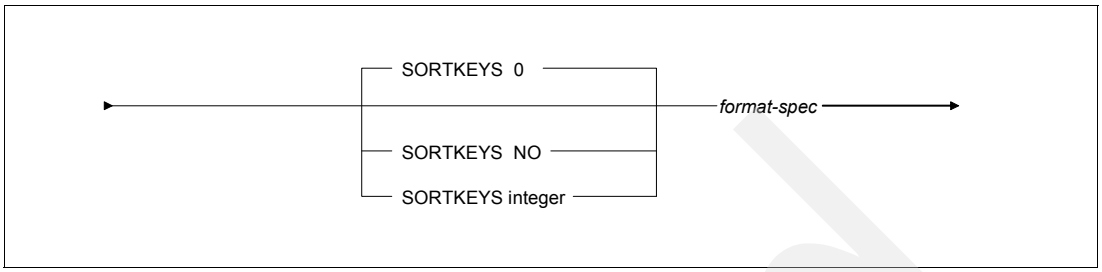


Figure 11-9 SORTKEYS syntax for LOAD

For a load of a table with a LOB, using LOAD REPLACE SHRLEVEL NONE, the SORTKEYS option will determine how the utility can be restarted.

- If you specify SORTKEYS NO, you can restart with either RESTART(CURRENT) or RESTART(PHASE).
- If you do not specify SORTKEYS NO, you can restart only with RESTART(PHASE).

For a LOAD RESUME YES SHRLEVEL NONE of a table with a LOB column, to be restartable, SORTKEYS NO *must* be specified. In this case you will be able to perform a RESTART(CURRENT).

11.5.2 Rounding of DECFLOAT by UNLOAD and LOAD

Both LOAD and UNLOAD are enhanced to handle the floating decimal data type.

The DECFLOAT input data type is compatible with output data types SMALLINT, INTEGER, BIGINT, DECIMAL, FLOAT, and DECFLOAT. While input data types SMALLINT, INTEGER, BIGINT, DECIMAL, FLOAT, and DECFLOAT are compatible with the DECFLOAT output data type. Both LOAD and UNLOAD allow users to specify which mode of rounding is to be used.

The new syntax block is the same for both utilities and is shown in Figure 11-10.

If DECFLOAT ROUNDMODE is not specified the action is determined by DSNHDECP.

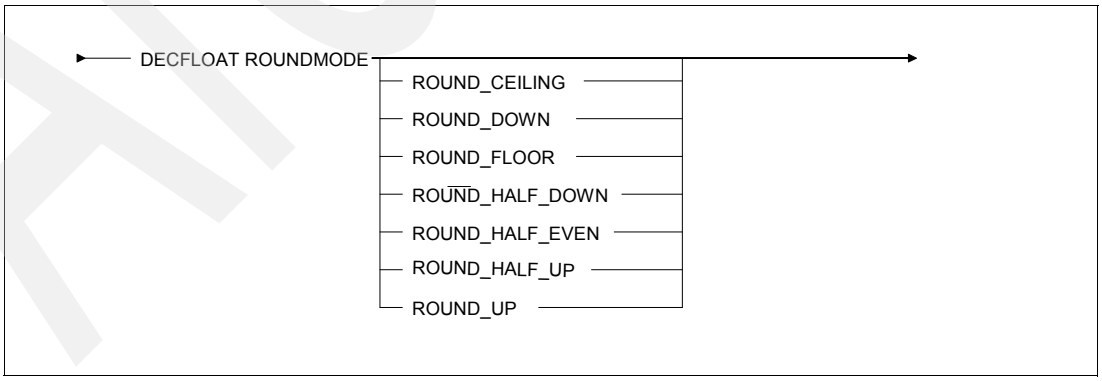


Figure 11-10 DECFLOAT ROUNDMODE syntax for LOAD and UNLOAD

Archived

The rounding modes supported are:

- ROUND_CEILING - Round towards +infinity. If all of the discarded digits are zero or if the sign is negative, the result is unchanged other than the removal of discarded digits. Otherwise, the result coefficient should be incremented by 1 (rounded up).
- ROUND_DOWN - Round towards 0 (truncation). The discarded digits are ignored.
- ROUND_FLOOR - Round towards -infinity. If all of the discarded digits are zero or if the sign is positive, the result is unchanged other than the removal of discarded digits. Otherwise, the sign is negative and the result coefficient should be incremented by 1.
- ROUND_HALF_DOWN - Round to nearest. If equidistant, round down. If the discarded digits represent greater than half (0.5) of the value of a one in the next left position, then the result coefficient should be incremented by 1 (rounded up). Otherwise (the discarded digits are 0.5 or less) the discarded digits are ignored.
- ROUND_HALF_EVEN - Round to nearest. If equidistant, round so that the final digit is even.If the discarded digits represent greater than half (0.5) the value of a one in the next left position, then the result coefficient should be incremented by 1 (rounded up). If they represent less than half, then the result coefficient is not adjusted (that is, the discarded digits are ignored). Otherwise (they represent exactly half), and the result coefficient is unaltered if its right most digit is even, or incremented by 1 (rounded up) if its right most digit is odd (to make an even digit).
- ROUND_HALF_UP - Round to nearest. If equidistant, round up. If the discarded digits represent greater than or equal to half (0.5) of the value of a one in the next left position, then the result coefficient should be incremented by 1 (rounded up). Otherwise the discarded digits are ignored.
- ROUND_UP - Round away from 0. If all of the discarded digits are zero the result is unchanged other than the removal of discarded digits. Otherwise, the result coefficient should be incremented by 1 (rounded up).

11.5.3 Skipping locked rows during UNLOAD

The ability to skip rows that are locked has been added to the UNLOAD utility. When UNLOAD is running with the SHRLEVEL CHANGE ISOLATION CS clauses, the additional option of SKIP LOCKED DATA is available.

With SKIP LOCKED DATA, the UNLOAD utility can skip rows on which incompatible locks are held by other transactions. This option applies for table spaces with either row level or page level locking.

11.6 Histogram statistics

When there is a query involving a range predicate, such as the following type:

```
SELECT * FROM SALES_TABLE
WHERE SALES_DATE BETWEEN ‘2007-01-15’ AND ‘2007-01-21’;
```

the single value-based frequency statistics sometimes cannot really help DB2 with the predicate selectivity, which may lead to undesirable access paths. These are the cases where the distribution can vary within ranges. One example can be sparse ranges versus dense ranges, like items sales during two weeks in March versus two weeks before Christmas. Another example is when some ranges are skipped or have no existing values.

Column frequency helps with fewer values. Histogram statistics collect the frequency statistics of the distinct values of a column cardinality over the entire range, giving better selectivity, and have the potential to allow better access path selection. This function is already provided by DB2 for Linux, UNIX, and Windows.

Figure 11-11 depicts a sample quantile distribution.

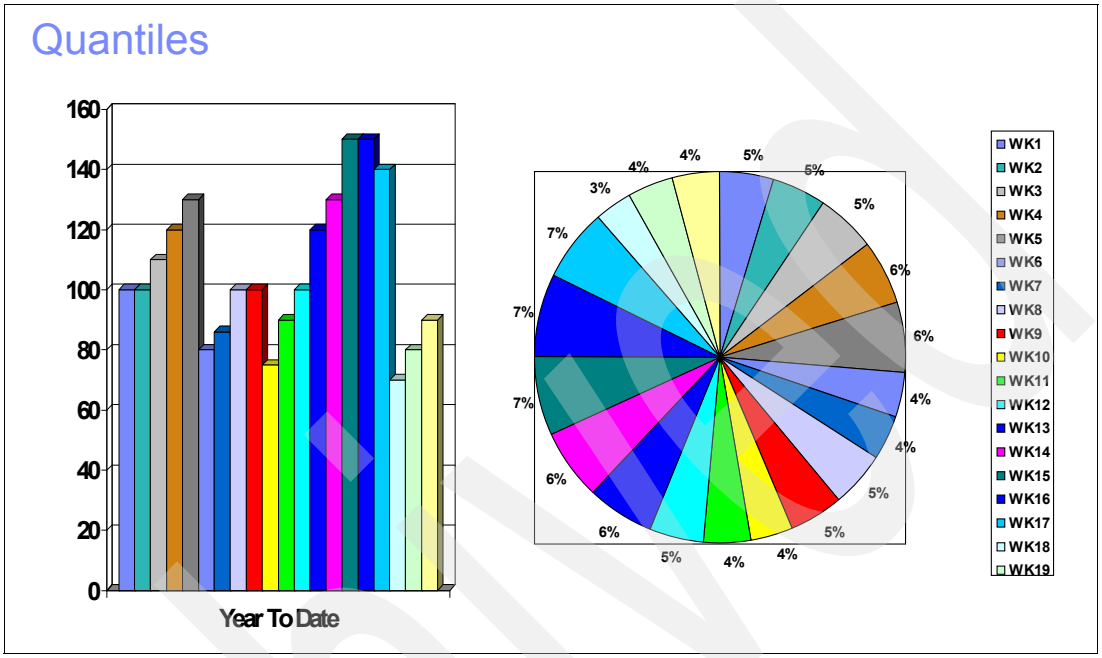


Figure 11-11 Histogram statistics

With DB2 9, RUNSTATS can now collect the information by what are called *quantiles*. DB2 allows up to 100 quantiles. The user can specify how many quantiles DB2 is to use from 1 to 100. Note that 1 is not meaningful. If a value N is specified for NUMQUATILES, the specified number will be used as guidance by RUNSTATS. However, the result may vary slightly: it could be from (N-1) to (N+1). Only when column cardinality is smaller than N, then the result may be equal to the column cardinality and thus less than N.

RUNSTATS will produce an equal-depth histogram (that is, each interval (range) will have about the same number of rows (not the same number of values)). As a general recommendation, specify 100 or allow the default and let DB2 arrive at a value. Lower values can be used when there is a good understanding of the application, for example, if the query ranges are always done on boundaries like 0–10%, 10–20%, 20–30%, and so on, and then NUMQUANTILES 10 maybe a better choice.

Figure 11-12 shows the RUNSTATS TABLESPACE added specifications. Similar syntax can be specified for RUNSTATS INDEX.

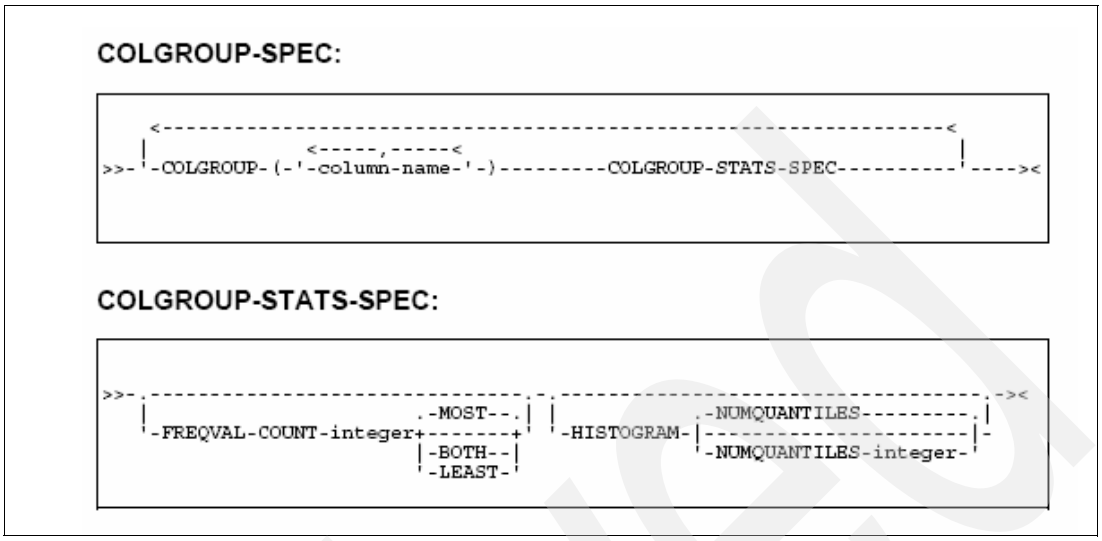


Figure 11-12 Histogram syntax for RUNSTATS

The predicate selectivity is more accurate if the searching range matches the boundary of any one quantile or any group of consecutive quantiles. Even if there is no perfect match, the predicate selectivity interpolation is done with one or two particular quantiles, which results in more accurate predicate evaluation.

Three new columns are added: QUANTILENO, LOWVALUE, and HIGHVALUE. The three new columns are present in the six catalog tables: SYSIBM.SYSCOLDIST, SYSIBM.SYSKEYTGTDIST, SYSIBM.SYSCOLDIST_HIST, SYSIBM.SYSCOLDISTSTATS, SYSIBM.SYSKEYTGTDIST_HIST, and SYSIBM.SYSKEYTGTDISTSTATS.

You can use histogram statistics to evaluate predicate selectivity: the better filter factor will benefit RANGE/LIKE/BETWEEN predicates for all fully qualified intervals, and interpolation of partially qualified intervals. It can also help in case of EQ, IS NULL, IN LIST, and COL op COL.

11.7 COPY improvements

We examine improvements related to the COPY utility. Note that the COPY utility now uses the most recently used (MRU) algorithm for the pages that it reads in, thus preventing trashing of the bufferpool.

11.7.1 COPY with CHECKPAGE

The CHECKPAGE option is generally recommended in best practices. Prior to V9, COPY has two disadvantages. The first one is the relatively expensive CPU overhead (about 5% for COPY INDEX and 14% for COPY TABLESPACE with DB2 V8). The second one is that, if COPY finds a broken page, it puts the table space or index space into COPY PENDING state and making it inaccessible to further user operations while it continues to look for broken pages till the end of the object.

With V9, work has been done to reduce the CPU overhead for COPY TABLESPACE with CHECKPAGE to be almost negligible. Furthermore, the buffer steal algorithm used by the

COPY utility has been changed from LRU to MRU buffer. This change not only reduces CPU time but also prevents the content of the buffer pool from being dominated by the image copy.

With V9, the CHECKPAGE option is always in operation. The COPY utility performs validity checking for each page, one page at a time, in the table space or index space. If it finds an error, COPY will issue message DSNU518I, which identifies the broken page and the type of error. If more than one error exists in a page, only the first error will be identified. COPY will continue checking the remaining pages, but will not copy them, in the table space or index space after it finds an error.

When the error has been detected, the object table space or index space will not be put in COPY pending state. Instead, a return code of 8 will be issued.

When COPY has detected a broken page, and all the following pages have been checked, a new SYSCOPY record will be written with an ICTYPE of 'T', an STYPE of ‘F’ (full), or ‘I” (incremental), and a TTYPE of 'B' (for broken page) for the table space or index space.

This entry will prevent subsequent incremental image copies from being taken. Incremental copies make no sense at this point because the previous full copy that encountered a checkpage error has not copied the full table space to the image copy. Furthermore, it has marked as not changed the pages that were already copied. So a subsequent incremental copy would rely on these pages already being available in a valid backup, but they are not available in a backup. Hence the need to suppress incremental copies.

Users are recommended to identify COPY utilities that terminate with a return code of 8 and fix the underlying problem as soon as practical.

11.7.2 SCOPE PENDING for COPY

The COPY utility has been enhanced to optionally copy only objects that are in copy pending or informational copy pending state by specifying the SCOPE PENDING option. See Figure 11-13. This could be very powerful when used with the LISTDEF support.

Note that when DSNUM ALL is specified, the entire table space or index space will be copied if one or more partitions are in copy or information copy states. To copy just those parts that are copy or informational copy pending, then this can be achieved using LISTDEF and the PARTLEVEL option.

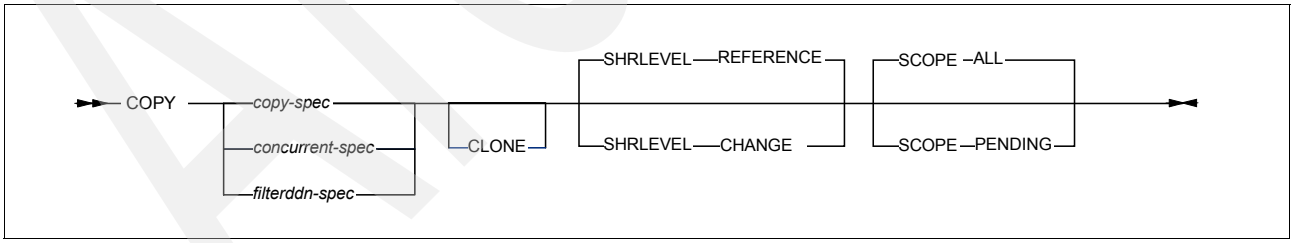


Figure 11-13 SCOPE syntax

11.8 TEMPLATE switching

The new template switching function allows image copies of varying sizes to have different characteristics. This provides significant flexibility in terms of the data set names and attributes, for example, device types.

Template switching is available for image copies produced by the following utilities:

- COPY
  - FULL YES/NO
  - CONCURRENT
- COPYTOCOPY
- MERGECOPY
- LOAD
- REORG

Both COPYDDN and RECOVERYDDN can support the switching of templates.

Template switching is controlled by the new template keyword LIMIT. The LIMIT keyword has two mandatory operands. The first is the maximum primary allocation that is permitted for the template. The second is the new template to be used when this maximum is reached. Note that you can only switch templates once.

In Example 11-2, if MY.SMALLTS table space is 20 cylinders in size, then it can be accommodated within the small template, specified by the COPYDDN keyword. The image copy data set is allocated on disk. If the MY.LARGETS table space is a 1000 cylinders, then it exceeds the limit specified by the COPYDDN keyword (small) and is switched to the large template. The image copy data set is then allocated on tape.

Example 11-2 Use of keyword LIMIT in TEMPLATE

```
//SYSIN DD *
TEMPLATE small DSN &DB..&TS..IC.D&DA..T&TI. UNIT=DASD LIMIT(100 CYL,large)
TEMPLATE large DSN &DB..&TS..IC.D&DA..T&TI. UNIT=TAPE
COPY TABLESPACE MY.SMALLTS COPYDDN(small)
COPY TABLESPACE MY.LARGETS COPYDDN(small)
```

11.9 MODIFY RECOVERY

The MODIFY RECOVERY utility is used to remove old entries from both SYSIBM.SYSCOPY and SYSIBM.SYSLGRNX. Removing these redundant entries saves disk space and improves performance.

Note: Besides removing old entries with MODIFY RECOVERY, also REORG both SYSCOPY and SYSLGRNX on a regular basis for best performance and to minimize lock contention.

Archived

Prior to V9 the user specified the deletion criteria either as before a specific date or by greater than a given age in days. V9 has an alternative, by which instead of deletion criteria, retention criteria can be specified. The new keyword RETAIN is added to the MODIFY RECOVERY syntax. See Figure 11-14.

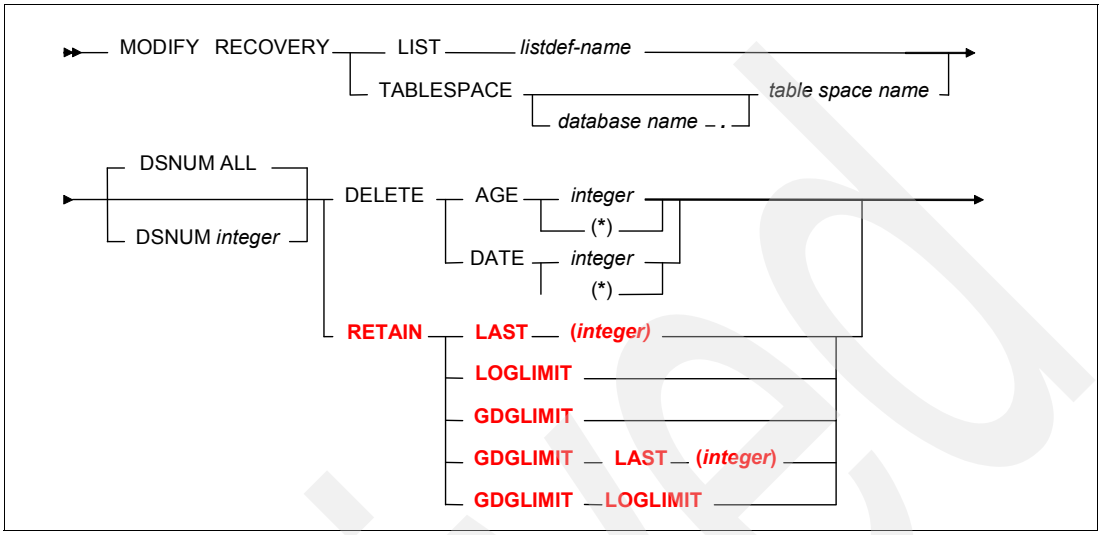


Figure 11-14 Modify recovery syntax

The RETAIN keyword has five possible settings. Note that for all these settings DB2 establishes a date before which entries can be deleted. In effect, DB2 chooses the most recent date that would satisfy the retention criteria. If there is more than one copy on that date then the result may be DB2 keeping additional copies. When the criteria is related to the number of copies, then DB2 only considers primary full copies (ICTYPE='F' and ICBACKUP=blank, in SYSCOPY).

- LAST (integer)

  This simply tells DB2 to select a date such that the number of full image copies kept is not less than the integer specified. If the integer is 10 and the tenth oldest copy is on a given date, then all entries prior to this date are deleted. Should the eleventh oldest copy be on the same day as the tenth oldest copy, it is retained.

- LOGLIMIT

  This option is used to delete copies for which DB2 no longer has log records to recover forward from. DB2 uses the BSDS to establish the oldest archive log time stamp. In a data sharing environment the BSDS from all members are checked, and value used is then the lowest of all the members (that is, the oldest). Users should be aware of this if they have a dormant member or a member that logs at a significantly different rate than other members in the group.

- GDGLIMIT

  This option allows the number of copies retained to be linked to the corresponding generation data group (GDG) base limit. DB2 checks the GDG base to determine the number of copies to keep. DB2 uses the GDG base associated with the last primary full image copy. Note that if the last primary full image copy is not a GDG, then *no action* is performed. DB2 chooses a date to use as a range for delete based on the GDG base limit and copies associated with this GDG only. If there are entries associated with a different GDG base or non-GDG entries, then DB2 ignores these in choosing the date.

Archived

- GDGLIMIT LAST (integer)

  This variation is a combination of the GDGLIMIT and LAST options discussed earlier. DB2 uses the GDG base limit if the last primary copy is a GDG. If not, it uses the integer specified. This is useful when both GDGs and non-GDGs are used and when using LISTDEF.

- GDGLIMIT LOGLIMIT

  This combines the GDGLIMIT and LOGLIMIT options previously detailed. If the most recent primary full copy is a GDG then the GDG limit is used. If the last copy is not a GDG then the BSDS is used to determine the date prior to which deletions will occur. This is useful when both GDGs and non-GDGs are used and when using LISTDEF.

An example of the SYSCOPY entries for a table space is shown in Figure 11-15.

```
---------+---------+---------+---------+---------+------
TSNAME    ICTYPE  ICDATE  ICTIME  DSNAME
---------+---------+---------+---------+---------+------
ROYTS3    F       070301  180335  PAOLOR9.COPYA3
ROYTS3    F       060918  191406  PAOLOR9.COPY341
ROYTS3    F       060912  134529  PAOLOR9.ROY33
ROYTS3    F       060912  134513  PAOLOR9.ROY32
ROYTS3    F       060911  191826  PAOLOR9.COPY.G0018V00
```

Figure 11-15 SYSCOPY entries ahead of MODIFY

We now run MODIFY to retain the GDG limit or the last three copies. The control cards for MODIFY are shown in Figure 11-16.

```
//SYSIN DD *
  MODIFY RECOVERY TABLESPACE ROY.ROYTS3
  RETAIN GDGLIMIT LAST  3
```

Figure 11-16 MODIFY RECOVERY with RETAIN LIMIT and GDG

In this example the most recent copy is not a GDG, so the GDG is not considered, even though an earlier copy was a GDG. The last three copies are kept. However, the fourth one was taken on the same day as the third (060912). The date chosen by MODIFY is 060912 (referred to as the *cleanup point* in the job output in Figure 11-17). Therefore, four out of the five SYSCOPY entries are retained.

```
DSNU000I    060 18:15:34.36 DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = MOD
 DSNU1044I   060 18:15:34.40 DSNUGTIS - PROCESSING SYSIN AS EBCDIC
 DSNU050I    060 18:15:34.40 DSNUGUTC -  MODIFY RECOVERY TABLESPACE ROY.ROYTS3 RETAIN GDGLIMIT LAST 3
 DSNU697I  -DB9B 060 18:15:34.42 DSNUMDEL - CLEANUP POINT 20060912 HAS BEEN DETERMINED
 DSNU517I  -DB9B 060 18:15:34.43 DSNUMDEL - SYSCOPY RECORD DELETED BY MODIFY UTILITY.
                     DSN=PAOLOR9.COPY.G0018V00, VOL=SER=(CATLG), FILESEQNO=0
 DSNU575I  -DB9B 060 18:15:34.45 DSNUMODA - MODIFY COMPLETED SUCCESSFULLY
 DSNU010I    060 18:15:34.45 DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

Figure 11-17 Job output from MODIFY

Archived

The confirmation from SYSCOPY entries is shown in Figure 11-18.

```
---------+---------+---------+---------+---------+-
TSNAME    ICTYPE  ICDATE  ICTIME  DSNAME
---------+---------+---------+---------+---------+-
ROYTS3    M       070301  181534  ROY.ROYTS3
ROYTS3    F       070301  180335  PAOLOR9.COPYA3
ROYTS3    F       060918  191406  PAOLOR9.COPY341
ROYTS3    F       060912  134529  PAOLOR9.ROY33
ROYTS3    F       060912  134513  PAOLOR9.ROY32
```

Figure 11-18 SYSCOPY entries after of MODIFY

Note the M record to indicate that MODIFY RECOVERY has run.

11.10 DSN1LOGP message enhancement

If a range is specified for DSN1LOGP to print, but the entire range is no longer recorded in the BSDS because, for example, the archive logs have rolled off, then there is no warning in DB2 V8. This can lead to confusion and hinder a recovery situation. DSN1LOGP has been enhanced to detect this situation and give a RC 4 along with the message shown in Example 11-3.

Example 11-3 New message from DSN1LOGP

```
DSN1224I csect-name SPECIFIED LOG type nnnnnnnnnnn COULD NOT BE
                      LOCATED FOR MEMBER mbn
```

Any missing archive logs can be provided by the ARCHIVE DD card in the DSN1LOGP JCL.

11.11 Online CHECK DATA

When integrity issues are suspected, the CHECK DATA utility can be run to validate the table or tables concerned. CHECK DATA looks at table check constraint, referential constraints, or compare LOB table spaces with the base table space. In V8 this reduces the access to the table or table spaces to read only for the duration of the utility executing. For large tables or complex RI environments, this can result in a significant outage of full access to the data.

If CHECK DATA finds a violation, then the table space is placed into CHECK pending state. This makes all the data unavailable even though as little as one row may have an issue. CHECK DATA resets CHECK-pending status if it finds no errors, or if the option is selected, all rows that contain violations were copied to exception tables and deleted.

In DB2 V9, CHECK DATA is enhanced to allow the new CLONE, LOBERROR, and XMLERROR options.

- CLONE

  Indicates that CHECK DATA is to check the clone table in the specified table space. Because clone tables cannot have referential constraints, the utility checks only constraints for inconsistencies between the clone table data and the corresponding LOB data. If you do not specify CLONE, CHECK DATA operates against only the base table.

- LOBERROR

  Specifies the action that CHECK DATA is to perform when it finds a LOB column check error. LOBERROR should not be specified if AUXERROR is specified. If both are specified, the keywords must match. LOBERROR is ignored for SCOPE XMLONLY since LOB checking is not being performed.

- XMLERROR

  Specifies the action that CHECK DATA is to perform when it finds an XML column check error. XMLERROR should not be specified if AUXERROR is specified. If both are specified, the keywords must match. XMLERROR is ignored for SCOPE XMLONLY since LOB checking is not being performed.

In DB2 V9, CHECK DATA is also enhanced to run in a SHRLEVEL CHANGE mode. The V8 processing is the default or it can be specified as SHRLEVEL REFERENCE. SHRLEVEL CHANGE CHECK DATA works on a copy of the data and indexes (shadows). The copy is taken by DB2 using the DFSMS ADRDSSU utility. For installations with data set level FlashCopy V2 enabled, this is extremely fast. Without the FlashCopy support the outage is still less with SHRLEVEL CHANGE. Note the utility must be able to drain the objects ahead of the copy. DRAIN WAIT options are available, similar to REORG. See Figure 11-19.

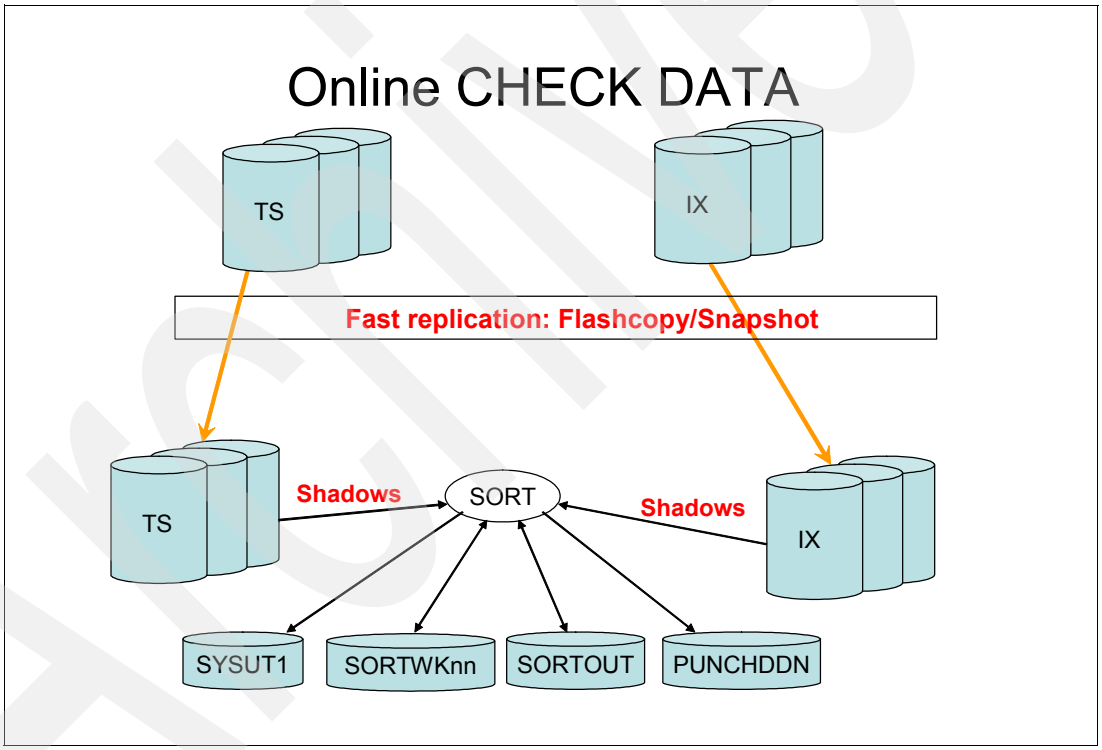


Figure 11-19 Online CHECK DATA

The integrity checks are now performed against the shadow copies of the data and indexes. If the utility detects violations it does not set the CHECK pending state, thus avoiding the removal of access from the whole table space. If SHRLEVEL CHANGE is run against objects already in a pending state and no problems are found, then REPAIR SET NOCHECKPEND or REPAIR SET NOAUXWARN must be run to remove these.

As CHECK DATA SHRLEVEL CHANGE runs against copies of the tables. Deletion of any rows with violations cannot be performed. The original table is in full read/write, and unpredictable results could arise. Instead, CHECK DATA writes to the PUNCHDDN, REPAIR

LOCATE DELETE cards that can be subsequently run against real table spaces. See also 11.13, “REPAIR LOCATE SHRLEVEL CHANGE” on page 410.

11.12 Online CHECK LOB

When CHECK LOB is run in V8, access is restricted to read only for the whole LOB table space for the duration of the utility. Due to their inherent size, this could be a significant amount of time. CHECK LOB also puts the LOB table space in a restrictive state, CHECK-pending or auxiliary-warning.

CHECK LOB has been enhanced to run in a SHRLEVEL CHANGE mode, in addition to the default SHRLEVEL REFERENCE (same as V8).

SHRLEVEL CHANGE CHECK LOB works on a copy of the LOB table space (shadow). The copy is taken by DB2 using the DFSMS ADRDSSU utility. For installations with data set level FlashCopy2 enabled, this will be used. The utility drains LOB table space before the copy. DRAIN WAIT logic is available, similar to that of REORG. See Figure 11-20.

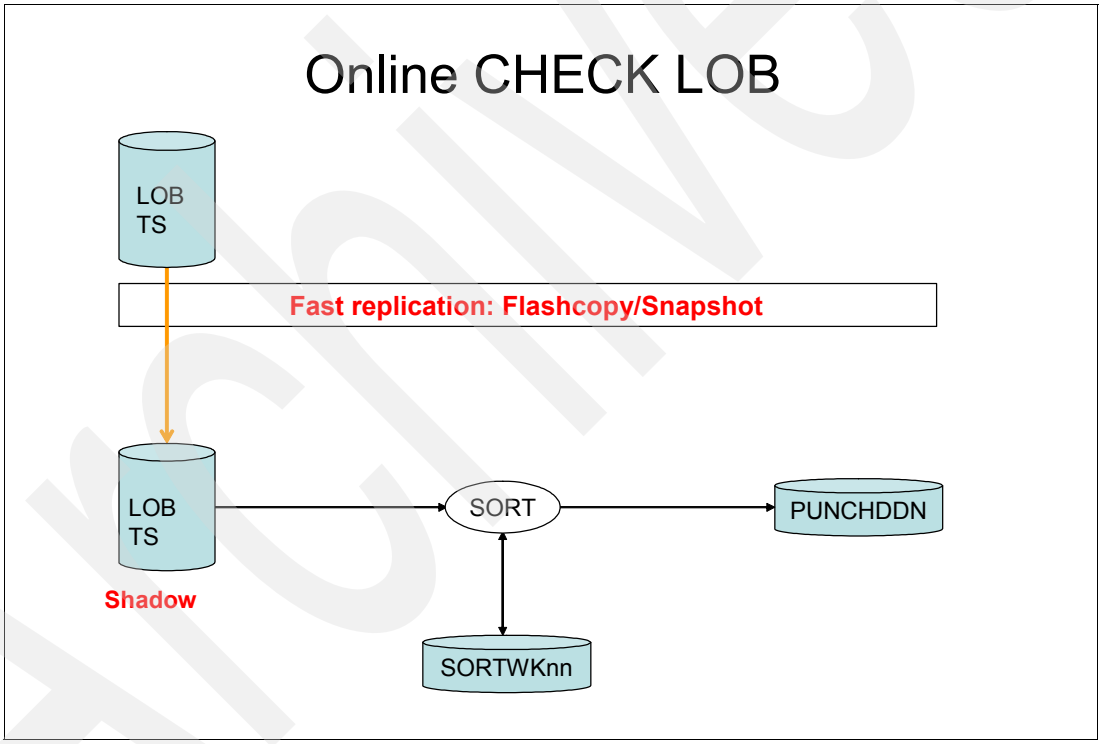


Figure 11-20 Online CHECK LOB

The shadow LOB table space is analyzed. As this is done against a copy, REPAIR statements are generated for the delete of the LOBs reported in error messages for the LOB table space.

REPAIR must be used to reset any CHECK pending or auxiliary warning states.

Archive

11.13 REPAIR LOCATE SHRLEVEL CHANGE

The REPAIR utility has been enhanced so that LOCATE can be run against indexes, index spaces, and table spaces with SHRLEVEL CHANGE. This does not apply to LOB table spaces.

If SHRLEVEL CHANGE is omitted, then the utility does not permit readers nor writers when a DELETE or REPLACE is specified. If SHRLEVEL CHANGE is not specified and only DUMP or VERIFY is used, then readers are permitted.

11.14 DB2I and DSNU CLIST enhancement

The DB2 Utilities Panel in DB2I has been enhanced to allow the SDSNLOAD library to be specified. This is of particular interest when more than one DB2 subsystem exists on an LPAR. See Figure 11-21.

```
DB2 UTILITIES                    SSID: DB9B
===>

Select from the following:

 1 FUNCTION ===> EDITJCL           (SUBMIT job, EDITJCL, DISPLAY, TERMINATE)
 2 JOB ID   ===> TEMP              (A unique job identifier string)
 3 UTILITY  ===> COPY              (CHECK DATA, CHECK INDEX, CHECK LOB,
                                    COPY, DIAGNOSE, LOAD, MERGE, MODIFY,
                                    QUIESCE, REBUILD, RECOVER, REORG INDEX,
                                    REORG LOB, REORG TABLESPACE, REPORT,
                                    REPAIR, RUNSTATS, STOSPACE, UNLOAD)
 4 STATEMENT DATA SET ===> UTIL

 5 RESTART  ===> NO                (NO, CURRENT, PHASE or PREVIEW)

 6 LISTDEF? (YES|NO) ===> NO       TEMPLATE? (YES|NO) ===> NO

 7 LIB ==>                                (BLANK or DB2 Library name).

 * The data set names panel will be displayed when required by a utility.

PRESS:  ENTER to process     END to exit    HELP for more information
```

Figure 11-21 DB2I utilities panel

For users of the DSNU CLIST, the LIB(data set name) parameter can be used to specify the SDSNLOAD library.

11.15 Online REBUILD INDEX

Users often create indexes with DEFER YES, particularly on existing large populated tables, to avoid large sorts using the workfile database. The REBUILD INDEX utility is then used to finish the creation of the index.

Even with the more efficient process of the REBUILD utility, this can still take a significant amount of time when the table involved is very large. During this time the utility has the table drained of writers, so access is denied to updating applications.

DB2 V9 has an enhancement to REBUILD INDEX so that it can run SHRLEVEL CHANGE. The REBUILD utility will now have a LOGPHASE when SHRLEVEL CHANGE is specified. There are DRAIN WAIT options similar to REORG to control the final drain of writers before the index can be made available. The existing V8 behavior is now referred to as SHRLEVEL REFERENCE and is the default.

Note: REBUILD INDEX is a function designed to create new indexes or recreate an index in recovery pending status. Do not use REBUILD INDEX to move the index to a new location. It is more efficient to use REORG INDEX.

11.16 Time stamps in messages

For operational clarity, all utility messages now include the day number from the Julian date, plus a time stamp. See Example 11-4.

Example 11-4 Utility message with day and time stamp highlighted

```
DSNU010I     242 18:13:37.30 DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

In this example the utility completion message was issued just after 18.13 on day 242.

11.17 Large format data sets

Large format data sets are greater than 65,535 tracks per disk volume. Once in New Function Mode, large format data sets are automatically supported when they are input data sets to utilities. They are supported as output for utilities if the DD card in V9 NFM specifies table spaces that are created with DSNTYPE=LARGE.

11.18 Support of large block interface

For tape data sets the large block interface is now supported (greater than 32760 bytes). This can significantly improve COPY and the restore phase of RECOVER.

11.19 DSN1COPY RESET enhancement

This is when a compressed table space is be copied from a non-data sharing DB2 subsystem to another non-data sharing DB2 subsystem by DSN1COPY RESET. DSN1COPY copies the data and resets the PGLOGRBA of each page, but it does not reset the dictionary version. The dictionary version still reflects the RBA of the source subsystem and can be much higher than the target subsystem.

There is a work around for V7 and V8, which is to REORG the copied table space with KEEPDICTIONARY=NO in the target subsystem.

To solve this problem DSN1COPY RESET is changed to reset the dictionary version, in V9.

Note that the issue does not arise in data sharing as the LRSN is used, rather than an RBA.

11.20 DSNJU004

The print log map utility (DSNJU004) runs as a batch job and enables you to print the bootstrap data set contents.

Example 11-5 shows the output for the checkpoint queue of a DB2 9 subsystem. Notice the addition of the store clock information.

Example 11-5 Checkpoint queue output

```
CHECKPOINT QUEUE
               09:21:33 JUNE 07, 2007
TIME OF CHECKPOINT       09:20:05 JUNE 07, 2007
BEGIN CHECKPOINT RBA            000012006E40
END CHECKPOINT RBA              000012009799
END CHECKPOINT STCK             C0B5DB450D4A
TIME OF CHECKPOINT       19:17:31 JUNE 06, 2007
BEGIN CHECKPOINT RBA            000011FDD090
END CHECKPOINT RBA              000011FDF641
END CHECKPOINT STCK             C0B51EF109DD
```

11.21 Enhancements to V8

The following functions are included in the code of the V9 base, but have also been made available to V8 via APARs.

11.21.1 CHECK INDEX SHRLEVEL CHANGE

The original V8 CHECK INDEX restricts data access to read only for the duration of utility. This can be a significant reduction in availability when large or large numbers of indexes require checking.

SHRLEVEL CHANGE support has been added. DFSMSdss™ ADRDSSU is used to copy the table space and indexes to shadow data sets, having temporary drained writers. This will use data set level FlashCopy V2 if available. On completing copy or logical copy, read / write access is returned to the applications.

Archived

Integrity checking is done on the shadows. A further enhancement allows parallel checking of indexes. This is done in a similar fashion to the parallelism used in REBUILD INDEX.

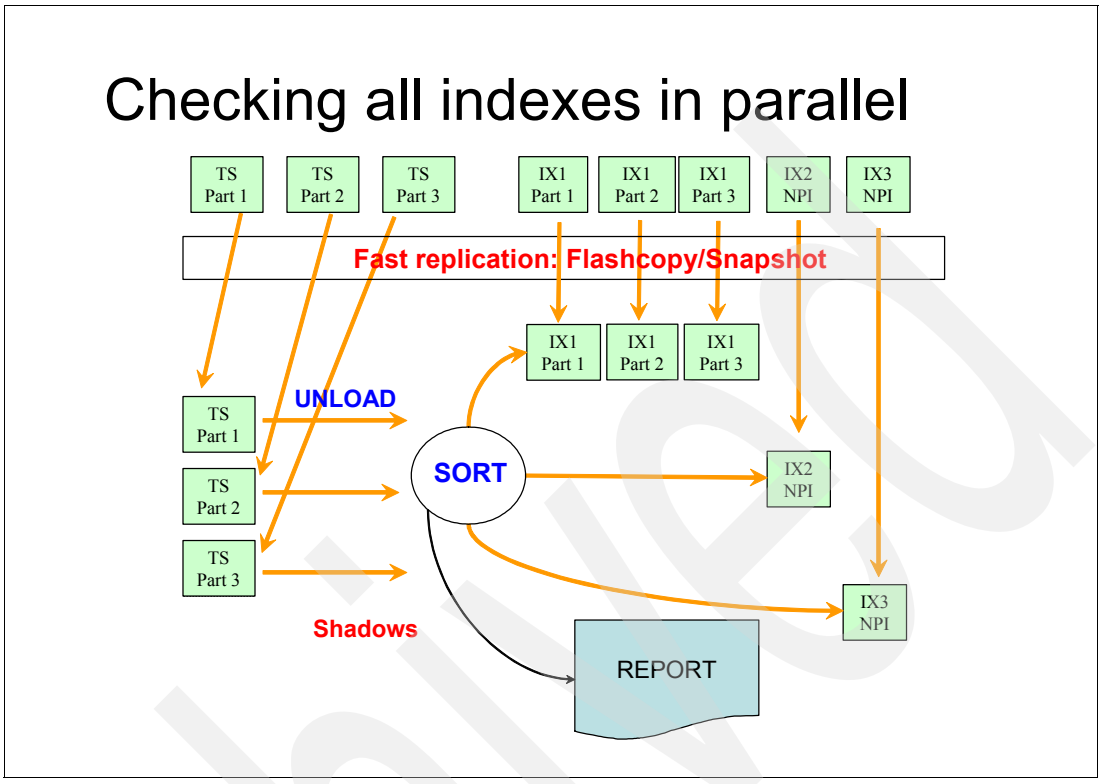


Figure 11-22 CHECK INDEX with parallel index checking

11.21.2 Cross loader support for LOBs

The cross loader capability of LOAD has been enhanced for handling LOBs. Previously, the processing of LOB columns was limited to 32767 bytes. Now restricted by memory above the 16 Mb line, consider increasing region size.

This enhancement is via APAR PQ90263 for both V7 and V8. It requires recycle of DB2 to become active.

Cross-loader also allows for conversion between CLOBs and DBCLOBs (not currently supported when loaded from file).

Archive

11.21.3 LOAD and UNLOAD of LOBs

LOAD and UNLOAD have been changed to allow LOBs to be loaded and unloaded into files (members in PDS, PDS/E, or HFS). The data set information is passed to LOAD in the control statements. See Figure 11-23.

```
//SYSREC   DD *
"000001","UN.DB1.TS1.RESUME(AI3WX3JT)","UN.DB1.TS1.PHOTO(AI3WX3JT)"
"000002","UN.DB1.TS1.RESUME(AI3WX5BS)","UN.DB1.TS1.PHOTO(AI3WX5BS)"
"000003","UN.DB1.TS1.RESUME(AI3WX5CC)","UN.DB1.TS1.PHOTO(AI3WX5CC)"
"000004","UN.DB1.TS1.RESUME(AI3WX5CK)","UN.DB1.TS1.PHOTO(AI3WX5CK)"


 LOAD DATA FORMAT DELIMITED
      INTO TABLE MY_EMP_PHOTO_RESUME
           (EMPNO    CHAR,
            RESUME   VARCHAR CLOBF,
            PHOTO    VARCHAR BLOBF)
```

new syntax

Figure 11-23 LOAD from reference variables

Similarly, UNLOAD places the LOBs in data sets. See Figure 11-24. Note that it is always one LOB per member or file. The output of UNLOAD provides details of the file that the LOBs are unloaded to.

UNLOAD is changed to store the value of a LOB column in a file and record the name of the file in the unloaded record of the base table.

```
TEMPLATE LOBFRV1 DSN 'UN.&DB..&TS..RESUME' DSNTYPE(PDS) UNIT(SYSDA)
TEMPLATE LOBFRV2 DSN 'UN.&DB..&TS..PHOTO'  DSNTYPE(PDS) UNIT(SYSDA)

UNLOAD DATA
FROM TABLE DSN8910.EMP_PHOTO_RESUME
(EMPNO  CHAR(6),
 RESUME VARCHAR(255)  CLOBF  LOBFRV1,
 PHOTO  VARCHAR(255)  BLOBF  LOBFRV2) DELIMITED
```

Creates PDS with member for each LOB

new syntax

Output:

```
"000001","UN.DB1.TS1.RESUME(AI3WX3JT)","UN.DB1.TS1.PHOTO(AI3WX3JT)"
"000002","UN.DB1.TS1.RESUME(AI3WX5BS)","UN.DB1.TS1.PHOTO(AI3WX5BS)"
"000003","UN.DB1.TS1.RESUME(AI3WX5CC)","UN.DB1.TS1.PHOTO(AI3WX5CC)"
"000004","UN.DB1.TS1.RESUME(AI3WX5CK)","UN.DB1.TS1.PHOTO(AI3WX5CK)"
…
```

Figure 11-24 Unload using reference variables

11.21.4 Automatic display of blocking claimers for REORG

REORG has been enhanced, so if it cannot drain objects it automatically lists the claimers. See Figure 11-25.

```
DSNUGUTC -  REORG TABLESPACE DB1.TS1 COPYDDN(SYSCOPY) SHRLEVEL NONE STATISTICS
DSNUGBAC - RESOURCE UNAVAILABLE
           REASON 00C200EA
           TYPE 00000200
           NAME DB1.TS1


DSNUGUTC -  REORG TABLESPACE DB1.TS1 COPYDDN(SYSCOPY) SHRLEVEL NONE STATISTICS
) **********************************
) * DISPLAY DATABASE SUMMARY
  *   GLOBAL CLAIMERS
) **********************************
…
NAME     TYPE PART  STATUS             CONNID   CORRID      CLAIMINFO
-------- ---- ----- ------------------ -------- ------------ --------
TS1      TS          RW,UTRO            BATCH    CAPP         (WR,C)
    -                   AGENT TOKEN 10
    -                   MEMBER NAME V91A
TS1      TS          RW,UTRO            BATCH    CAPP         (RR,C)
    -                   AGENT TOKEN 10
    -                   MEMBER NAME V91A
******* DISPLAY OF DATABASE DB1 ENDED **********************
) DSNTDDIS 'DISPLAY DATABASE' NORMAL COMPLETION
  DSNUGBAC - RESOURCE UNAVAILABLE
             REASON 00C200EA
             TYPE 00000200
             NAME DB1.TS1
```

Culprits

Figure 11-25 Claimers shown by REORG

When an online REORG cannot drain an object and the user has DISPLAYDB authority on the database containing the object, REORG will generate the command:

```
DISPLAY DATABASE(database-name) SPACENAM(space-name) CLAIMERS
```

The messages for the DISPLAY command will be written to the job output file before the DSNT500I message for the resource unavailable. The DISPLAY messages are shown before the RESOURCE UNAVAILABLE message in order to issue the DISPLAY as close as possible to the point of failure.

11.21.5 Online REORG enhancements

OLR has been improved to reduce the impact of the switch phase with COPY NO indexes. This involves the elimination of the need to update header pages.

A further enhancement has been to allow the utility status during the UTILTERM phase to become UTRW. This allows for improved concurrency.

Archived

12

Installation and migration

In this chapter we provide you with enough information for you to start evaluating the changes required by DB2 Version 9.1 for z/OS and to be able to start planning for installation or migration to this new release. We discuss the following topics:

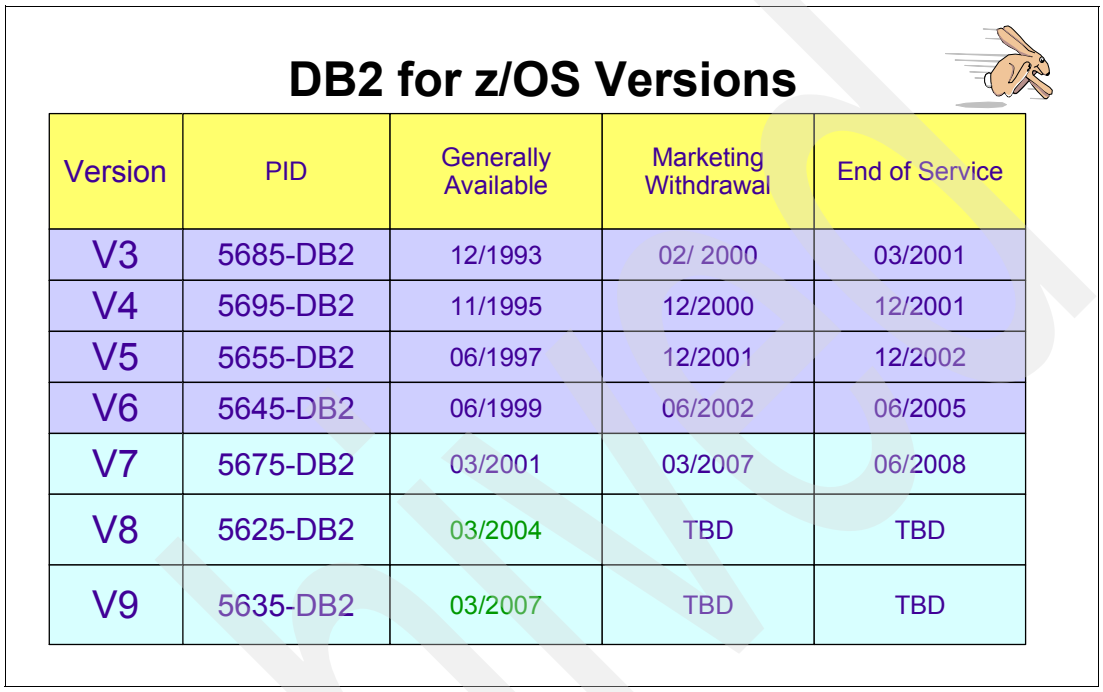
- Currency of versions and migration paths
- DB2 9 packaging
- Changes to installation and migration
- Installation and migration overview
- Migration and coexistence
- Fallback
- Other migration considerations
- Samples

Archived

© Copyright IBM Corp. 2007. All rights reserved.

12.1 Currency of versions and migration paths

Before you begin the installation/migration process, you should look at the big picture. You need to be aware of the major requirements to get to DB2 V9. You need to know where you are currently and where you need to be before you embark on this process. Figure 12-1 shows the life cycle of the versions of DB2 that are currently available.



Version	PID	Generally Available	Marketing Withdrawal	End of Service
V3	5685-DB2	12/1993	02/ 2000	03/2001
V4	5695-DB2	11/1995	12/2000	12/2001
V5	5655-DB2	06/1997	12/2001	12/2002
V6	5645-DB2	06/1999	06/2002	06/2005
V7	5675-DB2	03/2001	03/2007	06/2008
V8	5625-DB2	03/2004	TBD	TBD
V9	5635-DB2	03/2007	TBD	TBD

Figure 12-1 DB2 Versions

You can check the following Web site for the latest updates to these dates:

http://www.ibm.com/software/data/support/lifecycle/

Archive

Are you still on DB2 V7? If so, you will need to migrate to DB2 V8 new-function mode first. This can add a considerable amount of time to your project plan. What functions do you need from z/OS or hardware? Do you need to migrate z/OS 1.7 first? If you do, add another six months to a year to your plan. In Figure 12-2 you can see the versions of z/OS that are currently available.

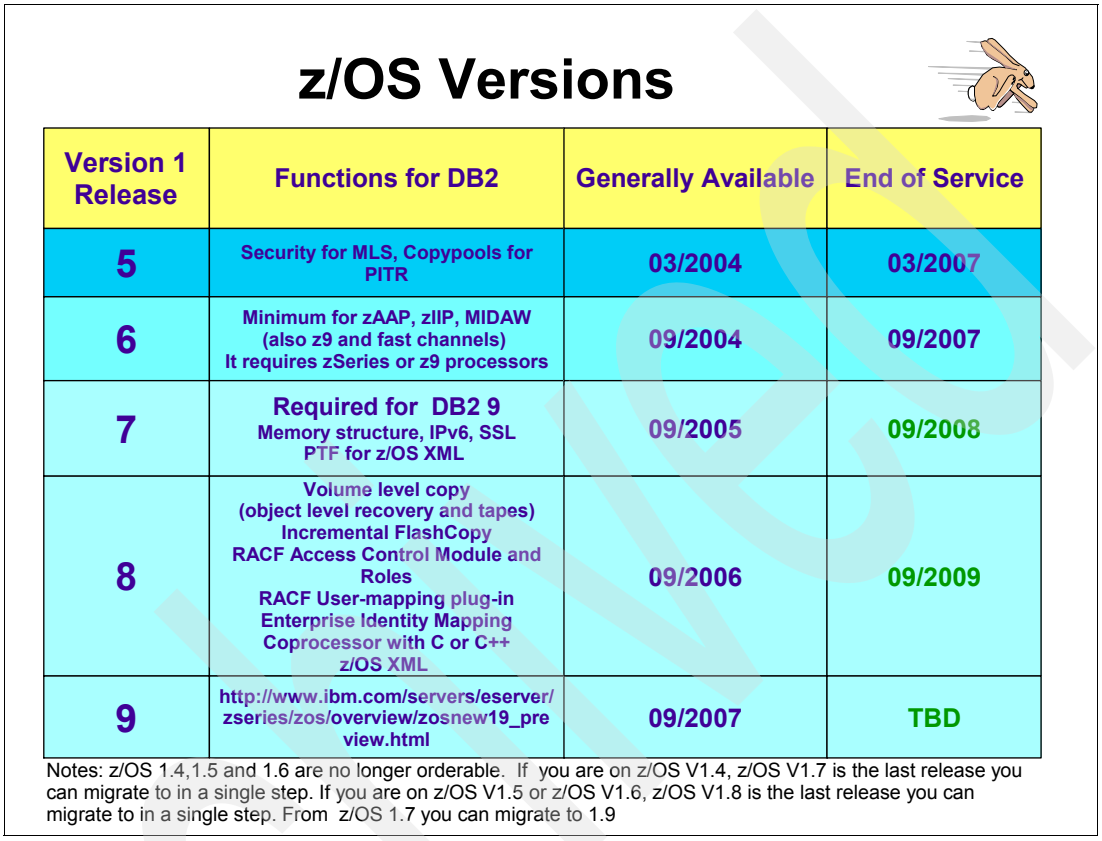


Figure 12-2 z/OS versions

Before you start to migrate your DB2 subsystems to V9, make sure that you are aware of the changes that will affect your new system. Below is a list of action items that you can start on now and that need to be completed in preparation for your migration. We recommend that these items be completed well in advance of your DB2 V9 migration project.

z/OS migration warning on RACF profiles

This warning is applicable when moving to or across z/OS 1.6. For details see *z/OS Migration Version 1 Release 7*, GA22-7499-08.

If you use RACF to protect connections to DB2 subsystems, you must have the DSNR class active and you might have to create additional RACF profiles in the DSNR class. Otherwise, connections to some DB2 subsystems might fail.

This change was introduced with z/OS V1R6 and it applies to migration from z/OS V1R5 and z/OS V1R4, before installing z/OS V1R7, if you use RACF to protect connections to any release of DB2 subsystems. The steps are:

1. Activate the DSNR class to activate RACF protection for DB2 connections.

   If you already have the DSNR class active, you can skip this step. To see whether you have the DSNR class active, issue the SETROPTS LIST command and look for DSNR

after the heading “ACTIVE CLASSES =”. If you wish to use RACF to protect connections to DB2, you must have the DSNR class active.

Starting with z/OS V1R6, if the DSNR class is inactive, connections are allowed to all DB2 subsystems from all environments. When the DSNR class is active, the profiles in the DSNR class determine whether a connection to a DB2 subsystem is allowed. To activate the DSNR class, issue the following RACF command:

```
SETROPTS CLASSACT(DSNR)
```

Before z/OS V1R6, if you had the DSNR class inactive, a connection to DB2 could be allowed or denied, depending on whether the DB2 subsystem was defined in the RACF router table. Starting with z/OS V1R6, the entry in the RACF router table is no longer required. If the DSNR class is inactive, connections are allowed to all DB2 subsystems from all environments. If the DSNR class is active, the profiles in the DSNR class determine whether the connection is allowed.

2. Create additional RACF profiles in the DSNR class to protect all DB2 subsystems.

   If you have the DSNR class active and you have all DB2 subsystems defined in the RACF router table (ICHRFR01) with an entry containing ACTION=RACF on a z/OS system before V1R6, you can skip this step.

   If your DB2 installation already uses DSNR protection to control access to some DB2 subsystems, you need to ensure that there are RACF profiles in the DSNR class to control access to all DB2 subsystems.

   In releases before z/OS V1R6, if you did not have an entry in the RACF router table (ICHRFR01) for a subsystem, RACF authorization checking was bypassed for that subsystem. Any user who tried to access that DB2 subsystem from any environment was allowed.

   Starting with z/OS V1R6, the router table entry is no longer required, so any user who tries to access that DB2 subsystem might or might not be allowed, depending on which RACF profiles are defined in the DSNR class. If you depended on the missing router table entry to allow access, starting with z/OS V1R6 you must create appropriate profiles in the DSNR class to allow access.

   For example, if you had no entry in the router table for subsystem DSND and have no profiles defined in the DSNR class that begin with “DSND”, starting with z/OS V1R6 you might receive error messages at DB2 startup when the DSNDSPAS address space attempts to connect to DB2, such as the following:

```
IEF403I DSNDSPAS -STARTED -TIME=15.31.42 DSNX980E DSNX9STP ATTEMPT TO PERFORM
CALL ATTACH FUNCTION CONNECT 047 FAILED WITH DB2 SUBSYSTEM DSND. RETURN CODE
=00000008 REASON CODE =00F30013 +DSNX965I DSNX9STP THE DB2-ESTABLISHED STORED
PROCEDURES ADDRESS 048 SPACE FOR SUBSYSTEM DSND IS STOPPING
```

DB2 V8 in NFM and z/OS 1.7

You can only migrate to DB2 V9 from a DB2 V8 system in new-function mode (NFM). DB2 V9 requires z/OS to be at least at V1.7 to run and at V1.8 if you wish to take advantage of volume-level copy and object-level recovery and tape. Make sure that you are running at these release levels and have a stable environment before you attempt to migrate to DB2 V9. DB2 V9 operates on any processor that supports 64-bit z/Architecture, including System z9, z990, z890, or a comparable processor such as z800 or z900. See the DB2 Program Directory for more information about system requirements.

Use of XML support requires z/OS XML System Services (XML z/OS), which is available either with z/OS V1.8 or with z/OS V1.7 with APAR OA16303. In addition, use of XML schemas requires IBM 31-bit SDK for z/OS, Java 2 Technology Edition V5 (5655-N98) (SDK5).

New DB2 V8 pre-migration health check job

Make sure that you run premigration job DSNTIJP9 on your DB2 V8 system to do a check prior to migrating to DB2 V9. This job can be run at any time during the premigration planing stages or any time you are interested in finding out what cleanup work needs to be done to DB2 V8 in order to get it in shape for migrating to DB2 V9. It is possible that DSNTIJP9 was not delivered with your DB2 V8 system software and initial install/migration. DSNTIJP9 was delivered via the service stream in APAR PK31841. We highly recommend that you apply this APAR and run this job during your early stages of DB2 9 migration planning.

Fall back SPE on DB2 V8

Make sure that the fallback SPE APAR PK11129 (PTF UK90008) is installed on all DB2 V8 subsystems and all members of the data sharing group to be migrated. You will need to ensure that each subsystem and all members of the data sharing group have been cycled in order to load the new code. DB2 stores this information in the BSDS and it is checked at startup time.

Migrating to DB2 V9 and fallback to DB2 V8 is only supported from DB2 V8 NFM with toleration and co-existence maintenance applied. Before attempting to migrate to DB2 V9 the DB2 V8 fallback SPE must be applied to the subsystem and the subsystem started. Without the DB2 V8 fallback SPE and subsystem startup the attempt to migrate to DB2 V9 will fail and the migration attempt will be terminated.

At DB2 startup time the code level of the DB2 subsystem will be compared to the code level required by the current DB2 catalog. If there is a code level mismatch between the two then the DSNX208E message is issued and the DB2 startup will be terminated:

```
DSNX208E csect-name ATTEMPTING TO START DB2 WITH AN UNSUPPORTED CODE LEVEL. REASON
CODE reason MISSING MAINTENANCE: maint CURRENT LEVEL current-code-level VALID
LEVEL(S) valid-code-levels
```

The same process is done for DB2 subsystems in a data sharing group with the addition that the code level check is also done to the participating DB2 subsystems that are started. If the starting system has a code level mismatch with the catalog or any of the DB2s that are running, then a message is issued and the subsystem startup is terminated. The startup termination message will be DSNX208E or DSNX209E:

```
DSNX209E csect-name MEMBER member-name IS AT A CODE LEVEL WHICH IS NOT COMPATIBLE
WITH THIS STARTING OR MIGRATING MEMBER. REQUIRED LEVEL: new-code-level KNOWN
LEVELS: known-code-levels
```

We recommend that only one DB2 subsystem be started at DB2 V9 in a data sharing group for the migration processing.

Note that you may need to apply additional maintenance due to the possibility of prerequisite APARs. Refer to Informational APAR II12423 for details of prerequisite maintenance for the fallback SPE.

The purpose of this APAR allows fallback from V9 CM to V8 NFM. Refer to the chapter “Falling back and Remigration” in the *DB2 V9 Installation Guide* for this procedure. After applying this PTF, you need to re-assemble and linkedit DSNHDECP to pick up the changes in module DSNARIB. Under DB2 data sharing, if each member of the group has its own DSNHDECP module, then you need to repeat the steps below for each member.

To update DSNHDECP:

1. Create a separate job containing only the DSNTIZP and DSNTIZQ steps from the job DSNTIJUZ previously edited by the install CLIST.

2. Re-assemble and linkedit the DSNHDECP load module by running your newly created job to produce a new DSNHDECP.
3. Stop and start your DB2 V8 systems.

Convert existing BSDSs

In order to migrate to DB2 9, your BSDSs must be in the new expanded format that became available beginning with DB2 V8 NFM. This expanded format allows for support of up to 10,000 archive log volumes and up to 93 active log data sets for each copy of the log. If you are running DB2 V8 in NFM and have not yet converted to the new BSDS format, you can do so by running the supplied utility (DSNJCNVB).

Note: We strongly recommend that you back up your BSDSs before running the conversion utility.

Convert all DB2 managed SPAS to WLM SPAS

DB2-managed SPAS were deprecated in DB2 V8. In DB2 V8 you could still run pre-existing (those that existed in releases prior to DB2 V8) stored procedures in a DB2 Managed SPAS, but all new stored procedures had to be run in a WLM SPAS. Any changes to existing stored procedures required you to convert them to run in a WLM SPAS. In DB2 V9, support for DB2 Managed SPAS is removed, so you can no longer run a DB2 Managed SPAS. Before migration to DB2 V9, you will need to convert all existing stored procedures that run in DB2 Managed SPAS to run in WLM SPAS. After migration to V9 CM, stored procedures defined with NO WLM ENVIRONMENT will fail with SQLCODE -471/00E7900A.

Consider increasing the size for catalog table spaces and indexes

As you can see in Figure 12-3, the DB2 Catalog continues to increase in size with every release. For DB2 V9, we have 33% more table spaces, 18% more indexes, 30% more columns, and more than 13% more table check constraints. These numbers do not include the objects required in support of the XML repository. Although this is not as large a change as that brought about with DB2 V8 (long name support and catalog data stored in Unicode), it is still a significant change to the catalog.

DB2 catalog evolution

DB2 Version	Table Spaces	Tables	Indexes	Columns	Table Check Constraints
V1	11	25	27	269	N/A
V3	11	43	44	584	N/A
V5	12	54	62	731	46
V6	15	65	93	987	59
V7	20	84	117	1212	105
V8	**21**	**87**	**128**	**1286**	**105**
V9	**28**	**106**	**151**	**1668**	**119**

Figure 12-3 DB2 catalog evolution: V9 counts do not include objects for XML repository

An extremely disorganized catalog can cause the catalog migration process to run slower and DB2 functions not to run as efficiently as possible. A REORG of your catalog prior to any migration is always recommended. It is good business practice to REORG your catalog at least once per year. We recommend that you increase the size of the underlying VSAM Clusters for your catalog and directory before migrating to DB2 V9. You should resize the catalog objects in order to consolidate space, eliminate extents, and provide for additional space for new columns and objects and for general growth. Additionally, before and after any migration, it is good business practice to check the integrity of your catalog and directory by checking for broken links (use DSN1CHKR). You should also check catalog and directory indexes for consistency (see sample job DSNTIJCX).

Note: As with previous releases of DB2, you can run the CATMAINT job (DSNTIJTC) while continuing to operate your production environment. We recommend that you do this during a non-peak time where the amount of catalog activity is low. Stay away from time frames when a lot of object changes are taking place.

12.2 DB2 9 packaging

DB2 Version 9.1 for z/OS incorporates or offers separately several features, which include tools for data warehouse management, Internet data connectivity, database management and tuning, installation, and capacity planning.

These features and tools work directly with DB2 applications to help you use the full potential of your DB2 system. When ordering the DB2 base product, you can select the free and chargeable features to be included in the package.

You must check the product announcement (USA 207-041, dated March 6, 2007) and the program directories for current and correct information about the contents of DB2 9 for z/OS package.

12.2.1 Removed features and functions

IBM added many data server capabilities in DB2 V9.1 for z/OS and reduced or removed support for some functions. As you prepare to upgrade your subsystems to DB2 9, you should be aware of the following changes:

- AIV Extender removed.
- Text Extender removed.
- Net Search Extender removed. Note: DB2 9 incorporates substantial text function into the base product. Use the new functions and indexing for text applications.
- Net.Data® removed. WebSphere is the strategic IBM solution for delivering DB2 data to Web applications.
- DB2 QMF Visionary Studio program removed from DB2 QMF Enterprise Edition.
- DB2 Estimator, available via Web download for DB2 V8, will not be provided for DB2 V9 for z/OS.
- Visual Explain for DB2 for z/OS, available via Web download for DB2 V8, will not be provided for DB2 9. Instead, IBM Optimization Service Center for DB2 for z/OS, part of the no-charge IBM DB2 Accessories Suite for z/OS (5655-R14), extends Visual Explain's capability and adds more SQL tuning functions.
- DB2-managed stored procedure support removed. Workload Manager (WLM) managed stored procedure address spaces are the strategic solution for stored procedure support,

Archive

and migration to WLM-managed stored procedure spaces is required for use of stored procedures in DB2 V9.1 for z/OS.

- Simple table space creation support removed. DB2 V9.1 for z/OS no longer implicitly creates simple table spaces nor allows you to create simple table spaces. However, DB2 V9.1 for z/OS continues to support simple table spaces created in previous versions.
- Online help facility replaced. The previous help support has been replaced by the Information Management Software for z/OS Solutions Information Center (Information Center). The Web-based Information Center is updated periodically during the life of each DB2 version, thus ensuring reader access to the most up-to-date information.
- JDBC/SQLJ Driver for OS/390 and z/OS support removed. All Java application programs, and Java routines that are currently written to work with JDBC/SQLJ Driver for OS/390 and z/OS, need to be modified to work with IBM DB2 Driver for JDBC and SQLJ (formerly known as the DB2 Universal JDBC Driver). The steps for migrating JDBC and SQLJ applications from the JDBC/SQLJ Driver for OS/390 and z/OS to IBM DB2 Driver for JDBC and SQLJ can be found in the *DB2 Version 9.1 for z/OS Application Programming Guide and Reference for JAVA* SC18-9842. In addition, all WLM-managed stored procedures address spaces that are set up to execute Java routines must be modified to reference IBM DB2 Driver for JDBC and SQLJ.
- DB2 Precompiler Flagger support was removed in DB2 for z/OS V8, and SQLFLAG is no longer a valid precompiler option.
- Plans and packages should be rebound on current releases to benefit from optimizer enhancements and DB2 9 virtual storage constraint improvements on SQL statement storage. For DB2 9, plans and packages from DB2 V4 or earlier will be automatically rebound when accessed by DB2 9.
- As noted in the Hardware requirements section, DB2 9 for z/OS requires long displacement support in the hardware. Due to greater use of long displacement instructions by DB2 9 than DB2 V8, column-intensive applications may experience shorter CPU time. However, these applications may take longer on systems that emulate long displacement (for example, z800 or z900). For further information about the long displacement facility, refer to *z/Architecture Principles of Operations*, SA22-7832-04, available at

  `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/r8pdf/zarchpops.html`

12.2.2 Deprecated features and functions

As part of ongoing efforts to deliver the most current technology and to remove those features that no longer provide our customers with strategic benefits, the following features are deprecated in DB2 9. This means that these functions will be dropped in the future. You should not create any new dependencies that rely on them, and if you have existing dependencies on them, you should start planning to remove those dependencies.

- DB2 Management Clients feature, which includes DB2 Administration Server, DB2 Control Center, and DB2 Development Center, is replaced by the new administration services.
- Msys for Setup DB2 Customization Center is being phased out and you should continue to use the install panels.
- Simple table spaces are deprecated. You will not be able to create new simple table spaces. The DDL for simple table space will create a segmented table space with default SEGSIZE 4. You should consider converting the large simple table spaces to partitioned by growth universal table spaces. See 3.6, “Universal table space” on page 62, for a detailed description of these new table spaces.

- XML Extender is still delivered, but we recommend replacing it with the new XML data type and pureXML support built into DB2 9.
- DB2 MQ XML user-defined functions and stored procedures are replaced by new XML functions.
- There is now a new interface to load DSNHDECP. To allow you, in a potential future release of DB2, to specify a module name other than DSNHDECP to be used by a DB2 instance, in a manner similar to that used for other DB2 parameters, DB2 9 lays the ground work with these changes:
  - RRS attach provides information about which DB2 subsystems are available on an LPAR and provides the address of the DSNHDECP load module (DSNHDECP is loaded in common storage (ECSA)) being used by a DB2 subsystem.
  - FCID 160 has been enhanced to provide the fully qualified DSNHDECP load module dataset name used by a DB2 subsystem.
  - The precompiler provides the fully qualified dataset name for the DSNHDECP load module used in a precompilation.
  - GETVARIABLE has been enhanced to provide all of the information that the DECP provides.
  - The fully qualified dataset name for the DB2 subsystem parameter module, DSNHDECP load module, and exits DSN3@ATH, DSN3@SGN, and DSNX@XAC are displayed as a message on START DB2.
- Private protocol has not been enhanced since DB2 V4 and is replaced by DRDA.

12.2.3 Base engine and no charge features

In this section we list the DB2 9 base and optional features.

Archived

We summarize these features in Figure 12-4. What you need to run DB2, from an external view, is the base engine and IRLM. However, if you use Java you will need the z/OS Application Connectivity as well, and if you use tools like the Control Center, you will need the Clients Management Package. So you will have to make sure to order the correct Function Module Identifiers (FMIDs).

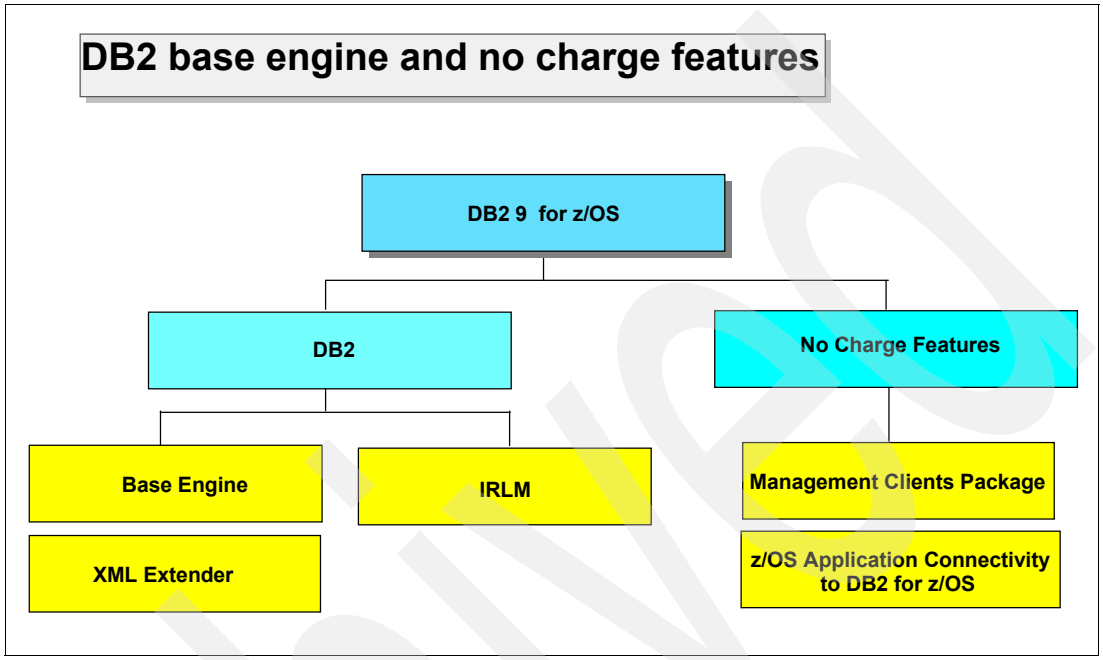


Figure 12-4 Base engine and optional no charge features

DB2 9 for z/OS

The DB2 9 for z/OS product, program number 5635-DB2, currently consists of the following FMIDs:

- Required FMIDs:
  - HDB9910 (contains DB2 Base, msys plug-in, REXX[1], MQSeries®, MQListener)
  - HIY9910 (IMS Attach)[2]
  - HIZ9910 (Subsystem Initialization)
  - HIR2220 (IRLM V2R2)
  - HDRE910 (DB2 RACF Authorization Exit)
  - JDB9914 (DB2 English Panels)
- Optional FMIDs:
  - JDB9912 (DB2 JDBC/SQLJ)
  - JDB9917 (DB2 ODBC)
  - JDB991X (DB2 XML Extender)
  - JDB9911 (DB2 Kanji Panels):

Note that DB2 Utilities Suite for z/OS Version 9.1 (program number 5655-N97) contains all of the IBM DB2 Utilities. However, all the utilities are shipped deactivated with the Base Engine. A valid separate product licence must be obtained to activate the utilities function. However, all utilities are always available for execution on the DB2 catalog and directory and the DB2 IVP objects.

[1] REXX language and REXX language stored procedure support are shipped as a part of the DB2 9 base code. The DB2 installation job DSNTIJRX binds the REXX language support to DB2 and makes it available for use.
[2] IMS Attach must be installed even if you do not have IMS.

DB2 Management Clients Package

The DB2 Management Clients Package is changed with DB2 9. This collection of workstation-based client tools is used to work with and manage your DB2 for z/OS environments. See Figure 12-5 on page 428 for its components.

The DB2 Management Clients Package consists of the following:

- Database Administration Server (DAS), *FMID HDAS910*:

  General mechanism for running z/OS level functions to support the IBM DB2 GUI Tools such as Control Center, Development Center, Replication Center, and so on.

- z/OS Enablement for DB2 Control Center, *FMID JDB991D*:

  The z/OS Enablement provides stored procedures, user-defined functions, and batch programs in the form of an SMP/E installable package.

- DB2 Connect Personal Edition Kit:

  This workstation CD-ROM kit includes:

  - IBM DB2 Control Center
  - IBM DB2 Development Center
  - IBM DB2 Replication Center
  - IBM DB2 Command Center
  - IBM Developer Workbench

  A limited-use copy of DB2 Connect 9.1 is included. This product is currently available for Linux and Windows operating systems, and is used to connect a single Windows operating system, or Linux workstation, to a host or iSeries database.

Archived

This feature is described in detail in the program directory GI10-8739-00.

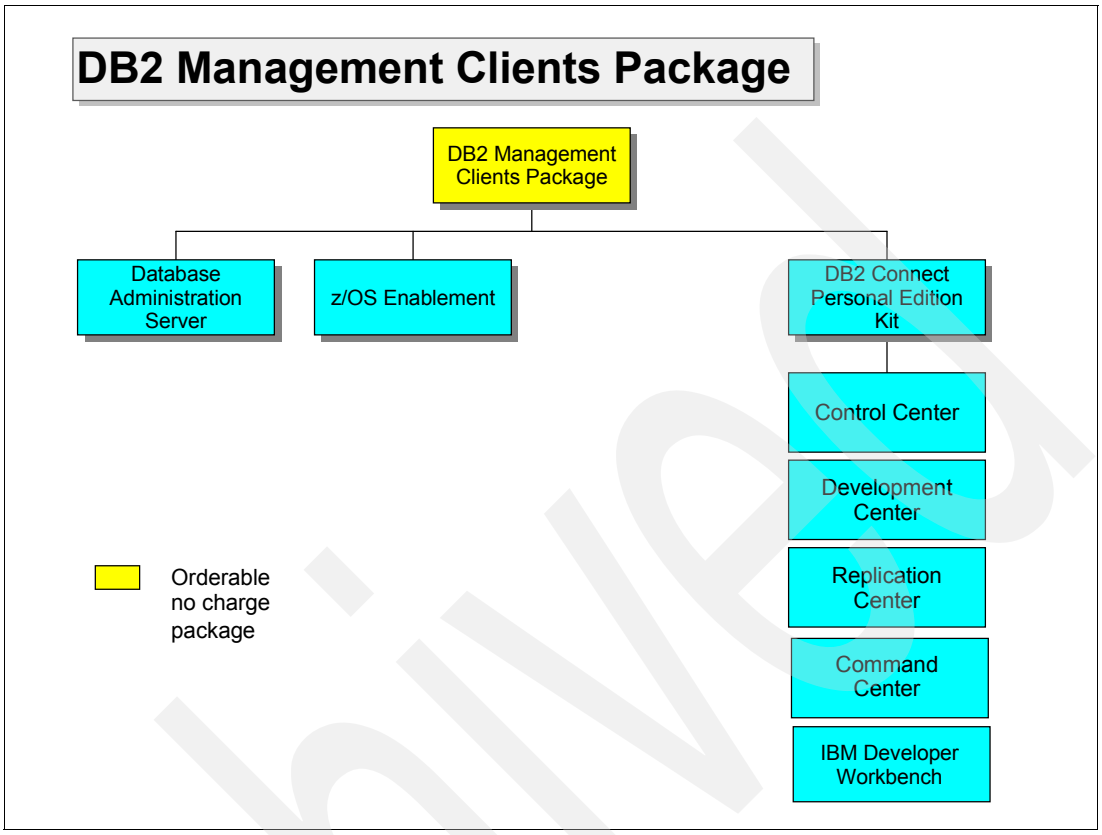


Figure 12-5 The Management Clients Package

New with DB2 9 is the IBM DB2 Developer Workbench. The Developer Workbench is a visual tool that aids in rapid development of DB2 business objects. This newly designed tool is based on the Eclipse framework and *replaces the DB2 Development Center* (that was based on Swing architecture). It runs on Linux and Windows.

Developer Workbench makes it easy to:

- Create, view, and edit database objects (such as tables and schemas).
- Explore and edit data in tables and rows.
- Visually build SQL and XQuery statements.
- Develop and deploy stored procedures, user defined functions (UDFs), routines, and scripts.
- Debug SQL and Java stored procedures.
- Develop SQLJ applications.
- Develop queries and routines for XML data.
- Perform data movement (such as load and extract).
- Collaborate and share projects with team members.
- Migrate projects from DB2 Development Center.

Archive

In addition to providing an interface for developing applications and objects on DB2 9 for Linux, UNIX, and Windows, Developer Workbench provides a common interface for working with DB2 on other platforms: DB2 for iSeries and DB2 for z/OS.

The Developer Workbench installation media is a CD with the DB2 Client Management Package, but it is also available as a free download, after registration, from the Web site:

https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?lang=en_US&source=swg-dm-db2dwb

z/OS Application Connectivity to DB2 for z/OS

This feature, with *FMID HDDA211*, consists of a component known as the DB2 Universal Database™ Driver for z/OS, Java Edition, a pure Java, type 4 JDBC driver designed to deliver high performance and scalable remote connectivity for Java-based enterprise applications on z/OS to a remote DB2 for z/OS database server.

This feature is described in detail in the program directory GI10-8738-00.

12.2.4 New optional product

We look at a separate no-charge product that offers functions complementary to DB2's.

DB2 Accessories Suite (no charge)

The DB2 Accessories Suite, program number 5655-R14, is a new no-charge product available for, but not part of, DB2 9. This collection of tools expand capabilities with the current functions (see Figure 12-6):

- DB2 Optimization Service Center, FMID *H2AG110*
- DB2 Spatial Support, *FMID J2AG110*
- The International Components for Unicode for DB2 for z/OS, FMID *H2AF110*

See also "DB2 Accessories Suite for z/OS" on page 542.

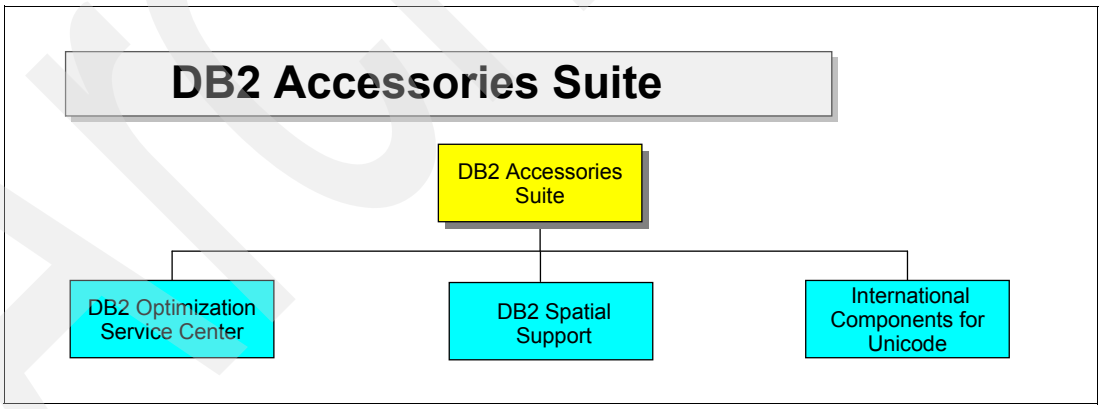


Figure 12-6 The Accessories Suite

12.2.5 Optional chargeable products

Optional chargeable products available with DB2 9 are:

- DB2 Version 9.1 for z/OS QMF
- DB2 Utilities Suite for z/OS Version 9.1

Archived

DB2 QMF Version 9.1

With this release, DB2 QMF is a completely redesigned, cross-platform workstation and Web-based solution, providing high-performance on demand access to data, reports, and interactive data analysis, using a rich desktop application or ordinary Web browser. Included in DB2 QMF V9 is:

- Support for DB2 9, with XML and enhancements to SQL.
- Reengineered Eclipse-based desktop application that extends QMF to numerous workstation operating systems, including Windows, Linux, Solaris, and HP-UX platforms.
- New OLAP query builder, complete with graphical designer, hierarchical and dimension filtering, informational SQL view, and fully interactive results grid.
- New object repository simplifies the storage, access control, and distribution of QMF objects across the enterprise.
  - *Offline* schemas and snapshots allow content authors to view filtered, role-specific database resources, such as tables, views, and procedures, without physically accessing the underlying database.
  - Independent user interface *perspectives* for QMF administrators, QMF users, and visual report designers.
- Redesigned QMF for WebSphere companion delivers key workstation functionality, using a pure HTML, browser-based client.

  Extends key QMF functionality to browser-based users across multiple platforms, including report authoring and drag-drop data analysis.
- Completely re-engineered form designer significantly simplifies the development of QMF forms/reports.
- Ability to develop and export query results and reports in a variety of formats including XML, HTML, and PDF.
- Improved table editor provides a means of directly inserting, deleting, and editing data in all supported database types.
- Newly developed data cache provides a means of caching query result sets with user or group-specific expiration rules.

QMF V9.1 consists of the following two priced optional features (see Figure 12-7 on page 431):

- DB2 QMF Enterprise Edition

  DB2 QMF Enterprise Edition provides the entire DB2 QMF family of technologies, enabling enterprise-wide business information across end-user and database platforms. DB2 QMF Enterprise Edition consists of these components:
  - DB2 QMF for TSO/CICS, *FMID HSQ9910*
  - DB2 QMF High Performance Option (HPO), *FMID HHP9910*
  - DB2 QMF for Workstation
  - DB2 QMF for WebSphere
- DB2 QMF Classic Edition

  DB2 QMF Classic Edition supports end users functioning entirely from traditional mainframe terminals and emulators (including IBM Host On Demand) to access DB2 databases. This edition consists of DB2 QMF for TSO/CICS, *FMID HSQ9910*.

Archive

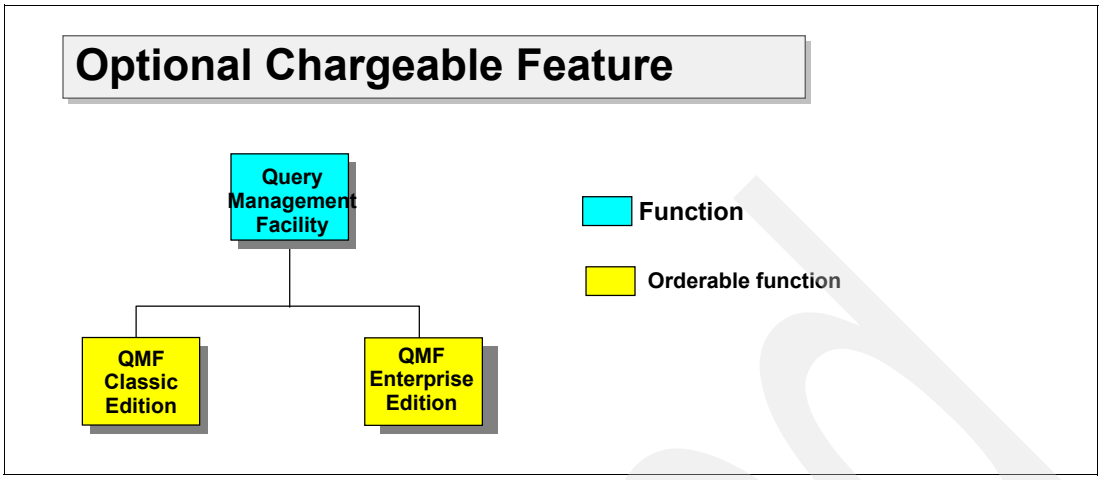


Figure 12-7 Optional chargeable function

DB2 Utilities Suite (charge)

A separate product, DB2 Utilities Suite for z/OS Version 9.1, program number 5655-N97, is available with DB2 9.

The following utilities are core utilities, which are included (at no extra charge) with Version 9 of DB2 for z/OS:

- CATENFM
- CATMAINT
- DIAGNOSE
- LISTDEF
- OPTIONS
- QUIESCE
- REPAIR
- REPORT
- TEMPLATE
- All DSN stand-alone utilities

All other utilities are made available with the DB2 Utilities Suite (program number 5655-N97). This Suite has FMID JDB991K and enables the following utilities:

- BACKUP SYSTEM
- CHECK DATA
- CHECK INDEX
- CHECK LOB
- COPY
- COPYTOCOPY
- EXEC SQL
- LOAD
- MERGECOPY
- MODIFY RECOVERY
- MODIFY STATISTICS
- REBUILD INDEX
- RECOVER
- REORG INDEX
- REORG TABLESPACE
- RESTORE SYSTEM
- RUNSTATS

Archived

- STOSPACE
- UNLOAD

All DB2 utilities operate on the catalog, directory, and sample objects, without requiring any additional products.

With DB2 V7and V8, the DB2 utilities were separated from the base product and offered as separate products licensed under the IBM Program License Agreement (IPLA). However, effective December 9, 2003, IBM withdrew from marketing the following programs licensed under the IBM International Program License Agreement (IPLA).

- DB2 Diagnostic and Recovery Utilities (5655-E62) for z/OS V7
- DB2 Operational Utilities for z/OS, V7 (5655-E63)

Note that DB2 Utilities Suite for z/OS V7 (5655-E98) continues to be available and contains all the capabilities of both of the withdrawn programs.

DB2 9 offers all of the utilities in one package. The only DB2 utility product is now the DB2 Utilities Suite. The DB2 Operational Utilities and the DB2 Diagnostic and Recovery Utilities offerings are no longer available. In V8, DB2 Utilities were separately orderable as a single feature but provided as two FMIDs, requiring two separate SMP/E installs. In DB2 V9 the two FMIDs are merged into a single FMID on a single tape, thus simplifying the SMP/E install process for customers by eliminating one tape, and sample SMP/E install jobs DSNRECVM, DSNRECVS, DSNAPPLM, DSNAPPLS, DSNACCPM, and DSNACCPS have been deleted.

See Appendix A, “DB2 Tools for z/OS” on page 537, for the list of DB2 utilities.

All DB2 utilities operate on catalog, directory, and sample objects, without requiring the additional product.

12.3 Changes to installation and migration

Installation is the process of installing a new DB2 subsystem. In this case you will not need to be concerned with the different modes/stages of the migration process, regression, or conversion of pre-existing data. With a new installed DB2 V9 subsystem, you can immediately take advantage of all new functions provided by the new release. In 12.5, “Installation and migration overview” on page 440, we discuss the major changes to the install process.

Migration is the process of converting an existing DB2 subsystem or data sharing group, catalog, directory, and user data to DB2 V9. This process is changed in V9 to allow for the new functions provided and functions deprecated.

The major changes to the installation and migration process are:

- Real Time Statistics (RTS) now becomes part of the DB2 catalog.
- The BSDS must be in the new expanded format.
- The TEMP database is no longer used and the WORKFILE database is now used for all temporary table spaces, both system and user defined.
- You can now use SMS managed data sets for installation data sets.
- You can now use a different prefix/suffix for non SMP/E installation data sets.
- DB2 Managed SPAS is no longer supported.
- Simple table spaces are deprecated.
- The DBPROTCL default is no longer supported.

- Enhanced support for XML.
- The migration process consists of three distinct phases and two new *fallback* phases.
  - CM*: Compatibility Mode*
  - ENFM*: Enable new-function mode*

12.3.1 Incompatibilities between Version 8 and Version 9

Be aware of the following changes that might affect your DB2 operations when migrating to V9. For more information about how migration might affect your DB2 operations, see the *DB2 Version 9.1 for z/OS Installation Guide,* GC18-9846.

Applications

The changes regarding applications are discussed here.

Changes to PL/I applications with no DECLARE VARIABLE statements

For PL/I applications with no DECLARE VARIABLE statements, the rules for host variables and string constants in the FROM clause of a PREPARE or EXECUTE IMMEDIATE statement have changed. A host variable must be a varying-length string variable that is preceded by a colon. A PL/I string cannot be preceded by a colon.

Commands

The changes regarding commands are discussed here.

Changes to -DISPLAY THREAD output length

In previous releases, if the -DISPLAY THREAD command was issued from an MVS console, DB2 displayed 255 lines of thread output per type specified, per member. In V9, the -DISPLAY THREAD command output displays the number of lines per member that is specified in the LIMIT keyword. The default value of LIMIT is 512 lines of output per type specified, per member.

Storage

The changes regarding storage are discussed here.

Changes to the format of the BSDS

In previous releases, the BSDS format allowed only 1,000 data sets per copy for archive logs and 31 data sets per copy for active logs. In V9, the BSDS must be in the format that supports up to 10,000 entries of volumes per copy for archive logs and 93 data sets per copy for active logs. Any unconverted BSDSs are converted in job DSNTIJUZ. BSDSs that you have already converted are unaffected. Note that the new format in supported by V8 NFM, and conversion can be run any time (with DB2 down) after V8 is moved to NFM and before the first V9 start.

The work file database is the only temporary database

In DB2 V9, the work file database is the only temporary database. The work file database is used for all temporary tables, both system and user defined, which simplifies DB2 systems management tasks. The TEMP database is no longer used by DB2.

Work file table spaces are now created on DB2-managed storage during installation

In previous releases, the DB2 installation process created work file table spaces on user-managed storage. In V9, the installation process creates work file table spaces on DB2-managed storage. The default storage group is SYSDEFLT. You can specify a different,

Archived

user-managed storage group by modifying the parameters in the last step of installation job DSNTIJTM.

SQL

The changes regarding SQL are discussed here.

Changes in BIND PACKAGE and BIND PLAN defaults

The default value for bind option CURRENTDATA is changed from YES to NO. This applies to the BIND PLAN and the BIND PACKAGE subcommands, as well as the CREATE TRIGGER for trigger packages, and the CREATE PROCEDURE and the ALTER PROCEDURE ADD VERSION SQL statements for SQL PL procedure packages. Specifying NO for CURRENTDATA is the best option for performance. This change does not affect the REBIND subcommand.

The default value for bind option ISOLATION is changed from RR to CS. This applies to the BIND PLAN and the remote BIND PACKAGE subcommands. For the BIND PACKAGE subcommand, the current default (plan value) stays. The default change does not apply to implicitly-built CTs (for example, DISTSERV CTs) or for the REBIND subcommand.

Although you can specify DBPROTOCOL(PRIVATE) for the DBPROTOCOL parameter of the BIND option, DB2 issues a new warning message, DSNT226I.

All BIND statements for plans and packages that are bound during the installation or migration process specify the ISOLATION parameter explicitly, except for routines that do not fetch data. The current settings are maintained for compatibility.

Plans and packages bound on DB2 Version 3 and before

If you have plans and packages that were bound on DB2 Version 3 and before and you specified YES or COEXIST in the AUTO BIND field of panel DSNTIPO, DB2 V9 autobinds these packages. Thus, you may experience an execution delay the first time that such a plan is loaded. Also, DB2 may change the access path due to the autobind, potentially resulting in a more efficient access path.

If you specified NO in the AUTO BIND field of panel DSNTIPO, DB2 V9 returns SQLCODE -908, SQLSTATE 23510 for each attempt to use such a package or plan until it is rebound.

Column names and labels in SQLDA SQLNAME field for statements involving UNION

Prior to DB2 V8, the result column name in a SQLNAME field of the SQLDA for a statement involving a UNION reflected the column name or label of the first sub-query in the statement. In V8, if labels are used, DB2 returns the label of the column in the first sub-query. If labels are not used, the result column name will only be returned if the column name is the same across all sub-queries in the statement. You can temporarily override this behavior by setting subsystem parameter UNION_COLNAME_7 to YES.

Changed behavior for ORDER BY clause in SELECT statement

If you order a query by a qualified column where the column name is the same as the AS NAME of the column in the select list, DB2 issues an error.

Changed behavior of the INSERT statement with the OVERRIDING USER VALUES clause

When the INSERT statement is specified with the OVERRIDING USER VALUES clause, the value for the insert operation is ignored for columns that are defined with the GENERATED BY DEFAULT attribute.

DESCRIBE no longer returns LONG type values

When you execute a DESCRIBE statement against a column with a LONG VARCHAR or LONG VARGRAPHIC data type, the DESCRIBE statement returns the values as VARCHAR or VARGRAPHIC data type.

DB2 enforces the restrictions about where a host variable array can be specified

The host-variable-array variable is the meta-variable for host variable arrays. The host-variable-array variable is included only in the syntax for multi-row FETCH, multi-row INSERT, multi-row MERGE, and EXECUTE in support of a dynamic multi-row INSERT or MERGE statement. The host-variable-array variable is not included in the general description of expression as a meta-variable, so host-variable-array cannot be used in other contexts. In previous releases, you could specify a host-variable-array in an unsupported context. In V9, if a host variable array is referenced in an unsupported context, DB2 issues an error.

For more information about where you can specify the host-variable-array variable, see DB2 SQL Reference.

DEBUGSESSION system privilege required for continued debugging of SQL procedures

After you migrate to new-function mode, users that debug external SQL procedures need the DEBUGSESSION system privilege. Only users of the new Unified Debugger-enabled client platforms need this system privilege. Users of the Version 8 SQL Debugger-enabled client platforms do not need this system privilege.

Changes to the result length of the DECRYPT function

The result length of the DECRYPT function is shortened to 8 bytes less than the length of the input value. If the result expands because of a difference between input and result CCSIDs, you must cast the encrypted data to a larger VARCHAR value before the DECRYPT function is run.

Utilities

The changes regarding utilities are discussed here.

Changes to DSN1LOGP

In previous releases, when you invoked DSN1LOGP with an RBA or LRSN range and the lower or higher bounds are not found in the available log files, DSN1LOGP returned return code 0. In V9, when you invoke DSN1LOGP with an RBA or LRSN range and the lower or higher bounds are not found in the available log files, DB2 issues a warning message to let you know that not all of the expected information can be formatted in the available logs and returns return code 4.

Changes to the REORG utility

The REORG utility has been updated to reduce the amount of virtual storage that is used to build dictionaries. Because the REORG utility needs less virtual storage, the 254 partition restriction for reorganizing compressed table spaces has been removed. REORG no longer issues message DSNU1146I.

REORG or LOAD REPLACE utility automatically converts table spaces to reordered row format

In previous releases, table spaces were created in basic row format (BRF). In V9, the REORG or LOAD REPLACE utility automatically converts table spaces to reordered row format (RRF).

If a table in the table space has a VALIDPROC, modify the VALIDPROC to null prior to running the REORG or LOAD REPLACE utility, and alter it back after the REORG or LOAD

REPLACE utility runs. For an EDITPROC on a table in a BRF table space, drop the table, convert the table space to RRF, and recreate the table. You might need to rewrite the VALIDPROC or EDITPROC so that it correctly interprets rows that are in RRF.

Changes to the REORG SHRLEVEL CHANGE utility

If you use the REORG SHRLEVEL CHANGE utility, you might need to change your operating procedures. If you ran concurrent REORG TABLESPACE SHRLEVEL CHANGE PART x on the same table space in Version 8, you need to change the way you submit the jobs so that only one executes at a time. If you allocate your own shadow data sets (for example, if you use user-managed VSAM data sets), you need to change your jobs to use the NPI data set naming convention and you need to allocate larger data sets for the NPI shadows. If you do not use the FASTSWITCH data set naming convention, you need to change your jobs to use that convention. Also, the BUILD2 phase of REORG TABLESPACE PART x SHRLEVEL CHANGE is eliminated.

If you have not been using RETRY with REORG SHRLEVEL CHANGE or REFERENCE, in V9, your REORGs retry after failed drain attempts, and if the retries are not successful, the jobs terminate with return code 8 instead of abending.

In V9, you do not need to run REORG INDEX SHRLEVEL CHANGE on all NPI data sets after you run REORG TABLESPACE PART x SHRLEVEL CHANGE because the NPI data sets are rebuilt by the REORG TABLESPACE PART.

During a REORG TABLESPACE PART SHRLEVEL CHANGE, applications that access partitions other than the ones being reorganized might now encounter time outs if they access NPI data sets during the last iteration of the LOG phase or during the SWITCH phase. Adjust the values of the DRAIN_WAIT and MAXRO parameters to minimize application time outs.

COLTYPE column in SYSIBM.SYSCOLUMNS and SYSIBM.SYSCOLUMNS_HIST for LONG column types

When new tables are created with LONG VARCHAR or LONG VARGRAPHIC columns, the COLTYPE values in SYSIBM.SYSCOLUMNS and SYSIBM.SYSCOLUMNS_HIST contain VARCHAR or VARGRAPHIC.

DB2 returns all DSNWZP output in the same format as DB2 parameters

In previous releases, DSNWZP returned the current setting of several system parameters in a format other than the one used by the system parameter macros. For example, DSN6SPRM expected the setting for EDMPOOL in kilobytes, and DSNWZP returned it in bytes. In Version 9, DB2 returns all DSNWZP output in the same format as DB2 parameters. Modify programs that call DSNWZP if they compensate for the format differences.

DB2 enforces the restriction that row IDs are not compatible with character strings when they are used with a set operator

In previous releases, DB2 did not always enforce the restriction that row IDs are not compatible with character strings. In V9, DB2 enforces the restriction that row IDs are not compatible with string types when they are used with a set operator (UNION, INTERSECT, or EXCEPT).

The GROUP ATTACH process is randomized

In V8, if more than one DB2 subsystem was running on a z/OS image in a data sharing group, incoming jobs went to the first active DB2 subsystem. In V9, the GROUP ATTACH process is randomized so that all members that are defined in a data sharing group and running on a z/OS image have an equal chance of getting the attach from a starting application if the application specifies a GROUP ATTACH name instead of an individual subsystem name.

You can no longer explicitly create a database name as DSNxxxxx

After you migrate to compatibility mode, if you explicitly create a database name with eight characters that begins with DSN and is followed by exactly five digits, DB2 issues an SQLCODE -20074 (SQLSTATE 42939).

SDSNLOAD must be a PDSE

The DB2 SDSNLOAD data set that contains most of the DB2 executable code must now be allocated (via the DSNALLOC job) as a PDSE data set. In DB2 V8, you had the option to convert this data set to a PDSE. If you did not take the opportunity during your DB2 V8 migration to convert this data se to a PDSE, you must now do so. When you allocate the DB2 SDSNLOAD data set as a PDSE, and use it in combination with the IMS Attach, you must apply the IMS toleration PTF UK10146. Also be aware of the operational differences between PDS and PDS/E data sets. These are explained in the DB2 9 program directory. Note, however, that SMP/E must know which type of data set it is managing. It will compress a PDS, but not compress a PDSE. If you change the data set organization, you must also update the SMP/E definition.

12.3.2 Enhanced functions

Be aware of the following changes for enhanced functions that might need appropriate settings to be taken advantage of. See the *DB2 Version 9.1 for z/OS Installation Guide,* GC18-9846.

Changes to the real-time statistics tables

In V9 new-function mode, the real-time statistics tables are part of the DB2 catalog. In V9 compatibility mode, the real-time statistics data is still in the user-defined tables. During enabling-new-function mode processing, job DSNTIJEN moves the real-time statistics data from your user-defined tables to catalog tables SYSIBM.SYSTABLESPACESTATS and SYSIBM.SYSINDEXSPACESTATS. If you revert to compatibility* mode, DB2 keeps the real-time statistics data in the catalog tables and does not use the user-defined tables. You can drop the user-defined tables after job DSNTIJEN moves the data to the catalog tables.

In previous releases, in the user-defined real-time statistics tables, the data types of TOTALROWS and SPACE in SYSTABLESPACESTATS and TOTALENTRIES in SYSINDEXSPACESTATS were FLOAT. In V9, they are the BIGINT data type.

Changes to resource limit facility tables

In previous releases, there was a single type of resource limit facility table with a name of the form authid.DSNRLSTnn. If the START RLIMIT command was executed and the resource limit facility table did not exist, the START RLIMIT command terminated. In V9, there are two types of resource limit facility tables: tables of the form authid.DSNRLSTnn and tables of the form authid.DSNRLMTnn. In V9 compatibility mode, if authid.DSNRLSTnn does not exist, START RLIMIT terminates. In V9 new-function mode, the START RLIMIT command can complete successfully if either type of resource limit facility table exists.

New limits for the number of work file table spaces and indexes

The number of table spaces in the work file database cannot exceed the limit of 500. The number of indexes that are defined on declared global temporary tables that belong to all agents on the local DB2 member cannot exceed the limit of 10000.

You can use the installation CLIST to add table spaces to the workfile database as part of the migration process

In previous releases, you could create table spaces in the work file database only in the INSTALL mode of the installation CLIST. In V9, you can create zero or more additional table spaces in the work file database in the MIGRATE mode of the installation CLIST. The segment size of the table spaces is restricted to 16 until DB2 enters new-function mode. For more information see “Work file database panel: DSNTIP9” in the Installation Guide.

You can specify 4 KB, 8 KB, 16 KB, or 32 KB buffer pools for use with indexes

In previous releases, you could specify only 4 KB buffer pools for user indexes. In V9, you can specify 4 KB, 8 KB, 16 KB, or 32 KB buffer pools for user indexes. Indexes that are created during compatibility mode require a 4 KB buffer pool, so you need to specify a 4 KB buffer pool in the BUFFERPOOL clause when you create an index. If you do not specify a 4 KB buffer pool in the BUFFERPOOL clause when you create an index in compatibility mode, DB2 issues SQLCODE = -676, ERROR: ONLY A 4 KB PAGE BUFFERPOOL CAN BE USED FOR AN INDEX.

Fewer SMP/E jobs are required

Because utilities are now contained in a single FMID, SMP/E jobs DSNRECVM, DSNRECVS, DSNAPPLM, DSNAPPLS, DSNACCPM, and DSNACCPS have been deleted.

12.3.3 Functions that are deprecated or no longer supported

We list here the functions no longer available in DB2 V9.

Java stored procedures no longer run in resettable JVMs

In previous releases, Java stored procedures could be run in resettable JVMs. In V9, Java stored procedures cannot be run in resettable JVMs.

Because Java stored procedures cannot be run in resettable JVMs, they are no longer automatically prevented from impacting the execution of future Java stored procedures in the following ways:

- Setting system properties
- Attempting to access or modify the security configuration objects
- Loading a native library

Such actions can impact the ability of other Java stored procedures to use that JVM. For more information about resettable JVMs and for a full list of actions that are prevented, see the Unresettable actions topic in the Persistent Reusable Java Virtual Machine User's Guide. Recommendation: After you migrate to V9, modify your applications that use functions of resettable JVMs to not run Java stored procedures in resettable JVMs.

DB2-managed SPAS no longer supported

DB2 managed stored procedure address spaces are no longer supported. Stored procedures must be moved to a WLM environment.

12.4 Removal of support for DB2-managed stored procedures

Starting with DB2 V8, you are no longer allowed to create new DB2-managed stored procedures (that is, procedures that were operating in the DSNSPAS address space under the control of program DSNX9STP). Our recommendation in *DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ... and More,* SG24-6079, was that you should plan to migrate all your DB2-established stored procedures as soon as possible to WLM-controlled processing.

Now, starting with DB2 V9 compatibility mode, if you try to invoke program DSNX9STP (that is, if you try to start the DSNSPAS address space), DB2 issues the new message DSNX930, shown in Figure 12-8, and the DSNSPAS address space does not start.

```
DSNX930: DB2-MANAGED STORED PROCEDURES ADDRESS SPACE NOT SUPPORTED.
Explanation: An address space referencing module DSNX9STP was started. In prior
releases of DB2, this was the module to execute DB2-managed stored procedures,
which were defined as NO WLM ENVIRONMENT.
This function is no longer supported by DB2.
System Action: The address space terminates and this message is issued.
Programmer Response: Use only WLM-managed stored procedures in V9. Do not
attempt to start an address space referencing DSNX9STP.
```

Figure 12-8 DSNX930 message when trying to start DSNSPAS address space

Note: Before you migrate to V9, modify your DB2-managed stored procedures to be managed by WLM.

JDBC/SQLJ Driver for OS/390 and z/OS is no longer supported

The JDBC/SQLJ Driver for OS/390 and z/OS is no longer supported. All Java application programs and Java routines that are currently written to work with the JDBC/SQLJ Driver for OS/390 and z/OS need to be modified to work with the IBM DB2 Driver for JDBC and SQLJ (formerly known as the DB2 Universal JDBC Driver). The steps for migrating JDBC and SQLJ applications from the legacy JDBC/SQLJ Driver for OS/390 and z/OS to the IBM DB2 Driver for JDBC and SQLJ can be found in the Application Programming Guide and Reference for Java.

All WLM-managed stored procedure address spaces that are set up to execute Java routines must be modified to reference the IBM DB2 Driver for JDBC and SQLJ. Existing WLM environments that are configured to use the JDBC driver fail when the address space initializes. Data sharing groups with members in V8 that are in coexistence with V9 in compatibility mode fail if a Java routine is invoked on any V9 members where the WLM-SPAS JCL does not reference the Universal JDBC driver.

Tip: Before you migrate to V9, modify your WLM-SPAS JCL to use the IBM DB2 Driver for JDBC and SQLJ.

Simple table spaces are deprecated

Simple table spaces are deprecated. Simple table spaces that were created in previous releases can still be read and updated. However, if DB2 implicitly creates a table space, or you explicitly create a table space without specifying the SEGSIZE, NUMPARTS, or MAXPARTITIONS options, DB2 creates a segmented table space instead of a simple table

space. By default, the segmented table space has a value of 4 for SEGSIZE and, for implicitly created table spaces, a value of ROW for LOCKSIZE.

DBPROTCL is no longer supported

The DBPROTCL subsystem parameter is no longer supported. The default for the DBPROTOCOL bind option is DRDA. You can still specify DBPROTOCOL(PRIVATE) explicitly when binding packages and plans, but DB2 issues a warning message if you perform any bind with this value.

Removed subsystem parameters

Subsystem parameter SUPPRESS_TS_CONV_WARNING has been removed. In V9, DB2 always operates as though SUPPRESS_TS_CONV_WARNING=NO. Subsystem parameters MAX_OPT_ELAP and TABLES_JOINED_THRESHOLD have been removed.

CAST FROM clause of CREATE FUNCTION statement for SQL functions is no longer supported

The CAST FROM clause of the CREATE FUNCTION statement for SQL functions is no longer supported. If you issue a CREATE FUNCTION statement for an SQL function with a CAST FROM clause, DB2 issues an error.

The RELCURHL=NO DSNZPARM option is removed

The RELCURHL=NO DSNZPARM option is removed. This is an incompatibility for applications dependent on retaining page or row locks across commits for a WITH HOLD cursor.

12.5 Installation and migration overview

You can install or migrate to DB2 V9 via the traditional method of a host-based installation process executed from TSO. This processes only differs from previous releases in that some panels have been changed in order to support new and changed features and functions as well as some deprecated functions. There are new, expanded, and removed fields and some changes to the panels. In this section we review the changes made to the TSO panels.

We discuss the following:

- Summary of changes
- Enhancements to the installation CLIST and panels
- Installation changes in support of the major enhancements:
  - XML
  - Converge temp space
  - Network trusted contexts
  - DBMA cleanup
  - Improve RLF usability
  - Private Protocol removal
  - Optimization Service Center
  - Discontinue SQLFLAG precompiler option
  - Discontinue DB2-managed stored procedures
  - BSDS Conversion required for V9 migration
- Installation troubleshooting tips

12.5.1 Summary of changes

With DB2 V9 we use the same general approach for installing and migrating DB2 as with DB2 V8. We have four CLIST modes: INSTALL, MIGRATE, ENFM, and UPDATE. You use the INSTALL option if you want to install a brand new DB2 V9 subsystem. The MIGRATE option is used if you have existing DB2 V8 systems that you want to migrate to DB2 V9. If you are in data sharing, you will need to run the MIGRATE option for each member of the data sharing group to get each member into DB2 V9 compatibility mode (CM). The ENFM option is used to enable a new function for a subsystem or an entire data sharing group. ENFM is a group-wide enablement. You only need to run it once for each data sharing group. The UPDATE option is used when you want to update or change DSNZPARMs.

DB2 V9 introduces two new migration modes in addition to the modes we had in V8 (compatibility mode (CM), enable-new-function mode (ENFM), and new-function mode (NFM)). The new modes in DB2 V9 are compatibility mode* (CM*) and enable new-function mode* (ENFM*). Refer to 12.6.1, “The migration process” on page 476, for details on these two new modes.

Note: The term *conversion mode* has recently replaced *compatibility mode* to better reflect the purpose of this mode during which some new function are made available but do not preclude a fall back to the previous version.

12.5.2 Enhancements to the installation CLIST and panels

There are several usability enhancements to the V9 installation CLIST.

Archive

Use different prefix and suffix as DB2 SMP/E data sets

Prior to V9, the installation CLIST assumed that the data sets defined during DB2 installation use the same prefix and suffix as the DB2 SMP/E libraries. However, many customers share the same SMP/E libraries across multiple DB2s. In V9, the main CLIST panel (DSNTIPA1) has prefix/suffix fields for both types of data sets. In Figure 12-9, SMP/E libraries have a prefix of DB9B9, but installation-defined libraries will have a prefix of DB9B9.V9MIGRAT.

```
DSNTIPA1 DB2 VERSION 9 INSTALL, UPDATE, MIGRATE, AND ENFM - MAIN PANEL
===>

Check parameters and reenter to change:
 1  INSTALL TYPE          ===> MIGRATE   Install, Migrate, ENFM, or Update
 2  DATA SHARING          ===> NO        Yes or No (blank for ENFM or Update)

Enter the data set and member name for migration only. This is the name used
from a previous Installation/Migration from field 9 below:
 3  DATA SET(MEMBER) NAME ===> DB8R8.SDSNSAMP(DB8RURG)

For DB2 SMP/E libraries (SDSNLOAD, SDSNMACS, SDSNSAMP, SDSNCLST, etc.), enter:
 4  LIBRARY NAME PREFIX   ===> DB9B9
 5  LIBRARY NAME SUFFIX   ===>

For install data sets (NEW.SDSNSAMP, NEW.SDSNCLST, RUNLIB.LOAD, etc.), enter:
 6  DATA SET NAME PREFIX  ===> DB9B9.V9MIGRAT
 7  DATA SET NAME SUFFIX  ===>

Enter to set or save panel values (by reading or writing the named members):
 8  INPUT MEMBER NAME     ===> DSNTIDXA  Default parameter values
 9  OUTPUT MEMBER NAME    ===> DSNTIDV9  Save new values entered on panels
PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-9 Main panel DSNTIPA1

Support of SMS-managed storage

Prior to V9, explicit device types and volume serial IDs were required for all data sets created during installation, migration, and verification. In V9, you can optionally specify the following:

- SMS data class
- SMS management class
- SMS storage class

Archived

If you specify a value, it will be used in the generated JCL accordingly. For example, you can specify SMS classes for installation-defined data sets such as prefix.RUNLIB.LOAD, prefix.DBRMLIB.DATA, and so on. In Figure 12-10 you can see the new panel fields for SMS options.

Note: DB2 does not provide guidance for SMS design, setup, or use. Most customers already use SMS for DB2 related data sets. You will need to coordinate with your storage administrator.

```
DSNTIPA3            MIGRATE DB2 - DATA PARAMETERS PANEL 2
===>

Check parameters and reenter to change:
 1 PERMANENT UNIT NAME    ==> 3390               Device type for MVS catalog
                                                 and partitioned data sets
 2 TEMPORARY UNIT NAME    ==> VIO                Device type for
                                                 temporary data sets


                                        ---------------- SMS -----------------
                            VOL/SER     DATA CLASS    MGMT CLASS    STOR CLASS
                            -------     ----------    ----------    ----------
 3 CLIST ALLOCATION       ==> SBOX9A  ==>           ==>           ==>
 4 NON-VSAM DATA          ==> SBOX9A  ==>           ==>           ==>
 5 VSAM CATALOG, DEFAULT, ==> SBOX9A  ==>           ==>           ==>
   AND WORK FILE DATABASE
 6 DIRECTORY AND CATALOG  ==> SBOX9A  ==>           ==>           ==>
   DATA
 7 DIRECTORY AND CATALOG  ==> SBOX9A  ==>           ==>           ==>
   INDEXES
 8 LOG COPY 1, BSDS 2     ==> SBOX9A  ==>           ==>           ==>
 9 LOG COPY 2, BSDS 1     ==> SBOX9A  ==>           ==>           ==>
PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-10 SMS-managed install data sets

12.5.3 Installation changes

In this section we discuss installation changes.

XML

New tasks in support of XML schema processor and XML decomposition are:

- Create supporting database for the XML schema processor.
- Create supporting stored procedures and UDFs.
- Bind packages for supporting procedures and UDFs.
- Grant access to PUBLIC.
- Establish limits to storage used by user and system.

These tasks are performed by:

- DSNTIJSG for new V9 installations
- DSNTIJNX for migrations to V9

  Tasks require NFM, and so cannot be performed by DSNTIJSG.

Note: In the past, some customers have manually edited job DSNTIJSG and run it during migrations. If you attempt to use this job during migration to V9, you may get SQLCODE -4700. You should be using DSNTIJNX during migrations instead.

- DSNTIJUZ changes in panel DSNTIPD to set up DSNZPARMs XMLVALA and XMLVALS.
  - XMLVALA: This specifies an upper limit for the amount of storage in KB that each user can have for storing XML values:

    `RANGE: 1-2097152, DEFAULT: 204800`
  - XMLVALS: This specifies an upper limit for the amount of storage in MB that the system can use for storing XML values:

    `RANGE: 1-51200, DEFAULT: 10240`

Converge temporary space

Prior to V9:

- Two databases for temporary data
  - WORKFILE database: used for Created Global Temporary Tables and SQL work space (result sorting, materializing views, and so on)
  - TEMP database: used for Declared Global Temporary Tables
- Installation creates work file table spaces on user-managed storage

In V9:

- One work database: The TEMP database is converged into WORKFILE.
- Installation creates work file table spaces (on DB2-managed storage).
- Migration permits adding new work file table spaces.

Archived

- A new DSNZPARM called MAXTEMPS regulates max temp storage available to a single agent. See option 7 in Figure 12-11.

```
DSNTIP9                MIGRATE DB2 - WORK FILE DATABASE
===>

Enter work file configuration options below:
 1 TEMP 4K SPACE      ===> 0           Amount of 4K-page work space (MB)
 2 TEMP 4K TBL SPACES ===> 0           Number of tablespaces for 4K-page data
 3 TEMP 4K SEG SIZE   ===> 16          Segment size of 4K-page tablespaces

 4 TEMP 32K SPACE     ===> 500         Amount of 32K-page work space (MB)
 5 TEMP 32K TBL SPACES===> 5           Number of tablespaces for 32K-page data
 6 TEMP 32K SEG SIZE  ===> 16          Segment size of 32K-page tablespaces

 7 MAX TEMP STG/AGENT ===> 100         Maximum MB of temp storage space
                                       that can be used by a single agent









PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-11 WORKFILE database panel

The space per table space cannot exceed 64 GB. For example, the amount of 4 KB work space divided by the number of 4 KB table spaces cannot exceed 64 GB. The same is true for the 32 KB table spaces. If you specify values that will cause each table space to be larger than 64 GB, the CLIST will reduce the space and issue a warning message DSNT55I. There is a maximum of 500 table spaces that can be created in the work file database. The sum of table spaces to be created cannot exceed 500 or you will receive error message DSNT556I. Note that this process has no knowledge of existing work file table spaces.

On panel DSNTIP9 in INSTALL mode, you must specify at least one 4 KB work file table space and one 32 KB work file table space. You can override the segment size to a value other than 16.

On panel DSNTIP9 in MIGRATE mode, by default, no additional workfile table spaces are to be created, but you can specify additional ones to be created. You cannot change the segment size. It is fixed at 16 prior to NFM. If you specify an amount of space greater than 0, then you must specify a number of table spaces greater than 0. Otherwise you will receive error message DSNT557I. If you specify a number of table spaces greater than 0, you must specify an amount of space greater than 0. Otherwise you will receive error message DSNT558I.

The parameters on this panel are used to generate the last step in job DSNTIJTM. This step executes a tool (DSNTWFG DB2 REXX exec) for creating the specified number of 4 KB or 32 KB workfile table spaces. Note that this is a REXX program and requires DB2 REXX Language support. In Figure 12-12 you can see a sample of this step and the description of its input parameters.

```
//*
//* DSNTWFG: ADD 4-KB AND 32-KB TABLE SPACES IN THE WORK FILE DB
//*   PARMS: 1. LOCAL DB2 SSID
//*          2. AUTHORIZATION ID TO PERFORM THE WORK
//*          3. NAME OF THE DB2 WORK FILE DATABASE
//*          4. NUMBER OF 4-KB TYPE TABLE SPACES TO BE ADDED
//*          5. PRIMARY SPACE IN MB FOR EACH 4-KB TYPE TABLE SPACE
//*          6. SEGMENT TYPE FOR 4-KB TYPE TABLE SPACES
//*          7. NAME OF BUFFERPOOL FOR 4-KB TYPE TABLE SPACES
//*          8. NAME OF DB2 STORAGE GROUP FOR 4-KB TYPE TABLE SPACES
//*          9. NUMBER OF 32-KB TYPE TABLE SPACES TO BE ADDED
//*         10. PRIMARY SPACE IN MB FOR EACH 32-KB TYPE TABLE SPACE
//*         11. SEGMENT TYPE FOR 32-KB TYPE TABLE SPACES
//*         12. NAME OF BUFFERPOOL FOR 32-KB TYPE TABLE SPACES
//*         13. NAME OF DB2 STORAGE GROUP FOR 32-KB TYPE TABLE SPACES
//*
//DSNTIST EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//SYSEXEC  DD  DISP=SHR,DSN=DSN!!0.SDSNCLST
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSTSIN  DD  *
  DSNTWFG DSN SYSADM DSNDB07 +
          1 16 16 BP0   SYSDEFLT +
          1  1 16 BP32K SYSDEFLT
```

Figure 12-12 DSNTWFG and description of its parameters

DSNTIJTM changes

Step DSNTICR creates the DB2 default STOGROUP SYSDEFLT. This was previously handled by job DSNTIJSG. This adds a new step DSNTIRX to bind the plan and packages for DB2 REXX Language support. This was previously handled by optional job DSNTIJRX. This added new step DSNTIST to execute the DSNTWFG REXX tool to create work file table spaces. We also removed AMS to define data sets for work file table spaces since they are now STOGROUP defined.

DSNTIJUZ changes

MAXTEMPS DSNZPARM was added to specify the maximum amount of temp storage available (in MB) to a single agent. This is on panel DSNTIP9 field MAX TEMP STG/AGENT. The valid range for this field is 0–2147483647, and this field defaults to 0, meaning no limit.

Migration and fallback consideration

After migration to CM, the TEMP database is no longer used. It is your responsibility to drop the TEMP database and resize the WORKFILE database accordingly.

Tip: Do not drop the TEMP database until you are committed to remaining on V9. Additionally, if you are in coexistence mode, the other members still use their TEMP databases.

Network trusted context

Provide a secure port for DRDA:

- DSNTIP5 (DDF panel 2) changes:

  Installation collects the setting from a new field, SECURE PORT. This is not a DSNZPARM. The default is blank. If specified, you must also specify a value in the RESYNCH PORT field.

- DSNTIJUZ changes:

  If the value specified in the DSNTIP5 SECURE PORT field is non-blank, it is used to set the SECPORT parameter of the DSNJU003 (change log inventory) utility in job step DSNTLOG (initialize/update the BSDS).

DBMA cleanup

This discontinues the DSN6SPRM.RELCURHL DSNZPARM. This was used to control whether at commit time DB2 should release a lock on which a cursor defined WITH HOLD is positioned. In V9, DB2 will always release the lock at commit. This allows greater concurrency.

- DSNTIP8 (Performance and Optimization panel) changes:

  The RELEASE LOCKS field is removed.

- DSNTIJUZ changes:

  The RELCURHL DSNZPARM is removed from the DSN6SPRM macro expansion.

Improve RLF usability

RLF usability enhancements in DB2 V9 require a new RLF table (DSNRLMTnn) and index (DSNMRLnn).

DSNTIJSG changes

This job creates the RLST database during installation (but not migration) of a DB2 subsystem or data sharing group. In V9, DSNTIJSG provides DDL to create the new DSNRLMTnn table and DSNMRLnn index. The DDL is provided regardless of installation or migration. However, the DDL is commented out because not all sites will want to use the new feature, and an empty DSNRLMTnn table increases RLF overhead with no benefit. The DSNTIJSG prolog calls attention to the need to uncomment the DDL if you want to deploy the new table.

Migration consideration

SQLCODE -204 will result if you attempt to create the new table without the RLF database.

Private Protocol removal

The DSN6SYSP.DBPROTCL DSNZPARM is eliminated so that you can no longer bind plans and packages for Private Protocol by default. A generic collection ID, DSNCOLLID, is defined to reduce administrative tasks to maintain collections of remote packages. We also provide a tool called DSNTP2DP to facilitate conversion from private protocol to DRDA.

Archived

DSNTIP5 (DDF panel 2) changes

The DATABASE PROTOCOL field is removed. DSN6SYSP.DBPROTCL was used to specify the default setting of the DBPROTOCOL bind option. In V9, DBPROTOCOL is always DRDA by default. Explicit coding DBPROTOCOL(PRIVATE) on your bind statements still works, but results in DSNT226I warning.

DSNTIJUZ changes

The DBPROTCL DSNZPARM is removed from the DSN6SYSP macro expansion.

DSNTIJSG changes

There is a new, optional, job step, DSNTICC, to grant access to DSNCOLLID. Note that DSNTICC is commented out but can be uncommented by sites that can accept remote generic binding at a server.

DSNTIJPD

This is a new job that drives the new DSNTP2DP tool. Consider running it on your V8 system.

DSNTP2DP scans the catalog and generates commands to convert all objects that have a Private Protocol dependency to DRDA. This job can be run anytime after DSNTIJTM, which binds DB2 REXX Language Support. DSNTIJPD does not run the commands generated by DSNTP2DP. You have to run these yourself at a later time.

Optimization Service Center

Several new job steps are added to support the Optimization Services Center (OSC):

- DSNTIJOS: a new job that creates the OSC database (optional)
- DSNTICU: drops the OSC database and STOGROUP (optional)
- DSNTIAS: creates the OSC STOGROUP, database, and 4-KB page size table spaces
- DSNTPRO: creates the OSC profile tables
- DSNTRUN: creates the OSC runtime information tables and LOB table space
- DSNTEXP: creates the OSC EXPLAIN tables
- DSNTOSC: creates the OSC statement cache database
- DSNTWCC: creates the WCC tables
- DSNTRIO: creates RUNTIME Info tables under DB2OSC
- DSNTEXO: creates EXPLAIN tables under DB2OSC
- DSNTWSA: creates the WSA tables
- DSNTALI: creates ALIASes on the OSC tables
- DSNTRTN: creates and bind the OSC routines
- DSNTJRT: creates the Language Environment environment options data set for the WLM environment used to execute OSC stored procedure SYSPROC.OSC_EXECUTE_TASK
- DSNTIJG: grants access on objects created by this job

Discontinue SQLFLAG precompiler option

In DB2 V9, we remove support for the precompiler SQLFLAG option, that is, FLAGGER. In earlier releases, DSNTIJSG binds a package and plan for FLAGGER called DSNHYCRD. The package and plan are freed as part of migration to V9.

In MIGRATE mode only, the installation CLIST appends a new job step, DSNTIJF, to the end of DSNTIJSG to free the DSNHYCRD plan and package. Expect return code 8 from this step if the plan or package does not exist.

Discontinue DB2-managed stored procedures

Support for DB2-managed stored procedures was deprecated in V8 and is eliminated in V9.

DSNTIPX (Routine parameters panel) changes:

The DB2 PROC NAME field is removed. It was used to set the DSN6SYSP.STORPROC DSNZPARM.

DSNTIJMV changes

The DSNSPAS address space procedure is eliminated.

DSNTIJUZ changes

The STORPROC DSNZPARM is removed from the DSN6SYSP macro expansion.

BSDS conversion required for V9 migration

DB2 V8 introduced a new, optional format for the BSDS that supports up to 10,000 archive log volumes and 93 active log data sets per log copy. The DSNJCNVB utility converts the BSDSs to the new format. In V9, the new BSDS format is mandatory.

In MIGRATE mode only, the installation CLIST inserts a new job step, DSNTCNVB, in job DSNTIJUZ to execute DSNJCNVB. Expect return code 888 from this step if the BSDSs have already been converted. DSNTIJUZ accepts a return code of 0 and 888 from step DSNTCNVB.

Archived

12.5.4 Panels in the MIGRATE flow

From the main panel, shown in Figure 12-13, you choose MIGRATE.

```
DSNTIPA1 DB2 VERSION 9 INSTALL, UPDATE, MIGRATE, AND ENFM - MAIN PANEL
===>

Check parameters and reenter to change:
 1  INSTALL TYPE          ===> MIGRATE   Install, Migrate, ENFM, or Update
 2  DATA SHARING          ===> NO        Yes or No (blank for ENFM or Update)

Enter the data set and member name for migration only. This is the name used
from a previous Installation/Migration from field 9 below:
 3  DATA SET(MEMBER) NAME ===> DB8R8.SDSNSAMP(DB8RURG)

For DB2 SMP/E libraries (SDSNLOAD, SDSNMACS, SDSNSAMP, SDSNCLST, etc.), enter:
 4  LIBRARY NAME PREFIX   ===> DB9B9
 5  LIBRARY NAME SUFFIX   ===>

For install data sets (NEW.SDSNSAMP, NEW.SDSNCLST, RUNLIB.LOAD, etc.), enter:
 6  DATA SET NAME PREFIX  ===> DB9B9.V9MIGRAT
 7  DATA SET NAME SUFFIX  ===>

Enter to set or save panel values (by reading or writing the named members):
 8  INPUT MEMBER NAME     ===> DSNTIDXA  Default parameter values
 9  OUTPUT MEMBER NAME    ===> DSNTIDV9  Save new values entered on panels
PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-13 Main panel DSNTIPA1

Figure 12-14 shows the first migration panel for ALIAS and SMS definitions. We specify NO for SMS.

```
DSNTIPA2            MIGRATE DB2 - DATA PARAMETERS PANEL 1
===>

Check parameters and reenter to change:

 1 CATALOG ALIAS       ===> DB8RU     Alias of VSAM catalog for DB2 subsystem
                                      data sets
 2 DEFINE CATALOG      ===> NO        YES or NO

 3 USE SMS             ===> NO        For installation-defined objects: NO, YES









PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-14 Panel DSNTIPA2

You then get to the second migration panel shown in Figure 12-15.

```
DSNTIPA3            MIGRATE DB2 - DATA PARAMETERS PANEL 2
===>

Check parameters and reenter to change:
 1 PERMANENT UNIT NAME    ==> 3390                  Device type for MVS catalog
                                                    and partitioned data sets
 2 TEMPORARY UNIT NAME    ==> VIO                   Device type for
                                                    temporary data sets

                                          ---------------- SMS -----------------
                              VOL/SER    DATA CLASS    MGMT CLASS    STOR CLASS
                              -------    ----------    ----------    ----------
 3 CLIST ALLOCATION       ==> SBOX9A  ==>           ==>           ==>
 4 NON-VSAM DATA          ==> SBOX9A  ==>           ==>           ==>
 5 VSAM CATALOG, DEFAULT, ==> SBOX9A  ==>           ==>           ==>
   AND WORK FILE DATABASE
 6 DIRECTORY AND CATALOG  ==> SBOX9A  ==>           ==>           ==>
   DATA
 7 DIRECTORY AND CATALOG  ==> SBOX9A  ==>           ==>           ==>
   INDEXES
 8 LOG COPY 1, BSDS 2     ==> SBOX9A  ==>           ==>           ==>
 9 LOG COPY 2, BSDS 1     ==> SBOX9A  ==>           ==>           ==>
PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-15 If you specity NO for field 3 USE SMS of panel DSNTIAP2

Archived

If we had specified SMS YES, we would have seen the panel in Figure 12-16.

```
DSNTIPA3            MIGRATE DB2 - DATA PARAMETERS PANEL 2
===>

Check parameters and reenter to change:
 1 PERMANENT UNIT NAME    ==> 3390                 Device type for MVS catalog
                                                   and partitioned data sets
 2 TEMPORARY UNIT NAME    ==> VIO                  Device type for
                                                   temporary data sets

                                        ---------------- SMS -----------------
                             VOL/SER    DATA CLASS    MGMT CLASS    STOR CLASS
                             -------    ----------    ----------    ----------
 3 CLIST ALLOCATION       ==> SBOX9A  ==>           ==>           ==>
 4 NON-VSAM DATA          ==> SBOX9A  ==>           ==>           ==>
 5 VSAM CATALOG, DEFAULT, ==> SBOX9A  ==>           ==>           ==>
   AND WORK FILE DATABASE
 6 DIRECTORY AND CATALOG  ==> SBOX9A  ==>           ==>           ==>
   DATA
 7 DIRECTORY AND CATALOG  ==> SBOX9A  ==>           ==>           ==>
   INDEXES
 8 LOG COPY 1, BSDS 2     ==> SBOX9A  ==>           ==>           ==>
 9 LOG COPY 2, BSDS 1     ==> SBOX9A  ==>           ==>           ==>
PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-16 If you specified YES for field 3 USE SMS of panel DSNTIAP2

Figure 12-15 on page 451 (If you specified NO for field 3 USE SMS of panel DSNTIAP2) and Figure 12-16 (If you specified YES for field 3 USE SMS of panel DSNTIAP2) look identical. However, in the first case DSNTIPA3 locks the fields for the SMS classes but in the second case they are open for entry.

Archive

The next panel is for bootstrap, logging and archiving data sets. See Figure 12-17.

```
DSNTIPH         MIGRATE DB2 - SYSTEM RESOURCE DATA SET NAMES
===>

Enter data below:

Bootstrap Data Sets (BSDS):
  1 COPY 1 NAME         ===> DB8RU.BSDS01
  2 COPY 2 NAME         ===> DB8RU.BSDS02

Active Logs:
  3 NUMBER OF COPIES    ===> 2        2 or 1.  Number of active log copies
  4 COPY 1 PREFIX       ===> DB8RU.LOGCOPY1
  5 COPY 2 PREFIX       ===> DB8RU.LOGCOPY2

Archive Logs:
  6 NUMBER OF COPIES    ===> 2        2 or 1.  Number of archive log copies
  7 COPY 1 PREFIX       ===> DB8RU.ARCHLOG1
  8 COPY 2 PREFIX       ===> DB8RU.ARCHLOG2
  9 TIMESTAMP ARCHIVES  ===> NO       NO, YES, or EXT (Extended date format)


PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-17 DSNTIPH

The next panel is about the allocation of DB2 libraries. See Figure 12-18.

```
DSNTIPT         MIGRATE DB2 - DATA SET NAMES PANEL 1
===>

Data sets allocated by the installation CLIST for edited output:
  1  TEMP CLIST LIBRARY ===> DB9B9.V9MIGRAT.NEW.SDSNTEMP
  2  SAMPLE LIBRARY     ===> DB9B9.V9MIGRAT.NEW.SDSNSAMP
Data sets allocated by the installation jobs:
  3  CLIST LIBRARY      ===> DB9B9.V9MIGRAT.NEW.SDSNCLST
  4  APPLICATION DBRM   ===> DB9B9.V9MIGRAT.DBRMLIB.DATA
  5  APPLICATION LOAD   ===> DB9B9.V9MIGRAT.RUNLIB.LOAD
  6  DECLARATION LIBRARY===> DB9B9.V9MIGRAT.SRCLIB.DATA
Data sets allocated by SMP/E and other methods:
  7  LINK LIST LIBRARY  ===> DB9B9.SDSNLINK
  8  LOAD LIBRARY       ===> DB9B9.SDSNLOAD
  9  MACRO LIBRARY      ===> DB9B9.SDSNMACS
 10  LOAD DISTRIBUTION  ===> DB9B9.ADSNLOAD
 11  EXIT LIBRARY       ===> DB9B9.SDSNEXIT
 12  DBRM LIBRARY       ===> DB9B9.SDSNDBRM
 13  IRLM LOAD LIBRARY  ===> DB9B9.SDXRRESL
 14  IVP DATA LIBRARY   ===> DB9B9.SDSNIVPD
 15  INCLUDE LIBRARY    ===> DB9B9.SDSNC.H

PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-18 DSNTIPT

Archived

The panel in Figure 12-19 is about allocation of languages.

```
DSNTIPU         MIGRATE DB2 - DATA SET NAMES PANEL 2
===>

Enter data set names below:
 1  IBM LE RUNTIME LIBRARY   ===> CEE.SCEERUN
 2  IBM LE LINK EDIT LIB     ===> CEE.SCEELKED
 3  IBM LE PRELINK MSG LIB   ===> CEE.SCEEMSGP

 4  HIGH LEVEL ASSEMBLER LIB ===>
 5  C/CPP COMPILER MODULE    ===> CBCDRVR
 6  C/CPP COMPILER LIBRARY   ===> CBC.SCBCCMP
 7  C/CPP HEADER LIBRARY     ===> CEE.SCEEH.H
 8  C/370 COMPILER MESSAGES  ===>
 9  CPP CLASS LIB HEADERS    ===> CBC.SCLBH.HPP
10  CPP AUTO CALL LIBRARY    ===> CEE.SCEECPP
11  CPP CLASS LIBRARY        ===> CBC.SCLBCPP
12  CPP PROCEDURE LIBRARY    ===> CBC.SCBCUTL
13  COBOL COMPILER LIBRARY   ===>
14  FORTRAN COMPILER LIBRARY ===>
15  FORTRAN LINK EDIT LIB    ===> SYS1.VSF2FORT
16  PL/I COMPILER LIBRARY    ===> IBMZ.SIBMZCMP

PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-19 DSNTIPU

We now list the allocation for other products to be used. See Figure 12-20.

```
DSNTIPW         MIGRATE DB2 - DATA SET NAMES PANEL 3
===>

Enter data set names below. Delete data set names for products not used.
 1  SYSTEM MACLIB        ===> SYS1.MACLIB
 2  SYSTEM PROCEDURES    ===> DB8RU.PROCLIB
 3  SORT LIBRARY         ===> SYS1.SORTLIB
 4  IMS RESLIB           ===>
 5  ISPF ISPLINK MODULE  ===> ISP.SISPLOAD
 6  GDDM MACLIB          ===> GDDM.SADMSAM
 7  GDDM LOAD MODULES    ===> GDDM.SADMMOD
 8  CICS LOAD LIBRARY    ===> CICSTS31.CICS.SDFHLOAD
 9  CICS MACRO LIBRARY   ===> CICSTS31.CICS.SDFHMAC
10  CICS COBOL LIBRARY   ===> CICSTS31.CICS.SDFHCOB
11  CICS PL/I  LIBRARY   ===> CICSTS31.CICS.SDFHPL1
12  CICS EXCI  LIBRARY   ===> CICSTS31.CICS.SDFHEXCI




PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-20 DSNTIPW

Archived

```
DSNTIPD         MIGRATE DB2 - SIZES
===>
Check numbers and reenter to change:

 1  DATABASES          ===> 200        In this subsystem
 2  TABLES             ===> 20         Per database (average)
 3  COLUMNS            ===> 10         Per table (average)
 4  VIEWS              ===> 3          Per table (average)
 5  TABLE SPACES       ===> 20         Per database (average)
 6  PLANS              ===> 200        In this subsystem
 7  PLAN STATEMENTS    ===> 30         SQL statements per plan (average)
 8  PACKAGES           ===> 300        In this subsystem
 9  PACKAGE STATEMENTS ===> 10         SQL statements per package (average)
10  PACKAGE LISTS      ===> 2          Package lists per plan (average)
11  EXECUTED STMTS     ===> 15         SQL statements executed (average)
12  TABLES IN STMT     ===> 2          Tables per SQL statement (average)
13  USER LOB VALUE STG ===> 10240      Max KB storage per user for LOB values
14  SYSTEM LOB VAL STG ===> 2048       Max MB storage per system for LOB values
15  USER XML VALUE STG ===> 40960      Max KB storage per user for LOB values
16  SYSTEM XML VAL STG ===> 10240      Max MB storage per system for LOB values
17  MAXIMUM LE TOKENS  ===> 20         Maximum tokens at any time.  0-50

PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-21 DSNTIPD

```
DSNTIP7         MIGRATE DB2 - SIZES PANEL 2
===>

Enter options for implicitly-created table spaces and indexes:
 1  DEFINE DATA SETS         ===> YES      When creating a table space or index
 2  USE DATA COMPRESSION     ===> NO       When creating a table space

Enter general options for table spaces below:
 3  TABLE SPACE ALLOCATION   ===> 0        Default space allocation in KB for
                                           table spaces
                                            (0 for DB2 default or 1-4194304)
 4  INDEX SPACE ALLOCATION   ===> 0        Default space allocation in KB for
                                            index spaces
                                            (0 for DB2 default or 1-4194304)
 5  VARY DS CONTROL INTERVAL ===> YES      Optimize VSAM CONTROL INTERVAL to
                                            page size for data set allocation
 6  OPTIMIZE EXTENT SIZING   ===> YES      Use sliding secondary quantity
                                            for DB2-managed data sets



PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-22 DNTIP7

```
DSNTIP9                MIGRATE DB2 - WORK FILE DATABASE
===>

Enter work file configuration options below:
 1 TEMP 4K SPACE      ===> 0           Amount of 4K-page work space (MB)
 2 TEMP 4K TBL SPACES ===> 0           Number of table spaces for 4K-page data
 3 TEMP 4K SEG SIZE   ===> 16          Segment size of 4K-page table spaces

 4 TEMP 32K SPACE     ===> 0           Amount of 32K-page work space (MB)
 5 TEMP 32K TBL SPACES===> 0           Number of table spaces for 32K-page data
 6 TEMP 32K SEG SIZE  ===> 16          Segment size of 32K-page table spaces

 7 MAX TEMP STG/AGENT ===> 0           Maximum MB of temp storage space
                                       that can be used by a single agent









PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-23 DSNTIP9

```
DSNTIP9                MIGRATE DB2 - WORK FILE DATABASE
===>

Enter work file configuration options below:
 1 TEMP 4K SPACE      ===> 0           Amount of 4K-page work space (MB)
 2 TEMP 4K TBL SPACES ===> 0           Number of table spaces for 4K-page data
 3 TEMP 4K SEG SIZE   ===> 16          Segment size of 4K-page table spaces

 4 TEMP 32K SPACE     ===> 500         Amount of 32K-page work space (MB)
 5 TEMP 32K TBL SPACES===> 5           Number of table spaces for 32K-page data
 6 TEMP 32K SEG SIZE  ===> 16          Segment size of 32K-page table spaces

 7 MAX TEMP STG/AGENT ===> 100         Maximum MB of temp storage space
                                       that can be used by a single agent









PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-24 DSNTIP9 with 32 KB table spaces

Archive

```
DSNTIP9                INSTALL DB2 - WORK FILE DATABASE
===>

Enter work file configuration options below:
 1 TEMP 4K SPACE      ===> 8000        Amount of 4K-page work space (MB)
 2 TEMP 4K TBL SPACES ===> 10          Number of table spaces for 4K-page data
 3 TEMP 4K SEG SIZE   ===> 16          Segment size of 4K-page table spaces

 4 TEMP 32K SPACE     ===> 500         Amount of 32K-page work space (MB)
 5 TEMP 32K TBL SPACES===> 5           Number of table spaces for 32K-page data
 6 TEMP 32K SEG SIZE  ===> 16          Segment size of 32K-page table spaces

 7 MAX TEMP STG/AGENT ===> 100         Maximum MB of temp storage space
                                       that can be used by a single agent



PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-25 DSNTIP 9 with Max Temp

When in INSTALL mode, you must enter a number greater than 0 for fields 1–6 in DSNTIP9.

```
DSNTIPE        MIGRATE DB2 - THREAD MANAGEMENT
===>

Check numbers and reenter to change:
 1  DATABASES             ===> 100       Concurrently in use
 2  MAX USERS             ===> 70        Concurrently running in DB2
 3  MAX REMOTE ACTIVE     ===> 200       Maximum number of active
                                          database access threads
 4  MAX REMOTE CONNECTED  ===> 200       Maximum number of remote DDF
                                          connections that are supported
 5  MAX TSO CONNECT       ===> 40        Users on QMF or in DSN command
 6  MAX BATCH CONNECT     ===> 20        Users in DSN command or utilities
 7  SEQUENTIAL CACHE      ===> BYPASS    3990 storage for sequential IO.
                                         Values are SEQ or BYPASS.
 8  MAX KEPT DYN STMTS    ===> 5000      Maximum number of prepared dynamic
                                         statements saved past commit points
 9  CONTRACT THREAD STG   ===> NO        Periodically free unused thread stg
10  MANAGE THREAD STORAGE ===> NO        Manage thread stg to minimize size
11  LONG-RUNNING READER   ===> 0         Minutes before read claim warning
12  PAD INDEXES BY DEFAULT===> NO        Pad new indexes by default
13  MAX OPEN FILE REFS    ===> 50        Maximum concurrent open data sets

PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-26 DSNTIPE

Archived

```
DSNTIP1         MIGRATE DB2 - BUFFER POOL SIZES - PANEL 1
===>
 1 DEFAULT  4-KB BUFFER POOL FOR USER DATA ===> BP0        BP0    - BP49
 2 DEFAULT  8-KB BUFFER POOL FOR USER DATA ===> BP8K0      BP8K0  - BP8K9
 3 DEFAULT 16-KB BUFFER POOL FOR USER DATA ===> BP16K0     BP16K0 - BP16K9
 4 DEFAULT 32-KB BUFFER POOL FOR USER DATA ===> BP32K      BP32K  - BP32K9
 5 DEFAULT BUFFER POOL FOR USER LOB DATA   ===> BP0        BP0    - BP32K9
 6 DEFAULT BUFFER POOL FOR USER XML DATA   ===> BP16K0     BP16K0 - BP16K9
 7 DEFAULT BUFFER POOL FOR USER INDEXES    ===> BP0        BP0    - BP32K9

Enter buffer pool sizes in number of pages.
 8 BP0  ==> 20000        18 BP10 ==> 0            28 BP20 ==> 0
 9 BP1  ==> 40000        19 BP11 ==> 0            29 BP21 ==> 0
10 BP2  ==> 40000        20 BP12 ==> 0            30 BP22 ==> 0
11 BP3  ==> 0            21 BP13 ==> 0            31 BP23 ==> 0
12 BP4  ==> 0            22 BP14 ==> 0            32 BP24 ==> 0
13 BP5  ==> 0            23 BP15 ==> 0            33 BP25 ==> 0
14 BP6  ==> 0            24 BP16 ==> 0            34 BP26 ==> 0
15 BP7  ==> 3000         25 BP17 ==> 0            35 BP27 ==> 0
16 BP8  ==> 0            26 BP18 ==> 0            36 BP28 ==> 0
17 BP9  ==> 0            27 BP19 ==> 0            37 BP29 ==> 0

PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-27 DSNTIP1

See Figure 12-28 for the continuation panel.

```
DSNTIP2         MIGRATE DB2 - BUFFER POOL SIZES - PANEL 2
===>

Enter buffer pool sizes in number of pages.

 1 BP30 ==> 0          18 BP47   ==> 0          35 BP16K4  ==> 0
 2 BP31 ==> 0          19 BP48   ==> 0          36 BP16K5  ==> 0
 3 BP32 ==> 0          20 BP49   ==> 0          37 BP16K6  ==> 0
 4 BP33 ==> 0          21 BP8K0  ==> 1000       38 BP16K7  ==> 0
 5 BP34 ==> 0          22 BP8K1  ==> 0          39 BP16K8  ==> 0
 6 BP35 ==> 0          23 BP8K2  ==> 0          40 BP16K9  ==> 0
 7 BP36 ==> 0          24 BP8K3  ==> 0          41 BP32K   ==> 250
 8 BP37 ==> 0          25 BP8K4  ==> 0          42 BP32K1  ==> 0
 9 BP38 ==> 0          26 BP8K5  ==> 0          43 BP32K2  ==> 0
10 BP39 ==> 0          27 BP8K6  ==> 0          44 BP32K3  ==> 0
11 BP40 ==> 0          28 BP8K7  ==> 0          45 BP32K4  ==> 0
12 BP41 ==> 0          29 BP8K8  ==> 0          46 BP32K5  ==> 0
13 BP42 ==> 0          30 BP8K9  ==> 0          47 BP32K6  ==> 0
14 BP43 ==> 0          31 BP16K0 ==> 500        48 BP32K7  ==> 0
15 BP44 ==> 0          32 BP16K1 ==> 0          49 BP32K8  ==> 0
16 BP45 ==> 0          33 BP16K3 ==> 0          50 BP32K9  ==> 0
17 BP46 ==> 0          34 BP16K3 ==> 0

PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-28 DSNTIP2

```
DSNTIPN         MIGRATE DB2 - TRACING PARAMETERS
===>

Enter data below:

 1 AUDIT TRACE        ===> NO          Audit classes to start. NO,YES,list
 2 TRACE AUTO START   ===> NO          Global classes to start. YES, NO, list
 3 TRACE SIZE         ===> 65536       Trace table size in bytes.  4K-396K
 4 SMF ACCOUNTING     ===> 1           Accounting classes to start. NO,YES,list
 5 SMF STATISTICS     ===> YES         Statistics classes to start. NO,YES,list
 6 STATISTICS TIME    ===> 15          Time interval in minutes.  1-1440
 7 STATISTICS SYNC    ===> 0           Synchronization within the hour. NO,0-59
 8 DATASET STATS TIME ===> 5           Time interval in minutes.  1-1440
 9 MONITOR TRACE      ===> NO          Monitor classes to start. NO, YES, list
10 MONITOR SIZE       ===> 256K        Default monitor buffer size. 256K-16M
11 UNICODE IFCIDS     ===> NO          Include UNICODE data when writing IFCIDS
12 DDF/RRSAF ACCUM    ===> NO          Rollup accting for DDF/RRSAF. NO,2-65535
13 AGGREGATION FIELDS ===> 0           Rollup accting aggregation fields, 0-17



PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-29 DSNTIPN

```
DSNTIPO         MIGRATE DB2 - OPERATOR FUNCTIONS
===>

Enter data below:

 1  WTO ROUTE CODES    ===> 1
                                  Routing codes for WTORs
 2  RECALL DATABASE    ===> YES     Use DFHSM automatic recall.  YES or NO
 3  RECALL DELAY       ===> 120     Seconds to wait for automatic recall
 4  RLF AUTO START     ===> NO      Resource Limit Facility.  NO or YES
 5  RLST NAME SUFFIX   ===> 01      Resource Limit Spec. Table (RLST)
 6  RLST ACCESS ERROR  ===> NOLIMIT NOLIMIT, NORUN, or 1-5000000
 7  PARAMETER MODULE   ===> DSNZPARM Name of DB2 subsystem parameter module
 8  AUTO BIND          ===> YES     Use automatic bind. YES, NO, or COEXIST
 9  EXPLAIN PROCESSING ===> YES     Explain allowed on auto bind? YES or NO
10  DPROP SUPPORT      ===> 3       1=NO  2=ONLY  3=ANY
11  SITE TYPE          ===> LOCALSITE   LOCALSITE OR RECOVERYSITE
12  TRACKER SITE       ===> NO      Tracker DB2 system.  NO or YES
13  READ COPY2 ARCHIVE ===> NO      Read COPY2 archives first.  NO or YES
14  REAL TIME STATS    ===> 30      RTS time interval in minutes. 1-1440


PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-30 DSNTIPO

Archive

```
DSNTIPF         MIGRATE DB2 - APPLICATION PROGRAMMING DEFAULTS PANEL 1
===>

Enter data below:
 1 LANGUAGE DEFAULT      ===> IBMCOB   ASM,C,CPP,IBMCOB,FORTRAN,PLI
 2 DECIMAL POINT IS      ===> .        . or ,
 3 STRING DELIMITER      ===> DEFAULT  DEFAULT, " or ' (IBMCOB only)
 4 SQL STRING DELIMITER  ===> DEFAULT  DEFAULT, " or '
 5 DIST SQL STR DELIMTR  ===> '        ' or "
 6 MIXED DATA            ===> NO       NO or YES for mixed DBCS data
 7 EBCDIC CCSID          ===> 37       CCSID of SBCS or mixed data. 1-65533.
 8 ASCII CCSID           ===> 437      CCSID of SBCS or mixed data. 1-65533.
 9 UNICODE CCSID         ===> 1208     CCSID of UNICODE UTF-8 data
10 DEF ENCODING SCHEME   ===> EBCDIC   EBCDIC, ASCII, or UNICODE
11 APPLICATION ENCODING  ===> EBCDIC   EBCDIC, ASCII, UNICODE, ccsid (1-65533)
12 LOCALE LC_CTYPE     ===>
13 DECFLOAT ROUNDING MODE===> ROUND_HALF_EVEN




PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-31 DSNTIPF

```
DSNTIP4         MIGRATE DB2 - APPLICATION PROGRAMMING DEFAULTS PANEL 2
===>

Enter data below:
 1  MINIMUM DIVIDE SCALE ===> NO       NO or YES for a minimum of 3 digits
                                       to right of decimal after division
 2  DECIMAL ARITHMETIC   ===> DEC15    DEC15, DEC31, 15, 31 or DPP.S
 3  USE FOR DYNAMICRULES ===> YES      YES or NO
 4  DESCRIBE FOR STATIC  ===> NO       Allow DESCRIBE for STATIC SQL. NO or YES
 5  DATE FORMAT          ===> ISO      ISO, JIS, USA, EUR, LOCAL
 6  TIME FORMAT          ===> ISO      ISO, JIS, USA, EUR, LOCAL
 7  LOCAL DATE LENGTH    ===> 0        10-254 or 0 for no exit
 8  LOCAL TIME LENGTH    ===> 0         8-254 or 0 for no exit
 9  STD SQL LANGUAGE     ===> NO       NO or YES
10  PAD NUL-TERMINATED   ===> NO       NO or YES









PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-32 DSNTIP4

```
DSNTIP8         MIGRATE DB2 - PERFORMANCE AND OPTIMIZATION
===>

Enter data below:
 1  CURRENT DEGREE       ===> 1        1 or ANY
 2  CACHE DYNAMIC SQL    ===> YES      NO or YES
 3  OPTIMIZATION HINTS   ===> YES      Enable optimization hints.  NO or YES
 4  VARCHAR FROM INDEX   ===> NO       Get VARCHAR data from index. NO or YES
 5  MAX DEGREE           ===> 0        Maximum degree of parallelism. 0-254
 6  UPDATE PART KEY COLS ===> YES      Allow update of partitioning key
                                       columns.  YES, NO, SAME
 7  LARGE EDM BETTER FIT ===> NO       NO or YES
 8  IMMEDIATE WRITE      ===> NO       NO or YES
 9  EVALUATE UNCOMMITTED ===> NO       Evaluate uncommitted data. NO or YES
10  SKIP UNCOMM INSERTS  ===> NO       Skip uncommitted inserts. NO or YES
11  CURRENT REFRESH AGE  ===> 0        0 or ANY
12  CURRENT MAINT TYPES  ===> SYSTEM   NONE, SYSTEM, USER, ALL
13  STAR JOIN QUERIES    ===> DISABLE  DISABLE, ENABLE, 1-32768
14  MAX DATA CACHING     ===> 20       0-512

PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-33 DSNTIP8

```
DSNTIPI         MIGRATE DB2 - IRLM PANEL 1
===>

Enter data below:

 1 INSTALL IRLM       ===> YES         IRLM is required for DB2. Should the
                                       IRLM distributed with DB2 be installed?
 2 SUBSYSTEM NAME     ===> ID8R        IRLM MVS subsystem name
 3 RESOURCE TIMEOUT   ===> 60          Seconds to wait for unavailable resource
 4 AUTO START         ===> YES         Start IRLM if not up. YES or NO
 5 PROC NAME          ===> DB8RIRLM    Name of start procedure for IRLM
 6 TIME TO AUTOSTART  ===> 300         Time DB2 will wait for IRLM autostart
 7 U LOCK FOR RR/RS   ===> NO          Lock mode for update cursor with
                                       RR or RS isolation. YES or NO
 8 X LOCK FOR SEARCHED U/D ===> NO     Use X lock for searched updates or
                                       deletes. NO, YES, or TARGET
 9 START IRLM CTRACE  ===> NO          Start IRLM component traces at startup
                                       Blank, NO, YES, or 10 - 255
10 IMS BMP TIMEOUT    ===> 4           Timeout multiplier for BMP. 1-254
11 DL/I BATCH TIMEOUT ===> 6           Timeout multiplier for DL/I. 1-254
12 RETAINED LOCK TIMEOUT ===> 0        Retained lock timeout multiplier. 0-254

PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-34 DSNTIPI

Archived

```
DSNTIPJ         MIGRATE DB2 - IRLM PANEL 2
===>

Enter data below:
 1  PAGE PROTECT           ===> YES        Page protect common modules (YES,NO)
 2  MAX STORAGE FOR LOCKS  ===> 2          Control block storage in GB (2-100)
 3  LOCKS PER TABLE(SPACE) ===> 1000       Max before lock escalation (0-100M)
 4  LOCKS PER USER         ===> 10000      Max before resource unavail (0-100M)
 5  DEADLOCK TIME          ===> 5          Detection interval (1-5 seconds or
                                           100-5000 milliseconds)

For DB2 data sharing ONLY enter data below:

 6  DEADLOCK CYCLE         ===> 1         Number of LOCAL cycles before GLOBAL
 7  MEMBER IDENTIFIER      ===> 1         Member ID for this IRLM (1-255)
 8  IRLM XCF GROUP NAME    ===> DXRGROUP  Name of IRLM XCF group
 9  LOCK ENTRY SIZE        ===> 2         Initial allocation in bytes (2,4,8)
10  NUMBER OF LOCK ENTRIES ===> 0         Lock table entries (0-1024)
11  DISCONNECT IRLM        ===> YES       Disconnect automatically  (YES, NO)


PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-35 DSNTIPJ

```
DSNTIP6         MIGRATE DB2 - DB2 UTILITIES PARAMETERS
===>

Enter system-level backup options for RESTORE SYSTEM and RECOVER below:
 1 SYSTEM-LEVEL BACKUPS ===> NO         As a recovery base: NO or YES
 2 RESTORE/RECOVER      ===> NO         From dump: NO or YES
 3 DUMP CLASS NAME      ===>            For RESTORE/RECOVER from dump
 4 MAXIMUM TAPE UNITS   ===> NOLIMIT    For RESTORE SYSTEM: NOLIMIT or 1-255

Enter other DB2 Utilities options below:
 5 TEMP DS UNIT NAME    ===> VIO        Device for temporary utility data sets
 6 UTILITY CACHE OPTION ===> NO         3990 storage for DB2 utility IO
 7 STATISTICS HISTORY   ===> NONE       Default for collection of stats history
 8 STATISTICS ROLLUP    ===> NO         Allow statistics aggregation: NO or YES
 9 STATISTICS CLUSTERING===> ENHANCED   For RUNSTATS (ENHANCED or STANDARD)
10 UTILITY TIMEOUT      ===> 6          Utility wait time multiplier




PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-36 DSNTIP6

```
DSNTIPP         MIGRATE DB2 - PROTECTION
===>

Enter data below:

 1  ARCHIVE LOG RACF  ===> NO         RACF protect archive log data sets
 2  USE PROTECTION    ===> YES        DB2 authorization enabled.  YES or NO
 3  SYSTEM ADMIN 1    ===> SYSADM     Authid of system administrator
 4  SYSTEM ADMIN 2    ===> SYSAD2     Authid of system administrator
 5  SYSTEM OPERATOR 1 ===> SYSOPR     Authid of system operator
 6  SYSTEM OPERATOR 2 ===> SYSOPR     Authid of system operator
 7  UNKNOWN AUTHID    ===> IBMUSER    Authid of default (unknown) user
 8  RESOURCE AUTHID   ===> SYSIBM     Authid of Resource Limit Table creator
 9  BIND NEW PACKAGE  ===> BINDADD    Authority required: BINDADD or BIND
10  PLAN AUTH CACHE   ===> 1024       Size in bytes per plan (0 - 4096)
11  PACKAGE AUTH CACHE===> 102400     Global - size in bytes (0-5M)
12  ROUTINE AUTH CACHE===> 102400     Global - size in bytes (0-5M)
13  DBADM CREATE AUTH ===> NO         DBA can create views/aliases for others
14  AUTH EXIT LIMIT   ===> 10         Access control exit shutdown threshold



PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-37 DSNTIPP

```
DSNTIPM         MIGRATE DB2 - MVS PARMLIB UPDATES
===>

Check data and reenter to change:

 1  SUBSYSTEM NAME       ===> DB8R      Name for connecting to DB2
 2  COMMAND PREFIX       ===> -DB8R     DB2 subsystem command prefix
 3  SUBSYSTEM MEMBER     ===> 00        xx in IEFSSNxx
 4  SUBSYSTEM SEQUENCE   ===> 88888888  Sequence number for insertion
 5  AUTH MEMBER          ===> 00        xx in IEAAPFxx APF member name
 6  AUTH SEQUENCE        ===> 88888888  Sequence number for insertion
 7  LINK LIST ENTRY      ===> 00        xx in LNKLSTxx for SDSNLINK
 8  LINK LIST SEQUENCE   ===> 88888888  Sequence number for insertion
 9  COMMAND SCOPE        ===> STARTED   SYSTEM, SYSPLEX, or STARTED
10  SUPPRESS SOFT ERRORS ===> YES       Suppress logrec recording. YES or NO




PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-38 DSNTIPM

Archive

```
DSNTIPL         MIGRATE DB2 - ACTIVE LOG DATA SET PARAMETERS
===>

Enter data below:

  1 NUMBER OF LOGS      ===> 3         Data sets per active log copy (2-31)
  2 OUTPUT BUFFER       ===> 4096000   Size in bytes (40K-400000K)
  3 ARCHIVE LOG FREQ    ===> 24        Hours per archive run
  4 UPDATE RATE         ===> 3600      Updates, inserts, and deletes per hour
  5 LOG APPLY STORAGE   ===> 0M        Maximum ssnmDBM1 storage in MB for
                                       fast log apply (0-100M)
  6 CHECKPOINT FREQ     ===> 50000     Log records or minutes per checkpoint
  7 FREQUENCY TYPE      ===> LOGRECS   CHECKPOINT FREQ units. LOGRECS, MINUTES
  8 UR CHECK FREQ       ===> 0         Checkpoints to enable UR check. 0-255
  9 UR LOG WRITE CHECK  ===> 0K        Log Writes to enable UR check. 0-1000K
 10 LIMIT BACKOUT       ===> AUTO      Limit backout processing. AUTO,YES,NO
 11 BACKOUT DURATION    ===> 5         Checkpoints processed during backout if
                                       LIMIT BACKOUT = AUTO or YES.  0-255
 12 RO SWITCH CHKPTS    ===> 5         Checkpoints to read-only switch. 1-32767
 13 RO SWITCH TIME      ===> 10        Minutes to read-only switch. 1-32767
 14 LEVELID UPDATE FREQ ===> 5         Checkpoints between updates. 0-32767

PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-39 DSNTIPL

```
DSNTIPA         MIGRATE DB2 - ARCHIVE LOG DATA SET PARAMETERS
===>

Enter data below:

  1  ALLOCATION UNITS ===> BLK       Blk, Trk, or Cyl
  2  PRIMARY QUANTITY ===>           Primary space allocation
  3  SECONDARY QTY.   ===>           Secondary space allocation
  4  CATALOG DATA     ===> YES       YES or NO to catalog archive data sets
  5  DEVICE TYPE 1    ===> 3390      Unit name for COPY1 archive logs
  6  DEVICE TYPE 2    ===>           Unit name for COPY2 archive logs
  7  BLOCK SIZE       ===> 28672     Rounded up to 4096 multiple
  8  READ TAPE UNITS  ===> 2         Number of allocated read tape units
  9  DEALLOC PERIOD   ===> 0         Time interval to deallocate tape units
 10  RECORDING MAX    ===> 1000      Number of data sets recorded in BSDS
 11  WRITE TO OPER    ===> YES       Issue WTOR before mount for archive
 12  WTOR ROUTE CODE  ===> 1,3,4
 13  RETENTION PERIOD ===> 9999      Days to retain archive log data sets
 14  QUIESCE PERIOD   ===> 5         Maximum quiesce interval (1-999)
 15  COMPACT DATA     ===> NO        YES or NO for data compaction
 16  SINGLE VOLUME    ===> NO        Single volume for DASD archives. NO or YES
PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-40 DSNTIPA

```
DSNTIPS         MIGRATE DB2 - DATABASES AND SPACES TO START AUTOMATICALLY
===>

Enter data below:

 1  ===> RESTART       RESTART or DEFER the objects named below.
    The objects to restart or defer may be ALL in item 2, a database
    name, or database name.space name.

 2  ==> ALL              14  ==>                  26  ==>
 3  ==>                  15  ==>                  27  ==>
 4  ==>                  16  ==>                  28  ==>
 5  ==>                  17  ==>                  29  ==>
 6  ==>                  18  ==>                  30  ==>
 7  ==>                  19  ==>                  31  ==>
 8  ==>                  20  ==>                  32  ==>
 9  ==>                  21  ==>                  33  ==>
10  ==>                  22  ==>                  34  ==>
11  ==>                  23  ==>                  35  ==>
12  ==>                  24  ==>                  36  ==>
13  ==>                  25  ==>                  37  ==>

PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-41 DSNTIPS

```
DSNTIPR         MIGRATE DB2 - DISTRIBUTED DATA FACILITY
===>
DSNT512I Warning: Enter unique names for LUNAME and LOCATION NAME
Enter data below:

 1 DDF STARTUP OPTION   ===> AUTO      NO, AUTO, or COMMAND
 2 DB2 LOCATION NAME    ===> DB8R              The name other DB2s use to
                                               refer to this DB2
 3 DB2 NETWORK LUNAME   ===> SCPDB8R   The name VTAM uses to refer to this DB2
 4 DB2 NETWORK PASSWORD ===>           Password for DB2's VTAM application
 5 RLST ACCESS ERROR    ===> NOLIMIT   NOLIMIT, NORUN, or 1-5000000
 6 RESYNC INTERVAL      ===> 2         Minutes between resynchronization period
 7 DDF THREADS          ===> ACTIVE    Status of a qualifying database access
                                       thread after commit. ACTIVE or INACTIVE.
 8 MAX INACTIVE DBATS   ===> 0         Max inactive database activity threads
 9 DB2 GENERIC LUNAME   ===>           Generic VTAM LU name for this DB2
                                       subsystem or data sharing group
10 IDLE THREAD TIMEOUT  ===> 0         0 or seconds until dormant server ACTIVE
                                       thread will be terminated (0-9999).
11 EXTENDED SECURITY    ===> NO        Allow change password and descriptive
                                       security error codes.  YES or NO.

PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-42 DSNTIPR

```
DSNTIP5         MIGRATE DB2 - DISTRIBUTED DATA FACILITY PANEL 2
===>

Enter data below:
 1 DRDA PORT           ===> 38180   TCP/IP port number for DRDA clients.
                                    1-65534 (446 is reserved for DRDA)
 2 SECURE PORT         ===>         TCP/IP port number for secure DRDA
                                    clients. 1-65534 (448 is reserved
                                    for DRDA using SSL)
 3 RESYNC PORT         ===> 38181   TCP/IP port for 2-phase commit. 1-65534
 4 TCP/IP ALREADY VERIFIED ===> NO  Accept requests containing only a
                                    userid (no password)? YES or NO
 5 EXTRA BLOCKS REQ    ===> 100     Maximum extra query blocks when DB2 acts
                                    as a requester. 0-100
 6 EXTRA BLOCKS SRV    ===> 100     Maximum extra query blocks when DB2 acts
                                    as a server. 0-100
 7 AUTH AT HOP SITE    ===> BOTH    Authorization at hop site. BOTH or RUNNER
 8 TCP/IP KEEPALIVE    ===> ENABLE  ENABLE, DISABLE, or 1-65534
 9 POOL THREAD TIMEOUT ===> 120     0-9999 seconds


PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-43 DSNTIP5

```
DSNTIPX         MIGRATE DB2 - ROUTINE PARAMETERS
===>

Scrolling backward may change fields marked with asterisks
Enter data below:

* 1 WLM PROC NAME    ===> DB8RWLM    WLM-established stored procedure JCL PROC
  2 NUMBER OF TCBS   ===> 8          Number of concurrent TCBs (1-100)
  3 MAX ABEND COUNT  ===> 0          Allowable ABENDs for a procedure (0-255)
  4 TIMEOUT VALUE    ===> 180        Seconds to wait before SQL CALL or
                                     function invocation fails (5-1800,NOLIMIT)
  5 WLM ENVIRONMENT  ===>                              Default WLM env name
  6 MAX OPEN CURSORS ===> 500        Maximum open cursors per thread
  7 MAX STORED PROCS ===> 2000       Maximum active stored procs per thread









PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-44 DSNTIPX

```
DSNTIPZ         MIGRATE DB2 - DATA DEFINITION CONTROL SUPPORT
===>

Enter data below:

 1 INSTALL DD CONTROL SUPT. ===> NO        YES - activate the support
                                           NO  - omit DD control support
 2 CONTROL ALL APPLICATIONS ===> NO        YES or NO
 3 REQUIRE FULL NAMES       ===> YES       YES or NO
 4 UNREGISTERED DDL DEFAULT ===> ACCEPT    Action for unregistered DDL:
                                           ACCEPT - allow it
                                           REJECT - prohibit it
                                           APPL   - consult ART
 5 ART/ORT ESCAPE CHARACTER ===>           Used in ART/ORT Searches
 6 REGISTRATION OWNER       ===> DSNRGCOL         Qualifier for ART and ORT
 7 REGISTRATION DATABASE    ===> DSNRGFDB         Database name
 8 APPL REGISTRATION TABLE  ===> DSN_REGISTER_APPL Table name
 9 OBJT REGISTRATION TABLE  ===> DSN_REGISTER_OBJT Table name

Note: ART = Application Registration Table
      ORT = Object      Registration Table

PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-45 DSNTIPZ

```
DSNTIPY         MIGRATE DB2 - JOB EDITING
===>

Enter data below:

 1  REMOTE LOCATION   ===>                Remote location for sample
                                          distributed applications

Enter job card information for install and sample jobs:

 2 ===> //DB8R JOB (999,NNN),'DB8R MIGRATE',CLASS=A,
 3 ===> // MSGCLASS=T,NOTIFY=MYUSER8,TIME=1440,REGION=0M
 4 ===> /*JOBPARM SYSAFF=SC63,L=9999
 5 ===> // JCLLIB ORDER=(DB8RU.PROCLIB)
 6 ===>
 7 ===>



PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-46 DSNTIPY

Archived

```
DSNTIPC         MIGRATE DB2 - CLIST CALCULATIONS - PANEL 1
===>

 You can update the DSMAX, EDMPOOL, EDMPOOL STATEMENT CACHE (if CACHE DYNAMIC
 is YES), EDM DBD CACHE, SORT POOL, and RID POOL sizes if necessary.
                                        Calculated   Override

   1  DSMAX - MAXIMUM OPEN DATA SETS    =      9960                  (1-100000)
   2  DSNT485I  EDMPOOL STORAGE SIZE    =     18142 K            K
   3  DSNT485I  EDM STATEMENT CACHE     =     56693 K            K
   4  DSNT485I  EDM DBD CACHE           =     11700 K            K
   5  DSNT485I  SKELETON POOL SIZE      =      5120 K            K
   6 DSNT485I  BUFFER POOL SIZE        =       101 M
   7 DSNT485I  SORT POOL SIZE          =      2000 K            K
   8 DSNT485I  RID POOL SIZE           =      8000 K            K
   9 DSNT485I  DATA SET STORAGE SIZE   =     17928 K
  10 DSNT485I  CODE STORAGE SIZE       =     38200 K
  11 DSNT485I  WORKING STORAGE SIZE    =     55800 K
  12 DSNT486I  TOTAL MAIN STORAGE      =       238 M
  13 DSNT487I  TOTAL STORAGE BELOW 16M =      1159 K WITH SWA ABOVE 16M LINE
  14 DSNT438I  IRLM LOCK MAXIMUM SPACE =         2 G, AVAILABLE =   2 G


PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-47 DSNTIPC

```
DSNT478I BEGINNING EDITED DATA SET OUTPUT
DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTIJMV)', INSTALL JCL
DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTIJIN)', INSTALL JCL
DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTIJTC)', INSTALL JCL
DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTIJTM)', INSTALL JCL
DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTIJIC)', INSTALL JCL
DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTIJVC)', INSTALL JCL
DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTIJSG)', INSTALL JCL
DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTIJOS)', INSTALL JCL
DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTIJEX)', INSTALL JCL
DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTIJPD)', INSTALL JCL
DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNTEMP(DSNU)',     CLIST
DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNTEMP(DSNH)',     CLIST
DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNTEMP(DSNHC)',    CLIST
DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNTEMP(DSNEMC01)', CLIST
DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTIJCX)', MIGRATE JCL
DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTIJRI)', INSTALL JCL
DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTIJFV)', FALL BACK JCL
DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTIJUZ)', INSTALL JCL
***
```

Figure 12-48 Output of CLIST calculations

```
DSNTIPA1 DB2 VERSION 9 INSTALL, UPDATE, MIGRATE, AND   EDITED JCL SUCCESSFULLY
===>

Check parameters and reenter to change:
 1  INSTALL TYPE          ===> MIGRATE   Install, Migrate, ENFM, or Update
 2  DATA SHARING          ===> NO        Yes or No (blank for ENFM or Update)

Enter the data set and member name for migration only. This is the name used
from a previous Installation/Migration from field 9 below:
 3  DATA SET(MEMBER) NAME ===> DB8R8.SDSNSAMP(DB8RURG)

For DB2 SMP/E libraries (SDSNLOAD, SDSNMACS, SDSNSAMP, SDSNCLST, etc.), enter:
 4  LIBRARY NAME PREFIX   ===> DB9B9
 5  LIBRARY NAME SUFFIX   ===>

For install data sets (NEW.SDSNSAMP, NEW.SDSNCLST, RUNLIB.LOAD, etc.), enter:
 6  DATA SET NAME PREFIX  ===> DB9B9.V9MIGRAT
 7  DATA SET NAME SUFFIX  ===>

Enter to set or save panel values (by reading or writing the named members):
 8  INPUT MEMBER NAME     ===>           Default parameter values
 9  OUTPUT MEMBER NAME    ===> DSNTIDV9  Save new values entered on panels
PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-49 DSNTIPA1

Archived

12.5.5 Panel in the INSTALL flow

Here we show the sequence of panels in case of a new DB2 installation. We start from the main panel and choose INSTALL. See Figure 12-50.

```
DSNTIPA1 DB2 VERSION 9 INSTALL, UPDATE, MIGRATE, AND ENFM - MAIN PANEL
===>
Check parameters and reenter to change:
 1  INSTALL TYPE          ===> INSTALL   Install, Migrate, ENFM, or Update
 2  DATA SHARING          ===> NO        Yes or No (blank for ENFM or Update)

Enter the data set and member name for migration only. This is the name used
from a previous Installation/Migration from field 9 below:
 3  DATA SET(MEMBER) NAME ===>

For DB2 SMP/E libraries (SDSNLOAD, SDSNMACS, SDSNSAMP, SDSNCLST, etc.), enter:
 4  LIBRARY NAME PREFIX   ===> DB9B9
 5  LIBRARY NAME SUFFIX   ===>

For install data sets (NEW.SDSNSAMP, NEW.SDSNCLST, RUNLIB.LOAD, etc.), enter:
 6  DATA SET NAME PREFIX  ===> DB9B9.V9MIGRAT
 7  DATA SET NAME SUFFIX  ===>

Enter to set or save panel values (by reading or writing the named members):
 8  INPUT MEMBER NAME     ===>          Default parameter values
 9  OUTPUT MEMBER NAME    ===> DSNTIDV9  Save new values entered on panels
PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-50 Main panel for INSTALL

```
DSNTIPC1       INSTALL DB2 - CLIST CALCULATIONS - PANEL 2
===>
1 DSNT488I  DATA SETS CREATED BY THE INSTALLATION CLIST
            WILL REQUIRE AT LEAST 1000 4K BLOCKS (83 TRACKS)
  2 DSNT488I  OTHER NON-VSAM DATA SETS CREATED BY INSTALLATION AND IVP JOBS
            WILL REQUIRE AT LEAST 1000 4K BLOCKS (83 TRACKS)
  3 DSNT488I  DATA SETS FOR STORAGE GROUPS CREATED BY INSTALLATION AND IVP JOBS
            WILL REQUIRE AT LEAST 2176250 4K BLOCKS (181354 TRACKS)
  4 DSNT488I  VSAM DATA SETS CREATED FOR THE DB2 CATALOG AND DIRECTORY
            WILL REQUIRE AT LEAST 45788 4K BLOCKS (3815 TRACKS)
  5 DSNT488I  VSAM DATA SETS CREATED FOR DB2 CATALOG AND DIRECTORY INDEXES
            WILL REQUIRE AT LEAST 49074 4K BLOCKS (4089 TRACKS)
  6 DSNT488I  VSAM DATA SETS CREATED FOR ACTIVE LOG 1 AND BSDS 2
            WILL REQUIRE AT LEAST 25995 4K BLOCKS (2166 TRACKS)
  7 DSNT488I  VSAM DATA SETS CREATED FOR ACTIVE LOG 2 AND BSDS 1
            WILL REQUIRE AT LEAST 25995 4K BLOCKS (2166 TRACKS)
 ------------------------------------------------------------------------
  8 DSNT488I  DATA SETS CREATED FOR DB2 INSTALLATION AND VERIFICATION
            WILL REQUIRE AT LEAST 2325102 4K BLOCKS (193758 TRACKS)

PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-51 DSNTIPC1

```
DSNT470I WARNING, ONLY ONE VOLUME WAS SPECIFIED
DSNT478I BEGINNING EDITED DATA SET OUTPUT
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTIJMV)', INSTALL JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTIJIN)', INSTALL JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTIJID)', INSTALL JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTIJDE)', RECOVERY JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTIJTC)', INSTALL JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTIJTM)', INSTALL JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTIJIC)', INSTALL JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTIJVC)', INSTALL JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTIJSG)', INSTALL JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTIJOS)', INSTALL JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTIJEX)', INSTALL JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTIJPD)', INSTALL JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNTEMP(DSNU)',     CLIST
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNTEMP(DSNH)',     CLIST
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNTEMP(DSNHC)',    CLIST
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNTEMP(DSNEMC01)', CLIST
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTESC)',  SAMPLE DATA
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTESD)',  SAMPLE DATA
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTESA)',  SAMPLE DATA
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTESE)',  SAMPLE DATA
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ0)',  SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ1)',  SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ1L)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ1P)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ1S)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ1T)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ1U)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ2A)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ2C)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ2D)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ2E)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ2F)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ2H)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ2P)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ3C)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ3P)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ3M)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ4C)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ4P)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ5A)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ5C)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ5P)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ6U)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ7)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ71)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ73)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ75)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ76)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ77)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTEJ78)', SAMPLE JCL
DSNT489I CLIST EDITING 'DB9B9.V9INSTAL.NEW.SDSNSAMP(DSNTIJUZ)', INSTALL JCL
***
```

Figure 12-52 Typical output of the install CLIST in INSTALL mode

12.5.6 Panels for ENFM flow

Note that data sharing field must be blank and you must specify your input parm member. See Figure 12-53.

```
DSNTIPA1 DB2 VERSION 9 INSTALL, UPDATE, MIGRATE, AND ENFM - MAIN PANEL
===>
Check parameters and reenter to change:
 1  INSTALL TYPE          ===> ENFM      Install, Migrate, ENFM, or Update
 2  DATA SHARING          ===>           Yes or No (blank for ENFM or Update)

Enter the data set and member name for migration only. This is the name used
from a previous Installation/Migration from field 9 below:
 3  DATA SET(MEMBER) NAME ===>

For DB2 SMP/E libraries (SDSNLOAD, SDSNMACS, SDSNSAMP, SDSNCLST, etc.), enter:
 4  LIBRARY NAME PREFIX   ===> DB9B9
 5  LIBRARY NAME SUFFIX   ===>

For install data sets (NEW.SDSNSAMP, NEW.SDSNCLST, RUNLIB.LOAD, etc.), enter:
 6  DATA SET NAME PREFIX  ===> DB9B9.V9MIGRAT
 7  DATA SET NAME SUFFIX  ===>

Enter to set or save panel values (by reading or writing the named members):
 8  INPUT MEMBER NAME     ===>           Default parameter values
 9  OUTPUT MEMBER NAME    ===>           Save new values entered on panels
PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-53 Main panel for ENFM

```
DSNTIPT           ENFM DB2 - DATA SET NAMES PANEL 1
===>
Data sets allocated by the installation CLIST for edited output:
  1  TEMP CLIST LIBRARY ===> DB9B9.V9MIGRAT.NEW.SDSNTEMP
* 2  SAMPLE LIBRARY     ===> DB9B9.V9MIGRAT.NEW.SDSNSAMP
Data sets allocated by the installation jobs:
  3  CLIST LIBRARY      ===> DB9B9.V9MIGRAT.NEW.SDSNCLST
  4  APPLICATION DBRM   ===> DB9B9.V9MIGRAT.DBRMLIB.DATA
  5  APPLICATION LOAD   ===> DB9B9.V9MIGRAT.RUNLIB.LOAD
  6  DECLARATION LIBRARY===> DB9B9.V9MIGRAT.SRCLIB.DATA
Data sets allocated by SMP/E and other methods:
  7  LINK LIST LIBRARY  ===> DB9B9.SDSNLINK
  8  LOAD LIBRARY       ===> DB9B9.SDSNLOAD
  9  MACRO LIBRARY      ===> DB9B9.SDSNMACS
 10  LOAD DISTRIBUTION  ===> DB9B9.ADSNLOAD
 11  EXIT LIBRARY       ===> DB9B9.SDSNEXIT
 12  DBRM LIBRARY       ===> DB9B9.SDSNDBRM
 13  IRLM LOAD LIBRARY  ===> DB9B9.SDXRRESL
 14  IVP DATA LIBRARY   ===> DB9B9.SDSNIVPD
 15  INCLUDE LIBRARY    ===> DB9B9.SDSNC.H
PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-54 The only field you can change here is field 2

```
DSNTIPOO               ENABLE NEW FUNCTION MODE FOR DB2
===>

Enter storage management properties for defining ENFM shadow data sets:
                     VOL/SER       DATA CLASS       MGMT CLASS       STOR CLASS
                     -------       ----------       ----------       ----------
 1 TABLE SPACES ===> SBOX9A   ===>             ===>             ===>
 2 INDEXES      ===> SBOX9A   ===>             ===>             ===>

Enter space for defining the ENFM shadow data sets for SYSOBJ:
 3 PRIMARY RECS ===> 708        SECONDARY RECS ===> 708

Enter space for defining the ENFM shadow data sets for SYSPKAGE:
 4 PRIMARY RECS ===> 1241       SECONDARY RECS ===> 1241

Enter storage management properties for defining ENFM image copy data sets:
                     DEVICE        DATA CLASS       MGMT CLASS       STOR CLASS
                     --------      ----------       ----------       ----------
 5 IMAGE COPY   ===> SYSALLDA ===>             ===>             ===>
Enter the data set prefix for the ENFM image copy data sets:
 6 PREFIX       ===> DB9B9.V9MIGRAT.IMAGCOPY

PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-55 Enable New Function Mode

```
DSNTIPC1           ENFM DB2 - CLIST CALCULATIONS - PANEL 2
===>

  1 DSNT488I  SHADOW DATA SETS CREATED FOR THE DB2 CATALOG AND DIRECTORY
              WILL REQUIRE AT LEAST 1949 4K BLOCKS (162 TRACKS)
  2 DSNT488I  SHADOW DATA SETS CREATED FOR DB2 CATALOG AND DIRECTORY INDEXES
              WILL REQUIRE AT LEAST 6833 4K BLOCKS (569 TRACKS)




 ------------------------------------------------------------------------------
  3 DSNT488I  DATA SETS CREATED FOR DB2 ENABLING NEW FUNCTION MODE
              WILL REQUIRE AT LEAST 8782 4K BLOCKS (731 TRACKS)


PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

Figure 12-56 CLIST calculations

Archived

See Figure 12-57 for the output.

```
 DSNT478I BEGINNING EDITED DATA SET OUTPUT
 DATASET DB9B9.V9MIGRAT.NEW.SDSNSAMP COMPRESSED AT 17:29:14
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTIJEN)', ENFM PROCESSING
DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTIJNF)', TURN NEW FUNCTION MODE
ON
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTIJNX)', CREATE XML SCHEMA
DATABASE AND ROUTINES THAT REQUIRE NEW-FUNCTION MODE
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTIJES)', DISABLE USE OF
NEW FUNCTION (ENFM*)
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTIJCS)', RETURN FROM ENFM OR
ENFM* TO CM*
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTESC)',  SAMPLE DATA
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTESD)',  SAMPLE DATA
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTESA)',  SAMPLE DATA
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTESE)',  SAMPLE DATA
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ0)',  SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ1)',  SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ1L)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ1P)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ1S)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ1T)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ1U)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ2A)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ2C)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ2D)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ2E)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ2F)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ2H)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ2P)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ3C)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ3P)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ3M)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ4C)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ4P)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ5A)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ5C)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ5P)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ6U)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ7)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ71)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ73)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ75)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ76)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ77)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTEJ78)', SAMPLE JCL
 DSNT489I CLIST EDITING 'DB9B9.V9MIGRAT.NEW.SDSNSAMP(DSNTIJNG)', UPDATE
DSNHDECP FOR ENFM
 ***
```

Figure 12-57 Typical output of install CLIST in ENFM

Table 12-1 shows the differences in the sequence of jobs to be executed for the installation of V8 and V9.

Table 12-1 Differences in installation jobs

DB2 V8 installation jobs	DB2 V9 installation jobs
DSNTIJCA	DSNTIJCA
DSNTIJCC	DSNTIJCC
	DSNTIJCF Fallback job
DSNTIJCL	DSNTIJCL
DSNTIJCM	DSNTIJCM
	DSNTIJCS
DSNTIJCX	DSNTIJCX
DSNTIJDE	DSNTIJDE
DSNTIJEN	DSNTIJEN
	DSNTIJES
DSNTIJEX	DSNTIJEX
DSNTIJFT	DSNTIJFT
DSNTIJFV	DSNTIJFV
DSNTIJGF	DSNTIJGF
	DSNTIJHM Health Monitor
DSNTIJIC	DSNTIJIC
DSNTIJID	DSNTIJID
DSNTIJIM	DSNTIJIM
DSNTIJIN	DSNTIJIN
	DSNTIJI2 IMS Stored Proc
DSNTIJLR	
DSNTIJMC	
DSNTIJML	DSNTIJML
DSNTIJMS	DSNTIJMS
DSNTIJMV	DSNTIJMV
DSNTIJMX	DSNTIJMX
DSNTIJM1	DSNTIJM1
DSNTIJM2	DSNTIJM2
DSNTIJNE	
DSNTIJNF	DSNTIJNF
DSNTIJNG	DSNTIJNG
DSNTIJNH	
DSNTIJNR	
	DSNTIJNX
DSNTIJN1	DSNTIJN1
DSNTIJN2	DSNTIJN2
	DSNTIJOS
	DSNTIJPD
DSNTIJPM	DSNTIJPM
DSNTIJRC	DSNTIJRC
	DSNTIJRI
DSNTIJRX	DSNTIJRX
DSNTIJSD	DSNTIJSD
DSNTIJSG	DSNTIJSG
DSNTIJS1	DSNTIJS1
DSNTIJS2	DSNTIJS2
DSNTIJTC	DSNTIJTC
DSNTIJTM	DSNTIJTM
DSNTIJUT	DSNTIJUT
DSNTIJUZ	DSNTIJUZ
DSNTIJVC	DSNTIJVC
DSNTIJWL XPLINK Web services UDFs	
DSNTIJWS	DSNTIJWS

Archived

12.5.7 Installation CLISTs general troubleshooting tips

Suppose an installation panel reports that a field contains an invalid value but the field is *protected* — the value cannot be changed. Usually this is because the default's input member does not have an entry for the field reporting the error or you did not follow the documented process for installing or migrating a non-originating member of a data sharing group. If you get into this situation, check the input member for the entry in error. If no entry is found, the input member is probably downlevel, and you will need to locate the appropriate member. You should closely follow the procedures in the installation/migration chapter of the *DB2 Version 9.1 for z/OS Data Sharing: Planning and Administration,* SC18-9845.

Suppose that the CLIST halts while editing a job. This is typically caused by a CLIST error. The most likely cause is that the CLIST is at a different service level from the template JCL being edited when the error occurred or the template JCL (or possibly the CLIST) has been modified by the user. You can use CLIST trace mode (CONTROL LIST CONLIST SYMLIST) in the vicinity of failure to isolate the failing statement. Make sure that you are pointing to the proper input data sets. If you still encounter problems, collect the template JCL job and notify IBM support for help.

Suppose that an installation/migration job fails. This is usually caused by not following procedures, an error in installation/migration job (errors in JCL, DDL), or a DB2 problem (S04E abends). You should review the steps taken prior to the failure for compliance with the documented installation procedures. If all procedures were followed properly and the job still fails, collect the job output and dump, if available, and report the problem to the support organization.

12.6 Migration and coexistence

In this section we examine the migration steps in more detail.

12.6.1 The migration process

Migration to DB2 V9 is permitted only from a system running DB2 for z/OS V8 in new-function mode (NFM).

Archived

The migration process comprises three progressive catalog levels: compatibility mode (CM), enabling-new-function mode (ENFM), and new-function mode (NFM). In Figure 12-58 you can see the steps involved in the migration process.

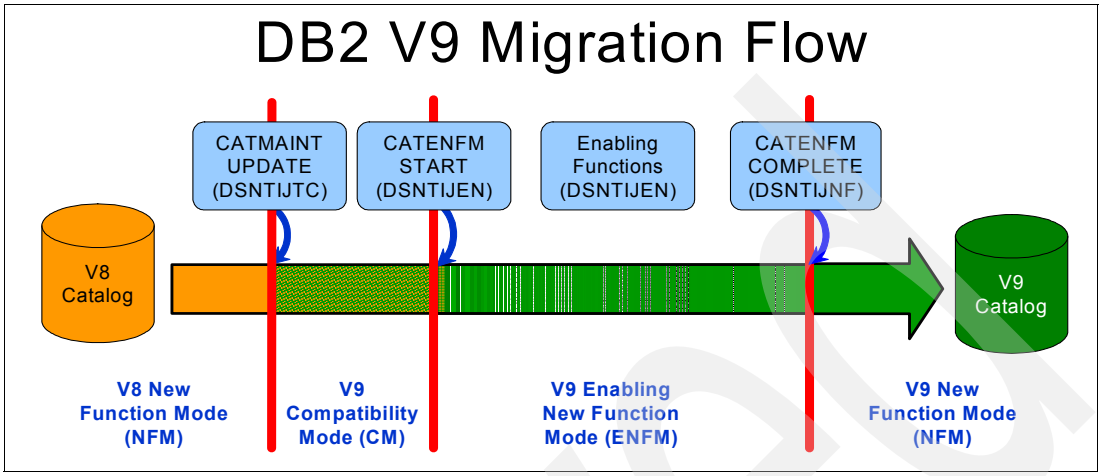


Figure 12-58 DB2 V9 migration flow

CM begins when catalog tailoring is complete (when job DSNTIJTC completes successfully), and continues until ENFM begins. In a DB2 data sharing group, members in CM can coexist with members that are still in V8 NFM. All members of a data sharing group must enter CM before DB2 can begin ENFM. CM is the phase where you test to make sure that there is no regression taking place. Some new functions may be available at this mode. You can remain in this phase for an extended period of time. You can only fall back to V8 NFM from V9 CM.

ENFM is a transitional state that indicates that the DB2 subsystem or data sharing group is in the process of enabling-new-function. This state begins when catalog conversion for new function starts (when you first run job DSNTIJEN successfully), and continues until DB2 enters NFM. After ENFM begins, data sharing coexistence is no longer allowed (no DB2 V8 members are allowed to join the group), and DB2 can no longer fall back to the previous release. You cannot return to V9 CM, but you can return to V9 CM*. This phase is a transitory phase, and you should only remain in this mode for a short period of time. Some new *functions* are available at this mode.

NFM is the state of the catalog after migration is complete. The catalog has been marked as being in new-function-mode. This state begins when you run job DSNTIJNF after successfully completing job DSNTIJEN. You can return to V9 CM* or V9 ENFM*. You cannot fall back to V8 from V9 NFM or coexist with V8 systems. NFM is the phase where all new functions of DB2 V9 for z/OS are available to be exploited.

DB2 has two additional modes that you might encounter during your migration to V9: compatibility-mode* (CM*) and enabling-new-function-mode* (NFM*).

Compatibility mode* (CM*) is similar to compatibility mode (CM), but the asterisk (*) indicates that at one time the DB2 subsystem or data sharing group was in ENFM, ENFM*, or NFM. You can still access objects that were created in ENFM or NFM. You cannot fall back to V8 from CM* or coexist with a V8 system. Data sharing groups cannot have any V8 members. Job DSNTIJCS takes DB2 to CM* from ENFM, ENFM*, or NFM.

Enabling-new-function mode* (ENFM*) is a transitional state that is similar to ENFM, but the * indicates that the at one time the DB2 subsystem or data sharing group was in NFM. You can still access objects that were created in NFM, but you cannot create any new objects that

require NFM. You can return to V9 CM*. You cannot fall back to V8 from ENFM* or coexist with a V8 system. Job DSNTIJES takes DB2 to ENFM* form CM* or NFM.

Figure 12-59 shows the allowed flow between all migration modes. Be aware that once you start the process for ENFM, you will no longer be able to fall back to DB2 V8.

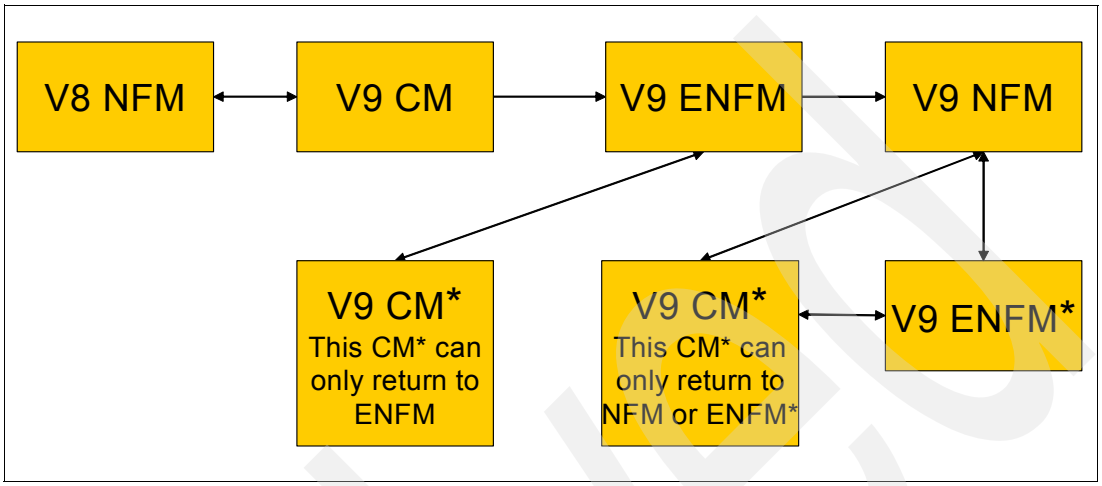


Figure 12-59 The allowed flows from one mode to another

Compatibility mode

We look at the actions in CM.

Fallback

Once you migrate to DB2 V9 CM, if you encounter any severe errors in operating your DB2 V9 system, you can fall back to DB2 V8 NFM.

Release coexistence

Standalone subsystems of various releases can communicate with each other. You can have DB2 V9 CM systems that communicate with DB2 V8 systems and prior releases. However, in a single data sharing group, you can only have DB2 V9 CM members and DB2 V8 NFM members. Before a data sharing group can enter DB2 V9 ENFM, all members of that data sharing group must be in DB2 V9 CM.

Enable new-function mode

When you run job DSNTIJEN with control statement option CATENFM START to convert from compatibility mode (CM) to enable-new-function mode (ENFM), the following changes are made:

- Alter the SYSPACKSTMT.SEQNO column from SMALLINT to INTEGER.
- Convert the SYSOBJ table space from 8 KB to 16 KB page size (REORG CONVERTV9).

  You may need to tailor DSNTIJEN. REORG requires VSAM data sets for shadows. If there are any user-defined indexes on the catalog, you will need to define those data sets. No action is required if your indexes are defined with STOGROUPS.
- Complete the definition of the new real time statistics (RTS) tables in the catalog (SYSINDEXSPACESTATS and SYSTABLESPACESTATS) by adding BIGINT and some VARCHAR columns.
- If the real time statistics database (DSNRTSDB) exists and is available, copy/load to the new RTS tables and signal the use of the new tables instead of the old ones.

DB2 remains in this mode until all enabling new functions are completed and until you run the job to switch to new-function mode.

Fallback

In DB2 V9, once you start the process of enable new-function mode (ENFM), you can only fall back to compatibility mode* (CM*). You will not be able to fall back to DB2 V8 NFM. To go back to CM* follow these steps:

- Run job DSNTIJCS with control statement option CATENFM CMON.
- If you have already run job DSNTIJNF, then after running DSNTIJCS, run DSNTIJNG with option DSNHDECP NEWFUN=NO to rebuild your DSNHDECP module.

Release coexistence

Standalone subsystems of various releases can communicate with each other. You can have DB2 V9 ENFM systems that communicate with systems in various stages of DB2 V9 and systems on prior releases. However, in a data sharing, ENFM is a group-wide event, and all members of the group will be at the same state (ENFM). Before a data sharing group can enter DB2 V9 NFM, all ENFM steps must be completed.

New-function mode

You can only get to new-function mode in DB2 V9 from a V9 system that was in ENFM or ENFM*. See the prior paragraphs for an overview of the major steps required to get to DB2 V9 ENFM or ENFM*.

Run job DSNTIJNF with control card option CAFTNFM COMPLETE.

Do not run this job until you have completed job DSNTIJEN. After running this job, run job DSNTIJNG to rebuild your DSNHDECP module. Be sure that DSNHDECM specifies NEWFUN=YES.

12.6.2 High-level migration sequence

Here is a high-level view of the major steps required to migrate to DB2 V9. Refer to the DB2 V9 installation guide for a complete detailed list and step-by-step process required for migration.

1. Ensure that all preparation steps have been completed:
   - DB2 V8 in NFM
   - BSDS converted to new expanded format available in DB2 V8
   - Fallback SPE installed everywhere, DSNHDECP re-assembled everywhere, *all* DB2 systems cycled once to pick up the code, and MVS IPLed to pick up ERLY code
   - z/OS at least at v1.7
   - SMP/E setup work done and new DB2 V9 libraries ready
   - Catalog reorged/resized and catalog and index consistency checked
   - DB2 Managed SPAS converted to WLM SPAS
2. Pick any member of your data sharing group to be the first member to migrate, or choose the first subsystem that you want to migrate.

   Run migration job DSNTIJIN to define new catalog objects for that data sharing group or subsystem.

Archive

3. Bring up the member or subsystem that you are migrating in MAINT mode and pointing to the DB2 V9 code library.
   - You will get a 00C900A6 message. This is okay and merely tells you that you are running V9 code with a V8 catalog.
   - Run the CATMAINT job DSNTIJTC with the CATMAINT UPDATE control statement from the member or subsystem that you are migrating. All the changes are done in one commit scope, so changes are all or nothing.
4. Cycle the member or subsystem you are migrating.
   - This member or subsystem is now in DB2 V9 Compatibility Mode (CM).
   - Cycle the rest of the members of the data sharing group to pick up the DB2 V9 code library. All members of the data sharing group must be running with the DB2 V9 code (in CM mode) before you can continue with the next step.
5. Image copy your entire catalog (see sample job DSNTIJIC) and check the catalog/directory for consistency.
6. Run the CATMAINT job DSNTIJEN with the CATENFM START control statement to convert from Compatibility Mode (CM) to *enable new function mode* (ENFM). This job also reorgs table space SYSOBJ to convert it from an 8 KB to a 16 KB page size. ENFM is a group-wide change. All data sharing members of that group will be in ENFM after this job completes successfully.

   You may need to tailor DSNTIJEN. REORG requires VSAM data sets for shadows. If there are any VCAT defined user indexes on SYSOBJ, you will need to define those data sets. No action is required if your user indexes on SYSOBJ are STOGROUP defined.
7. After DSNTIJEN completes successfully, you run CATMAINT job DSNTIJNF with the CATENFM COMPLETE control statement to put DB2 in new-function mode (NFM). This is a group-wide change. All data sharing members of that group will be in NFM.

12.6.3 BSDS conversion required before V9 migration

In DB2 9, the BSDS must be in the format that supports up to 10,000 entries per copy for archive log volumes and 93 per copy for active log data sets. The BSDS is converted with an optional conversion job supplied in the V8 DSNTIJUZ procedure.

Prior to DB2 9, whenever a new BSDS is created and formatted using DSNJU003, it will be in the old format that allows only 31 active logs and 1,000 archive logs. If an installation wants to use the new format for a new BSDS, it must create the BSDS, format it using DSNJU003, and then run the DSNJCNVB program to convert it to the new format.

When migrating from DB2 V8 to DB2 V9, running the conversion program DSNJCNVB is required if not previously done when running DB2 V8 in new-function mode. If the conversion of the BSDS in not performed prior to migration and if the BSDS is in the old format, a new message, DSNJ157I, will be issued during DB2 startup and startup will be terminated.

Also, the MAXARCH DSNZPARM now defaults to 10,000 for new installs.

Important: We recommend that you make a backup copy of your BSDS before running the conversion utility.

There are no fallback or coexistence considerations because DB2 V8 NFM has always supported the new format.

12.6.4 Convergence of temporary space

Prior to DB2 V9, users did not have a means of controlling how much temporary workspace was utilized by an individual agent. Large sorts, materialized views, and so on could monopolize the defined temporary space at the expense of other work and cause other work to fail due to unavailable resources (lack of temporary workspace). With DB2 V9, it is now possible to control temporary space utilization at the agent level. A new ZPARM, MAXTEMPS, is added to DSN6SYSP to specify the maximum amount of space that can be used by a single agent at any single time. You can specify this parameter in either MB or GB. If you specify a value of 0 (the default), then no limit is enforced. If any given agent exceeds MAXTEMPS, a resource unavailable condition is raised and the agent is terminated.

With DB2 V9, it is possible to monitor temporary space utilization at the DB2 subsystem level, helping you to manage the space allocated to the work file database. You can now obtain information about how many times the MAXTEMPS limit has been exceeded and details about each agent that exceeded the MAXTEMPS limit. You can monitor the following details:

- The current total storage used, in MB
- The high watermark — the maximum space ever used — in MB
- The maximum allowable storage limit (MAXTEMPS) for an agent, in MB
- The number of times the maximum allowable storage limit per agent was exceeded and for each agent:
  - The authorization ID.
  - The package collection ID and name.
  - The plan name.
  - The MAXTEMPS DSNZPARM value.
  - The current total system-wide usage of workfile storage.
  - The maximum total system-wide usage of workfile storage.
- The current total 4 KB-page table space storage used, both in MB and in KB
- The current total 32 KB-page table space storage used, both in MB and in KB
- How many times a 32 KB-page table space was used when a 4 KB one was preferable but not available
- How many times a 4 KB-page table space was used when a 32 KB one was preferable but not available

How this change is implemented

Prior to DB2 V9, DB2 supported two databases for temporary files and temporary tables (the WORKFILE database and the TEMP database). The 'AS' clause of the CREATE DATABASE statement with either the 'WORKFILE' or 'TEMP' subclause indicated that the database to be created was either a WORKFILE database or a TEMP database. Each DB2 subsystem or data sharing member had one WORKFILE database and may have had one TEMP database.

The WORKFILE database was used by DB2 for storing Created Global Temporary Tables and as storage for work files for processing SQL statements that required temporary working space (for SORTs, materializing views, triggers, nested table expressions, and others).

The TEMP database was used by DB2 for storing the user-defined Declared Global Temporary Tables and the DB2 Declared Global Temporary Tables for Static Scrollable Cursor implementation.

Archive

Characteristics of the WORKFILE and TEMP databases prior to DB2 V9

Prior to DB2 V9, the WORKFILE database and the TEMP database had the following characteristics that are relevant as background for this enhancement:

- DB2 created the WORKFILE database and table spaces in it by using installation-time parameters.
- There was no installation support for creating the TEMP database and the table spaces in it. Users had to create and drop them.
- WORKFILE table spaces are neither partitioned nor simple. They used a pseudo-segmented table space organization, but users could not specify SEGSIZE while creating table spaces in the WORKFILE database.
- The table spaces in the Temp database are segmented. SEGSIZE is either specified by the user or SEGSIZE 4 is used as the default for a table space created in a TEMP database.
- The WORKFILE database supports only 4 KB and 32 KB page sizes. The 8 KB and 16 KB page sizes are not supported. An installation panel allows users to specify the total amount of physical storage to be used for table spaces to be created in the WORKFILE database with 4 KB and 32 KB page sizes, and the number of table spaces for each type of table space. If the user creates table spaces in the WORKFILE database, the 8 KB and 16 KB page sizes cannot be specified.
- The TEMP database supports 4 KB, 8 KB, 16 KB, and 32 KB page sizes.
- The size of the temporary tables using the WORKFILE database is not limited to the size of the table space. The work files (WORKFILE tables) have the capacity to span multiple (up to 254) table spaces, if the WORKFILE database has multiple table spaces defined in it. The current maximum size of a table space in the WORKFILE database is 64 GB, with a maximum 2 GB per data set of up to 32 data sets. To support large work files (mainly for SORT), DB2 allocates a single logical work file to several physical WORKFILE table spaces. So the maximum table size for tables in the WORKFILE database is approximately 16 TeraBytes (64 GB per table space * 254 table spaces).
- The size of the temporary tables using the TEMP database is limited to the size of the table space since these tables cannot span multiple table spaces as the WORKFILE tables can. The current maximum size of a table space in the TEMP database is 64 GB, with a maximum of 2 GB per data set of up to 32 data sets.
- The number of work files that can be used for DB2 sorts is limited only by the buffer pool size. The WORKFILE database does not use global DBD management to assign and release table IDs. Each work file has its own local DBD, copied from the global DBD. (*Global* here means global within a single DB2 subsystem.) The global DBD has only the table space descriptor objects for each table space defined in the database. The work file's local DBD has the table space descriptors and the table descriptor for just that table that is associated with that work file. Since the DBD is not globally shared (within a DB2 subsystem), there is no limit to the number of tables in the WORKFILE database.
- The number of tables that can be defined in the TEMP database at any given time is less than 32 KB per DB2 subsystem. The DBD for the TEMP database is shared by all agents on the system, and therefore it contains descriptors for all table spaces, tables, indexes, and related objects defined in the database. Since the DBD for the TEMP database is globally shared (within a DB2 subsystem), the number of tables that can be defined in the TEMP database per DB2 subsystem is limited to less than 32 KB.

Characteristics of the WORKFILE database in DB2 V9

In DB2 V9, we converge the two (WORKILE and TEMP) temporary databases previously supported by DB2 into one. Now you can define a single database (the WORKFILE database) to be used as storage for all temporary files and tables. We preserve all the

functionality of the temporary tables as they existed prior to DB2 V9, but provide additional benefits.

Here is a list of the changes made in order to discontinue the use of the TEMP database and switch the temporary tables that currently use the TEMP database to use the WORKFILE database:

- The CREATE DATABASE and CREATE TABLESPACE statements are modified to support this. This is effective in both Compatibility and New-Function Modes (NFM):
  - Disallow the creation of database as TEMP.
  - Allow DROP of table spaces in an existing TEMP database.
  - Allow DROP of an existing TEMP database.
  - Do not allow creation of additional table spaces in an existing TEMP database.
  - Change the Declared Global Temp Table and Scrollable Cursor implementation to use the WORKFILE Database instead of the TEMP database.
- To compensate for the removal of the TEMP database, which supports the SEGSIZE clause for creation of a table space in it, the restriction that SEGSIZE cannot be specified for creating a table space in the WORKFILE database is removed in new-function mode only.
  - In Compatibility Mode, the SEGSIZE clause will continue to be rejected in the CREATE TABLESPACE statement for creating table spaces in the WORKFILE database.
  - For table spaces that existed in a WORKFILE database before migration from DB2 Version 8, for those created during migration, and for those created in the Compatibility Mode of DB2 Version 9, the SEGSIZE column of the catalog table SYSTABLESPACE will continue to show 0 as the segment size, indicating that these table spaces were created prior to the enablement of the Version 9 new-function mode (NFM). However, DB2 will treat these table spaces as segmented, with the default segment size of 16, both in Compatibility and new-function modes.

System planning and installation

In DB2 V9 the two temporary databases are converged, leaving a single database, the WORKFILE database. The WORKFILE database is used for all temporary tables, both external and for use by DB2, simplifying DB2 systems management tasks. The TEMP database is no longer used by DB2.

If you migrate from DB2 V8 and wish to reclaim the storage associated with an existing TEMP database, it is *your* responsibility to drop the TEMP database and redefine the storage to be used for the WORKFILE database or for something else. If storage is not a problem, then it may be better not to DROP the TEMP database until you are sure that you will not be falling back to DB2 V8. Otherwise, you will have to recreate the TEMP database.

Tip: We recommend that you do not drop the TEMP database until you are sure that you will not be falling back to DB2 V8. In DB2 V9, you should re-allocate your WORKFILE database to add the equivalent space previously defined to the TEMP database.

Before DB2 V9, the DB2 installation process defines user-managed data sets for use by table spaces in the WORKFILE database. The V9 installation process now generates the DDL to create table spaces in the WORKFILE database using DB2-managed storage in the SYSDEFLT storage group. If you want the WORKFILE table spaces on user-managed data sets, then you need to provide your own AMS statements to define them, and modify the installation-generated DDL to specify the VCAT clause in place of the STOGROUP clause.

You also have the option during installation and migration to create additional table spaces in the work file database.

We recommend that you set the DSNZPARM 'DSVCI' to YES. This will enable DB2 to create the DB2-managed data sets with a VSAM control interval that matches the page size for the table spaces.

Keep in mind that 8 KB and 16 KB page sizes are still not supported for table spaces in the WORKFILE database. This means that Declared Global Temporary Tables and Static Scrollable Cursor result tables require a table space with page size of 32 KB.

Note: If you use Declared Global Temporary Tables or Statics Scrollable Cursors, ensure that the WORKFILE database exists with at least one 32 KB page size table space.

Security and auditing

Rules and authorization actions associated with creating the TEMP database in releases prior to DB2 V9 will now apply to creating the WORKFILE database. In other words, the authorization privileges needed by the privilege set for CREATE TABLESPACE for the prior supported TEMP database now apply for CREATE TABLESPACE in the WORKFILE database.

PUBLIC implicitly has the following privileges without GRANT authority for temporary tables created by application processes:

- All table privileges on the table and the authority to drop the table
- The CREATETAB privilege to define temporary tables in the WORKFILE database
- The USE privilege to use the table spaces in the WORKFILE database

No additional privileges are needed to access the WORKFILE database and the temporary tables in it.

Table privileges for temporary tables are *not* grantable/revokable. As with prior releases of DB2, users cannot grant the USE privilege for table spaces that are for Declared Global Temp Tables.

Note: These implicit privileges for declared temporary tables are not recorded in the DB2 Catalog.

Instrumentation

The counters for monitoring temporary space utilization at the DB2 subsystem level are included in IFCID 2 (Statistics record). A new IFCID 343 trace record will be written when the MAXTEMPS DSNZPARM limit for an agent is exceeded. The new IFCID 343 trace record is classified as trace type PERFM CLASS 3 and STAT CLASS(4).

12.6.5 DBMA clean-up

Prior to V9, you had the option (via DB2 DSNZPARM DSN6SPRM.RELCURHL) to control whether at commit time DB2 should release a lock on which a cursor defined WITH HOLD is positioned. This was set from installation panel DSNTIP8, field 5, with the RELEASE LOCKS option. The default setting is YES.

In DB2 V9 this is no longer optional, and DB2 will always release the lock at commit. This allows for greater concurrency.

This is a release incompatibility for customers who use RELCURHL=NO in DB2 Version 8.

12.6.6 DBETE state

This new state is used in the event that there is a problem recreating DBET states during DB2 restart.

DBETE is a new DBET advisory state. DBETE is set during DB2 restart to flag objects whose DBET state cannot be determined from the DB2 logs during restart processing. This is done in order to allow DB2 to come up quickly and provide access to other objects not affected by this problem.

A new message DSNI046I is issued. You can use the DISPLAY DATABASE(*) SPACE(*) ADV command to see the objects in DBETE state (message DSNT392I). Use normal recovery methods to get these objects out of restrictive states.

12.7 Fallback

The fallback process is when you have completed the CATMAINT process (the migration job DSNTIJTC completed successfully) but *fall back* to using the prior releases code. You do this only if a severe error occurs either during the subsequent migration steps or during the operation of DB2 V9 CM. You can only fall back from DB2 V9 CM. For example, you migrate your DB2 V8 NFM system to DB2 V9 CM and later decide to fall back to the V8 code. You cannot fall back from DB2 V9 CM*, ENFM, ENFM*, or NFM.

Here is s short overview of the steps to fallback to V8:

1. Stop DB2 V9.
2. Run Job DSNTIJFV to rename the cataloged procedures (point back to the V8 libraries).
3. Start DB2 V8.
4. Run the V8 installation verification jobs DSNTEJxx.

For a more detailed description of the steps required to fall back, see “Falling back” in the *DB2 Version 9.1 for z/OS Installation Guide,* GC18-9846.

Things to watch out for

Watch out for:

- The SEGSIZE clause indicates that the table space is segmented and how many pages are assigned to each segment.
  - The current restriction that “SEGSIZE clause cannot be specified for a table space in work file database is removed, in new-function mode only.
  - If the SEGSIZE clause is omitted for a table space created in the WORKFILE database in new-function mode, then SEGSIZE 16 should be used as the default and the SEGSIZE column of the catalog table SYSTABLESPACE should reflect the actual (either user-specified or default) value used for the table space.
  - The SEGSIZE clause will continue to be rejected in Compatibility Mode (unrevised SQLCODE -620, SQLSTATE 53001). The SEGSIZE column of the catalog table SYSTABLESPACE will continue to record 0 as the segment size for table spaces created in the WORKFILE database in Compatibility Mode, to indicate that these table spaces were created in the Compatibility mode. However, DB2 will treat them as segmented, with the default segment size 16.
- Due to the new enhanced V9 control block infrastructure (some storage above the bar and some storage below the bar), plans and packages bound on V9 will be incrementally rebound when run on V8, so the sections will be all be below the bar for V8 to process.

Code is also added to ensure that V8 does not ask a V9 member to assist a query for sysplex query parallelism. Otherwise, V9 would have to puff the received storage from V8 into two pieces (above and below bar). Sysplex query parallelism is enabled for members on the same version only. Sysplex query parallelism is not allowed across DB2 versions. A V8 member in the same data sharing group will not assist a query originated on V9.

12.8 Other migration considerations

In this section we discuss other migration considerations.

Compatibility mode versus new-function mode

In this section we discuss compatibility mode versus new-function mode.

Functions available in CM

These are:

- New histogram statistics are available in Compat mode.
- WLM Buffer Pool management is available in Compat mode.
- 64-bit DDF is available in Compat mode.
- Archive log BSAM reads instead of BDAM.
- Built-in monitor to detect CPU stalls that result in latch contention and functionality to attempt to eliminate the contention. Built-in monitor to warn when virtual storage consumption below the 2 GB bar reaches threshold.
- WLM assisted buffer pool management.

Functions available in NFM

These:

- Can create up to 500 user-defined indexes on the catalog. The prior limit was 250.
- The WLM environment name for stored procedures can now be 32 characters long. This is not supported in CM.
- Extensions for SQL Procedures.

Name resolution differs in CM and NFM

DSNHxxxx messages that were issued in prior releases for external SQL procedures are replaced with SQLCODEs in NFM for native SQL procedures.

Behavior for automatically create of objects changes in NFM

This is:

- If a containing table space was implicitly created, you cannot explicitly drop any required objects (for example, unique index for primary key), whether or not these objects were implicitly or explicitly created prior to V9 (except for LOB table spaces).
- CREATE LOB TABLESPACE: It is not possible to create a LOB table space in an implicitly created database.
- DROP TABLE for an auxiliary table will fail when the base table's table space is implicitly created.
- DROP TABLESPACE for an implicitly created LOB table space will fail.
- DROP INDEX for an enforcing primary key/unique key/ROWID index will fail when the containing table space is implicitly created.

Archived

- For CREATE TABLE, if no IN clause or table space name is specified, DB2 will choose a buffer pool size based on row length only if the default buffer pool is not large enough.
- For CREATE TABLE, if no IN clause or table space name is specified, the DSNZPARM IMPTSSEG will determine the type of table space created (segmented or partition by growth), but LOCKSIZE defaults to ROW instead of ANY. If the table is dropped, the containing table space will also be dropped.
- If you have implemented a UDF with any of the following names, if the function name is unqualified when invoked, then the new built-in function is invoked instead of the UDF:
  - ASCII
  - TIMESTAMP
  - SOUNDEX
  - DIFFERENCE
  - TIMESTAMP_ISO
  - EXTRACT
  - MONTHS_BETWEEN
  - VARCHAR_FORMAT
  - TIMESTAMP_FORMAT
  - RID
  - DECFLOAT
  - QUANTIZE
  - IS_IDENTICAL_TO
  - NORMALIZE_DECFLOAT

Improve performance for RLST table searching

The RLST database is created once during installation of a brand, new DB2 subsystem, or data sharing group by job DSNTIJSG. In V9, DSNTIJSG creates the DSNRLSTnn table and DSNARLnn index as in previous releases. V9 DSNTIJSG also provides DDL to create the new DSNRLMTnn table and DSNMRLnn index, but the DDL is commented out because not all customers will want to use the new feature, and an empty DSNRLMTnn table increases RLF overhead with no benefit.

DB2 migration, the DDL for creating the RLST, is typically removed from DSNTIJSG by the installation CLIST. For Version 9 migration, the installation CLIST preserves only the commented-out DDL for creating the new DSNRLMTnn table and DSNMRLnn index.

New Catalog tables

These are:

- SYSCONTEXT – trusted context.
- SYSROLES – roles.
- SYSIBM.SYSOBJROLEDEP lists the dependent objects for each role.
- SYSIBM.SYSCTXTTRUSTATTRS holds the attributes for a given context.
- SYSIBM.SYSCONTEXTAUTHIDS stores the authorization IDs that can be “switched to” in a trusted connection.
- SYSDEPENDENCIES – object dependencies.
- SYSENVIRONMENT – bind/decp options for the DDLs which has its own statement text stored in the catalog.
- SYSKEYTARGETS – extended indexes, index on expression.
- SYSJAVAPATH – Java stuff.
- SYSROUTINESTEXT – native SQL stored procedures.

Archived

- New table space DSNDB06.SYSRTSTS.
- SYSTABLESPACESTATS and SYSINDEXSPACESTATS – Real Time Statistics.
  - Add the RELCREATED column.
  - Add the DSNRTX01 and DSNRTX02 indexes.
- SYSXMLSTRINGS and SYSXMLRELS - XML.
- SYSIBM.SYSPENDINGDDL with indexes.
  - DSNADL01 (non-unique)
  - DSNADL02 (unique)

Changes to advisory and restrictive states (DBET states)

There are changes to the states from V8. These changes take effect in DB2 V9 compatibility mode:

- Altering from COMPRESS YES to COMPRESS NO or COMPRESS NO to COMPRESS YES puts the INDEX in advisory REBUILD-pending.
- Altering NOT PADDED or PADDED puts the index in ARBDP.

The LGDISCLR field in the DSNDQJ00 macro has been removed in DB2 Version 9. This could result in incompatibility when migrating into DB2 Version 9. Thus, any application that relies upon LGDISCLR in the DSNDQJ00 mapping macro (or IFI306) to determine whether a log record is a compensation log record needs to be changed to reference LRHCLR instead.

Changes to the group attachment process

In DB2 Version 9, the group attachment process attempts to attach to data sharing members on the same LPAR in random order. You can prevent GROUP ATTACH from being attempted by specifying the option GROUPOVERRIDE = 'NOGROUP'. All other ATTACH types must specify the 'GROUP' option to automatically attempt the GROUP attach. All the ATTACHs that use GROUP ATTACH could be affected. In particular, an application that ATTACHs multiple times as part of its normal process and expects to always return to the same DB2 will probably fail. This problem exists today if the number of active DB2s on an LPAR change with a new active being lower on the GROUP ATTACH list, but will be more pronounced with the new randomizing protocols.

IFCID 306 enhancement to avoid decompression errors

The change keeps the old dictionary in memory so DPROP can decompress log records.

Support for DB2-managed stored procedures removed

DB2-established stored procedure address spaces are no longer supported. You cannot run a stored procedure in a DB2-established stored procedures address space.

Automatic creation of objects

The default for MGEXTSZ has changed from NO to YES. YES indicates that DB2-managed data sets secondary extent allocation is set according to a sliding scale.

Run DDF without SNA when using TCP/IP only

If you do not plan to communicate with remote sites with SNA/APPC, you do not need to define VTAM to DB2 if you update the BSDS DDF record with an IPNAME value. When you update the BSDS DDF record with an IPNAME value, DDF does not activate its SNA/APPC communications support. DB2 TCP/IP communications uses the IPNAME value and a character representation of the TCP/IP resync port (RESPORT) hexadecimal value to identify units of work. For information about how to update the BSDS IDDF record, see DB2 Utility Guide and Reference. If the BSDS DDF record does not contain an IPNAME value, you must

Archived

define VTAM to DB2 because DB2 TCP/IP communications uses the VTAM network ID and LUNAME to identify units of work.

Dropping of indexes when a unique constraint is dropped

In previous releases of DB2, if a user dropped a unique constraint, DB2 did not drop the index that enforced uniqueness. In DB2 Version 9, if a table is in an implicitly created table space, and a unique constraint on that table is dropped, DB2 drops the index that enforces uniqueness.

New column added to SYSPARMS

You can now have multiple versions of a routine. Also, some of the defaults (like ISOLATION and CURRENTDATA) have changed.

Changes made to the way traces output

Most traces used to default to OPX. Now each trace started with the destination OPX will attempt to use the next available buffer that is not in use. All monitor programs accessing OP buffers may need to adjust assignment strategies. Well-behaved monitors will likely need no changes. More specifically, monitor programs assuming MONITOR class traces will always pick OP1 when an OPX dest is specified will need to be changed. In addition, monitors can no longer assume that OP traces are left in a disabled state and can be reactivated by starting another trace to a previously used buffer. All automation based on DSNW128I or DSNW129I message format will have to be changed.

Catalog changes

These are:

- Real Time Statistics tables are moved from user-defined tables to the Catalog.
- SYSTABLESPACESTATS.TOTALROWS and SYSINDEXSPACESTATS.TOTALENTRIES have changed from FLOAT to BIGINT. Also, we now have support for Cloned tables, new columns are added for SYSTABLESPACESTATS.INSTANCE, and SYSINDEXSPACESTATS.INSTANCE.:SYSTABLESPACESTATS.DATASIZE contains the total number of bytes occupied by row data used in the partition or table space. Current queries of the RTS tables may need to consider the additional columns in SYSTABLESPACESTATS and SYSINDEXSPACESTATS. Current queries of the TOTALROWS and TOTALENTRIES may need to consider the change from FLOAT to BIGINT.
- To accommodate the maximum size of a LOB table space, the data type of SYSTABLESPACESTATS.SPACE is changed form INT to BIGINT.

The default for bind option CURRENTDATA - the new default is NO

This applies to both the BIND PLAN and the BIND PACKAGE subcommands, as well as the CREATE TRIGGER for trigger packages, and the CREATE PROCEDURE and the ALTER PROCEDURE ADD VERSION SQL statements for SQL PL procedure packages. The default for bind option ISOLATION is now changed from RR to CS. This applies to the BIND PLAN and the remote BIND PACKAGE subcommands. For the BIND PACKAGE subcommand, the current default (plan value) stays. The default change does not apply to implicitly built CTs (for example, DISTSERV CTs) because that would impact existing customers. DB2I and SPUFI are updated to reflect the change in default settings for CURRENTDATA and ISO. Add new index DSNOPX04 to SYSIBM.SYSPARMS. The foreign keys in SYSIBM.SYSTABLESPACESTATS and SYSIBM.SYSINDEXSPACESTATS (DSNDP@RK and DSNDI@RK) are removed.

12.9 Samples

DB2 samples are provided for several new V9 functions. For details see *DB2 Version 9.1 for z/OS Installation Guide,* GC18-9846.

- New sample for native SQL procs, job DSNTEJ66, which does the following:
  - Creates a sample native SQL procedure called DSN8.DSN8ES3 that generates and returns (by result set) a CREATE PROCEDURE statement for a given stored procedure
  - Prepares and executes a sample caller of DSN8ES3 called DSN8ED9
  - Shows how to use ALTER PROCEDURE ... ADD VERSION to create a version V2 of DSN8ES3 that does the same thing as the original version but also adds a terminating semicolon at the end of the generated CREATE PROCEDURE statement
  - Shows how to ALTER ACTIVATE version V2 to make it the active version of DSN8ES3
  - Shows how to DEPLOY DSN8ES3 at a remote site
- LOBs
  - Job DSNTEJ7 now includes the use of the DB2 LOAD utility to populate a LOB table.
  - The DSN8DLPL sample application demonstrates how to use LOB locators to populate a LOB column.
  - DSNTIAUL supports a LOBFILE option for unloading LOBs to data sets with a specific naming convention.

  See *LOBs with DB2 for z/OS: Stronger and Faster*, SG24-7270, for more information about LOBs with DB2 V9.
- Private to DRDA protocol REXX tool (DSNTP2DP)

  To help you convert your plans and packages from using private protocol to DRDA protocol, DB2 provides the private to DRDA protocol REXX tool, DSNTP2DP, which scans your catalog and generates the necessary commands to convert all objects that have a private protocol dependency to DRDA.
- DB2-provided stored procedures

  The XML schema support and XML decomposition stored procedures have been added. See *DB2 Version 9.1 for z/OS Application Programming and SQL Guide,* SC18-9841, for details.
- XML

  There is no full IVP, but some functions have examples:
  - DSNTEJ1 creates objects that contain XML columns.
  - DSNTEJ2H shows how to SELECT, INSERT, UPDATE, and DELETE.

12.9.1 DSNZPARM change summary

These parameters are eliminated:

- DSN6SPRM.RELCURHL: Release at COMMIT any page or row locks on which a cursor WITH HOLD is positioned. The field RELEASE LOCKS has been removed from panel DSNTIP8.
- DSN6SYSP.DBPROTCL: Default protocol (DRDA or PRIVATE) to be used when DBPROTOCOL is not explicitly specified for the bind of a plan or a package. The field DATABASE PROTOCOL has been removed from panel DSNTIP5.

- DSN6SYSP.STORPROC: Name of the address space procedure for the DB2-established SPAS. The field DB2 PROC NAME has been removed from panel DSNTIPX.
- DSN6SPRC.SPRMIFS: Maximum allocated storage area for IFI READS.
- DSN6SPRM.PARAPAR1: Enables optimizer enhancements added by APAR PQ87352.
- DSN6SPRM.TABLES_JOINED_THRESHOLD: Minimum number of table joins in a query to cause DB2 to monitor the resources consumed when determining the optimum access path for that query.
- DSN6SPRM.MAX_OPT_ELAP: Maximum amount of elapsed time in seconds to be consumed by the DB2 Optimizer.
- DSN6SPRM.SJMXPOOL: Max MB of the virtual memory pool for star join queries. The field STAR JOIN MAX POOL has been removed from panel DSNTIP8.
- DSN6SPRM.SUPPRESS_TS_CONV_WARNING: Suppress table space conversion warnings (hidden keyword).

These parameters are modified:

- DSN6SPRM.MGEXTSZ: Use sliding secondary quantity for DB2-managed data sets. Default changes from NO to YES. On panel DSNTIP7, OPTIMIZE EXTENT SIZING.
- DSN6SYSP.IDXBPOOL: Default bufferpool for CREATE INDEX. Now accepts 8 KB, 16 KB, and 32 KB page size BPs in addition to 4 KB.
- DSN6SYSP.WLMENV: Name of the default WLM environment for DB2. Length has increased from 18 to 32 characters.

These parameters are added:

- DSN6SYSP.IMPDSDEF: Define the data set when creating an implicit base table space or implicit index space. On panel DSNTIP7, DEFINE DATA SETS. Range: YES or NO. Default: YES.
- DSN6SYSP.IMPTSCMP: Enable data compression an implicit base table space. On panel DSNTIP7, USE DATA COMPRESSION. Range: NO or YES. Default: YES.
- DSN6SPRM.MAXTEMPS: Max MB of temp storage in the work file DB that can be used by a single agent at any given time for all temporary tables. On panel DSNTIP9, MAX TEMP STG/AGENT. Range: 0–2147483647. Default: 0 (meaning no limit ~ previous releases).
- DSN6SYSP.MAXOFILR: Maximum number of open files for LOB file references. On panel DSNTIPE, MAX OPEN FILE REFS. Range: 0–value of MAX USERS field. Default: 100.
- DSN6SYSP.TBSBP8K: Default 8 KB BP for CREATE TABLESPACE. On panel DSNTIP1, DEFAULT 8-KB BUFFER POOL FOR USER DATA. Range: BP8K0–BP8K9. Default: BP8K0.
- DSN6SYSP.TBSBP16K: Default 16 KB BP for CREATE TABLESPACE. On panel DSNTIP1, DEFAULT16-KB BUFFER POOL FOR USER DATA. Range: BP16K0–BP16K9. Default: BP16K0.
- DSN6SYSP.TBSBP32K: Default 32 KB BP for CREATE TABLESPACE. On panel DSNTIP7, DEFINE DATA SETS. Range: YES or NO. Default: YES.
- DSN6SPRM.SYSTEM_LEVEL_BACKUPS: Whether the RECOVER Utility should use system-level backups as a recovery base for object level recoveries. On panel DSNTIP6, SYSTEM-LEVEL BACKUPS. Range: NO or YES. Default: NO.
- DSN6SPRM.RESTORE_RECOVER_FROMDUMP: This specifies for the RESTORE SYSTEM and the RECOVER Utilities whether the system-level backup that has been chosen as the recovery base should be used from the disk copy of the system-level

backup (NO), or from the dump on tape (YES). On panel DSNTIP6, RESTORE/RECOVER. Range: NO or YES. Default: NO.

- DSN6SPRM.UTILS_DUMP_CLASS_NAME: DFSMShsm dump class for RESTORE SYSTEM to restore from a system-level backup dumped to tape. On panel DSNTIP6, DUMP CLASS NAME. Range: Valid SMS class name. Default: blank.
- DSN6SPRM.RESTORE_TAPEUNITS: Max tape units that RESTORE SYSTEM can allocate to restore a system-level backup from tape. On panel: DSNTIP6, MAXIMUM TAPE UNITS. Range: 0–256 or NOLIMIT. Default: NOLIMIT.
- DSN6SPRM.STATCLUS: RUNSTATS clustering statistics type. On panel: DSNTIP6, STATISTICS CLUSTERING. Range: ENHANCED or STANDARD. Default: ENHANCED.
- DSN6SPRM.MXDTCACH: Max MB of virtual memory allocated for data caching. On panel DSNTIP8, MAX DATA CACHING. Range: 0–512. Default: 20.
- DSN6SPRM.EDM_SKELETON_POOL: Minimum size in KB of the EDM skeleton pool. On panel DSNTIPC, EDM SKELETON POOL SIZE. Range: 5000–1048576. Default: 102400.
- DSN6SPRM.OPTXQB: Enable/Disable global query optimization. No panel. Range: ON or OFF. Default: ON
- DSNHDECM.DEF_DECFLOAT_ROUND_MODE: Default rounding mode for decimal floating point data type. On panel DSNTIPF, DECFLOAT ROUNDING MODE. Range: ROUND_CEILING, ROUND_DOWN, ROUND_FLOOR, ROUND_HALF_DOWN, ROUND_HALF_EVEN, ROUND_HALF_UP, ROUND UP. Default: ROUND_HALF_EVEN
- DSN6SPRM.ADMTPROC: JCL proc name used to start the task address space for the DB2 admininstrative scheduler.
- DSN6SPRM.HONOR_KEEPDICTIONARY: This specifies whether DB2 should honor or ignore the KEEPDICTIONARY parameter on LOAD REPLACE or REORG request that converts a table space from basic row format to reordered row format. Valid settings are NO and YES. The default is NO.
- DSN6SPRM.MAX_CONCURRENT_PKG_OPS: This specifies the maximum number of concurrent package requests that can be processed simultaneously.
- DSN6SYSP.TBSBPLOB: Name of the bufferpool to be used by LOB table spaces that are implicitly created. Any valid DB2 bufferpool name is accepted. The default is BP0.
- DSN6SYSP.TBSBPXML: Name of the bufferpool to be used by XML table spaces that are implicitly created. Any valid 16K BP name is accepted. The default is BP16K0.

Archived

13

Performance

DB2 9 is an exciting new version with many improvements in performance and very little regression. Changes in index management have reduced CPU utilization for most utilities. DB2 9 improves availability and security and adds greatly to SQL and XML function. Optimization improvements include more SQL functions, improved data for the optimizer, better optimization techniques, and a new approach to providing information for tuning.

For details see the companion book *DB2 9 for z/OS Performance Topics*, SG24-7473.

In this chapter we introduce the DB2 9 for z/OS functions that are closely related to performance improvements.

This chapter describes the following:

- Expectations and best practices
- Enhanced sequential key insert
- REOPT AUTO based on parameter marker change
- RLF enhancements
- Histogram statistics
- Utilities CPU reduction
- Global query optimization
- Generalized sparse index and in-memory data cache
- Dynamic index ANDing for star join queries
- Large Object (LOB/XML) flow optimization
- LOBs performance improvements
- WLM-assisted buffer pool management

© Copyright IBM Corp. 2007. All rights reserved.

13.1 Expectations and best practices

Your mileage will definitely vary, but Figure 13-1 shows a common shape for a performance plan. Starting with zero for the V7 base line, once you move to V8, CPU time generally increases from 5% to 10%, shown as 7 in our example. Start with REORG and collect improved statistics for non-uniform distribution of data on non-indexed columns. The V8 compatibility mode performance plan REBINDs the primary packages, long-term page fixes the I/O intensive buffer pools (high ratio of numbers of pages read and written over number of buffers), and adjusts DSNZPARMs. During the CM, a zIIP is added if your peak workload includes DRDA SQL, parallel queries, or LOAD, REORG, and REBUILD. The REBIND process can provide most of the improved access paths. Data sharing batching helps CM.

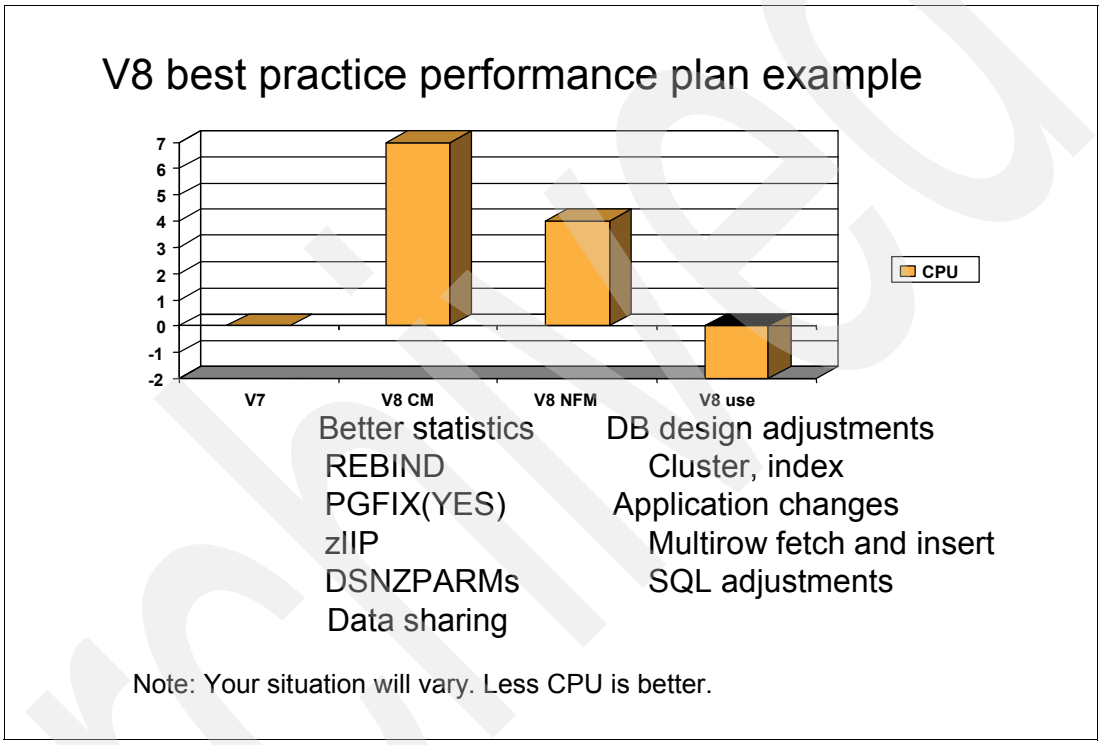


Figure 13-1 V8 best practice performance plan example

In moving to NFM, additional DSNZPARMs are adjusted and all of the plans and packages are rebound. Database designs can start taking advantage of new clustering and indexing options, such as NOT PADDED, for large varchar indexed columns. After making the design changes, REORG the data and REORG or REBUILD the indexes, get the improved statistics, and REBIND. When the data sharing group is quiesced, protocol 2 locking is used. If your environment does not use variable length indexes and/or data sharing, performance in NFM are pretty similar to CM.

V8 use takes wider advantage of the V8 performance improvements. MQTs, DPSI, more not-padded indexes, multi-row Fetch, cursor Update, cursor Delete, Insert, and use of other SQL improvements make V8 use less CPU intensive than V7. The workload grows, but much of the growth can use the zIIP.

For details and recommendations on V8 performance see *DB2 UDB for z/OS Version 8 Performance Topics*, SG24-6465.

Let us now have a preview of the DB2 9 performance plan. Again, your mileage will vary, but Figure 13-2 on page 495 shows a common shape for a performance plan, starting with zero

for the V8 base line. When you first move to V9, CPU time will generally stay the same (+/-3%) for z9 and z990 systems, shown in this case as -3. Utility performance can help right away. You should start with REORG and collect improved histogram statistics. The V9 CM performance plan REBINDs the primary packages and adjusts DSNZPARMs. The REBIND process can provide most of the improved access paths. On z900 the initial CPU expectation is 5 to 10%, due to the hardware not fully supporting long displacement intructions, so making the adjustments is more important.

In moving to NFM, additional DSNZPARMs are adjusted and all of the plans and packages are rebound. After making the design changes, REORG the data and REORG or REBUILD the indexes, get the improved statistics, and REBIND. Generally NFM will perform equally or better that CM.

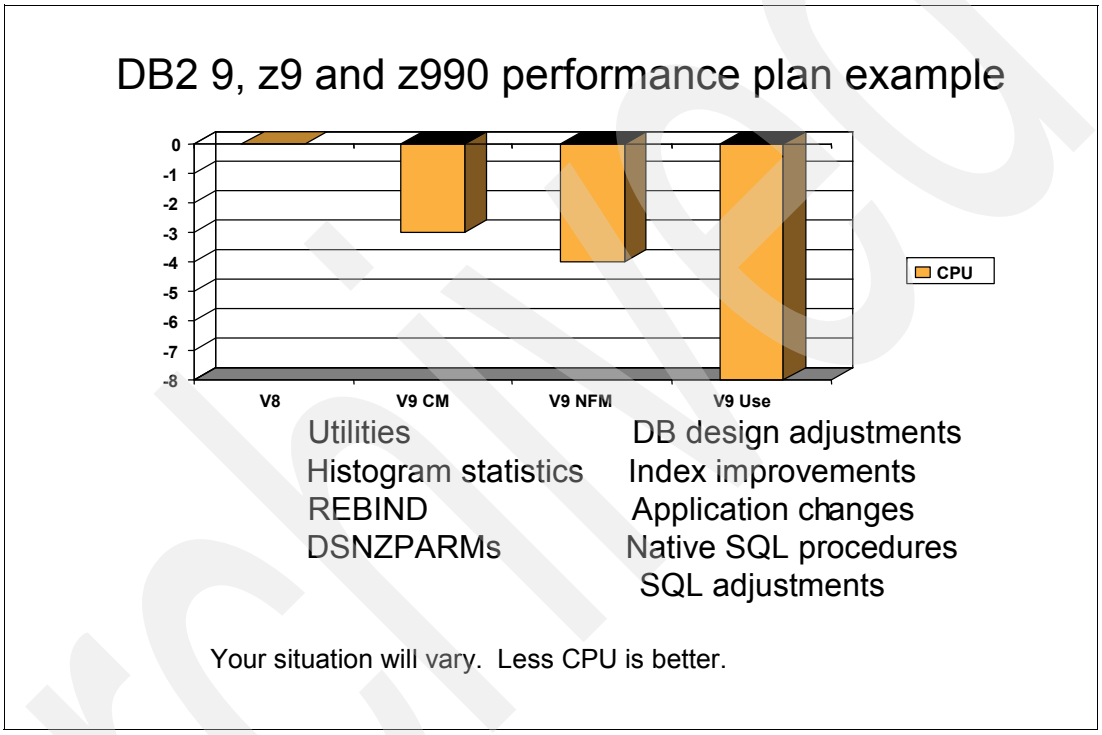


Figure 13-2 DB2 9 z9 and z990 performance plan example

The V9 use line is taking wider advantage of the V9 performance improvements. Database designs start taking advantage of new indexing options, such as index compression, index on expression, and larger pages. Native SQL procedures, added use of zIIP, and improved SQL continue the improvements in this phase.

13.2 Enhanced sequential key insert

Currently, an index page splits with a (roughly) 50/50 ratio so that half of the index keys on the splitting page remain on this page and half move to the new page unless the split is triggered by an insert that is higher than any existing key in the index. The insertion of the highest key in an index can trigger a special asymmetric split of the last leaf page in the index so that all existing keys remain on the splitting page, leaving room for future insertions on the new page. However, in an index with sequential inserts into ever-increasing or decreasing ranges of values in the middle of the index, the current splitting behavior can leave pages in the index that remain 50% empty after splitting, since no further inserts would occur on these pages. The 50/50 split along with the current 4-KB index page size limit can also cause more

frequent splits to occur in the index and consequently increase index tree contention. Non partitioned indexes (NPIs) updated by LOAD utilities running in parallel against multiple partitions are especially susceptible to these problems, since such a scenario can cause sequential inserts to occur into multiple ranges in the NPI.

This enhancement alleviates the problems described above by introducing:

- Asymmetric splitting for index pages (other than the last index leaf page) to accommodate varying patterns of inserts into an index
- Larger than 4 KB page sizes for indexes

13.2.1 Details of enhancement

With the roughly 50/50 index page split available today, an index with sequential insert patterns such as the one in Figure 13-3 would experience frequent page splits that would leave half of the splitting pages empty.

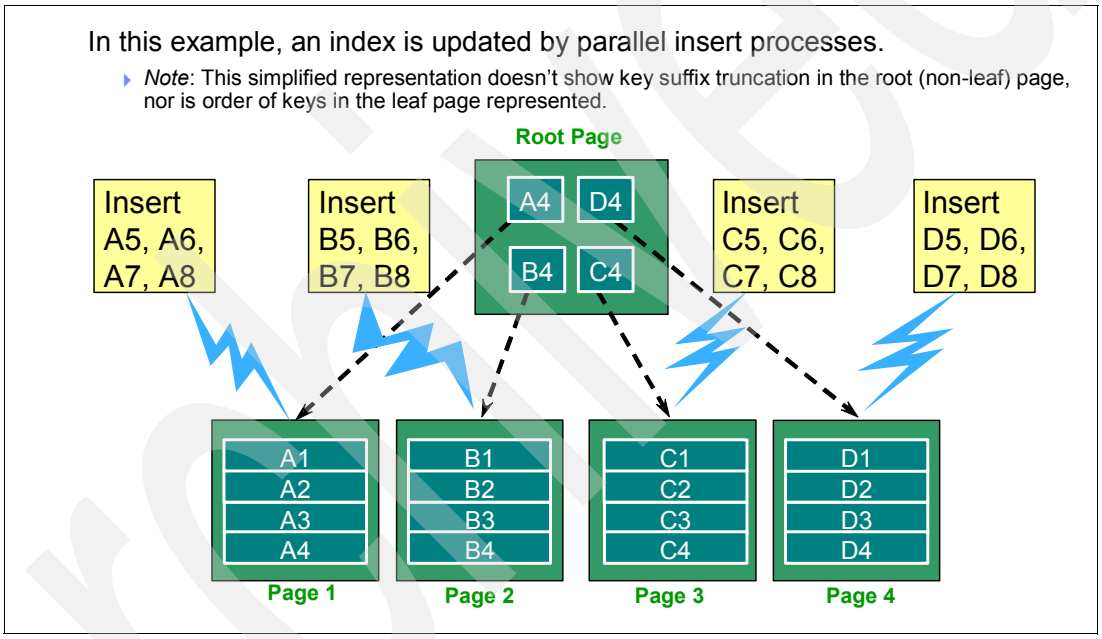


Figure 13-3 Index with multiple increasing sequential insert patterns

Archived

Figure 13-4 shows the index after pages 1 and 2 have split using the 50/50 split.

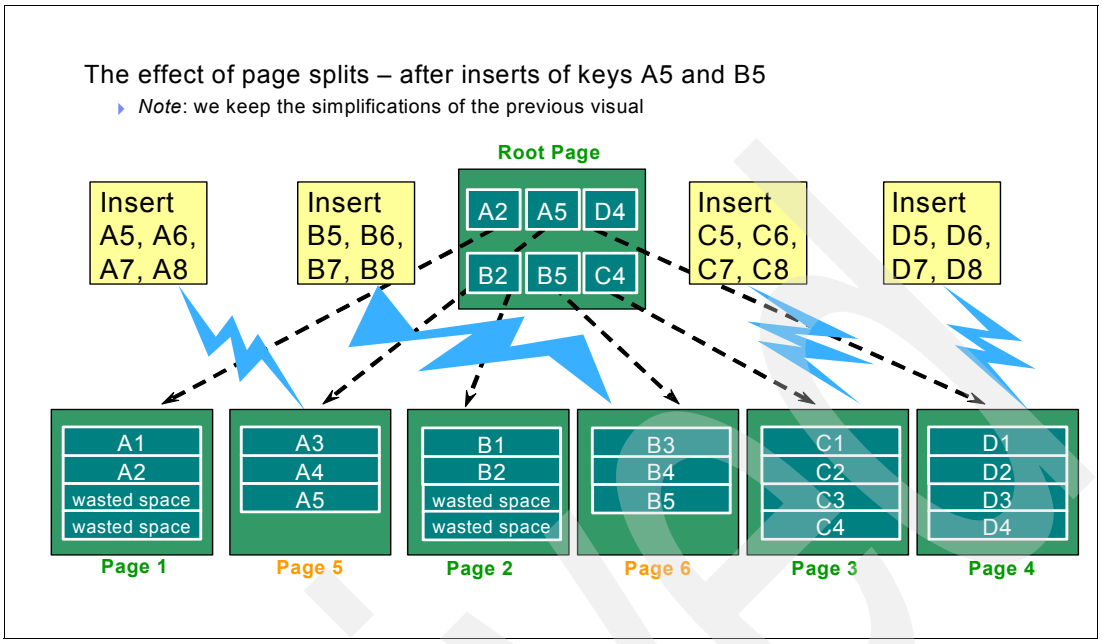


Figure 13-4 Index after pages 1 and 2 have split using the 50/50 split algorithm

An index experiencing this type of insert pattern can benefit from asymmetric page splits. If pages 1 and 2 in the index from Figure 13-4 were to split asymmetrically, the index would experience fewer page splits because most of the keys originally on pages 1 and 2 would remain on these pages after the splits.

With this enhancement DB2 detects various insert patterns in an index and split appropriately for a given insert pattern by choosing from among several index page split algorithms.

If an ever-increasing sequential insert pattern is detected for an index, DB2 splits index pages asymmetrically so that most existing keys stay on the original index page, thereby leaving more space on the new index page for future inserts and preventing pages that are 50% empty from being left in the index.

Conversely, if an ever-decreasing sequential insert pattern is detected in an index, DB2 splits index pages asymmetrically so that most existing keys are moved to the new index page, thereby making room on the original index page for future inserts.

If a random insert pattern is detected in an index, DB2 splits index pages with a roughly 50/50 ratio.

Figure 13-5 shows the index after pages 1 and 2 have split asymmetrically.

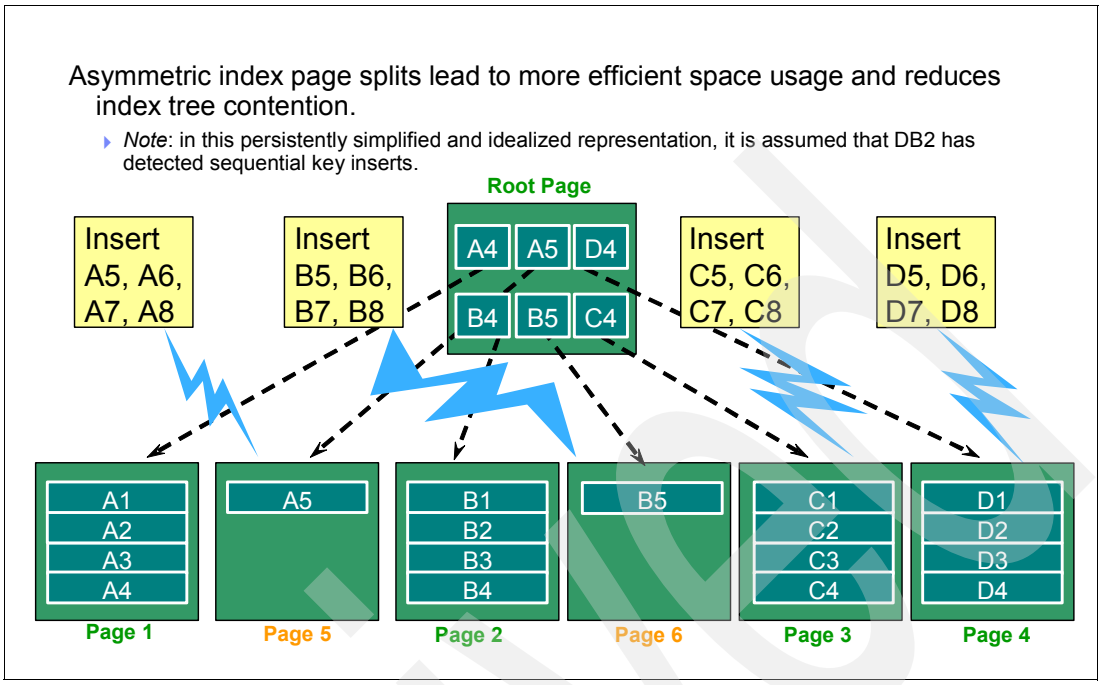


Figure 13-5 Index after pages 1 and 2 have split asymmetrically

Currently, you can only specify a 4-KB buffer pool on the CREATE INDEX statement for a particular index. To allow indexes to have pages that are greater than 4 KB in size, you can now specify 8 KB, 16 KB, and 32 KB buffer pools on the CREATE INDEX statement. Also, you can now identify a 4 KB, 8 KB, 16 KB, or 32 KB index buffer pool on the CREATE DATABASE and ALTER DATABASE statements when using the INDEXBP option. These options are available in DB2 for z/OS Version 9 new-function mode only.

13.3 REOPT AUTO based on parameter marker change

For an SQL statement with input host variables static or dynamic, the access path chosen by the optimizer during bind time (before the values of host variables are available) may not always be optimal.

The bind option REOPT(ALWAYS) solves this problem by preparing the statement again at run time when the input variable values are available, so that the optimizer can re-optimize the access path using the host variable values. However, for frequently called SQL statements that take very little time to execute, re-optimization using different input host variable values at each execution time is expensive, and it may affect the overall performance of applications.

The REOPT(ONCE) bind option introduced in V8 tries to combine the benefits of REOPT(ALWAYS) and dynamic statement caching. The idea of REOPT(ONCE) is to re-optimize the access path only once (using the first set of input variable values) no matter how many times the same statement is executed. The access path chosen based on the set of input variable values is stored in the dynamic statement cache and used for all later executions (as with normal dynamic statement caching). This solution is based on the assumption that the chosen set of host variable values at run time is better than the default one chosen by the optimizer at bind time.

The three options REOPT(ONCE), REOPT(NONE), and REOPT(ALWAYS) can be specified in the BIND and REBIND commands for plans and packages. REOPT(NONE) is the default option.

REOPT(ONCE) only applies to dynamic SQL statements and is ignored if you use it with static SQL statements. DB2 for z/OS caches only dynamic statements. If a dynamic statement in a plan or package that is bound with REOPT(ONCE) runs when dynamic statement caching is turned off (DSNZPARM CACHEDYN=NO), the statement runs as though REOPT(ONCE) is not specified.

13.3.1 REOPT enhancement in V9

Despite all of the above enhancements, the impact of host variables is still very visible and can result in less than efficient access paths. The requirement is to come up with the optimal access path in the minimum number of prepares.

If you specify the new REOPT(AUTO) bind option, DB2 automatically determines whether a new access path is required to further optimize the performance of a statement for each execution. REOPT(AUTO) only applies to dynamic statements that can be cached. If dynamic statement caching is turned off and DB2 executes a statement that is bound with REOPT(AUTO), no reoptimization occurs.

Consider using the REOPT(AUTO) bind option to achieve a better balance between the costs of reoptimization and the costs of processing a statement. You might use the REOPT(AUTO) bind options for many statements for which you could choose either the REOPT(ALWAYS) or REOPT(NONE) bind options, and especially in the following situations:

- The statement is a dynamic SQL statement and can be cached.
- The SQL statement sometimes takes a relatively long time to execute, depending on the values of referenced parameter markers, especially when parameter markers refer to non-uniform data that is joined to other tables. For such SQL statements, the performance gain from a new access path that is chosen based on the input variable values for each might or might not be greater than the performance cost of reoptimization when the statement runs.

13.4 RLF enhancements

The DB2 Resource Limit Facility (RLF) is used to govern DB2 resources (processor time). The current RLF reactive and predictive governor governs resources using a plan, package, or collection name in conjunction with AUTHID or LUNAME. There are some workloads/applications that do not have a unique AUTHID, plan name, or package name. This presents some difficulty in governing these resources. For example, middleware servers do not access DB2 with unique plans, collections, packages, and authids:

- Middleware servers accessing DB2 through DDF use the same plan name, DISTSERV.
- Middleware servers using CLI and JDBC APIs use a common set of packages.
- Middleware servers connect using a single authorization ID.

The DB2 RLF function is enhanced to provide the middleware servers with a better way to govern their resources. The enhancements are provided by using a new Resource Limit Middleware Table (RLMT) and a unique index in the existing database DSNRLST.

Archived

13.4.1 The new Resource Limit Middleware Table and unique index

Figure 13-6 shows the DDL for creating the new RLMT table and unique index.

The New RLMT Table and Index

```
CREATE TABLE authid.DSNRLMTnn
(RLFFUNC         CHAR(1)   NOT NULL WITH DEFAULT,
 RLFEUAN         CHAR(32)  NOT NULL WITH DEFAULT,
 RLFEUID         CHAR(16)  NOT NULL WITH DEFAULT,
 RLFEUWN         CHAR(18)  NOT NULL WITH DEFAULT,
 RLFIP           CHAR(254) NOT NULL WITH DEFAULT,
 ASUTIME         INTEGER,
 RLFASUERR       INTEGER,
 RLFASUWARN      INTEGER,
 RLF_CATEGORY_B CHAR(1)    NOT NULL WITH DEFAULT)
IN DSNRLST.DSNRLSnn;

CREATE UNIQUE INDEX authid.DSNMRLnn
ON DSNRLMTnn
(RLFFUNC, RLFEUAN DESC, RLFEUID DESC,
 RLFEUWN DESC, RLFIP DESC)
CLUSTER CLOSE NO;
```

Figure 13-6 Resource Limit Middleware Table DSNRLMTnn and Unique Index DSNMRLnn

In both the table and index names *nn* is any two-character alphanumeric value, and *authid* is specified when DB2 is installed. Because the two characters nn must be entered as part of the START command, they must be alphanumeric (no special or DBCS characters). The nn in the index name (DSNMRLnn) must match the nn in the table name (DSNRLMTnn).

Use the SQL statements INSERT, UPDATE, MERGE, and DELETE to populate the RLMT. The limit that exists when a job makes its first dynamic SELECT, INSERT, UPDATE, MERGE, TRUNCATE, or DELETE statement applies throughout the life of the job. If you update the RLMT while a job is executing, that limit for that job does not change. Instead, the updates are effective for all new jobs and for those that have not issued their first dynamic SELECT, INSERT, UPDATE, MERGE, TRUNCATE, or DELETE statement. To insert, update, merge, or delete from the RLMT, you need only the usual table privileges on the RLMT. No higher authority is required.

13.4.2 The Resource Limit Middleware Table columns

Here is a description for all columns in the RLMT. The columns that you must populate depend on what function is performed by that row (determined by the value in RLFFUNC) and how narrowly you want to qualify values. For example, you can qualify broadly by leaving the RLFEUAN column blank, which means that the row applies to all user IDs. Or you can qualify very narrowly by specifying a different row for each end user ID for which the function applies.

Search order: DB2 tries to find the most exact match when it determines which row to use for a particular function. The search order depends on which function is being requested

(reactive or predictive governing). The search order is described under each of those functions.

RLFFUNC specifies how the row is used. The values that have an effect are:

- ’8’

  The row reactively governs dynamic SELECT, INSERT, UPDATE, MERGE, TRUNCATE, or DELETE statements by client information (RLFEUID, RLFEUAN, RLFEUWN, and RLFIP).

- ’9’

  The row predictively governs dynamic SELECT, INSERT, UPDATE, MERGE, TRUNCATE, or DELETE statements by client information (RLFEUID, RLFEUAN, RLFEUWN, and RLFIP).

All other values are ignored.

RLFIP specifies the IP address of the location where the request originated. A blank value in this column represents all locations.

RLFEUAN specifies an application name. A blank value in this column means that the row applies to all application names from the location specified in RLFIP.

RLFEUID specifies an end user's user ID. A blank value means that the limit specifications in this row apply to every user ID for the location that is specified in RLFIP.

RLFEUWN specifies an end user's workstation name. A blank value in this column means that the row applies to all workstation names from the location that is specified in RLFIP.

ASUTIME specifies the number of processor service units allowed for any single dynamic SELECT, INSERT, UPDATE, MERGE, TRUNCATE, or DELETE statement. Use this column for reactive governing. If ASUTIME is null it indicates No limit. If ASUTIME is 0 (zero) or a negative value it indicates that no dynamic SELECT, INSERT, UPDATE, MERGE, TRUNCATE, or DELETE statements are permitted.

RLFASUERR is used for predictive governing (RLFFUNC= '9') and only for statements that are in cost category A. This specifies the error threshold number of processor service units allowed for a single dynamic SELECT, INSERT, UPDATE, MERGE, TRUNCATE, or DELETE statement. If the predicted processor cost (in service units) is greater than the error threshold, an SQLCODE -495 is returned to the application. If RLFASUERR is null it indicates no error threshold. If RLFASUERR is 0 (zero) or a negative value it indicates that all dynamic SELECT, INSERT, UPDATE, MERGE, TRUNCATE, or DELETE statements receive SQLCODE -495.

RLFASUWARN is used for predictive governing (RELFFUNC= '9'), and only for statements that are in cost category A. This specifies the warning threshold number of processor service units that are allowed for a single dynamic SELECT, INSERT, UPDATE, MERGE, TRUNCATE, or DELETE statement. If the predicted processor cost (in service units) is greater than the warning threshold, an SQLCODE +495 is returned to the application. If RLFASUERR is null it indicates no warning threshold. If RLFASUWARN is 0 (zero) or a negative value it indicates that all dynamic SELECT, INSERT, UPDATE, MERGE, TRUNCATE, or DELETE statements receive SQLCODE +495.

Make sure that the value for RLFASUWARN is less than that for RLFASUERR. If the warning value is higher, the warning is never reported. The error takes precedence over the warning.

RLF_CATEGORY_B is used for predictive governing (RLFFUNC='9'). Tells the governor the default action to take when the cost estimate for a given statement falls into cost category B,

which means that the predicted cost is indeterminate and probably too low. You can tell if a statement is in cost category B by running EXPLAIN and checking the COST_CATEGORY column of the DSN_STATEMNT_TABLE.

COST_CATEGORY A indicates that DB2 had enough information to make a cost estimate without using default values. COST_CATEGORY B indicates that some condition exists for which DB2 was forced to use default values. The value in column REASON indicates why DB2 was unable to put this estimate in cost category A. Table 13-1 shows the possible reasons for COST_CATEGORY B.

Table 13-1 Reasons for COST_CATEGORY B

HAVING CLAUSE	A subselect in the SQL statement contains a HAVING clause.
HOST VARIABLES	The statement uses host variables, parameter markers, or special registers.
REFERENTIAL CONSTRAINTS	Referential constraints of the type CASCADE or SET NULL exist on the target table of a DELETE statement.
TABLE CARDINALITY	The cardinality statistics are missing for one or more of the tables that are used in the statement.
TRIGGERS	Triggers are defined on the target table of an INSERT, UPDATE, or DELETE statement.
UDF	The statement uses user-defined functions.
MATERIALIZATION	Statistics are missing because the statement uses materialized views or nested table expresses.

The acceptable values for RLF_CATEGORY_B are:

- blank

  By default, prepare and execute the SQL statement.
- Y

  Prepare and execute the SQL statement.
- N

  Do not prepare or execute the SQL statement. Return SQLCODE -495 to the application.
- W

  Complete the prepare, return SQLCODE +495, and allow the application logic to decide whether to execute the SQL statement.

Archive

13.4.3 Example of query limits

Table 13-2 shows examples of reactive and predictive client-based query limits in an RLMT.

Table 13-2 Qualifying rows for DSNRLMTxx

RLFFUNC	RLFEUAN	RLFEUID	RLFEUWN	RLFIP	ASUTIME
8	APP1	PAUL	(blank)	9.30.72.223	12000
9	(blank)	(blank)	WORKSTA10	(blank)	7000

The first row in Table 13-2 shows that when PAUL runs the APP1 application from the 9.30.72.223 IP address, the resource limit facility limits dynamic queries run by APP1 to 12000 service units (SUs) each. The second row shows that any queries that are requested from work station 10 are predictively limited to 7000 SUs each.

13.4.4 Providing client information to RLF

To use the enhanced RLF functions mentioned above, in addition to populating the new RLMT table, the user must provide client information to DB2 using various client API interfaces.

Middleware servers using CLI set client information by the "sqleseti" API. Details on providing client information to the DB2 server can be found in *IBM DB2 Administrative API Reference,* SC10-4231.

Set Client Information passed in:

```
SQLE_CLIENT_INFO_USERID,
SQLE_CLIENT_INFO_WRKSTNNAME,
SQLE_CLIENT_INFO_APPLNAME
```

Middleware servers using JDBC use the following methods to provide client information to the DB2 server. Details on providing CLI information to the DB2 server can be found in the "DB2 for z/OS Application Programming Guide and Reference for Java" publication.

```
void setDB2ClientApplicationInformation(String info)
void setDB2ClientUser(String user)
void setDB2ClientWorkstation(String name)
```

13.4.5 Operational considerations

You can create several resource limit tables (RLST and RLMT) and start and stop them as needed to support the type and amount of processing that occurs at different times. Activate any particular resource limit table by using the DB2 command START RLIMIT ID=*nn*, where *nn* is the two-character identifier that you specified on the name DSNRLSTnn or DNSRLMTnn. This command gives you the flexibility to use different resource limit tables at different times. However, only one pair of resource limit tables, one RLST and one RLMT, can be active at any given time.

You can also use the DISPLAY RLIMIT and STOP RLIMIT commands as in prior versions. You require SYSOPR, SYSCTRL, or SYSADM authority to issue the three commands.

If in Compatibility Mode, when the -START RLF command is issued:

- If the DSNRLSTnn table does not exist, then existing messages DSNT706I and DSNT708I are issued, and the command fails.

- If the DSNRLSTnn table exists but the new DSNRLMTnn table has not yet been created, then new message DSNT727I is issued to indicate that DSNRLMTnn will not be used by RLF.

  Existing message DSNT704I is issued, indicating that RLF started successfully using the DSNRLSTnn table.

If in new-function mode, when the –START RLF command is issued, if neither of the DSNRLSTnn or DSNRLMTnn tables exist, then the existing DSNT706I and DSNT708I messages are issued for each table and index respectively. The command fails.

- If one of the tables exists but not both, then new message DSNT727I is issued for the missing table and the existing DSNT704I message is issued to indicate that RLF has started successfully using the one table that exists.
- If both tables exist, two DSNT704I messages are issued to indicate that RLF has started successfully with both tables.

At installation time, you can specify a default resource limit table to be used each time DB2 is restarted.

If the governor is active and you restart it without stopping it, any jobs that are active continue to use their original limits, and all new jobs use the limits in the new table.

If you stop the governor while a job is executing, the job runs with no limit, but its processing time continues to accumulate. If you later restart the governor, the new limit takes effect for an active job only when the job passes one of several application checkpoints. A typical dynamic statement, which builds a result table and fetches from it, passes those checkpoints at intervals that can range from moments to hours. As a result, your change to the governor might not stop an active job within the time you expect.

Use the DB2 command CANCEL THREAD to stop an active job that does not pick up the new limit when you restart the governor.

While the resource limit facility (governor) is active, you cannot execute the following SQL statements on the resource limit table, or the table space and database that contain the resource limit table:

- DROP DATABASE
- DROP INDEX
- DROP TABLE
- DROP TABLESPACE
- RENAME TABLE

You cannot stop a database or table space that contains an active resource limit table, nor can you start the database or table space with ACCESS(UT).

13.4.6 Installation and migration

The DSNRLST database is created once during installation of a new DB2 subsystem or data sharing group, by job DSNTIJSG. In V9, DSNTIJSG creates the DSNRLSTnn table and DSNARLnn index as in previous releases. V9 DSNTIJSG also provides DDL to create the new DSNRLMTnn table and DSNMRLnn index, but the DDL is commented out because not all sites want to use the new feature, and an empty DSNRLMTnn table increases RLF overhead with no benefit.

For DB2 migration, the DDL for creating the RLST is typically removed from DSNTIJSG by the installation CLIST. For V9 migration, the installation CLIST preserves only the commented-out DDL for creating the new DSNRLMTnn table and DSNMRLnn index.

13.5 Histogram statistics

RUNSTATS normally collects frequency statistics for a single-column or single multi-column data set. Because catalog space and bind time performance concerns make the collection of these types of statistics on every distinct value found in the target column or columns very impractical, such frequency statistics are commonly collected only on the most frequent or least frequent, and therefore most biased, values. Such limited statistics often do not provide an accurate prediction of the value distribution because they require a rough interpolation across the entire range of values.

Table 13-3 shows example frequency statistics for the YRS_OF_EXPERIENCE column in the EMPLOYEE table.

Table 13-3 Frequency statistics for YRS_OF_EXPERIENCE column in EMPLOYEE table

VALUE	FREQUENCY
2	10%
25	15%
26	15%
27	7%
12	0.02%
13	0.01%
40	0.0001%
41	0.00001%

Some example predicates on values in this table include:

- Range predicate:

```
SELECT T.EMPID FROM EMPLOYEE T
WHERE T.YRS_OF_EXPERIENCE BETWEEN 5 AND 10;
```

- Non-local predicate:

```
SELECT T1.EMPID FROM EMPLOYEE T1, OPENJOBS T2
WHERE T1.SPECIALTY = T2.AREA AND T1.YRS_OF_EXPERIENCE > T2.YRS_OF_EXPERIENCE;
```

For each of the above predicates, distribution statistics for any single value cannot help DB2 to estimate predicate selectivity, other than by uniform interpolation of filter factors over the uncollected part of the value range. The result of such interpolation might lead to inaccurate estimation and undesirable access path selection. You can improve access path selection by specifying the histogram statistics option HISTOGRAM in RUNSTATS.

Archive

13.5.1 Histogram statistics collection

Histogram statistics is a way of summarizing data distribution. This technique divides up the range of possible values in a data set into intervals, such that each interval contains approximately the same percentage of the values.

When you use RUNSTATS to collect statistics on a column that contains such wide-ranging frequency values, specify the HISTOGRAM statistics option to collect more granular distribution statistics that account for the distribution of values across the entire range of values. This helps to improve access path selection.

The number of intervals is specified by the value of NUMQUANTILES when you use the HISTOGRAM option of RUNSTATS. Each interval has an identifier value QUANTILENO, and values, the LOWVALUE and HIGHVALUE columns, that bound the interval. DB2 collects distribution statistics for each interval.

Table 13-4 shows the result of collecting histogram statistics for the YRS_OF_EXPERIENCE column values in the EMPLOYEE table. In this example, the statistics have been collected with 7 intervals.

Table 13-4 Histogram statistics for YRS_OF_EXPERIENCE column in EMPLOYEE table

QUANTILENO	LOWVALUE	HIGHVALUE	CARDF	FREQUENCYF
1	0	3	4	14%
2	4	15	8	14%
3	18	24	7	12%
4	25	25	1	15%
5	26	26	1	15%
6	27	30	4	16%
7	35	40	6	14%

Histogram statistics enable DB2 to improve access path selection by estimating predicate selectivity from value-distribution statistics that are collected over the entire range of values in a data set. RUNSTATS cannot collect histogram statistics on randomized key columns.

DB2 chooses the best access path for a query based on predicate selectivity estimation, which in turn relies heavily on data distribution statistics. Histogram statistics summarize data distribution on an interval scale by dividing the entire range of possible values within a data set into a number of intervals.

DB2 creates equal-depth histogram statistics, meaning that it divides the whole range of values into intervals that each contain about the same percentage of the total number rows.

Note the following characteristics of histogram statistics intervals:

- Each interval includes approximately the same number, or percentage, of the rows. A highly frequent single value might occupy an interval by itself.
- A single value is never broken into more than one interval, meaning that the maximum number of intervals is equal to the number of distinct values on the column. The maximum number of intervals cannot exceed 100, which is the maximum number that DB2 supports.
- Adjacent intervals sometime skip values that do not appear in the table, especially when doing so avoids a large range of skipped values within an interval. For example, if the

value 30 above has 1% frequency, placing it in the seventh interval would balance the percentage of rows in the sixth and seventh intervals. However, doing so would introduce a large skipped range to the seventh interval.

- HIGHVALUE and LOWVALUE can be inclusive or exclusive, but an interval generally represents a non-overlapped value range.
- NULL values, if any exist, occupy a single interval.
- Because DB2 cannot break any single value into two different intervals, the maximum number of intervals is limited to the number of distinct values in the column, and cannot exceed the DB2 maximum of 100 intervals.

Histogram statistics are beneficial when gaps exist in ranges. Figure 13-7 shows an example.

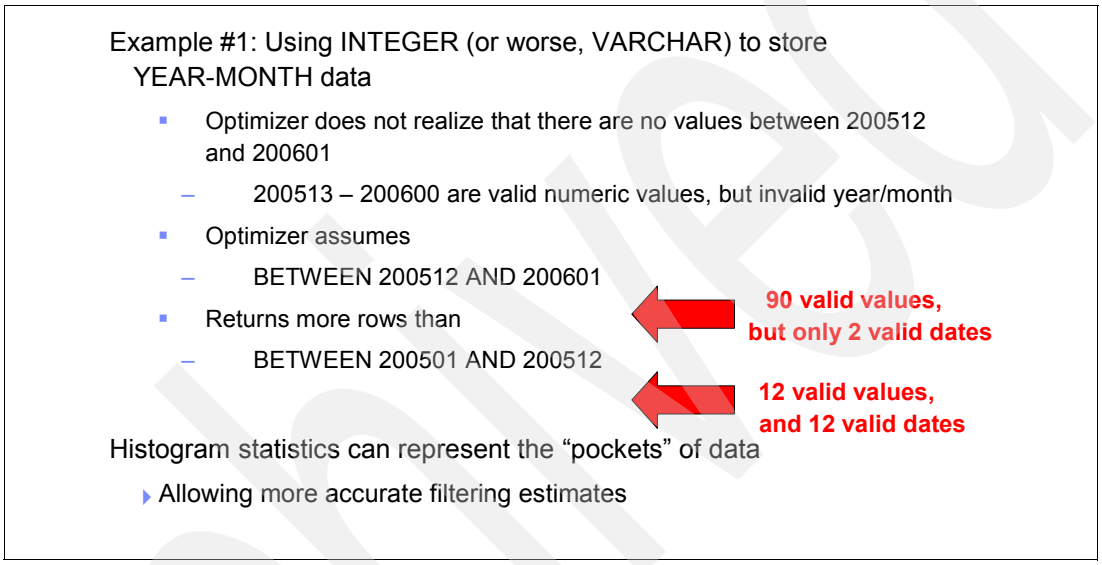


Figure 13-7 Example of histogram statistics benefits when gaps exist in ranges

Figure 13-8 and Figure 13-9 on page 508 show RUNSTATS syntax changes.

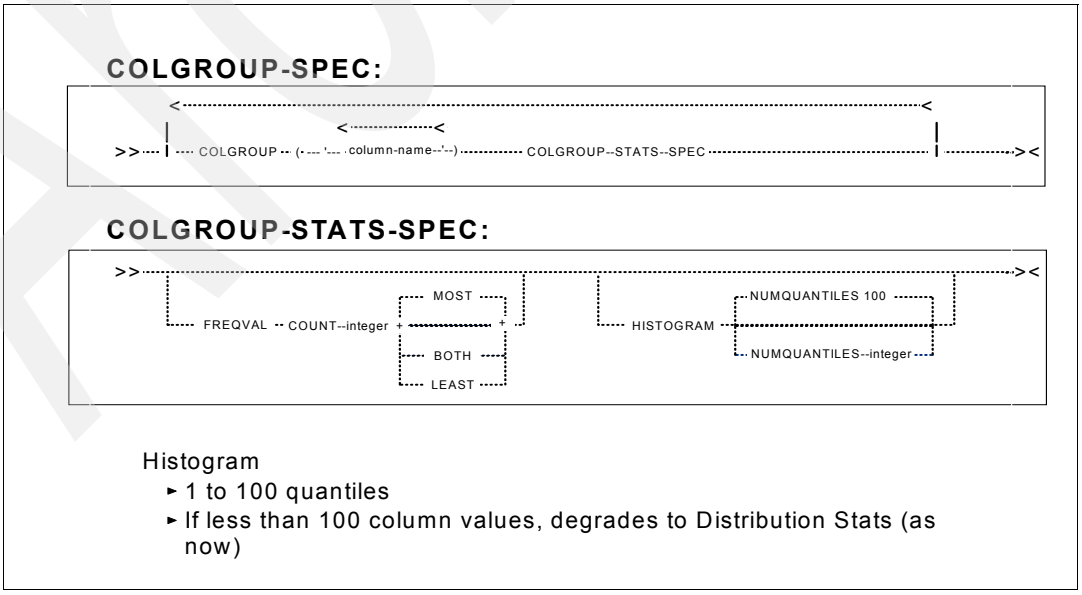


Figure 13-8 RUNSTATS syntax changes 1

When you run RUNSTATS TABLESPACE, you can use the HISTOGRAM option, with the COLGROUP option, to indicate that histogram statistics are to be gathered for the specified group of columns. When you run RUNSTATS INDEX, histogram statistics can only be collected on the prefix columns with the same order. Key columns with a mixed order are not allowed for histogram statistics.

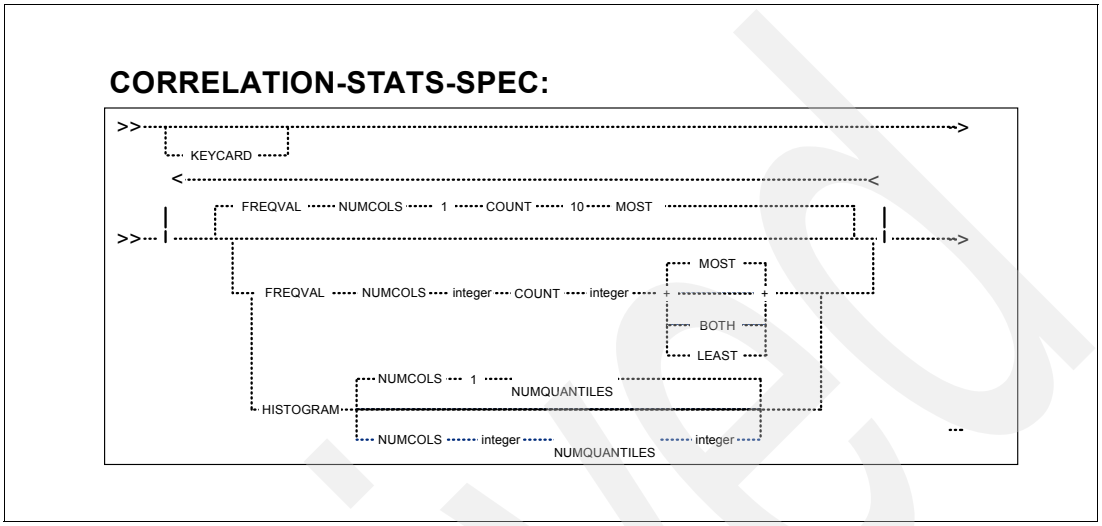


Figure 13-9 RUNSTATS syntax changes 2

You can control the catalog growth by controlling how many additional histogram statistics you want to collect. RUNSTATS wipes off the old histogram statistics for a particular column or column set before populating the new histogram statistics.

13.6 Utilities CPU reduction

Performance measurements show good CPU time reduction for the utilities CHECK INDEX, COPY, LOAD, REBUILD INDEX, RECOVER INDEX, RECOVER TABLESPACE, REORG, and RUNSTATS INDEX. See Figure 13-10.

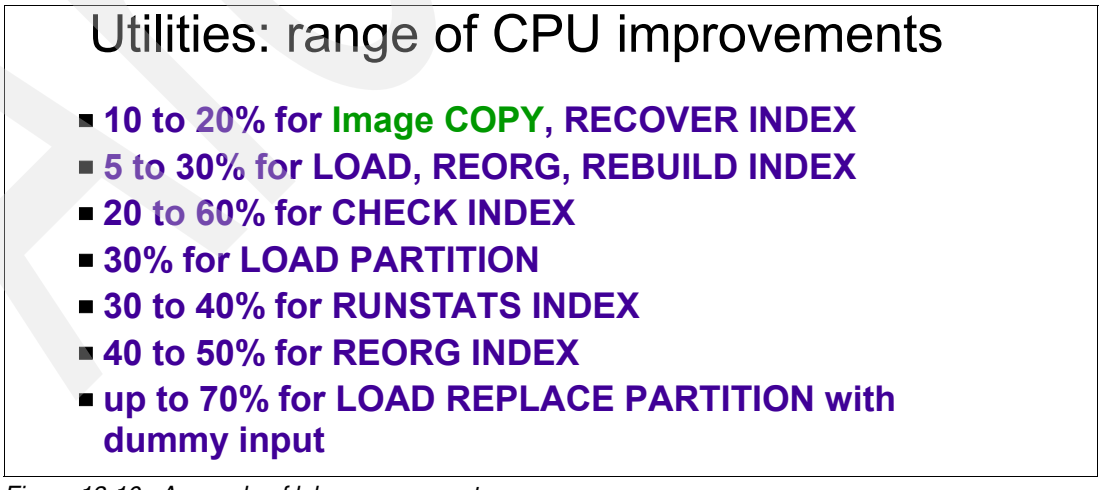


Figure 13-10 A sample of lab measurements

Archived

The primary improvement is in index processing and applies to all utilities shown in Figure 13-10 on page 508 except for Image Copy. You will probably find larger improvements if you have more indexes.

If a table has mostly variable keys, then performance could improve up to 24% in load phase. If a table has average of 8% variable length in the index key, performance could improve 1–2% CPU in load phase.

One exception to the CPU time improvement is the online reorganization for a partition with non-partitioning indexes. Eliminating the BUILD2 phase provides a dramatic improvement in availability, but can increase the CPU time and the elapsed time when one or a few partitions are reorganized. For this process the non-partitioning indexes are copied to the shadow data set.

There is a smaller percentage improvement for LOAD, REORG, and REBUILD if the work is using a zIIP on both V8 and V9.

13.7 CHECK INDEX SHRLEVEL REFERENCE parallelism

With DB2 9, when more than one index is specified in the CHECK SHRLEVEL REFERENCE option, CHECK INDEX checks the indexes in parallel unless constrained by lack of storage or sort work files. Checking indexes in parallel can reduce the elapsed time by sorting the index keys and checking multiple indexes in parallel, rather than sequentially.

Users are notified that parallelism is active by message DSNU395I. A sample output would include:

```
DSNU395I DSNUKPIK - INDEXES WILL BE BUILT IN PARALLEL, NUMBER OF TASKS = 3
```

13.8 Global query optimization

To understand the enhancements introduced for global query optimization, it is useful to have some background information.

13.8.1 Background information

By understanding how DB2 processes subqueries, you can estimate the best method to use when writing a given query when several methods can achieve the same result.

A subquery is a SELECT statement within the WHERE or HAVING clause of an INSERT, UPDATE, MERGE, or DELETE SQL statement. In many cases two or more different SQL statements can achieve identical results, particularly those that contain subqueries. The statements have different access paths, however, and probably perform differently.

Correlated and non-correlated subqueries

Different subqueries require different approaches for efficient processing by DB2. All subqueries can be classified into two categories: correlated and non-correlated.

Correlated subqueries contain a reference to a table or column that is outside of the scope of the subquery.

Example: In the following correlated subquery, the correlation name X is a value from a table that is not listed in the FROM clause of the subquery. The inclusion of X illustrates that the subquery references the outer query block:

```
SELECT * FROM DSN8910.EMP X
WHERE JOB = 'DESIGNER' AND EXISTS
(SELECT 1 FROM DSN8910.PROJ
 WHERE DEPTNO = X.WORKDEPT
 AND MAJPROJ = 'MA2100');
```

Non-correlated subqueries do not refer to any tables or columns that are outside of the scope of the subquery.

Example: The following non-correlated subquery refers only to tables that are within the scope of the FROM clause:

```
SELECT * FROM DSN8910.EMP
WHERE JOB = 'DESIGNER'
  AND WORKDEPT IN (SELECT DEPTNO
                   FROM DSN8910.PROJ
                  WHERE MAJPROJ = 'MA2100');
```

Nested subqueries

Subqueries might also contain their own subqueries. Such nested subqueries can be either correlated or non-correlated. DB2 uses the same processing techniques with nested subqueries that it does for non-nested subqueries, and the same optimization techniques apply. For example:

```
SELECT * FROM T1
WHERE T1.C1 IN (SELECT T2.C1 FROM T2, T3
                WHERE T2.C1 = T3.C1
                 AND EXISTS (SELECT 1 FROM T4, T5
                             WHERE T4.C1 = T5.C1
                              AND T4.C2 = T1.C2
                             AND T4.C3 = T3.C3))
```

In this example, the nested subquery involving (T4, T5) is correlated to T1 and T3, making it a correlated subquery. Since the (T4, T5) subquery is nested within the subquery involving (T2, T3), and since the reference to T1 is outside the scope of the (T2, T3) subquery, this makes the (T2, T3) subquery a correlated subquery.

Conditions for DB2 to transform a subquery into a join

For a SELECT, UPDATE, or DELETE statement, DB2 can sometimes transform a subquery into a join between the result table of the subquery and the result table of the outer query.

For a SELECT statement, DB2 transforms the subquery into a join if the following conditions are true:

- The transformation does not introduce redundancy.
- The subquery appears in a WHERE clause.
- The subquery does not contain GROUP BY, HAVING, or aggregate functions.
- The subquery has only one table in the FROM clause.
- For a correlated subquery, the comparison operator of the predicate containing the subquery is IN, = ANY, or = SOME.
- For a noncorrelated subquery, the comparison operator of the predicate containing the subquery is IN, EXISTS, = ANY, or = SOME.

Archived

- For a noncorrelated subquery, the subquery select list has only one column, guaranteed by a unique index to have unique values.
- For a noncorrelated subquery, the left side of the predicate is a single column with the same data type and length as the subquery's column. (For a correlated subquery, the left side can be any expression.)
- Query parallelism is *not* enabled.

For an UPDATE or DELETE statement, or a SELECT statement that does not meet the previous conditions for transformation, DB2 transforms a correlated subquery into a join if the following conditions are true:

- The transformation does not introduce redundancy.
- The subquery is correlated to its immediate outer query.
- The FROM clause of the subquery contains only one table, and the outer query for SELECT, UPDATE, or DELETE references only one table.
- If the outer predicate is a quantified predicate with an operator of =ANY or an IN predicate, and the following conditions are true:
  - The left side of the outer predicate is a single column.
  - The right side of the outer predicate is a subquery that references a single column.
  - The two columns have the same data type and length.
- The subquery does not contain the GROUP BY or DISTINCT clauses.
- The subquery does not contain aggregate functions.
- The SELECT clause of the subquery does not contain a user-defined function with an external action or a user-defined function that modifies data.
- The subquery predicate is a Boolean term predicate.
- The predicates in the subquery that provide correlation are stage 1 predicates.
- The subquery does not contain nested subqueries.
- The subquery does not contain a self-referencing UPDATE or DELETE.
- For a SELECT statement, the query does not contain the FOR UPDATE OF clause.
- For an UPDATE or DELETE statement, the statement is a searched UPDATE or DELETE.
- For a SELECT statement, parallelism is not enabled.

For a statement with multiple subqueries, DB2 transforms only the last subquery in the statement that qualifies for transformation.

Example: The following subquery can be transformed into a join because it meets the first set of conditions for transformation:

```
SELECT * FROM EMP
WHERE DEPTNO IN
(SELECT DEPTNO FROM DEPT
 WHERE LOCATION IN ('SAN JOSE', 'SAN FRANCISCO')
 AND DIVISION = 'MARKETING');
```

If a department in the marketing division has branches in both San Jose and San Francisco, the result of the SQL statement is not the same if a join were performed. The join makes each employee in this department appear twice because it matches once for the department of location San Jose and again of location San Francisco, although it is the same department. Therefore, it is clear that to transform a subquery into a join, the uniqueness of the subquery

Archived

select list must be guaranteed. For this example, a unique index on any of the following sets of columns would guarantee uniqueness:

- (DEPTNO)
- (DIVISION, DEPTNO)
- (DEPTNO, DIVISION).

The resultant query is:

```
SELECT EMP.* FROM EMP, DEPT
WHERE EMP.DEPTNO = DEPT.DEPTNO
 AND DEPT.LOCATION IN ('SAN JOSE', 'SAN FRANCISCO') AND
     DEPT.DIVISION = 'MARKETING';
```

Example: The following subquery can be transformed into a join because it meets the second set of conditions for transformation:

```
UPDATE T1 SET T1.C1 = 1
WHERE T1.C1 =ANY
 (SELECT T2.C1 FROM T2
  WHERE T2.C2 = T1.C2);
```

13.8.2 Version 9 optimizer enhancements

In this section we discuss Version 9 optimizer enhancements.

Subqueries that DB2 cannot transform to joins

When a subquery does not meet the conditions for transformation into a join, the DB2 optimizer might instead transform the subquery, for more efficient processing, by correlating or de-correlating the queries.

Virtual tables

For optimization purposes, DB2 represents a subquery, which cannot be transformed to a join, as a virtual table. This is similar to the way in which materialized views and table expressions are represented. However, unlike materialized views and table expressions, there may not be any work files involved with a virtual table. The use of virtual tables applies to nested subqueries just as it does to non-nested subqueries.

Early Out processing

Early Out is a technique used to ensure that duplicate rows are not returned by a query when they should not be and without incurring the cost of sorting.

Correlating and de-correlating subqueries

Correlated and non-correlated subqueries have different processing advantages. Where possible, the DB2 optimizer transforms the query to the most efficient type.

The DB2 optimizer might transform a correlated subquery to a non-correlated (that is, de-correlate the subquery) to improve processing efficiency. Likewise, the DB2 optimizer might transform a non-correlated subquery to a correlated (that is, correlate the subquery). When a correlated and non-correlated subquery can achieve the same result, the most efficient way depends on the data. The DB2 optimizer chooses to correlate or de-correlate subqueries based on cost. Correlated subqueries allow more filtering to be done within the subquery. Non-correlated subqueries allow more filtering to be done on the table whose columns are being compared to the subquery result.

Archived

DB2 might correlate a non-correlated subquery, or de-correlate a correlated subquery, that cannot be transformed into a join to improve access path selection and processing efficiency.

Example 1

In the non-correlated subquery below, the BIG_TABLE is materialized (that is, it is scanned and put into a work file). The SMALL_TABLE is joined with the work file. This is how the query is handled in V8.

```
SELECT * FROM SMALL_TABLE A
WHERE A.C1 IN (SELECT B.C1 FROM BIG_TABLE B)
```

In V9, DB2 can transform the above non-correlated subquery into a correlated subquery. With this transformation, DB2 accesses SMALL_TABLE first, accesses the BIG_TABLE using index, and avoids the work file.

```
SELECT * FROM SMALL_TABLE A
WHERE EXISTS (SELECT 1 FROM BIG_TABLE B
              WHERE B.C1 = A.C1)
```

Example 2

In the correlated subquery below, the outer table BIG_TABLE is scanned to obtain the A.C1 value (rather than using matching index access), and a matching index is used to access the SMALL_TABLE. This is how the query is handled in V8.

```
SELECT * FROM BIG_TABLE A
WHERE EXISTS (SELECT 1 FROM SMALL_TABLE B
              WHERE A.C1 = B.C1)
```

In V9, DB2 can transform the above correlated subquery into a non-correlated subquery. With this transformation, DB2 scans the SMALL_TABLE and puts it in work file (it is okay to put a small table into the work file). This allows for a more efficient matching index access on BIG_TABLE.

Some queries cannot be transformed from one form to another. Most set functions and grouping functions make it difficult to transform a subquery from one form to another. Expressions that can prevent such transformation include:

- Set functions and grouping functions

  Most set functions and grouping functions make it difficult to transform a subquery from one form to another.

  Example: In the following query, the non-correlated subquery cannot be correlated to T1 because it would change the result of the SUM function. Consequently, only the non-correlated form of the query can be considered.

  ```
  SELECT * FROM T1
  WHERE T1.C2 IN (SELECT SUM(T2.C2) FROM T2, T3
                  WHERE T2.C1 = T3.C1
                 GROUP BY T2.C1)
  ```

- Correlated ranges and <> comparisons

  Some range comparisons involving correlated columns make it difficult to de-correlate the subquery. This is because when a correlated subquery is de-correlated duplicates may have to be removed in order to consider the virtual table in the outer position (see “Early Out processing” on page 512). This duplicate removal requires a set of *equal-join* predicate columns as the key. Without equal-join predicates the early out process breaks down and does not work. This means that the virtual table can only be considered in correlated form (as the inner table of the join).

Example: DB2 cannot de-correlate the following query and use it to access T1 because removing duplicates on the T2.C2 subquery result does not guarantee that the range predicate correlation does not qualify multiple rows from T1.

```
SELECT * FROM T1
WHERE EXISTS (SELECT 1 FROM T2, T3
              WHERE T2.C1 = T3.C1 AND T2.C2 > T1.C2 AND T2.C2 < T1.C3)
```

13.8.3 EXPLAIN output

The EXPLAIN output in the PLAN_TABLE is modified to show virtual tables that are materialized to a work file. The table name for a virtual table uses a naming convention similar to that used for mini-query blocks (MQBs). The name includes an indicator of the query block number of the associated subquery (that is, DSNFWQB(02)).

The table type for virtual tables that are materialized is W for work file.

Virtual tables that are not materialized are not shown in the EXPLAIN output.

Also, a new column is added to the PLAN_TABLE called "PARENT_PLANNO". For correlated subqueries it corresponds to the plan number in the parent query block where the correlated subquery is invoked. For non-correlated subqueries it corresponds to the plan number in the parent query block that represents the work file for the subquery.

Figure 13-11 shows an example of EXPLAIN output, which includes use of virtual table.

SELECT * FROM T1 WHERE T1.C2 IN
(SELECT T2.C2 FROM T2, T3 WHERE T2.C1 = T3.C1)

QBNO	PLAN-NO	METHOD	TNAME	AC-TYPE	MC	AC-NAME	SC-JN	PAR_QB	PAR_PNO	QB_TYPE	TB_TYPE
1	1	0	DSNFWQB (02)	R	0		N	0	0	SELECT	W
1	2	1	T1	I	1	T1_IX_C2	Y	0	0	SELECT	T
2	1	0	T2	R	0		N	1	1	NCOSUB	T
2	2	1	T3	I	1	T3_X_C1	N	1	1	NCOSUB	T

Figure 13-11 EXPLAIN - Use of virtual table in noncorrelated subquery

In Figure 13-11, the row corresponding to QBNO=2 and PLAN-NO=1 has PARENT_PLANNO (abbreviated as PAR_PNO) = 1 and PARENT_QBNO (abbreviated as PAR_QB) = 1. This means that the row corresponding to QBNO=1 and PLAN-NO=1 is the parent row. The sequence of execution flows from parent to child, then back to parent after the child rows are exhausted. In the example above, this means that the sequence of execution is (QBNO, PLANNO): (1,1), (2,1), (2,2), (1,2).

13.8.4 Other considerations

In this section we discuss additional considerations.

Predicates generated through transitive closure

When the set of predicates that belongs to a query logically imply other predicates, DB2 can generate additional predicates to provide more information for access path selection.

Example of transitive closure for an inner join:

```
SELECT * FROM T1, T2
   WHERE T1.C1=T2.C1
   AND T1.C1>10;
```

DB2 generates an additional predicate to produce this query, which is more efficient:

```
SELECT * FROM T1, T2
   WHERE T1.C1=T2.C1 AND
   T1.C1>10 AND
   T2.C1>10;
```

The correlated form of the subquery now involves correlation references to multiple tables. However, these references are redundant. The DB2 optimizer recognizes this redundancy and removes the predicate that is not needed to support the chosen access path.

Example of transitive closure for an outer join:

```
SELECT * FROM
   (SELECT T1.C1 FROM T1 WHERE T1.C1>10) X
   LEFT JOIN
   (SELECT T2.C1 FROM T2) Y
   ON X.C1 = Y.C1;
```

The before join predicate, T1.C1>10, meets the conditions for transitive closure, so DB2 generates a query that has the same result as this more-efficient query:

```
SELECT * FROM
   (SELECT T1.C1 FROM T1 WHERE T1.C1>10) X
   LEFT JOIN
   (SELECT T2.C1 FROM T2 WHERE T2.C1>10) Y
   ON X.C1 = Y.C1;
```

UNION and UNION ALL

The UNION and UNION ALL operators are allowed in subqueries, table expressions, and views, and do not limit the DB2 optimizer's ability to correlate or de-correlate these expressions.

INSERT, UPDATE, DELETE

INSERT, UPDATE, and DELETE statements that contain the types of subqueries discussed previously are handled the same as SELECT statements that contain these subqueries.

Optimization hints

Optimization hints are supported. Since the EXPLAIN output is enhanced to show the virtual tables and the position in which the virtual table is accessed, that information can be fed into the DB2 Optimizer as a hint using the existing Optimization hints support. For example, this means that users have the ability to request that a non-correlated subquery be processed in its correlated form, or that a correlated subquery be processed in its de-correlated form. This allows for greater control over how a query is processed without requiring a change to the way in which the query is coded.

Archived

Limits

The maximum number of tables that can be specified in a single SQL statement is 225. However, the generation of virtual tables can cause the total number of tables to exceed 225. This is okay as long as the total number of tables after generation of virtual tables does not exceed 512. In the unlikely case that the total number of tables exceeds 512, SQLCODE –101 is returned.

13.8.5 Subquery tuning

DB2 automatically performs some subquery tuning by subquery to join transformation and through subquery correlation and de-correlation. However, you should be aware of the differences among the subqueries like those in the following examples. You might need to code a query in one of the ways below for performance reasons that stem from restrictions to DB2 in transforming a given query, or restrictions to the DB2 optimizer accurately estimating the cost of the various transformation choices.

Each of the following three queries retrieves the same rows. All three retrieve data about all designers in departments that are responsible for projects that are part of major project MA2100. These three queries show that you can retrieve a desired result in several ways.

Query A: a join of two tables:

```
SELECT DISTINCT DSN8910.EMP.*
FROM DSN8910.EMP, DSN8910.PROJ
WHERE JOB = 'DESIGNER'
 AND WORKDEPT = 'DEPTNO'
 AND MAJPROJ = 'MA2100';
```

Query B: a correlated subquery:

```
SELECT * FROM DSN8910.EMP X
WHERE JOB = 'DESIGNER' AND EXISTS
  (SELECT 1 FROM DSN8910.PROJ
   WHERE DEPTNO = X.WORKDEPT
   AND MAJPROJ = 'MA2100');
```

Query C: a noncorrelated subquery:

```
SELECT * FROM DSN8910.EMP
WHERE JOB = 'DESIGNER'
 AND WORKDEPT IN (SELECT DEPTNO FROM DSN8910.PROJ
                  WHERE MAJPROJ = 'MA2100');
```

If you need columns from both tables EMP and PROJ in the output, you must use the join. Query A might be the one that performs best, and as a general practice you should code a subquery as a join whenever possible. However, in this example, PROJ might contain duplicate values of DEPTNO in the subquery, so that an equivalent join cannot be written. In that case, whether the correlated or non-correlated form is most efficient depends upon where the application of each predicate in the subquery provides the most benefit.

When looking at a problematic subquery, check whether the query can be rewritten into another format, especially as a join, or if you can create an index to improve the performance of the subquery. Consider the sequence of evaluation for the different subquery predicates and for all other predicates in the query. If one subquery predicate is costly, look for another predicate that could be evaluated first to reject more rows before the evaluation of problem subquery predicate.

13.9 Generalized sparse index and in-memory data cache

A sparse index is an index pointing to a range of n contiguous rows rather than a specific RID.

13.9.1 Pre-V9 scenario

DB2 V4 introduced sparse index for non-correlated subquery materialized work files. DB2 V7 introduced sparse index on the join key for the materialized work files of star schema queries. DB2 V8 replaced sparse index with in-memory data caching for star schema queries, with runtime fallback to sparse index when enough memory is not available.

The characteristics of star schema sparse indexes are:

- In-memory index occupies up to 240 KB
- Probed through an equal-join predicate
- Binary search for the target portion of the table
- Sequential search within the target portion if it is sparse

The characteristics of in-memory data caching (also known as in-memory work file) are:

- Memory pool size controlled by DSNZPARM SJMXPOOL
- Entire work file is in-memory (and is thus *not* sparse)
- Searched using binary search (as per sparse index)

Figure 13-12 shows how sparse index works.

The work file is sorted in the join column sequence (T2.C). While sorting, the (sparse) index is built containing the key (join) columns and the RID. The sparse index structure is flat rather than a B-tree structure used by normal indexes on data tables. The index is probed through an equi-join predicate (T1.C = T2.C). A binary search of the index is utilized to find the target row (or segment of rows). In case the index is sparse (not all entries in the work file have an index entry), a sequential search of the work file is subsequently initiated within the target segment to find the corresponding join row. Note that the query tries to use data caching (in-memory work files) first, and that the use of the sparse index is only a fallback plan.

Sparse index may be a subset of work file. For example:

- The work file may have 10,000 entries.
- The sparse index may have enough space (240 KB) for 1,000 entries.
- Thus, the sparse index is *binary searched* to find the target location of the search key.
- At most 10 work file entries are scanned.

Archived

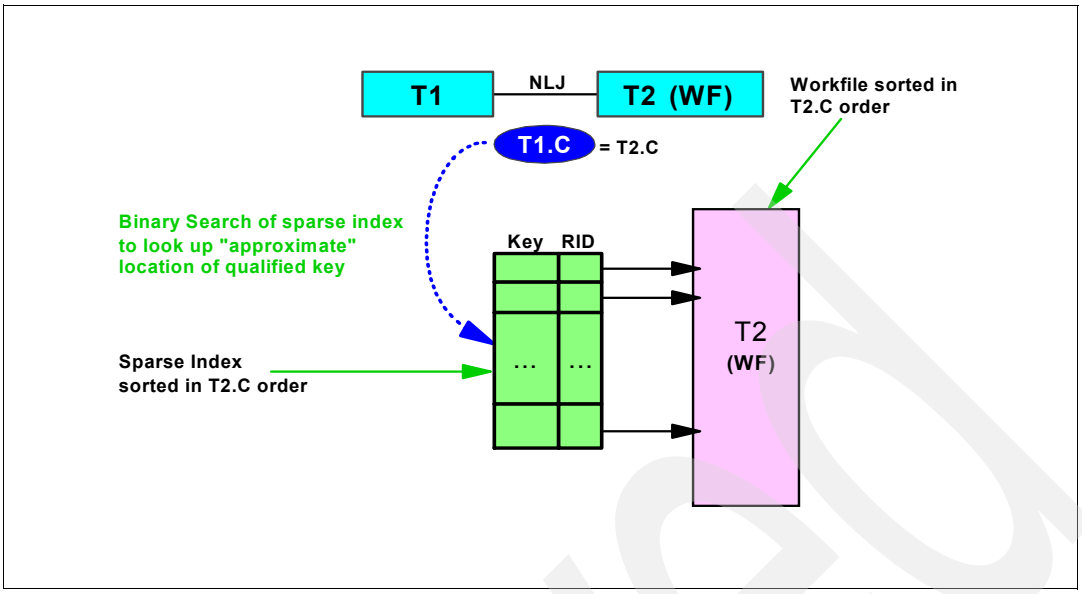


Figure 13-12 How does sparse index work?

13.9.2 DB2 V9 in memory enhancement

In DB2 V9 in-memory data caching is extended to joins other than star join.

In DB2 V8 in-memory work files are stored in a new dedicated storage pool that is called a star join pool. The DB2 DSNZPARM SJMXPOOL specifies its maximum size, which defaults to 20 MB (maximum 1 GB). It resides above the 2 GB bar and is only in effect when star join processing is enabled through DSNZPARM STARJOIN. When a query that exploits star join processing finishes, the allocated blocks in the star join pool to process the query are freed.

DB2 V9 uses a local pool above the bar instead of a global pool. Hence, data caching storage management is associated with each thread, which can reduce the potential storage contention.

Benefits of in-memory data caching

In theory, all tables that lack an appropriate index or enough statistics could benefit from sparse index/in-memory data caching:

- Temporary tables
- Table expressions
- Materialized view

This new access method and join context compete with other options, and then the most efficient access path is chosen. In-memory data caching access costing is not done in Access Path Selection since data caching is a runtime decision. If in-memory data caching could be used, it is a bonus (less costly than sparse index access).

DB2 V9 supports sparse index with multi-column keys. Sparse index/in-memory data caching search may be more efficient when there is more than one join predicate between two tables, because of the support for multi-column keys.

Archive

13.9.3 New DSNZPARM parameter MXDTCACH

A new installation parameter MXDTCACH is introduced to specify the maximum memory allocated for data caching. This corresponds to the field MAX DATA CACHING in the installation panel INSTALL DB2 - PERFORMANCE AND OPTIMIZATION (DSNTIP8).

- The unit of a value specified for MXDTCACH is in MB (1 MB is 1048576 bytes). For example, if a value 20 is specified, the total amount of up to 20 MB memory can be allocated for data caching per thread.
- The default value is 20. That is, 20 MB is the default maximum extent for data caching.
- When the value of 0 (zero) is specified, the data caching method is not used during query execution. Only sparse index (key+rid) can be applied.
- The possible range of the values is between 0 and 512.
- When data records are cached, the required memory amount is:

```
(number of rows)*
((maximum length of the keys) +
 (total maximum length of all the relevant columns))
```

  If the required memory exceeds the limit that MXDTCACH specifies or the memory allocation fails, query continues without using data caching. Sparse index is used instead.
- DB2 allocates memory from above the 2 GB bar pool up to the specified size for data caching.
- When thread terminates, the allocated memory for data caching is freed from the local pool above the 2 GB bar.

Attention: The existing installation parameter SJMXPOOL is replaced by MXDTCACH. Accordingly, the data caching space for star join is moved to the local pool above the 2 GB bar.

13.10 Dynamic index ANDing for star join queries

Currently, DB2 z/OS uses the following methodology to support Star Join:

- Cartesian Join on the dimension tables prior to join with the Fact Table
- Multi-column index access on the Fact Table with index feedback

The problems with the current method are:

- The Cartesian Join could become the attribute of performance degradations since there is no filtering in each dimension table.
- The user needs to create a multi-column index on the Fact table with the index columns to support joining from different dimension tables. It is difficult to create a suitable multi-column index without knowing the join sequence of the dimension tables.

This enhancement introduces a different join approach within a star join group, called Pair-Wise Join. It requires a one-column index on the Fact table to support each dimension table join column. The proposed join method relies heavily on the resource of Rid Pool.

13.10.1 Pair-Wise Join

Each dimension table is joined with a Fact table separately and returns a set of Fact table RIDs in rid pool. Figure 13-13 shows how the filtering is done in parallel.

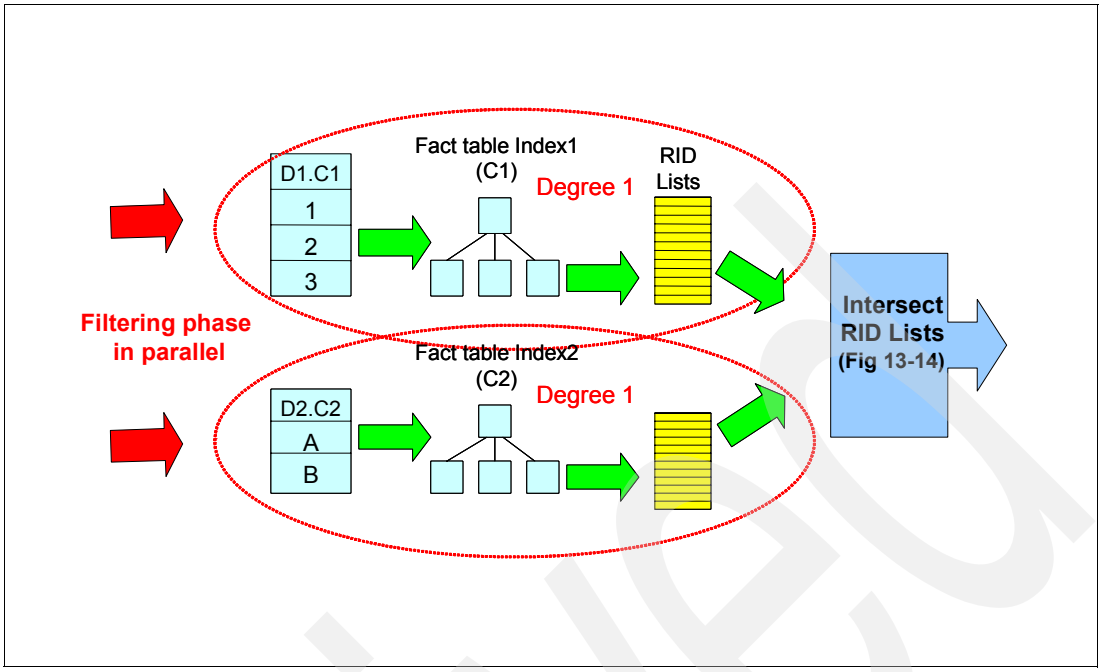


Figure 13-13 Star Join Parallel Filtering of RID Lists

Figure 13-13 shows the two dimensions D1.C1 (column 1) and D2.C2 (column 2). We can access these in parallel independently.

We then apply filtering to Dimension 1 and then use Dimension 1 to access the Fact table. We then do the same thing with Dimension 2.

Multiple processes are then going on at same time building the RID lists.

All the sets of Fact table RIDs that are returned from the join of each dimension table with the Fact table are ANDed together to produce a final RID List. The final Fact table RID list is used to retrieve data from the Fact table and join back to the dimension table as necessary for obtaining data from dimension tables.

Archive

Figure 13-14 shows the ANDing of RID lists.

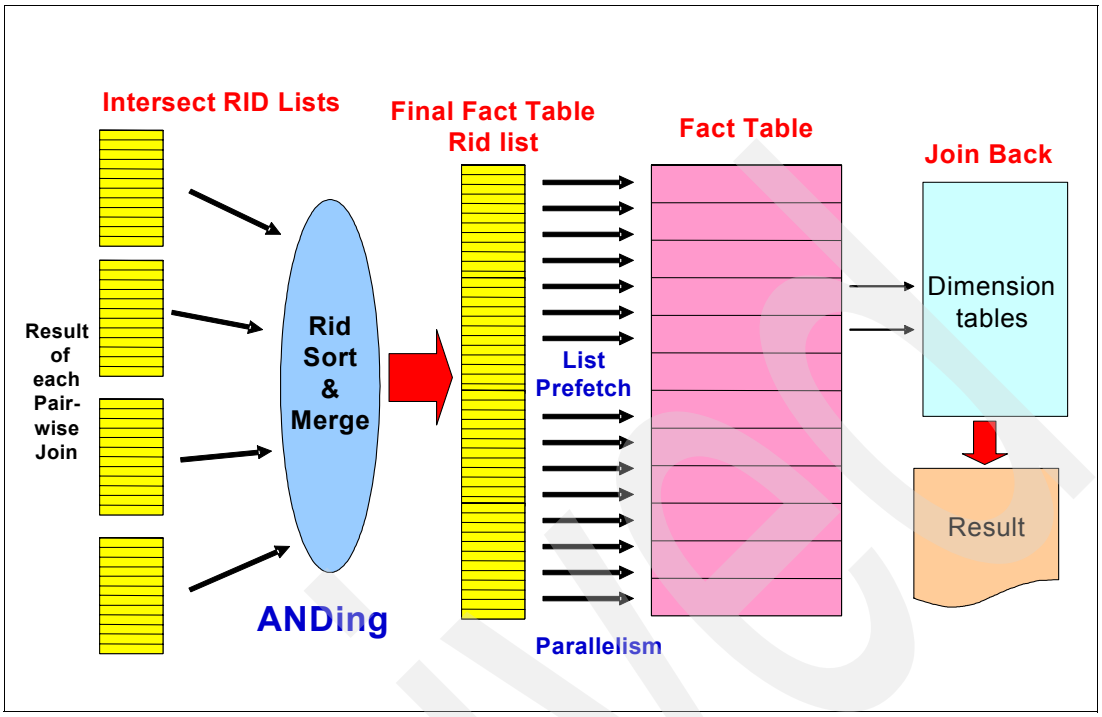


Figure 13-14 Star Join - Intersect RID List, Access Fact Table, and Join Back

13.10.2 Pair-Wise example

For example, if the query is as shown below:

```
SELECT PRODNAME, SUM(SALES), ...
FROM F, PROD P, CUST C, TIME T, STORES S
 WHERE F.PID = P.ID
   AND F.TID = T.ID
   AND F.SID = S.ID
   AND T.MONTH IN ('JAN', 'FEB') ~~ 17%
   AND S.LOCATION IN ('SAN JOSE', 'DALLAS') ~~ 2%
   AND P.TYPE IN ('FOOD', 'SODA') ~~ 6%
GROUP BY ...
```

In the Pair-Wise Join, each dimension table is joined with the FACT table separately using RID access through an index on Fact Table (in our example (T, F) and (S, F) and (P,F)), then the result fact table RID lists are ANDed together to form the final RID set of RID List.

Assuming that the join reduction between each dimension table and Fact table pair (T, F), (S, F), and (P, F) is about 17%, 2%, and 6%, respectively, and also assuming that the size of Fact table is about 500 million records after applying the join reduction, the estimated number of records is approximately around 85 million, 10 million, and 30 million. Figure 13-15 illustrates the relationship.

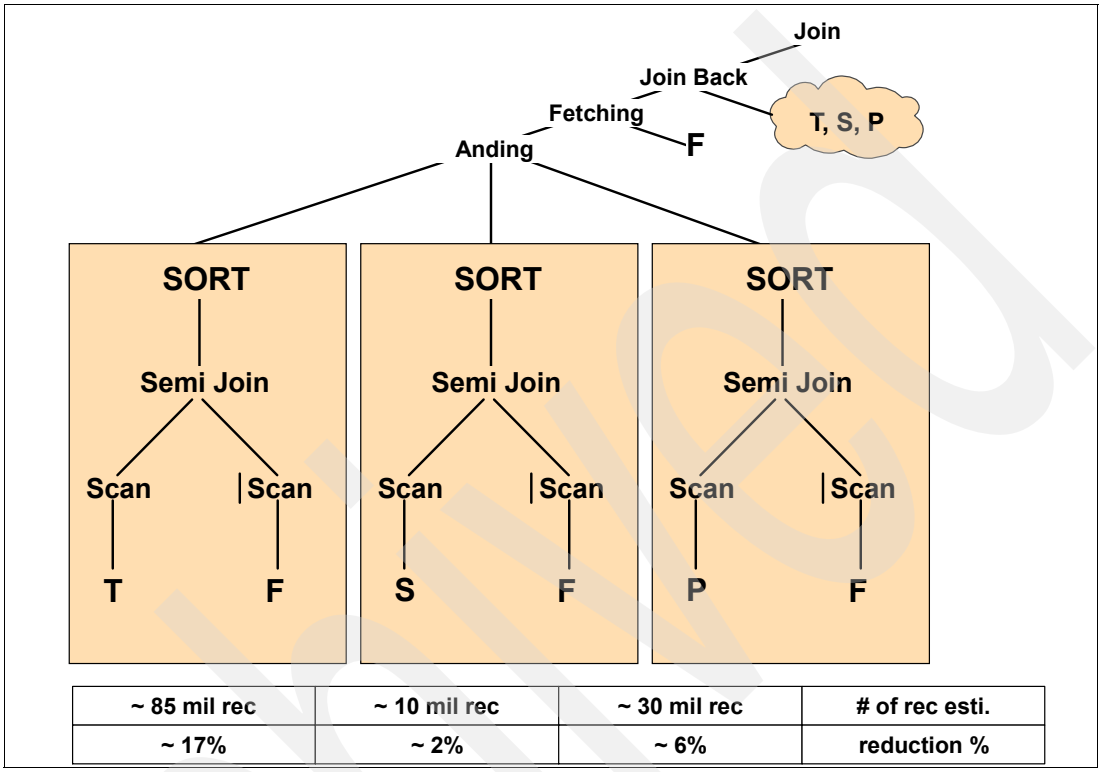


Figure 13-15 Star Join with Dynamic Index ANDing Scheme

After ANDing all the Fact table candidate result Rids returned from each join pair, the final set of least common Rids does not exceed 10 million records (2% of filtering factor), and the least common Rids are used to fetch data from the Fact table. If there is any column in the selection list from the dimension table, then a Join Back is performed to retrieve data from dimension tables.

To summarize, the pair-wise join is performed as follows:

1. Join each dimension table with the Fact Table through the index independently.
2. Perform Rid Sort and Rid Merge (ANDing) to form the final fact table RID list.
3. Fetch data from the FACT table using the final RID list.
4. Join back the dimension table only if it has to select out columns from dimension tables.

If any of the pair-wise join result Fact table RID List is unable to fit in the rid pool due to the rid pool resource shortage, either due to physical constraint (no more physical storage available to store the RIDs) or logical constraint (RID map is full), this particular pair of join result RIDs is written into a work file. The final Fact table rid list is obtained by ANDing the RID Lists from the rid pool and the work file together.

13.10.3 Runtime optimization

Occasionally due to non-availability of statistics, the plan bound at bind time may not reflect optimal access plan. This enhancement implements runtime optimization that allows a long-running star join query a second chance to re-evaluate the join plan.

13.11 Large Object (LOB/XML) flow optimization

Database applications increasingly store character data in large object (LOB) columns. LOB columns provide additional capacity for the data to grow, as compared to varchar or long varchar columns. LOB columns may be used to store small character strings, serialized Java objects, and XML documents.

Figure 13-16 shows reasons for increasing LOB usage.

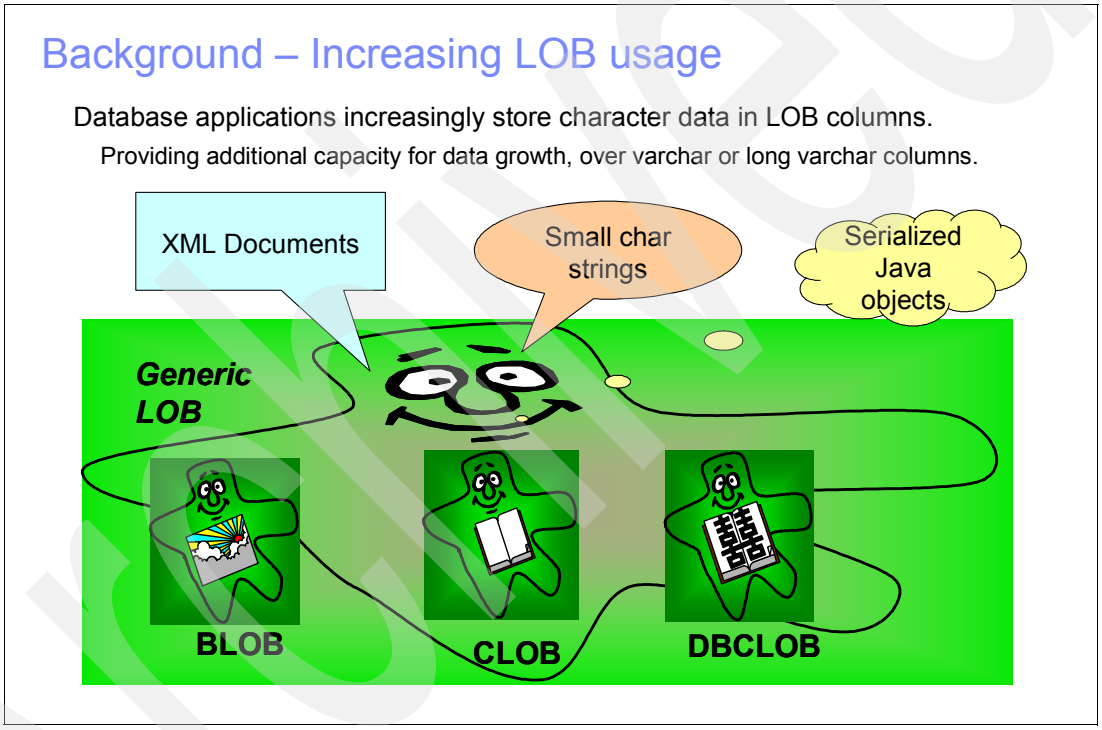


Figure 13-16 Increasing LOB usage

The processing of LOBs in a distributed environment with Java Universal Driver on the client side has been optimized for the retrieval of larger amounts of data. This dynamic data format is only available for the JCC T4 driver (Type 4 Connectivity). The Call Level Interface (CLI) of DB2 for Linux, UNIX, and Windows also has this client-side optimization. Many applications effectively use locators to retrieve LOB data regardless of the size of the data being retrieved. This mechanism incurs a separate network flow to get the length of the data to be returned so that the requester can determine the proper offset and length for SUBSTR operations on the data to avoid any unnecessary blank padding of the value. For small LOB data, returning the LOB value directly instead of using a locator would be more efficient (that is, the overhead of the underlying LOB mechanisms can tend to overshadow the resources required to achieve the data retrieval).

For these reasons, LOB (and XML) data retrieval in DB2 9 has been enhanced so that it is more effective for small and medium size objects, and still efficient in its use of locators to

retrieve large amounts of data. For small LOBs, the performance should approximate that of retrieving a varchar column of comparable size. This functionality is known as progressive streaming. Within the overall dynamic data format of progressive streaming, progressive reference is the mechanism that supports the category of large LOB data retrieval.

With the JCC Type 4 driver, a LOB value is associated to one of three categories depending on its size:

- Small LOBs - DRDA (server) default is 32 KB. The driver can override it by setting smaller values.
- Medium LOBs - Greater than 32 KB and up to 1 MB size.
- Large LOBs - Greater than 1 MB and up to 2 GB size.

The large threshold is set as DRDA parameter MEDDTASZ whose default is 1 MB. This threshold should be set to the maximum storage size the application region can handle, but not less than 32 KB. There is no means to override the small threshold (currently set for performance reasons at 12 KB) set by the Type 4 driver.

Based on the size, the decision is made about how to transfer the data. The structure of the data query block varies according to LOB size. When small-sized LOBs are used, they are treated exactly as VARCHAR type and transferred as part of the row, thus gaining performance close to that of retrieving a varchar. When medium-sized LOBs are used, the non-LOB data is placed in the query data block and the LOBs are placed in overflow blocks. These blocks are called externalized data blocks. This way, all LOBs are retrieved at once and cached on a client for subsequent processing. For large LOBs, it was found that locators are still the most efficient flow method. Thus, the locators are transmitted in a data query block with the rest of the data, avoiding a need to materialize the entire LOB at once. This explanation is depicted in Figure 13-17.

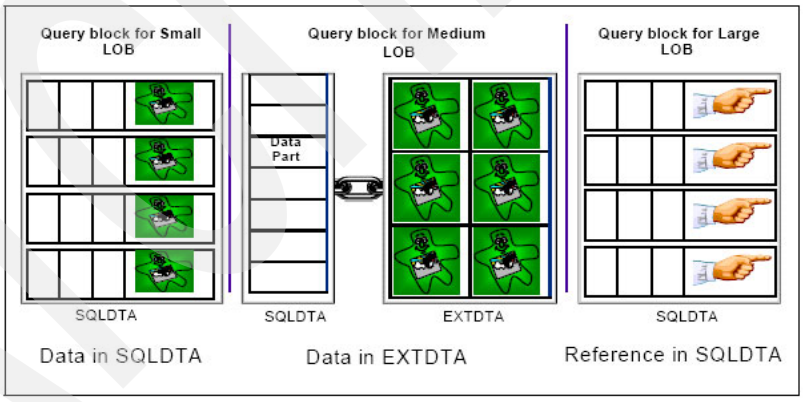


Figure 13-17 Progressive Reference Return of LOB Data

Note: Data streaming is architected in DRDA but is not implemented in a peer-to-peer DB2 connection where it requires the Java Universal Driver on the client side.

Inserting data

LOBs are streamed to DB2 by the client and completely materialized on the server side. The object is *assembled* in the DDF address space and then moved on to the DBM1 address space for further processing.

This mechanism provides a very adaptable flow mechanism to seamlessly change the logic used to retrieve data for real LOBs, where the data length is indeed large, and data stored within LOB columns that does not really satisfy the description of large. This is particularly useful for data stores where the length of data is subject to wide variations, with DB2 adapting on the fly to minimize the effort to serve the workload it receives.

13.11.1 IBM DB2 Driver for JDBC and SQLJ

Java applications explicitly request what to return in one of two ways:

- Fully materialized data (fullyMaterializedLobData = true)
  - All data is returned at the same time.
  - Best for *not-so-huge* large objects.
- Locators (fullyMaterializedLobData = false)
  - Only a locator is returned for the request.
  - Separate flow to ask for (partial) data.
  - Best for *huge* large objects.

The applications may not have prior knowledge of whether large objects are *not-so-huge* or *huge* until full data is retrieved. The locators are implemented as random-access based (which may not optimize in data streaming scenario) and transaction based (which may hog resources if transaction lasts too long). Progressive streaming is used to address these issues.

Figure 13-18, Figure 13-19 on page 526, and Figure 13-20 on page 526 show the details of how the progressive streaming works.

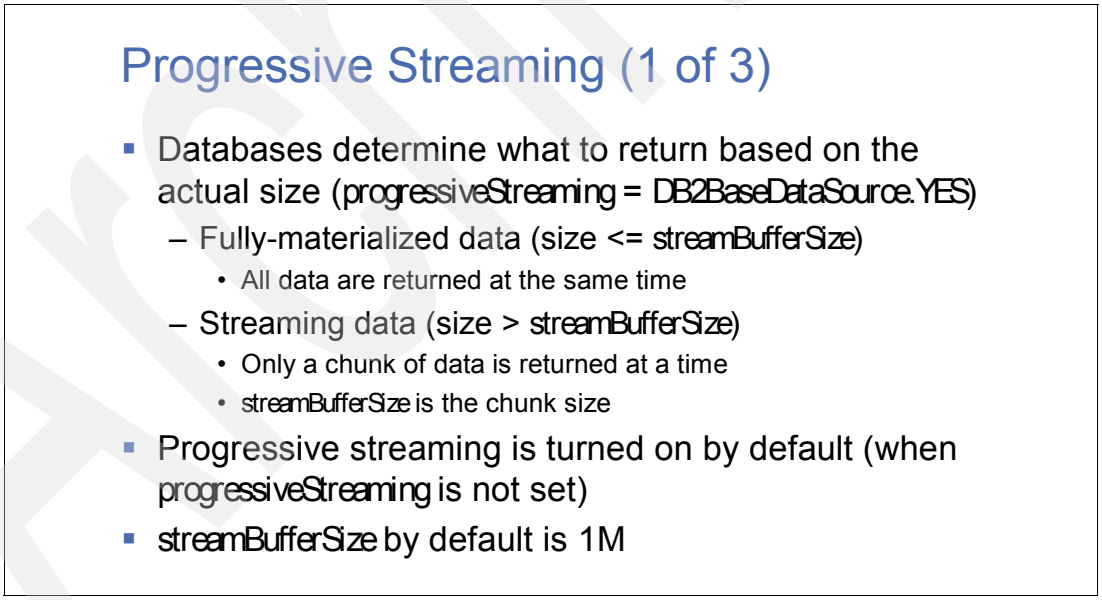


Figure 13-18 Progressive Streaming (1 of 3)

The IBM DB2 Driver for JDBC and SQLJ includes all of the LOB support in the JDBC 3.0 and earlier specifications. This driver also includes support for LOBs in additional methods and for additional data types.

If the database server supports progressive streaming, the IBM DB2 Driver for JDBC and SQLJ can use progressive streaming to retrieve data in LOB or XML columns. With progressive streaming, the database server dynamically determines the most efficient mode

in which to return LOB or XML data, based on the size of the LOBs or XML objects. Progressive streaming behavior is the default for connections to database servers that support progressive streaming.

Progressive Streaming (2 of 3)

progressiveStreaming \ fullyMaterializeLobData		true	false
YES or NOT_SET	data size <= streamBufferSize	Fully-materialized data	Fully-materialized data
	data size > streamBufferSize	Streaming data	Streaming data
NO		Fully-materialized data	Locators

(YES, NO, and NOT_SET are constants defined on DB2BaseDataSource)

Figure 13-19 Progressive Streaming (2 of 3)

If the IBM DB2 Driver for JDBC and SQLJ progressiveStreaming property is DB2BaseDataSource.YES or DB2BaseDataSource.NOT_SET (the default), and the connection is to a database server that supports progressive streaming, such as DB2 Version 9.1 for z/OS or later, your application uses progressive streaming behavior. When progressive streaming is enabled, you can control when the JDBC driver materializes LOBs with the streamBufferSize property. If a LOB or XML object data size is less than or equal to the streamBufferSize value, the object is materialized. See Figure 13-20.

Progressive Streaming (3 of 3)

- Streaming data is
  - Sequential-access based (which optimizes in data streaming scenario)
  - Cursor based (which does not hog resources even if transaction lasts too long)
- The more diversified the data is, the better performance improvement

Figure 13-20 Progressive Streaming (3 of 3)

Archived

Important: With progressive streaming, when you retrieve a LOB or XML value from a ResultSet into an application variable, you can manipulate the contents of that application variable until you move the cursor or close the cursor on the ResultSet. After that, the contents of the application variable are no longer available to you. If you perform any actions on the LOB in the application variable, you receive an SQLException.

13.11.2 DRDA LOB flow optimization performance

Some indication of the performance advantages can be seen from test cases where the LOB length is varied around the streamBufferSize threshold, which effectively compares the old non-progressive mechanism with the new progressive streaming, using varying sized LOB workloads.

Figure 13-21 shows the effect of varying LOB data sizes with differing flow mechanisms.

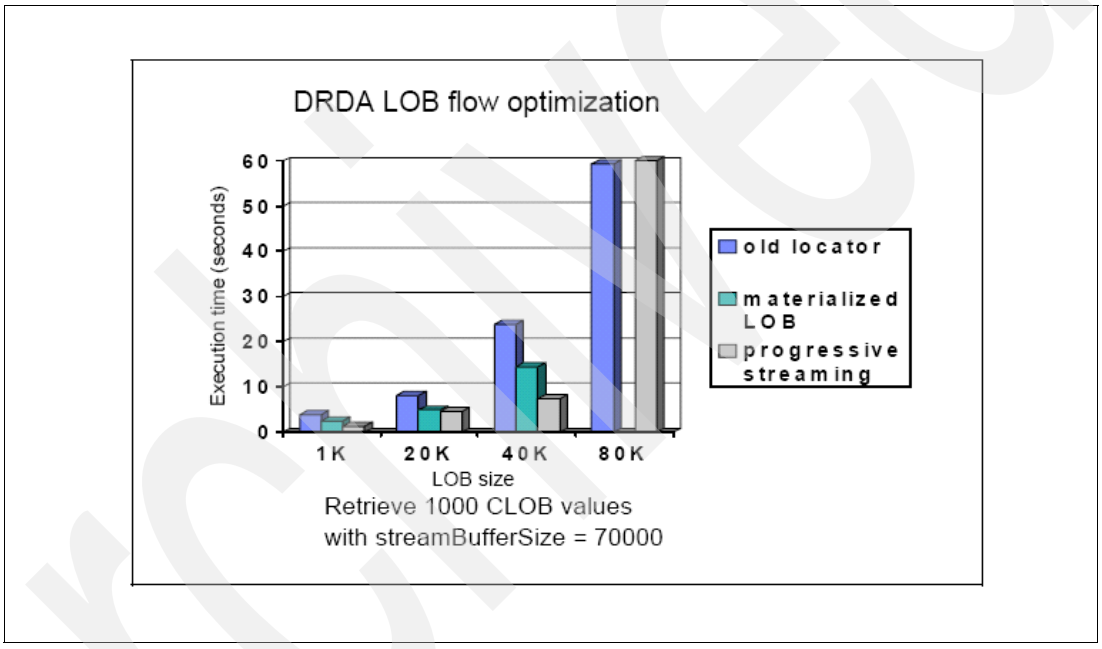


Figure 13-21 Performance of LOB Progressive Streaming

For small LOB sizes, the longest run time comes from using LOB locators. It is somewhat more efficient in terms of elapsed time to materialize the LOB instead of using a locator, because a degree of overhead is removed from DB2, and a reduction occurs in the number of data flows. Using progressive streaming further improves the performance by eliminating overhead involved with invoking LOB specific flow mechanisms. However, when the LOB size is increased to 80 KB, which is above our specified value of 70 KB, the same performance is seen, because the mechanisms used are the same.

Note: The highest performance improvement is seen when selecting a large number of rows in one cursor. For singleton selects, only minimal performance improvement is seen.

Archived

13.12 LOBs performance improvements

The large objects (LOBs) were introduced in DB2 V6 as new database data type that stores large bit and byte string data that has a size limit of 2 GB. We have the ability to store data objects up to 2 GB in size, including audio, documents, and pictures. We can store data that will not fit into 32 KB maximum page size in BLOBs, CLOBs, and DBCLOBs.

Figure 13-22 shows how LOB values are stored in DB2.

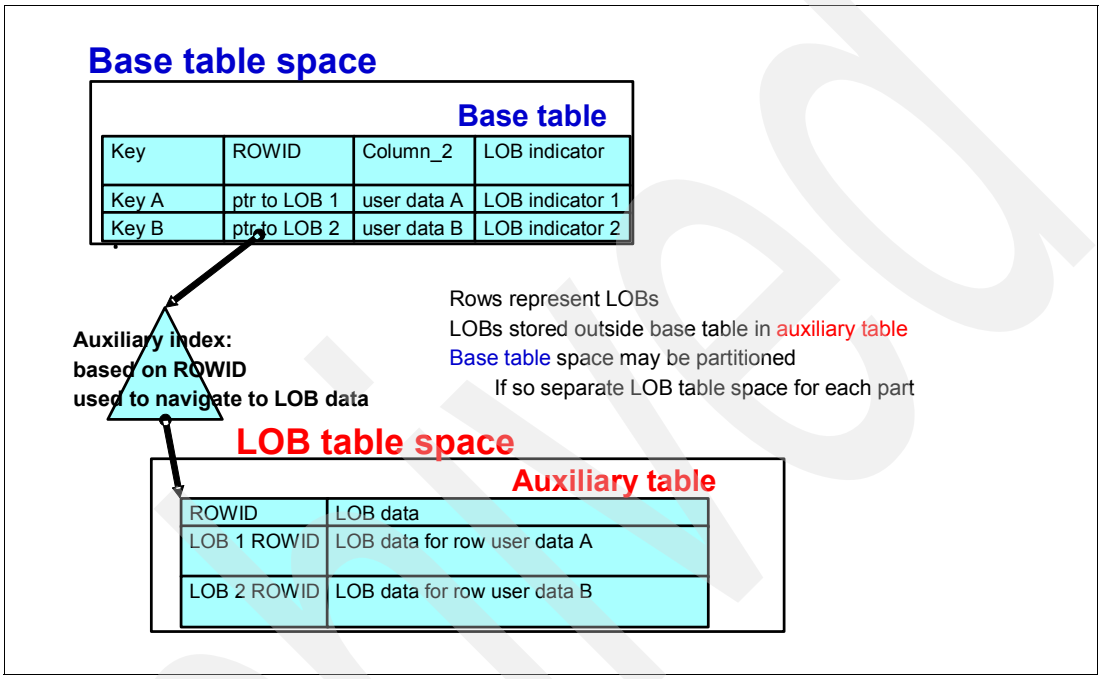


Figure 13-22 LOB overview

Since DB2 V6, business applications can store, manipulate, and retrieve multimedia objects from database systems. As the size of the database grows, timely and frequent access to large objects also becomes an integral part of the complex business applications.

From DB2 Version 6 until DB2 V8, DB2 uses two types of locks for ensuring a LOB's integrity: the S-LOB and the X-LOB locks. They are very similar to the common S- and X-Locks. There are no U-LOB locks, because of a different update mechanism taking place for LOBs. Updating a LOB for DB2 means deallocation of used data pages, and allocating and inserting new data pages, which contain the new LOB value.

Assuming LOCKSIZE LOB, selecting a LOB acquires an S-LOB lock on the accessed LOB, even if you use ISOLATION (UR) in a package or WITH UR in your SQL statement. This is because of the new table space format where physical correctness has to be guaranteed while retrieving the LOB value, which might be spanned over many pages. Acquiring S-LOB locks prevents the application from retrieving only partial LOB data.

Deleting a LOB also requests an S-LOB lock on a specific LOB. But how does this work with other transactions selecting the same LOB? A transaction can delete a LOB that another transaction is reading at the same time, but the space is not reused until all readers of the LOB have committed their work.

S-LOB locks do not prevent a LOB from being deleted.

Inserting a LOB acquires an X-LOB lock on the new LOB, and the lock is released at COMMIT. If a LOB is locked by an X-LOB lock, no readers can access the LOB before it is committed.

Because every LOB lock is like a row lock, the number of acquired locks can increase dramatically, especially when mass-updates occur on LOBs, or many LOBs are inserted using subselect.

To summarize, in DB2 V8 S-LOB lock is acquired and freed in each FETCH (both non-UR and UR readers) and DELETE call. That is, S-LOB lock is not held till commit. On the other hand, X-LOB lock acquired in INSERT or space allocation is held till commit.

13.12.1 LOB locking enhancements

Figure 13-23 shows how LOB locks are handled in DB2 9.

LOB Operations and LOB Locks

Operation	Action
INSERT and UPDATE	Continues to hold lock on the base row or page as in V8. However, for Inserting or updating the LOB value, X LOB lock taken is held only until the duration of the insert or update (that is, not until commit)
DELETE	Continues to hold lock on the base row or page as in V8. However, for deleting the LOB value no LOB lock (S or X) is taken.
SELECT	Continues to hold any locks on the base row or page as in V8. However, no S LOB lock is taken while building the data descriptor on the LOB data.
UR readers	Should request for S LOB lock in order to serialize with concurrent Insert or update operation.

Figure 13-23 LOB operations and LOB Locks

Beginning with DB2 9, the locking technique for LOBs has changed.

Locks on LOBs are taken when they are needed for an INSERT or UPDATE operations and released immediately at the completion of the operation. LOB locks are not held for SELECT and DELETE operations. A LOB lock that is acquired in the case of an application that uses uncommitted read only tests the LOB for completeness and releases immediately after it is acquired.

For normal SELECT operations, DB2 avoids acquiring any LOB lock since the holding of the current lock on a base row or a data page is good enough to ensure the access of a correct copy of LOB data. But for a SELECT with ISO(UR) readers, if LOB columns are to be selected, the UR reader must acquire an S LOB lock to serialize with concurrent updater/insert operations.

Instead of using LOB locks, DB2 now uses the READ LSN value to determine whether LOB storage can be reused.

To summarize, in DB2 9 S-LOB lock is acquired and freed in each FETCH call as in DB2 V8. The change from DB2 V8 is that for non-UR readers and DELETE calls, no S-LOB lock is acquired. For INSERT, X-LOB lock acquired is released after allocation completes and is not held till commit. There is no change for UPDATE.

Availability improves because locks are not held until commit.

Attention: In data sharing, there is now a need to ensure that changed pages are forced out before X-LOB lock is released. Therefore, use GBPCACHE(CHANGED) for better performance.

13.12.2 Database design and implementation

IBM recommends that LOB users perform the following setups to increase the value of this enhancement in their DB2 systems:

- Have a dedicated buffer pool for LOB objects to improve the efficiency of the use of Read LSN value, especially in a data sharing environment.
- Set the GBP castout thresholds to a low value to reduce the need for having a large GBP for LOB objects.
- Consider use of the GBPCACHE CHANGED option instead of the current default option of GBPCACHE SYSTEM for the LOB table space. Due to the usage pattern of LOB, the use of GBPCACHE CHANGED can avoid excessive and synchronous write I/Os to DASD and GBP, particularly for relatively small LOBs. For large LOBs, the default GBPCACHE SYSTEM may be more appropriate.

13.12.3 Performance monitoring and tuning

The new locking scheme is only used in V9 with New-Function Mode. V8 coexistence mode uses the old locking scheme, and V9 Compatibility Mode (CM) also uses the old locking scheme. In data sharing, like for protocol 2 in V8, it needs a group shut down to become active.

13.13 WLM-assisted buffer pool management

In z/OS V1R8, DB2 buffer pool management by WLM is introduced. This function is at its early stages. You need to verify implementation and maintenance with your DB2 support.

The aim is to help data managers benefit from the performance-driven storage management of WLM. Buffer pools occupy large amounts of main storage. With this change, buffer pool management is integrated with WLM's management of storage resources in order to use storage resources more efficiently. This is useful in adjusting the size of a buffer pool. Buffer pool management is now based on the goal achievement of work according to the WLM policy. The buffer pool size is coordinated dynamically with the requirements of data requesters and the systems sysplex workload.

In support of DB2 buffer pool management in z/OS V1R8, a new delay type is being reported to WLM. A new mechanism is introduced to identify the user of buffer pools and to report the current state of a user.

The prerequisites for buffer pool management with WLM and SRM (systems resource manager) are that the buffer pool is registered with WLM and the existence of a Performance

Block for the Database Manager. The buffer pool registration informs the data collection exit and adjustment exits of size limits using the new services.

A new mechanism is introduced to associate a buffer pool with users. Buffer pool users are CICS or IMS transactions represented by a performance block (PB) and DDF enclaves represented by an enclave.

A new mechanism is introduced to track and report the user's current state. These new delays are reported to dedicated performance blocks of data requesters suffering performance delays while waiting for data that still has to be loaded into the buffer pool. WLM tracks when users have to wait for data in the buffer pool.

Figure 13-24 shows a typical sequence of a database manager reporting buffer pool delays that have been triggered by an application's request.

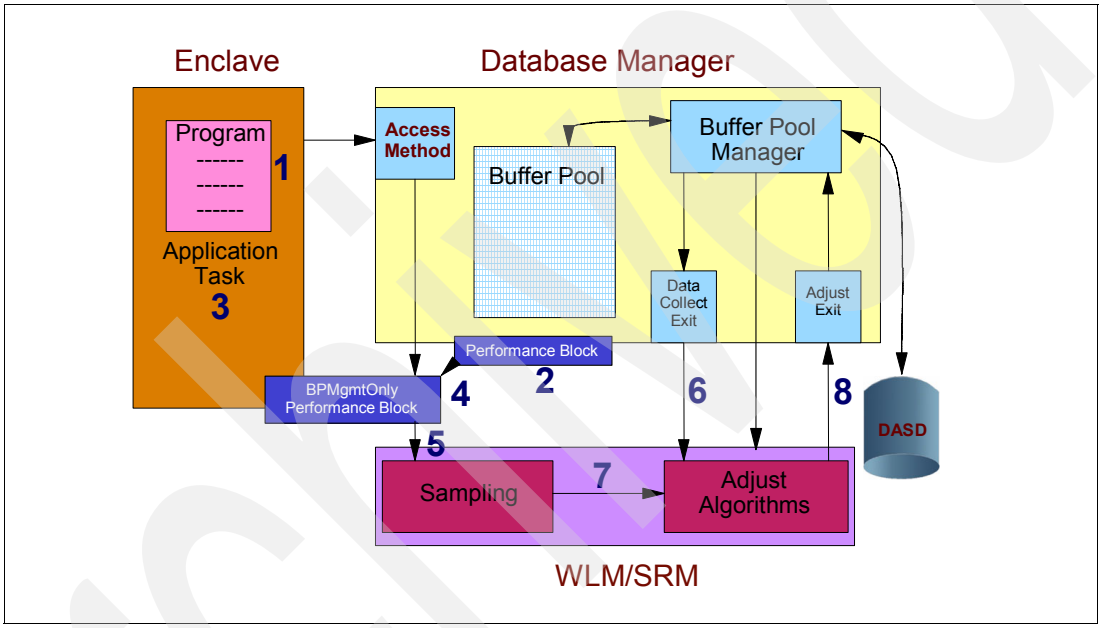


Figure 13-24 Buffer Pool Management by WLM

Steps 1 and 2

Prerequisites for buffer pool management with WLM are that the buffer pool is registered with WLM and the creation of a performance block for the database manager. The buffer pool vector table gathers all the buffer pool management blocks. The buffer pool management blocks are created at registration time. The buffer pool registration informs about data collection and adjustment exists as well as of size limits.

The application task joins the enclave.

The application starts to use the buffer pool. At this time, the buffer pool performance block is related to the enclave performance block. During this period, state changes are recorded in the enclave's performance block.

WLM collects performance block states.

Steps 3, 4, and 5

The database manager creates the performance block that will be exclusively used to record buffer pool delays. The performance block is associated with the requesting enclave and the buffer pool performance block is related to the dedicated enclave's performance block.

WLM collects buffer pool size and hit ratio.

WLM algorithms verify buffer pool size.

Steps 6 and 7

The database manager reports any buffer pool delays in the buffer pool performance block. Delays are caused when requested data is not available in the buffer pool that must be retrieved by the buffer pool manager through I/O operations. The delays are considered in the performance calculations of the enclave's classification. These will directly affect the enclave's service class period PI.

If necessary, WLM calls the adjustment exit to change the buffer pool size.

Step 8

After having satisfied the enclave's request, the dedicated buffer pool performance block is released from its association with the enclave. The buffer pool performance block can be reused for reporting states of another requesting enclave.

Once delay values are collected, WLM gathers performance data to recognize whether a buffer pool size change is necessary. WLM collects the performance block states. The collected states are reported to the adjustment algorithms. Periodically, WLM polls the current buffer pool size. If buffer pool delays are a problem, a calculation is made and the buffer pool manager is instructed to adjust the buffer pool size.

13.14 Performance scalability

Insert, Update, and Delete performance is, and has always been, one of the most challenging issues in any database management system. DB2 9 for z/OS adds dramatic performance/scalability improvement in this area.

Figure 13-25 shows the performance scalability enhancements.

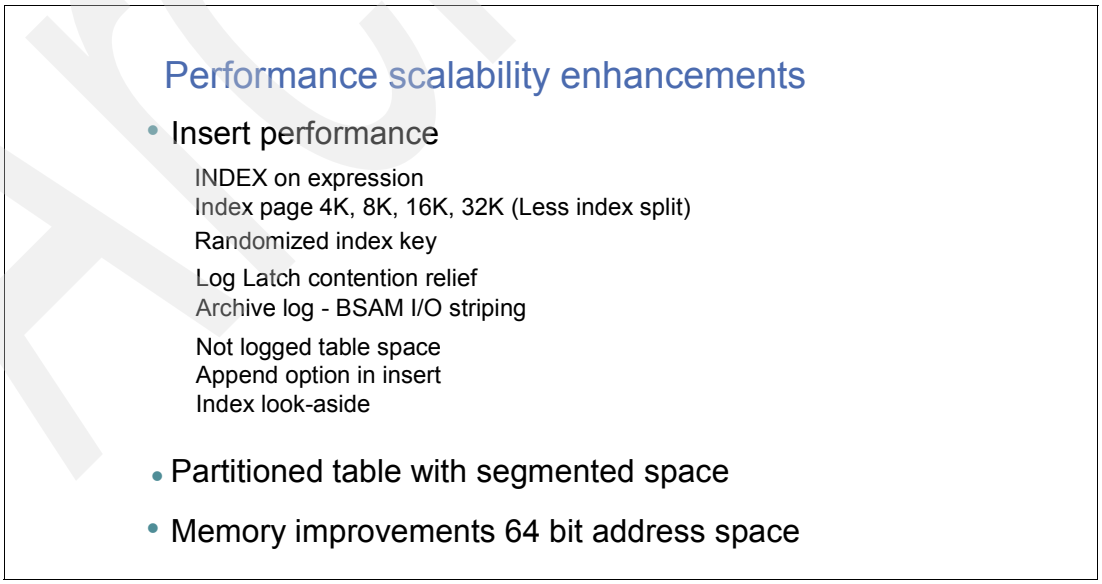


Figure 13-25 Performance scalability enhancements

Archived

Insert performance increases substantially, through a wide range of improvements. Logging performance is improved with latching improvements and striped archiving. The newer disk and channel changes (DS8000 Turbo, 4 GB per second channels, MIDAW) improve the data rates substantially. Indexes are improved, with larger page sizes to reduce the number of page splits and also a better page split. Where performance should be optimized for inserts, rather than for later retrieval, the APPEND option can be used. If the data needs to be randomized to avoid insert hot spots, the new randomized index key is useful.

The segmented space structure is more efficient in many situations, so adding that space structure for the large partitioned table spaces helps DB2 scale more efficiently.

Memory improvements continue the work from V8, with memory shared above the bar between the DDF and DBM1 address spaces. The shared memory can be used to avoid moving data from one address space to the other. More data structures from the EDMPOOL and dynamic statement cache are moved above the bar.

A randomized index key to avoid hot spots can be beneficial for data sharing because of index page P-lock contention. It is a trade-off between contention relief and additional getpage, read/write I/O, and lock request. This performs better for indexes resident in buffer pool. The use of the NOT LOGGED table space option where appropriate can help reduce log latch contention. The APPEND option in insert can be used to reduce a longer chain of spacemap page search as the table space keeps getting bigger.

The objective of the index look-aside technique is to minimize the number of getpage operations that are generated when an individual SQL statement or DB2 process is executed repeatedly and makes reference to the same or nearby pages. Index look-aside results in a significant reduction in the number of index and data page getpage requests when an index is accessed in a sequential, skip-sequential, or hot spot pattern. In DB2 V8, the index look-aside technique is used for clustering index only in insert, and not for delete. The enhancement is that index look-aside is possible for more indexes in both insert and delete.

Archive

13.15 Other performance improvements

Figure 13-26 shows other performance improvements.

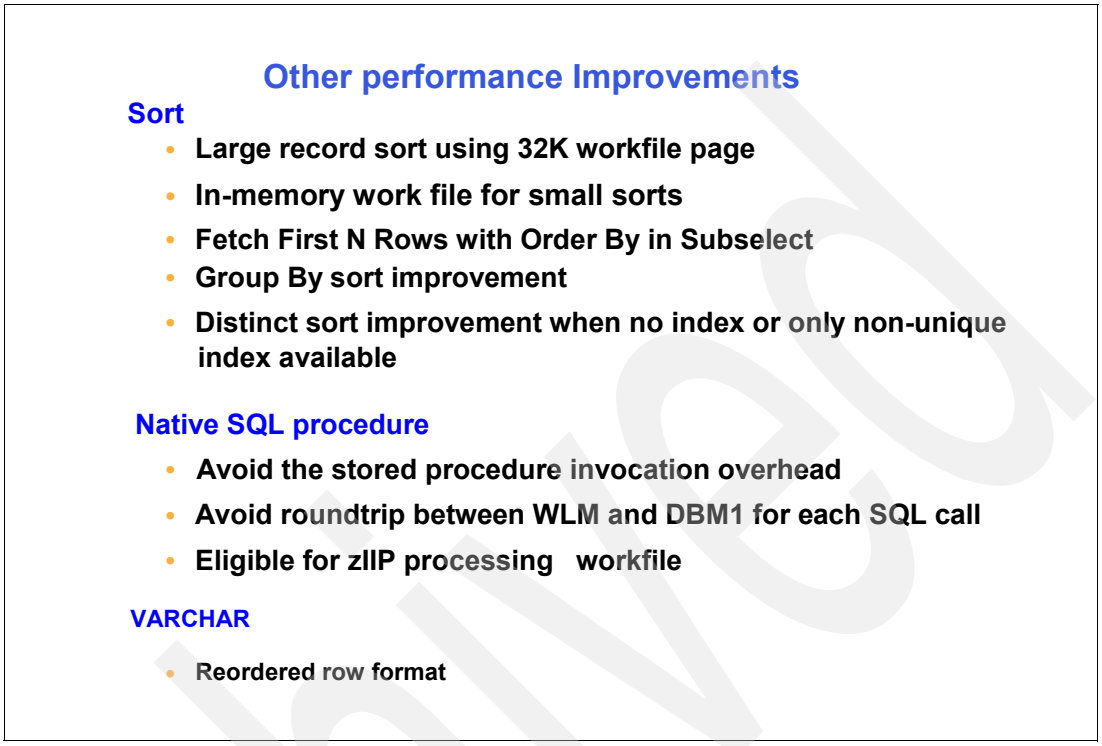


Figure 13-26 Other performance improvements

The various sort improvements listed result in reduction in CPU time. In-memory work files are beneficial for online transactions with relatively short-running SQL calls in which the number of rows sorted can be small. With the support for Fetch First N rows with ORDER BY (also in subselects), the tournament tree sort is avoided for small N. Group By sort improvement occurs when there is no column function and no index is available.

Performance improvement of using native SQL procedure can vary from 0 to 80% compared to external SQL procedure. For long-running SQL, no improvement may be noticed. DB2 V8 SQL procedures were not eligible to run on the zIIP, but changing to use the native SQL Procedure Language makes this work eligible for zIIP processing.

Remember the tuning recommendation for rows with many columns with any varchar present? In DB2 9 the rows are stored as per this recommendation. Prior to DB2 9, any column after the first varchar requires offset calculation. In DB2 9 all columns are directly accessible. Processing of varying length data can improve substantially if there is a large number of varying length columns. The improvement also applies for vargraphic columns. There is no difference if varchar or vargraphic columns are not present in the row.

Parallelism for DPSIs

In previous releases of DB2, if the access path for a query used a data-partitioned secondary index that provided data ordering, DB2 could not use parallelism when it executed the query. In DB2 V9, DB2 considers parallelism when it executes this type of query.

For example, suppose that table T1 has two columns, C1 and C2, and that a data-partitioned secondary index is defined on C1.

In DB2 Version 8, DB2 did not consider parallelism for the following query:

```
SELECT * FROM TABLE1 ORDER BY C1;
```

In DB2 Version 9, DB2 considers parallelism for this query.

Archived

Archived

A

DB2 Tools for z/OS

DB2 Tools for z/OS help reduce manual tasks, automatically generate utility jobs, capture and analyze performance data, make recommendations to optimize queries, and maintain high availability by sensing and responding to situations that could result in database failures and system outages. The DB2 tools are grouped into six functional categories to match your database and system requirements:

- Database administration tools to help you administer DB2
- Utilities management tools to help you optimize the use of your DB2 utilities
- Performance management tools to help you optimize the performance of your DB2 databases
- Recovery management tools to let you easily and efficiently back up and recover your DB2 databases
- Information integration tools to help you meet a diverse range of data integration requirements for business intelligence and business integration
- Application management tools to help you manage your DB2 applications

Together, the array of tools that IBM offers can help you meet performance standards and control IT costs. You can combine a selection of tools from any category to build a complete, flexible, and affordable solution that is tailored to meet your specific needs. A wide variety of tools that work with DB2 V9.1 for z/OS are available and ready to support all major new V9.1 functions.

For more information about DB2 tools see:

`http://ibm.com/software/data/db2imstools/`

In this appendix we mention the new and enhanced tools made available with DB2 9 for z/OS as described in the USA Software Announcement 207-040 on 3/6/2007.

© Copyright IBM Corp. 2007. All rights reserved.

Overview of tools Announcement 207-040

Three new products and one new version offer many innovative features to support DB2 9 for z/OS and assist database administrators in managing their DB2 environment, as well as assisting business analysts to leverage new business intelligence from their database data:

- DB2 Optimization Expert for z/OS, V1.1 (5655-S19) offers a comprehensive set of index and statistics advisors to improve system performance and lower total cost of ownership.
- DataQuant for z/OS, V1.1 (5697-N64) is a powerful analytical tool that delivers business intelligence capabilities including dashboards and data visualization.
- DB2 Accessories Suite for z/OS, V1.1 (5655-R14) features components that enhance SQL tuning and support spatial data and Unicode.
- DB2 Utilities Suite for z/OS, V9.1 (5655-N97) supports all major new functions and structural changes in DB2 9.

DB2 Optimization Expert for z/OS

DB2 Optimization Expert for z/OS, V1.1 (program number 5655-S19) supports monitoring and tuning of SQL statements that run as part of a workload on your DB2 for z/OS subsystem. You can:

- View query activity to find performance problems:
  - Quickly *snap the cache* of your DB2 subsystem to obtain:
    - What dynamic queries have been running
    - Which of those queries might be causing performance problems
  - View query activity from a number of other sources, for example, the DB2 catalog.
- Get expert tuning advice to improve the performance of an SQL query. You can easily get and implement expert tuning recommendations. After identifying a problem query, you can run any or all of the expert advisors.
  - Statistics advisor: recommends statistics to update or collect to improve the performance of a query.
    - Control statements are generated for you to collect and update needed statistics with the RUNSTATS utility.
    - The RUNSTATS utility can be invoked directly from your workstation.
    - DB2 Optimization Expert generates statistics beyond those provided by RUNSTATS ALL.
  - Query advisor: recommends ways to rewrite an SQL query to improve performance.
    - Considers many different conditions.
    - Recommends best-practice fixes to common query-writing mistakes
  - Access path advisor: Alerts you to problematic access paths that might cause poor performance in the access plan for the query.
  - Index advisor: recommends new indexes to enhance the performance of an SQL query. In addition, CREATE INDEX statements are generated that you can:
    - Use to implement the recommendations.
    - Execute directly from your workstation on your DB2 for z/OS subsystem.
  - Advanced tuning is supported with the powerful optimization tools provided by the Optimization Service Center functions. All the Optimization Service Center functions of DB2 Accessories Suite are included with DB2 Optimization Expert. In summary, they allow your experienced DBAs to understand, analyze, format, and optimize SQL

Archived

statements. Refer to the DB2 Accessories Suite section for descriptions of Optimization Service Center functions.

- View the activity of groups of SQL statements to find performance problems. You can monitor the health of the SQL workloads. After the performance of SQL queries is optimized, you can create monitor profiles that:
  - Monitor the health of SQL processing on the subsystem.
  - Alert you when problems develop or when more tuning activities are advised.
- Create monitor profiles for:
  - Normal processing to record information about the normal execution of static and dynamic SQL statements
  - Exception processing to record information about the execution of SQL statements when the execution exceeds specific thresholds
- Get expert tuning advice to improve the performance of an SQL workload. Expert advisors look beyond the steps to take to improve the performance of a single SQL query and recommend tuning activities to improve the overall performance of an entire DB2 SQL workload.
  - Workload statistics advisor: recommends statistics to collect or update to improve the overall performance of the statements that make up an SQL workload.
    - Control statements are generated for you to collect and update the statistics needed by the RUNSTATS utility.
    - The RUNSTATS utility can be invoked directly from your workstation.
  - Workload index advisor: recommends indexes to create or modify to improve the overall performance of the statemennts that make up an SQL workload. In addition, the CREATE INDEX statements are generated that you can:
    - Use to implement the recommendations.
    - Execute directly from your workstation.
  - Workload query advisor: recommends ways to rewrite specific SQL queries to improve performance of a workload.
    - Workload query reports enable you to monitor queries as a health check.
    - Allow you to drill down to those queries with more serious query rewrite problems.
    - Consider many different conditions.
    - Recommend best-practice fixes to common query-writing mistakes.

DB2 Optimization Expert is built on Eclipse, an award-winning, open-source platform for the construction of powerful software development tools and rich desktop applications. Leveraging the Eclipse plug-in framework to integrate technology on the desktop saves technology providers time and money. They can focus their efforts on delivering differentiation and value for their offerings. For more information about Eclipse visit:

http://www.eclipse.org

DataQuant for z/OS

DataQuant for z/OS, V1.1 (program number 5697-N64) delivers a comprehensive query, reporting, and data visualization platform for both Web and workstation-based environments. While remaining compatible with the IBM Query Management Facility (QMF) product line, DataQuant introduces a variety of powerful business intelligence capabilities, including:

- Powerful graphical reporting environment that allows rapid development and deployment of executive dashboards, information portals, and interactive data visualization solutions.

Archived

- Drag-and-drop development of OLAP analytics, SQL queries, tabular reports, graphical reports, pivot tables, and data analysis views.
- Dozens of charts, controls, and graphical primitives to visual reports, along with the ability to embed subqueries that provide supporting details.
- Full compatibility with QMF infrastructure and objects, including the ability to create, open, edit, delete, and save QMF queries, procedures, forms, and visual reports.
- Over 100 built-in mathematical and analytical functions accessible in both dashboards and visual solutions as well as in printable reports.
- Rich security infrastructure, providing personalization of both the query and reporting environment as well as business intelligence content distributed within it. This infrastructure includes the following:
  - Supports single sign-on and optionally interfaces with existing LDAP directories or internally defined DataQuant user directory.
  - Tailors look and feel, including available reports, visualizations, and data, on a per-user or group basis. For example, technical resources may see a traditional database-centric view, while business users see a role-specific view of relevant reports and dashboards.
  - Optionally logs on to data sources using accounts mapped from LDAP/internal security accounts.
  - Tailors offline schemas to each user or group so that individuals see only those tables and columns relevant to their job function.
- Eclipse-based rich desktop environment that supports numerous workstation operating systems, including Windows, Linux, Solaris, and AIX.
- High-performance, WebSphere-based environment that extends key DataQuant functions to browser-based users across multiple platforms, including querying, content authoring, and drag-drop data analysis.
- Service-oriented architecture (SOA) infrastructure, enabling enterprises to share specific business intelligence content and data with partners over secure Internet or intranet connections.
- Support for DB2 and Informix, as well as other popular database systems.
- Building and deploying information-rich business intelligence solutions that concurrently draw data from one or more heterogeneous relational databases.

With DataQuant you can derive maximum value from your data and rapidly build and distribute comprehensive business intelligence solutions across your enterprise.

Like DB2 Optimization Expert, DataQuant is built on Eclipse.

DB2 Utilities Suite for z/OS

DB2 Utilities Suite for z/OS, V9.1 (program number 5655-N97) supports all major new functions and structural changes in the DB2 for z/OS product, including the following functions and enhancements:

- The BACKUP SYSTEM utility includes new options to assist in the offloading of the system-level backups to tape.
- The following utilities support the clone table function:
  - CHECK DATA
  - CHECK INDEX
  - CHECK LOB

  - COPY
  - COPYTOCOPY
  - DIAGNOSE
  - LISTDEF
  - MERGECOPY
  - MODIFY RECOVERY
  - QUIESCE
  - RECOVER
  - REORG INDEX
  - REORG TABLESPACE
  - REPAIR
  - REPORT
  - UNLOAD
- The following utilities deliver higher data availability while running:
  - CHECK DATA
  - CHECK LOB
  - REBUILD INDEX
  - REORG LOB (now allows read access)
  - LOAD REPLACE with Clone Tables function
  - REPAIR
  - REORG TABLESPACE PARTITION (The BUILD2 phase has been eliminated.)
- The COPY utility includes SCOPE PENDING support to improve usability. The utility also allows you to limit operations to partitions and list items that are in copy pending or informational copy pending status.
- The LOAD and UNLOAD utilities support new data types and a new rounding mode.
- The MODIFY RECOVERY utility offers alternative ways to remove entries from SYSCOPY and SYSLGRNX.
- The RECOVER utility is enhanced to:
  - Recover to any point in time with consistency.
  - Select which image copy to restore for the recovery.
  - Ensure that the utility uses the most recent image copy prior to the recovery point.
  - Recover at the object level using system-level backups.
- The RESTORE SYSTEM utility gives you more tape control options, enabling the restore of system-level backups from tape with parallelism.
- The RUNSTATS utility collects new statistics, allowing DB2 to select a better access path.
- You can control the data set attributes of image copy output based on the projected size of the data set (for example, tape or disk).
- The UNLOAD utility supports skipping rows that are locked for transaction updates.
- Large block interface is now supported for tapes, resulting in a substantial performance improvement.
- The LOAD, REORG, CHECK INDEX, and REBUILD INDEX utilities include path length (CPU) reductions.

DB2 V9 Utility Packaging Changes

In V8 DB2 Utilities were separately orderable as a single feature but provided as two FMIDs, requiring two separate SMP/e installs. In DB2 V9 the two FMIDs are merged into a single FMID on a single tape, thus simplifying the SMP/e install process for customers by eliminating one tape, and sample SMP/e install jobs DSNRECVM, DSNRECVS, DSNAPPLM, DSNAPPLS, DSNACCPM, and DSNACCPS have been deleted.

Archived

DB2 Accessories Suite for z/OS

DB2 Accessories Suite for z/OS, V1.1 (program number 5655-R14) is made up of a series of independent components designed to enhance your use of the DB2 for z/OS data server. In this initial release of the suite, three components deliver innovative new capabilities for your DB2 for z/OS environment, including enhanced SQL tuning, support for spatial data, and support for Unicode and internationalization operations.

- *Optimization Service Center for DB2 for z/OS* improves the performance of DB2 SQL queries at both the individual query and full SQL workload level. For individual queries, Optimization Service Center can:
  - Annotate the query.
  - Draw an access plan graph.
  - Generate query reports.
  - Facilitate generation of optimization hints.
  - Suggest statistics needed by SQL Optimizer.

  At your request, Optimization Service Center can send query-related information back to IBM service. For SQL workloads, Optimization Service Center can gather workloads from various kinds of sources such as the dynamic statement cache, the catalog tables, the files and directories, the QMF tables, the profile monitoring, and so on. SQL queries can be captured by Optimization Service Center immediately or at a scheduled time, either once or periodically. Optimization Service Center can also suggest important statistics to be collected to improve performance for the entire workload. Profile monitoring is a lightweight exception monitoring mechanism. Only those exception statements are pushed out by DB2, and Optimization Service Center fully supports the use and management of profile monitoring.
- *Spatial Support for DB2 for z/OS* contains a set of spatial data types, user-defined functions, and stored procedures for spatial related queries. You can invoke these spatial queries for local and remote clients to answer questions based on geographic relationships. You can use relationships such as contains, crosses, equals, intersects, overlaps, touches, within, distance, and envelope intersect in a WHERE clause of the SELECT statement with predicate functions. Spatial Support creates spatial indexes on spatial columns, which can improve query performance when using spatial predicate functions. Stored procedure interfaces enable you to manage geographic coordinate systems, spatial indexes, and spatial column usages. Spatial Support includes an ODBC program to enable and disable the spatial feature. This program can invoke each of the stored procedures through a set of command line arguments.
- *The International Components for Unicode for DB/2 for z/OS* (ICU) is a set of C/C++ and Java libraries for Unicode support and software internationalization. ICU is an open source project sponsored by IBM that provides full-fledged Unicode services on a wide variety of platforms. It is called by selected features of DB2 for z/OS and the Accessories Suite, such as Spatial Support, that require these Unicode and internationalization functions.

Like DB2 Optimization Expert and DataQuant, Optimization Service Center is built on Eclipse.

Abbreviations and acronyms

AC	Autonomic computing
ACS	Automatic class selection
AIX	Advanced Interactive eXecutive from IBM
APAR	Authorized program analysis report
API	Application programming interface
AR	Application requester
ARM	Automatic restart manager
AS	Application server
ASCII	American National Standard Code for Information Interchange
B2B	Business-to-business
BCDS	DFSMShsm backup control data set
BCRS	Business continuity recovery services
BI	Business Intelligence
BLOB	Binary large objects
BPA	Buffer pool analysis
BSDS	Boot strap data set
CBU	Capacity BackUp
CCA	Channel connection address
CCA	Client configuration assistant
CCP	Collect CPU parallel
CCSID	Coded character set identifier
CD	Compact disk
CDW	Central data warehouse
CEC	Central electronics complex
CF	Coupling facility
CFCC	Coupling facility control code
CFRM	Coupling facility resource management
CICS	Customer Information Control System
CLI	Call level interface
CLOB	Character large object
CLP	Command line processor
CMOS	Complementary metal oxide semiconductor
CP	Central processor
CPU	Central processing unit
CRCR	Conditional restart control record
CRD	Collect report data
CRUD	Create, retrieve, update or delete
CSA	Common storage area
CSF	Integrated Cryptographic Service Facility
CTE	Common table expression
CTT	Created temporary table
CUoD	Capacity Upgrade on Demand
DAC	Discretionary access control
DASD	Direct access storage device
DB	Database
DB2	Database 2™
DB2 PE	DB2 Performance Expert
DBA	Database administrator
DBAT	Database access thread
DBCLOB	Double-byte character large object
DBCS	Double-byte character set
DBD	Database descriptor
DBID	Database identifier
DBM1	Database master address space
DBRM	Database request module
DCL	Data control language
DDCS	Distributed database connection services
DDF	Distributed data facility
DDL	Data definition language
DES	Data Encryption Standard
DLL	Dynamic load library manipulation language
DML	Data manipulation language
DNS	Domain name server
DPSI	Data partitioning secondary index
DRDA	Distributed Relational Data Architecture
DSC	Dynamic statement cache, local or global
DSNZPARMs	DB2's system configuration parameters
DSS	Decision support systems
DTT	Declared temporary tables
DWDM	Dense wavelength division multiplexer

© Copyright IBM Corp. 2007. All rights reserved.

DWT	Deferred write threshold
EA	Extended addressability
EAI	Enterprise application integration
EAS	Enterprise Application Solution
EBCDIC	Extended binary coded decimal interchange code
ECS	Enhanced catalog sharing
ECSA	Extended common storage area
EDM	Environmental descriptor manager
EJB™	Enterprise JavaBean
ELB	Extended long busy
ENFM	Enable-new-function mode
ERP	Enterprise resource planning
ERP	Error recovery procedure
ESA	Enterprise Systems Architecture
ESP	Enterprise Solution Package
ESS	Enterprise Storage Server
ETR	External throughput rate, an elapsed time measure, focuses on system capacity
EWLC	Entry Workload License Charges
EWLM	Enterprise Workload Manager
FIFO	First in first out
FLA	Fast log apply
FTD	Functional track directory
FTP	File Transfer Program
GB	Gigabyte (1,073,741,824 bytes)
GBP	Group buffer pool
GDPS®	Geographically Dispersed Parallel Sysplex™
GLBA	Gramm-Leach-Bliley Act of 1999
GRS	Global resource serialization
GUI	Graphical user interface
HALDB	High Availability Large Databases
HPJ	High performance Java
HTTP	Hypertext Transfer Protocol
HW	Hardware
I/O	Input/output
IBM	International Business Machines Corporation
ICF	Internal coupling facility
ICF	Integrated catalog facility
ICMF	Integrated coupling migration facility
ICSF	Integrated Cryptographic Service Facility
IDE	Integrated development environments
IFCID	Instrumentation facility component identifier
IFI	Instrumentation Facility Interface
IFL	Integrated Facility for Linux
IMS	Information Management System
IORP	I/O Request Priority
IPLA	IBM Program Licence Agreement
IRD	Intelligent Resource Director
IRLM	Internal resource lock manager
IRWW	IBM Relational Warehouse Workload
ISPF	Interactive system productivity facility
ISV	Independent software vendor
IT	Information technology
ITR	Internal throughput rate, a processor time measure, focuses on processor capacity
ITSO	International Technical Support Organization
IVP	Installation verification process
J2EE™	Java 2 Enterprise Edition
JDBC	Java Database Connectivity
JFS	Journaled file systems
JNDI	Java Naming and Directory Interface
JTA	Java Transaction API
JTS	Java Transaction Service
JVM	Java Virtual Machine
KB	Kilobyte (1,024 bytes)
LCU	Logical Control Unit
LDAP	Lightweight Directory Access Protocol
LOB	Large object
LPAR	Logical partition
LPL	Logical page list
LRECL	Logical record length
LRSN	Log record sequence number
LRU	Least recently used
LSS	Logical subsystem
LUW	Logical unit of work
LVM	Logical volume manager
MAC	Mandatory access control
MB	Megabyte (1,048,576 bytes)

Archive

MBps	Megabytes per second
MLS	Multi-level security
MQT	Materialized query table
MTBF	Mean time between failures
MVS	Multiple Virtual Storage
NALC	New Application License Charge
NFM	New-function mode
NFS	Network File System
NPI	Non-partitioning index
NPSI	Nonpartitioned secondary index
NVS	Non volatile storage
ODB	Object descriptor in DBD
ODBC	Open Database Connectivity
ODS	Operational Data Store
OLE	Object Link Embedded
OLTP	Online transaction processing
OP	Online performance
OS/390	Operating System/390®
OSC	Optimizer service center
PAV	Parallel access volume
PCICA	Peripheral Component Interface Cryptographic Accelerator
PCICC	PCI Cryptographic Coprocessor
PDS	Partitioned data set
PIB	Parallel index build
PPRC	Peer-to-Peer Remote Copy
PR/SM™	Processor Resource/System Manager
PSID	Pageset identifier
PSP	Preventive service planning
PTF	Program temporary fix
PUNC	Possibly uncommitted
PWH	Performance Warehouse
QA	Quality Assurance
QMF	Query Management Facility
QoS	Quality of Service
QPP	Quality Partnership Program
RACF	Resource Access Control Facility
RAS	Reliability, availability and serviceability
RBA	Relative byte address
RBLP	Recovery base log point
RDBMS	Relational database management system
RDS	Relational data system
RECFM	Record format
RI	Referential Integrity
RID	Record identifier
ROI	Return on investment
RPO	Recovery point objective
RR	Repeatable read
RRS	Resource recovery services
RRSAF	Resource recovery services attach facility
RS	Read stability
RTO	Recovery time objective
RTS	Real-time statistics
SAN	Storage area networks
SBCS	Store single byte character set
SCUBA	Self contained underwater breathing apparatus
SDM	System Data Mover
SDP	Software Development Platform
SLA	Service-level agreement
SMIT	System Management Interface Tool
SOA	Service-oriented architecture
SOAP	Simple Object Access Protocol
SPL	Selective partition locking
SQL	Structured Query Language
SQLJ	Structured Query Language for Java
SRM	Service Request Manager
SSL	Secure Sockets Layer
SU	Service Unit
TCO	Total cost of ownership
TPF	Transaction Processing Facility
UA	Unit Addresses
UCB	Unit Control Block
UDB	Universal Database
UDF	User-defined functions
UDT	User-defined data types
UOW	Unit of work
UR	Unit of recovery
vCF	Virtual coupling facility
VIPA	Virtual IP Addressing
VLDB	Very large database
VM	Virtual machine
VSE	Virtual Storage Extended
VSIP	Visual Studio® Integrator Program

Archive

VWLC	Variable Workload License Charges
wizards	Web-based assistants
WLC	Workload License Charges
WLM	Workload Manager
WSDL	Web Services Definition Language
WTO	Write to operator
XA	Extended Architecture
XML	Extensible Markup Language
XRC	EXtended Remote Copy
z800	ZSeries 800
z890	ZSeries 890
z990	ZSeries 990
zAAP	ZSeries Application Assist Processor
zELC	ZSeries Entry License Charge
zIIP	IBM System z9 Integrated Information Processor

Archived

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 549. Note that some of the documents referenced here may be available in softcopy only.

- *DB2 9 for z/OS Performance Topics,* SG24-7473
- *Securing DB2 and Implementing MLS on z/OS*, SG24-6480
- *LOBs with DB2 for z/OS: Stronger and Faster*, SG24-7270
- *DB2 UDB for z/OS Version 8 Performance Topics*, SG24-6465
- *Disaster Recovery with DB2 UDB for z/OS*, SG24-6370
- *DB2 for z/OS Stored Procedures: Through the CALL and Beyond*, SG24-7083
- *DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ... and More*, SG24-6079
- *Communications Server for z/OS V1R8 TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7339

Other publications

These publications are also relevant as further information sources:

- *DB2 Version 9.1 for z/OS Administration Guide,* SC18-9840-02
- *DB2 Version 9.1 for z/OS Application Programming and SQL Guide,* SC18-9841-01
- *DB2 Version 9.1 for z/OS Application Programming Guide and Reference for JAVA* SC18-9842-02
- *DB2 Version 9.1 for z/OS Codes,* GC18-9843-02
- *DB2 Version 9.1 for z/OS Command Reference,* SC18-9844-02
- *DB2 Version 9.1 for z/OS Data Sharing: Planning and Administration,* SC18-9845-01
- *DB2 Version 9.1 for z/OS Diagnosis Guide and Reference,* LY37-3218-01
- *DB2 Version 9.1 for z/OS Diagnostic Quick Reference,* LY37-3219-00
- *DB2 Version 9.1 for z/OS Installation Guide,* GC18-9846-03
- *DB2 Version 9.1 for z/OS Introduction to DB2,* SC18-9847-01
- *DB2 Version 9.1 for z/OS Licensed Program Specifications,* GC18-9848-00
- *DB2 Version 9.1 for z/OS Messages,* GC18-9849-02
- *DB2 Version 9.1 for z/OS ODBC Guide and Reference,* SC18-9850-01
- *DB2 Version 9.1 for z/OS Performance Monitoring and Tuning Guide,* SC18-9851-03
- *DB2 Version 9.1 for z/OS RACF Access Control Module Guide,* SC18-9852-01

© Copyright IBM Corp. 2007. All rights reserved.

- *DB2 Version 9.1 for z/OS Reference for Remote DRDA Requesters and Servers,* SC18-9853-01
- *DB2 Version 9.1 for z/OS Reference Summary,* SX26-3854-00
- *DB2 Version 9.1 for z/OS SQL Reference,* SC18-9854-03
- *DB2 Version 9.1 for z/OS Utility Guide and Reference,* SC18-9855-02
- *DB2 Version 9.1 for z/OS What's New?,* GC18-9856-00
- *DB2 Version 9.1 for z/OS XML Extender Administration and Programming,* SC18-9857-01
- *DB2 Version 9.1 for z/OS XML Guide,* SC18-9858-03
- *Optimization Service Center for DB2 for z/OS Getting Started*, GC19-1143-02
- *Spatial Support for DB2 for z/OS User's Guide and Reference*, GC19-1145-01
- *OmniFind Text Search Server for z/OS Installation, Administration, and Reference*, GC19-1146-00
- *Program Directory for IBM DB2 9 for z/OS,* GI10-8737-00
- *Program Directory for IBM DB2 V9.1 for z/OS DB2 Management Clients Package,* GI10-8738-00
- *Program Directory for IBM DB2 V9 for z/OS z/OS Application Connectivity to DB2 for z/OS*, GI10-8739-00
- *Program Directory for IBM DB2 Accessories Suite for z/OS*, GI10-8749-01
- *z/OS Support for Unicode: Using Conversion Services*, SA22-7649
- *z/OS Migration Version 1 Release 7*, GA22-7499-08
- *z/Architecture Principles of Operations*, SA22-7832-04
- *z/OS C/C++ Programming Guide*, SC09-4765
- *z/OS V1R6.0 DFSMS Implementing System Managed Storage*, SC26-7407-02
- *DB2 V9.1 for Linux, UNIX, and Windows Administrative API Reference,* SC10-4231
- *DB2 SQL Reference for Cross-Platform Development, Version 3,* available from:

  ftp://ftp.software.ibm.com/ps/products/db2/info/xplatsql/pdf/en_US/cpsqlrv3.pdf

Online resources

These Web sites are also relevant as further information sources:

- DB2 for z/OS

  http://www.ibm.com/software/data/db2/zos/
- ESRI Web site

  http://www.esri.com/
- DB2 Utilities Suite V9.1 for z/OS

  http://www.ibm.com/software/data/db2imstools/db2tools/db2utilsuite/
- Information on zIIP

  http://www.ibm.com/systems/z/ziip/

Archive

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Archived

Archived

Index

Symbols
+434 225

Numerics
4096 partitions 62
5655-N97 426, 431, 540
5655-S19 22, 538
5697-N64 21, 539
-7008 139
93 active log data sets 449

A
access control 20, 137, 278
ACTIVATE VERSION 68, 230, 232
Adaptive Multi-Stream Prefetching 7
ADD VERSION 68, 226, 230, 232
ALTER ACTIVE VERSION 230
ALTER BUFFERPOOL 47–48
ALTER GROUPBUFFERPOOL 125
ALTER TABLE 76, 81, 83, 179–182, 214, 273
  ALTER COLUMN 215
APIs 16, 200, 257, 357, 366, 499
APPL % 31
application 6, 15, 50, 73, 127, 129, 131, 178, 180, 302
APPLICATION ENCODING SCHEME 224
application programming 223, 352
Architecture 344, 420
argument 156, 178, 188, 276
ASCII 18, 132, 156, 212, 460, 487
ASCII_CHR 155–156
ATOMIC 140–141, 144, 254
Atomic values 283
Atomization 285
attribute 47, 133, 187–188, 270
Attributes 270
Audit Trace 71
auxiliary column 194
auxiliary index 168, 173–174, 194
  DDL 174
auxiliary LOB table 151
auxiliary table 168, 173–174, 194, 486
AUXW 54
axis 289

B
BACKUP 15, 106, 391, 393, 431
BACKUP SYSTEM 391–393
Base engine 425
base table 6, 51, 53–54, 81–82, 85, 137–138, 150, 174, 179, 194, 390, 407, 486, 491
BIGINT 5, 132, 155–156, 355, 399, 437, 478, 489
Binary Large Objects 193
BIND
  DEPLOY option 233
BIND DEPLOY 222
BIT 132–133, 178, 186, 230
BLOB 132, 153, 197–198, 271, 279
BLOB_FILE 198
BLOBs 193, 528
BMP 461
Boolean term predicate 511
BOTH 253, 466
boundary whitespace 283
BPOOL 48
broken page 117–118
BSDS 105–106, 342–343, 351, 405, 407, 421–422, 432–433, 440
buffer pool management 47–48, 486, 530–531
buffer pool management by WLM 530
buffer pools 20, 48, 71, 123, 172, 438, 494, 498, 530
BUILD2 2, 386–388, 436, 509

C
CACHE 392, 457, 461–462, 468
CACHEDYN 499
Call Level Interface 523
cancel DB commands 119
CARDF 506
CARDINALITY 502
carriage returns 283
CAST 132, 162, 164, 440
CATENFM 431, 478–480
CATMAINT 16, 111–113, 423, 431, 480, 485
CCSID 138, 156, 160, 178, 183, 186, 199, 460
  1208 460
  367 156
CF 10, 123, 128
channel adapter 4
Channel efficiency 5
Character Large Objects 193
check constraints 81, 92, 112–113, 422
CHECK DATA 57, 92, 407, 410, 431
CHECK INDEX 93, 410, 412, 431, 508–509
CHECK LOB 93, 409–410, 431
CHECKPAGE 118, 402
chunk 205
chunking 390
CICS xxviii, 17, 71, 335, 363, 430, 454, 531
class 28, 215, 267
CLI 16, 499, 503, 523
client reroute 358
Client SRB 14
CLIST modes 441
CLOB 4, 153, 158, 186, 197, 266, 269
CLOB_FILE 198
clone table 81–82, 84

© Copyright IBM Corp. 2007. All rights reserved.

CLUSTER 76, 80, 83, 85
Clusters 423
CM 59, 421, 433, 441, 477
CM* 477
COEXIST 434, 459
COLGROUP 508
Collation 183–185
COLLATION_KEY 155, 186, 188–189
COLLID 225
colon 278
COLTYPE 215, 436
Column
  XML 282
COMMENT 254
Comments xxxi, 271
Communications Server 69
Compatibility mode 478, 485–486
Compatibility mode* 477
components 4, 183, 265
compound statement 224, 238–240
compression 13, 3, 59, 63, 66, 171, 361, 491, 495
CONCAT 177, 236
condition 35, 56, 72, 74, 77, 112, 124, 138–139, 141–142, 179, 217, 293, 481, 502
condition names 35, 238, 240, 244
CONNECT 344, 457
constructor 284
Control Center 277, 424, 426–427
Coprocessor 371
COPY 15, 51, 57, 66, 79, 86, 230, 260, 349, 379, 388, 392, 394, 431, 443, 451–452, 508
COPYDDN 404
COPYPOOL 393–394
copypool 391, 396–397
COPYTOCOPY 93, 404, 431
COUNT 105, 107, 193, 466
Coupling Facility 10, 12, 128
CPACF 371
CPU Activity for zIIP 29
CRCR 105–107
CREATE TABLE
  LIKE 97
CREATEIN 218
CREATOR 83–84
CS 69, 179, 192, 253, 355, 400, 434, 489
CSA 71
CT 47
CTHREAD 202
cultural sort 3, 183
CURRENT DATA 223
CURRENT DEBUG MODE 222, 249
CURRENT MAINT TYPES 461
CURRENT REFRESH AGE 461
CURRENT ROUTINE VERSION 68, 230–231, 233, 249–250
CURRENT RULES 173
CURRENT SCHEMA 189–191, 364
CURRENT SQLID 189–191, 363
CURRENT_VERSION 92
CURRENTAPPENSCHEM 200
cursor definitions 243
cursor stability 179, 192, 223
CVT 14
CYCLE 29, 462

D

DAS 427
Data length 199
data model 265
data propagation 181
data sharing xxvii, 13, 59, 71, 101, 103, 106, 113, 121–122, 126, 224, 343, 351, 405, 411, 421, 432, 436, 441, 494, 504, 530
Database Administration Server 427
Database name 467
DATACLAS 16, 216–218
DataQuant for z/OS 21, 539
DATASIZE 489
DATE 12, 29, 98, 102, 108, 147, 163–164, 225, 460
DB2 authorization 366, 374–375, 463
DB2 Connect 13, 363, 427
DB2 Customization Center 424
DB2 Development Center 424
DB2 Management Clients Package 427
DB2 Optimization Expert for z/OS 22, 538
DB2 Optimization Service Center 22, 429
DB2 Spatial Support 429
DB2 Utilities Suite 426, 429, 431
DB2 Utilities Suite for for z/OS, V9.1 540
DB2 XML Extender
  See XML Extender
DB2_ROW_NUMBER 141
DBCLOB 153, 158, 186, 197, 279
DBCLOB_FILE 198
DBCS 237, 460, 500
DBD 54, 86, 90–91, 119, 274, 468, 482
DBD01 98
DBET 76, 113–114, 485, 488
DBETE 114, 485
DBINFO 225
DBMA 440, 447, 484
DBMS 19, 213
DBNAME 83, 97, 102
DBRM 344, 346, 350, 453, 472
DDF 11, 26, 44, 335, 343, 447–448, 457, 459, 465, 486, 499, 524, 531
DDL 77, 109, 447, 467, 476, 500, 504–505
DEBUG MODE 222–223
DEBUG mode 251
DECFLOAT 17, 4, 132, 178, 355, 460, 487, 492
decimal 17, 4, 98, 129, 225, 284
DECLARE VARIABLE 433
Decomposition 323, 443
decomposition 183, 266
default namespace 286
DEGREE 223
DELETE 6, 12, 43, 137, 177
delete 6, 56, 63, 74, 80, 137, 146, 179, 356, 367, 378, 405, 409, 500, 528, 533
DEPLOY 68, 222, 233

descendant 289
DETERMINISTIC 226
Development Center 232, 262, 424, 427
DFSMS 16, 59, 216–217, 408–409
DFSMShsm 391, 395, 492
DIFFERENCE 155, 157, 487
DISABLE DEBUG MODE 222
DISPLAY PROCEDURE 251
DISPLAY THREAD 60–61, 119, 433
DISPLAY THREAD TYPE ACTIVE 341
DIST address space 11, 44–45
Distributed xxviii, 21, 32, 44, 344
Distributed Data Facility 11, 44
Distributed Relational Database Architecture™ 16
DPSI 122, 494
DRDA 7, 11, 16, 44, 335, 344, 363, 425, 440, 447, 494, 524, 527
DRDA clients 466
DROP STORAGE 151
DROP VERSION 231
DS8000 17, 3
DSMAX 202, 468
DSN1COMP 66–67
DSN1COPY 6, 66, 68, 79–80, 389, 411
DSN1COPY RESET 411
DSN1LOGP utility 96
DSN1PRNT 64–65, 68, 117
DSN6SPRM 436, 447, 484, 490
DSN6SYSP 447–449, 481
DSNDB06 51, 98, 112, 170, 488
DSNESP07 255
DSNHDECP 156, 168, 399, 421, 425, 474, 479
  NEWFUN 479
DSNI005I 125
DSNI046I 114, 485
DSNJ157I 480
DSNJCNVB 422, 449, 480
DSNJU003 103, 105–106, 342, 447, 480
DSNJU004 59, 104, 342–343
DSNR class 419
DSNSPAS 439, 449
DSNT397I 54, 86, 90–91, 119, 274
DSNT408I 215, 282, 311
DSNT500I 116, 415
DSNT55I 445
DSNTEJ1 471, 474
DSNTEJ3M 471, 474
DSNTEJ7 471, 474
DSNTEJ76 471, 474
DSNTEJ77 471, 474
DSNTEJ78 471, 474
DSNTEP2 203, 252, 302
DSNTEP4 252
DSNTIDXA 442, 450
DSNTIJEN 437, 474–475
DSNTIJID 471, 475
DSNTIJIN 468, 471, 475, 479
DSNTIJLR 475
DSNTIJMC 475
DSNTIJMS 475
DSNTIJNE 475, 477
DSNTIJNF 474–475, 477, 479
DSNTIJNG 474–475, 479
DSNTIJNH 475
DSNTIJNR 475
DSNTIJP9 421
DSNTIJPM 475
DSNTIJSG 351, 443, 504–505
DSNTIJTC 423, 468, 471, 475, 477, 480
DSNTIJTM 434, 446
DSNTIJUZ 222, 421, 433, 446
DSNTIP00 473
DSNTIP4 460
DSNTIP5 447–448, 466, 490
DSNTIP6 391–392, 462, 491–492
DSNTIP7 455, 491
DSNTIP9 438, 445
DSNTIPA1 442, 450, 469–470
DSNTIPA2 450
DSNTIPC 468, 492
DSNTIPD 455
DSNTIPE 202, 457, 491
DSNTIPF 460, 492
DSNTIPI 461
DSNTIPN 459
DSNTIPO 434, 459
DSNTIPP 463
DSNTIPR 465
DSNTIPT 453, 472
DSNTIPX 222
DSNTYPE=LARGE 60, 411
DSNU116I 118, 397
DSNU1504I 54–55
DSNU1505I 56
DSNU1553I 101–102
DSNUM 118, 403
DSNV447I 341
DSNV448I 341
DSNW128I 489
DSNW129I 489
DSNWZP 436
DSNX208E 421
DSNX209E 421
DSNX930 439
DSNX9STP 439
DSNY011I 44
DSNY012I 44
DSNZPARM 58, 112, 168, 171, 222, 391, 440, 499, 517–518
  MXDTCACH 519
  RELCURHL 440
DSSIZE 63, 75, 171
DSVCI 484
dump classes 393
DYNAMIC RESULT SETS 226
DYNAMICRULES 142, 190, 223, 226, 460

E

EARLY code 107
ECSA 44

EDITPROC 436
EDMPOOL 47, 226, 436, 468, 533
efficiency 183, 312, 512, 530
EIM 363, 366
element 41, 269
Elements 270–271, 285
Enabling-new-function mode* 477
Enclave SRB 14
ENCODING 460
Encoding scheme 199
ENDRBA 98, 103, 105
ENFM 59–60, 433, 441–442, 477
ENFM* 477
Enterprise Identity Mapping 363, 366–367, 375
environment xxvii, 7, 14, 50, 101, 121, 129, 137, 267
EXCEPT 6, 152–153, 155, 436
EXCLUDE 70–71, 93
EXECUTION DELAYS % 32
EXISTS 155, 510, 513
EXPLAIN 141, 191, 224, 251, 448, 459, 502, 514–515
  QUERYNO 141
Explain 459
EXPLAIN output 514
expression 3, 93, 134, 276
extensions 4, 268
EXTRACT 155, 487
EXTRACT date values 157
EXTRACT time values 158

F

Fact table 519, 522
Fallback 475, 478–479
Fallback SPE 479
fallback SPE 421
Fast log apply 125–126
FENCED 255
FETCH 7, 146–147, 179–180, 202, 344, 435, 529
fetch 18, 147, 179, 203, 206, 356, 434, 522
FETCH CONTINUE 203–204
FETCH CURRENT 204–206
FETCH FIRST n ROWS ONLY 147
FETCH WITH CONTINUE 203–205
FICON 17, 3–4
FIELDPROC 134
File name length 198
file option variable 200
file reference variables 10, 194, 197–198
FINAL TABLE 143–144, 146
FlashCop 3
FlashCopy 15, 4, 391, 393
flexibility 7, 129, 216, 255, 265, 267, 305, 403, 503
flow optimization 523, 527
FMID 14, 426–427, 429
fn
  abs 306
  compare 306
  concat 307
  contains 307
  empty 306
  number 308
  round 310
FOR BIT DATA 132–133
FOR SQL statement 235
FOR UPDATE CLAUSE 225
function 2, 3, 69, 80, 122, 132, 271
Functional comments 253

G

GBPCACHE 122, 530
GBPCACHE SYSTEM 530
GENERATED ALWAYS 7, 179, 181
GENERATED BY DEFAULT 7, 168, 174, 179, 434
generations 396
GET DIAGNOSTICS 140, 234
GI10-8737-00 548
GRAPHIC 187–188
GRECP 113, 125–126
group attachment 123, 488
GRS 202

H

H2AF110 429
H2AG110 429
handle 2, 14, 299
handler declarations 244, 246
health check 421
HFS 201, 260, 366, 414
Hierarchical File System 201
HISTOGRAM 505–506, 508
Histogram statistics 19, 506–507
host adapter 4
host variable 140, 144–145, 178, 186, 195, 282, 433, 435, 498
host variables 45, 144, 194, 197–198, 433, 498–499, 502
HTML 3, 265, 430

I

IBM DB2 Command Center 427
IBM DB2 Control Center 427
IBM DB2 Development Center 427
IBM DB2 Replication Center 427
IBM Developer Workbench 427
IBM System z9 Integrated Information Processor 18, 4
IBMREQD 238
ICF 12, 216
ICHRFR01 420
ICTYPE 53, 82, 84, 87, 398, 403, 405
IDCAMS 215
IEAOPTxx 23
IFCID
  148 71
  172 115
  3 484
IFCID 306 488
IFCID217 45
IFCID225 45
IFL 12
II12423 421

II14203 45
II14219 16, 23
Image Copy 115
IMMEDWRITE 224
IMPDSDEF 171, 491
IMPLICIT 168
Implicit database 168
implicit table authorization 174
implicit table space authorization 175
implicitly defined objects 173
IMPLICITLY HIDDEN 182
IMS 17, 335, 363, 426, 437, 454, 461, 475, 531
Incompatibilities 433
incremental FlashCopy 391, 395, 397
index xxvii, 3, 14, 51, 79, 122, 133, 236
index ANDing 27, 519
Index compression 122
index compression 20, 66–67
Index maintenance 12, 22
Index on expression 19, 175
  Errors 177
  restrictions 178
  result length 177
index on expressions 236
index page size 66, 68, 71, 495
Infinity 134
in-memory data cache 19, 517
INSERT 6, 10, 12, 43, 45, 55, 64, 97, 110, 133, 137, 181–182, 247, 271, 277, 282, 344, 367, 370, 434, 500–502
insert xxvii, 5, 48, 59, 75, 77, 81, 97, 122, 137, 139, 177, 180, 198, 266, 271, 282, 356, 367, 378, 434, 495–497
installation 4, 23, 58, 111, 321
installation CLIST 438, 441, 449, 505
INSTANCE 82
INSTEAD OF triggers 6, 137–138, 367
International Components for Unicode 8, 542
International Components for Unicode for DB2 for z/OS 429
Internet Protocol Version 6 69
INTERSECT 6, 152–153, 155, 436
IRLM 10, 119, 123, 127, 426, 453, 461, 468
IS 65, 102, 107, 197, 199, 205, 282, 287, 311, 402, 421, 460
IS_IDENTICAL_TO 487
ISO 164, 212, 237, 274, 460, 489, 529
ISOLATION LEVEL 224
ITERATE 241, 247
IVP 426, 453, 470, 472
IX 86, 127, 274

J

Java 11, 129, 132, 134, 249, 262, 335, 351–352, 426, 429, 438, 487, 503, 523–524
Java stored procedures 438
JCC 199, 259, 351, 523
JDB991K 431
JDBC 16, 19, 203, 258, 335, 351–352, 363, 426, 439, 499, 503, 525
JDBC driver 352, 429, 439, 526
JOIN_TYPE 141

K

KEEP DYNAMIC 224
keyword 12, 16, 55, 74, 313

L

LANGUAGE SQL 222, 225, 231
LARGE 60, 62, 80, 411, 461
large objects 2, 197, 267, 525, 528
LDAP 366
LEAST 470, 473
LENGTH 54, 86, 90–91, 119, 132, 200, 215, 253, 274, 460
let xxvii, 10, 298
LIKE 97, 183, 402
list 26, 57, 59, 68, 93, 123, 126, 138, 143, 155, 184, 206, 216, 273, 277, 346, 378, 391, 398, 419, 425, 432, 434, 438, 511–512
LISTDEF 93, 102, 391, 403, 431
LOAD 18, 15, 49–50, 55, 74, 78, 81, 150, 182, 201, 218, 388, 392, 398, 431, 435, 442–443, 450, 494, 496, 508
LOB 2, 45, 50–52, 76–78, 122, 138, 150, 157, 160, 171, 173, 276, 278, 314, 356, 358, 390, 398, 431, 448, 455, 523–525
LOB column 83, 174, 193–194, 399, 529
LOB data 10, 77, 92–93, 122, 193, 195–196, 523–524, 527
LOB file reference variables 197–199
LOB indicator 194
LOB locator 195–197
LOB locators 194–196, 527
LOB lock 10, 528–530
LOB locking 122, 529
LOB locks 10, 528–529
LOB table 10, 51–53, 77, 93, 122, 151, 168, 174, 194, 390, 407, 409, 486, 489, 530
LOB table space 52–54, 194–195, 390, 409, 486
LOBs 5, 45, 50, 122, 193, 195, 344, 361, 390, 398, 523–525
  allocation 390
  processing 10, 390, 524
  update 528
LOCATE 94, 409–410
locator 10, 195–197, 523, 525, 527
locators 10, 194–196, 523–525
  types 197
locking 7, 80, 122, 192, 400, 494, 529–530
locks 10, 72, 81, 122, 126–127, 178–179, 191, 400, 440, 490, 528–529
LOCKSIZE LOB 528
LOG 50–53, 100, 105–107, 125, 150, 343, 386, 388, 390, 436, 443, 451, 463
Log Truncation 103
LOGAPPLY 101, 117, 386, 388, 397
LOGGED 12, 43, 48–50, 118, 533
logging xxvii, 9, 48, 50, 122, 125, 390, 453
logging attribute 49–51
LOGICAL_PART 118

LPL 54–55, 57, 113
LRSN 53–55, 59, 102–103, 106, 114, 122, 398, 412, 435
lt 271

M

maintenance 4, 14, 45, 230, 421
map page 80
mass delete 6, 80, 150
materialization 195
materialized query tables 112
MAXARCH 480
MAXOFILR 202, 491
MAXTEMPS 445–446, 481
MEMLIMIT 45
MERGE 5, 102, 131, 138–139, 410, 435, 500–501, 509
MERGECOPY 93, 431
MGEXTSZ 488, 491
MGMTCLAS 16, 216–218
Migration 2, 432, 442, 444, 446–447
migration paths 418
Mixed data 237
MODIFY 12, 93, 404, 406, 431
MONTHS_BETWEEN 155, 487
MQT 49, 110
MSTR address space 44, 60, 106, 115
multilevel comparison algorithm 184
Multiple Allegiance 17, 3
MXDTCACH 492, 519

N

namespace 286
Namespaces 284
NaN 134
native SQL stored procedures 18, 129, 222, 248, 487
NCName 278
NEWFUN 479
New-function mode 479
NFM 59, 77, 124, 420–421, 441, 477
NO WLM ENVIRONMENT 422, 439
node 269
NORMALIZE_DECFLOAT 134, 487
NOSYSREC 386
NOT LOGGED 49–51
NOT LOGGED table spaces 53, 56–57
NOT PADDED 488, 494
NOT VARIANT 226
NPI 72, 387–389, 436, 496
NULL 64, 155, 173, 176, 179, 272, 402, 502, 507
NUMPARTS 10, 62, 80, 170, 439

O

OA03194 123
OA13499 28
OA16303 420
OA17314 395
OA18461 xxxiv
OA20045 24
Object 44, 197, 391, 467, 523
ODBC 8, 200, 203, 214, 352, 363, 426
OLDEST_VERSION 92, 118
OPTHINT 224
Optimistic locking 178
optimistic locking 178
optimization 3, 11, 224, 267, 361, 461, 492, 498, 509, 512
Optimization hints 515
Optimization Service Center 8, 429, 440, 448
Optional chargeable features 429
options 10, 13, 55, 76, 333
ORDER 7, 141, 147–148, 188, 279, 300, 434, 467, 534
ORDER BY 13, 147, 149
ORDER BY clause 149–150, 153
OWNER FROM 112

P

PACKAGE OWNER 222
Packaging 541
PADDED 488, 494
page size 9, 62–63, 66, 75, 172–173, 195, 448, 455, 478, 480, 484, 491, 495, 528
Pair-Wise Join 519, 521
Parallel Access Volume 17, 3
PARAMETER STYLE GENERAL WITH NULLS 225
PART 54–55, 57, 63, 74, 77–78, 86, 102, 124, 274, 386, 388, 436, 461
PARTITION 30, 76, 134, 171, 388
partition by growth 10, 81, 487
partition by range 10
Partition Data Set 201
Partition Data Set Extended 201
partition-by-growth table spaces 74–76
PARTITIONED 10
partitioned table space 51, 74, 150
Partitioned-by-Growth 63
Partitioned-by-Range 63
Partitioning 63
partitioning 9, 27, 62–63, 74, 103, 134, 461, 509
partitions 9, 20, 51, 62–63, 74–76, 194–195, 224, 386, 388, 436, 496, 509
PARTLEVEL 403
PBG 10, 63
PBR 10, 63
PDSE data set 437
Performance xxvii, 12, 15, 59, 69, 71, 270, 346–348, 368, 430, 447, 508
performance xxvii, 2, 3, 44–45, 48, 80, 121–122, 126, 129, 132, 150, 191, 195, 263, 266–267, 270, 344, 346, 351, 361, 367, 374–375, 385, 393, 404, 429–430, 434, 487, 493–494
PGFIX 48
PK06616 25
PK10278 201
PK11129 421
PK13287 202
PK19920 16
PK22910 201
PK29281 202
PK43315 332

PORT 261, 447, 466
POSITION 133
POSSTR 133
PQ27578 19
PQ90263 413
precision 17, 4
predicate 8, 132, 293
prefetching 5
prefix 286
prestaging 5
primary key index 173
Private Protocol 344–346, 440, 447
PROGRAM TYPE 225
Progressive Streaming 527
progressive streaming 524–526
PSRBD 76, 114
PT 47
pureXML xxvii, 2, 129, 263

Q

QMF xxvii, 18, 197, 429–430, 457
  Classic Edition 430
  TSO/CICS 430
  WebSphere 20, 430
  Windows 20, 430
QName 295
qualified name 288
QUALIFIER 190, 222, 233
quantiles 19, 401–402
QUANTIZE 134, 487
query 4, 19, 50, 92, 138, 267
Query parallelism 511
query performance 8, 175
QUERYNO 141
QUIESCE 58, 93, 100, 410, 431, 464
Quiet NaN 134

R

RACF 174, 259, 261, 351, 363, 365, 367, 375, 426, 463
RACF profiles 419
RANDOM 176
range-partitioned table space 80, 92
RBA 54–55, 102–105, 398, 411, 435
RBDP 114, 133
READS 71, 231, 233, 491
Real 392, 432, 488–489
REBALANCE 386
REBUILD INDEX 22, 57, 79, 94, 177, 410–411, 413, 431, 508
Rebuild Pending 66
rebuild pending 114
REBUILD-pending 488
RECOVER 15, 49, 54–55, 79, 94, 98, 218, 391, 395, 431, 462, 508
RECOVER-pending 54–55
recovery pending 125
RECP 57, 114
REFRESH TABLE 55
REGENERATE VERSION 230
REGION 188, 261, 467
RELCREATED 236, 488
RELCURHL 440, 447, 484, 490
RELEASE AT 224
remote native SQL procedures 17, 4
RENAME COLUMN 109–111, 215
RENAME INDEX 109, 215
REOPT 223–224, 498–499
reordered row format 64–65
REORG 2, 15, 50, 56–57, 77–79, 180, 182, 218, 379, 386–388, 423, 431, 435–436, 494–495, 508
REORG TABLESPACE 57, 94, 410, 431, 436
REPAIR 94, 113, 408–410, 431
REPAIR VERSIONS 94
REPEAT 133, 178
REPLACE ACTIVE VERSION 230
Replication Center 427
REPORT 32, 61–62, 67, 94, 410, 431
REPORT TABLESPACESET 95
REPORT utility 95
repository 20, 321
requirements 21, 26, 44, 47, 185, 216, 270, 335, 365, 367, 375, 395, 418, 420, 530
RESET 411
RESIGNAL 244, 247
Resource Limit Facility 16, 459, 499
Resource Limit Middleware Table 499–500
resource unavailable 74, 481
RESTART 105–107, 398, 410, 465
RESTORE 15, 103, 106, 391–392, 394, 431, 462
RESTORE SYSTEM 391–392, 462, 491–492
RESTRICT 151
RESYNC 465–466
RETAIN search keyword 16
return 8, 28, 52, 89, 151, 270
REUSE STORAGE 151–152
RID 7, 155, 179, 468, 487, 517, 520
RIDs 68, 519, 522
RLF 16, 440, 447, 459, 499, 503
RLFFUNC 500–501, 503
RMF 16, 29, 45
RMF Distributed Data Server 32
RMF Monitor III 32, 36
RMF Partition Data Report 30
RMF Service Class Periods Report 30
ROLLBACK 58, 151
ROUNDING 225
ROW CHANGE TIMESTAMP 7, 182
row format 13, 63–65, 435
ROWID 143, 145, 168, 171, 174, 486
RRSAF 62, 375, 459
RTS 432, 459, 478, 489
RUN OPTIONS 225
RUNSTATS 89, 379, 392, 401–402, 431, 462, 492, 505–507

S

SAP 3, 11–12, 44, 367, 375
SAP workload 20
SAVEPOINT 58, 345

SBCS 156, 160, 230, 238, 460
SCA 113
SCHEMA 16, 111–113, 189–191, 363, 474
Schema 5, 20, 111, 270
schema 2, 13, 73, 156, 265
schema evolution 2, 9
SCHEMA SWITCH 111
SCOPE PENDING 388
SDSNEXIT 105–106, 453, 472
SDSNLINK 453, 463, 472
SDXRRESL 453, 472
Search 270, 500
Secure Socket Layer 14, 363, 376
SECURITY 55, 225
segmented table space 74, 127, 150–151
SELECT from DELETE 142, 146
SELECT from MERGE 142–143
SELECT from UPDATE 142, 145–146
SELECT SUBSTR 163, 165
SEQ 5
SEQCACH 5
SEQCACH BYPASS 5
SEQCACH SEQ 6
sequential files 201
Server 17, 3, 265, 268, 336, 338, 363, 367, 375–376, 424, 427
SET 58, 94, 98, 102, 125, 137–138, 141, 144, 189–190, 193, 345, 356, 408, 410, 442, 450, 453, 502, 512
setDB2ClientApplicationInformation 503
setDB2ClientUser 503
setDB2ClientWorkstation 503
SETROPTS CLASSACT(DSNR) 420
SETROPTS LIST 419
shared memory object 44
SHOWDSNS 95
shredding 4, 266, 323
SHRLEVEL 51, 57, 79, 101, 386, 388, 436, 509
side 142, 150, 201, 511, 523–524
SIGNAL 244, 247
Signaling NaN 134
simple table space 74, 150–151
singleton 527
SJMXPOOL 491, 517–519
SKCT 47
SKIP LOCKED DATA 191–193, 400
SKPT 47
S-LOB 528–529
S-LOB lock 528, 530
SMF 23, 459
SMF type 70 33
SMF type 72 34
SMF type 79 36
SMP/E 70, 427, 432, 437–438
SMS classes 216–218, 443
SNA 17, 335, 341, 351, 488
SORTDEVT 386, 388
SORTKEYS 398
SORTNUM 388
SOUNDEX 155, 157, 160, 487
sparse index 19, 517–518
Spatial 8, 129, 429, 542
special open 127
SPUFI 141, 197, 203, 231, 252, 282, 302, 489
SPUFI defaults panel 255
SQL 2, 4, 45, 80, 129, 131, 267
SQL Debugger 257, 435
SQL PATH 224
SQL procedures 17, 16, 68, 218
SQL statement 47, 87, 271
SQL statement text 46
SQL stored procedure 218, 231, 242
SQL stored procedures 7, 18, 129, 238, 248, 487
SQL variable declarations 240, 242
SQL/XML 40, 274–275, 283
SQLCA 205
SQLCODE 53, 65, 92, 110, 169, 197, 201, 282, 301, 338–339, 422, 434, 437–438, 444, 447, 501–502, 516
SQLCODE +434 225
SQLCODE -104 231
SQLCODE -4498 359
SQLCOMNT 252
SQLDA 46, 203–205, 434
SQLFORMAT 254
SQLJ 335, 351–352, 358, 426, 439, 525–526
SQLJ applications 439
SQLNAME 204, 434
SQLPL 252
SQLSTATE 240, 245, 282, 301, 434, 437, 485
SQLVAR 204–206
SSID 253, 255, 349, 410
SSL 14, 354, 363, 365, 376, 466
STAR 461, 491
Star schema 517
star schema 19, 19, 517
STARJOIN 518
START DATABASE 85, 87–88, 125
-START/STOP PROCEDURE 251
STATCLUS 392
statement 11, 46, 75, 137, 270
statistics 19, 23, 51, 89, 333
Statistics Clustering 392
STATUS 24, 54, 86, 90–91, 274
STAY RESIDENT 225
STMT 455
STORCLAS 16, 216–218
stored procedure
  REBIND PACKAGE 251
  special registers 249
STORES 521
STOSPACE 410, 432
string value 160, 271
STYPE 53, 87, 89
SUBSTR 10, 133, 141, 163, 165, 177, 198, 523
SYSADM 111, 139, 142, 174, 218, 367, 375, 463, 503
SYSCOLUMNS 215, 348, 436
SYSCOPY 53–54, 82, 84, 87, 101, 390
  STYPE 53
SYSCTRL 139, 175, 218, 503
SYSIBM.SYSCOLUMNS 215, 436
SYSIBM.SYSCOLUMNS_HIST 436

SYSIBM.SYSCOPY 89, 114, 118
SYSIBM.SYSDUMMY1 132–133, 156, 276, 278, 287
SYSIBM.SYSENVIRONMENT 236, 254
SYSIBM.SYSINDEXES 176
SYSIBM.SYSINDEXSPACESTATS 437, 489
SYSIBM.SYSROUTINES 232, 236
SYSIBM.SYSTABLEPART 75
SYSIBM.SYSTABLES 52, 97, 103
SYSIBM.SYSTABLESPACE 51, 53, 75, 77, 80
SYSIBM.SYSTABLESPACESTATS 437, 489
SYSIN 98, 102, 105–106, 116, 254, 388, 404
SYSINDEXSPACESTATS.LASTUSED 122
SYSLGRNX 53, 92–93, 126, 404
SYSOPR 61, 119, 463, 503
SYSPACKAGE 226
SYSPACKSTMT 478
SYSPITR 103, 107
SYSPRINT 105–106, 116, 254
SYSTABLEPART 75
System Assistant Processor 12
System z9 17, 3, 8, 128
System z9 Application Assist Processor 12, 25

T

table space scans 195
tables 4, 19, 49, 73, 127, 137, 266
TABLES_JOINED_THRESHOLD 440, 491
TBNAME 83–84
TCP/IP 7, 14, 44–45, 335, 338, 376, 378, 466, 488
TCP/IP port number 466
TEMP database 432–433, 444, 446
TEMPLATE 404, 410, 431
text node 277
TIME 29, 58, 98, 102, 105, 107, 159, 225, 261, 388, 398, 459–461, 521
TIMESTAMP 7, 106, 179–180, 453, 487
TIMESTAMP_FORMAT 155, 162–163, 487
TIMESTAMP_ISO 156, 162–163, 487
TIMESTAMPDIFF 18, 155, 160, 162
TIMEUSED 14
TOTALROWS 437, 489
trace filtering 15, 367
traces 70, 346, 368, 370, 461, 489
TRACKMOD 63
transitive closure 515
tree structure 289
triggers 6, 81, 92, 112, 136–138, 214, 367, 481
TRUNCATE 5, 55, 80, 150–151, 500–501
trusted connections tracing 368
TS 54, 86, 90–91, 274, 404
TYPE 26, 54, 61, 70, 75, 80, 83–84, 197, 199, 205, 274, 292–293, 442, 450, 459, 464, 521
Type 2 199, 352
Type 4 199, 352–353, 523–524

U

UDFs 15, 443, 475
UK10146 437
UK25744 332
UK90008 421
UNICODE 156, 459–460
Unicode 2, 4, 132, 160, 183, 185, 272, 276, 278, 306, 422, 429
Unicode Collation Algorithm 185, 188
Unified Debugger 257, 260, 435
UNION ALL 149, 153–154, 515
UNIQUE 173, 177, 313
unique key index 173
Universal Driver 523–524
Universal Table Space 10
universal table space 62, 74, 150–151
UNLOAD 18, 96, 192, 201, 388, 399, 432
unused indexes 122
UPDATE 6, 12, 43, 45, 112–113, 137, 179–181, 344, 356, 367, 370, 441–442, 450, 461, 464, 500–502
UPPER and LOWER 189
URI 286
USAGE 197
USING % 32
USING VCAT 112
UTF-16 186
UTF-8 44, 178, 222, 231, 271–272, 276, 460
UTRW 415
UTS 10, 63, 74

V

VALIDATE RUN 224
VALIDPROC 55, 152, 435
VALUE 182, 282, 455, 466, 505
VALUES 64, 110, 137, 141, 144, 181, 236, 277, 282, 434
VARBINARY 5, 132–133, 155, 188, 193, 355, 358
VARCHAR 64, 66, 110, 132–133, 137, 176, 178, 181, 266, 269, 315, 435–436, 461, 524
VARCHAR_FORMAT 156, 163–164, 487
VARGRAPHIC 188, 214, 435–436
variable 13, 45, 132, 282
VCAT SWITCH 112
VERSION 29, 68, 222, 226, 434, 442, 450, 469–470, 472
Version xxvii, 1, 4, 63, 109, 131, 221, 267–268, 335–336, 344, 365, 376, 395, 417, 423, 426, 429, 433–434, 512, 526
versions 3, 94, 188, 225, 389, 394, 396, 418, 486, 489, 503
Virtual Shared Object 44
Virtual storage constraint 10, 44
VPSIZE 48
VTAM 342, 351, 465

W

WebSphere 15, 265, 363, 367, 375, 430
well-formed XML 270
whitespace 271, 283
WITH 67, 102, 138, 203–204, 355, 378, 421, 440, 447, 468, 484, 490, 528
WITH IMMEDIATE WRITE 224
WLM 17, 3, 47–48, 60, 123, 222–223, 225, 422, 438–439, 466, 530–532
WLM ENVIRONMENT 223, 251, 422, 466

WLM ENVIRONMENT FOR DEBUG MODE 222
WLM-managed stored procedures 439
workfile 15, 438, 445–446, 481
write performance 60, 123
write-down 152

X

XES 123
X-LOB 529
X-LOB lock 529–530
XML 11, 51, 53, 56, 76–77, 129, 138, 150–151, 153, 171, 178, 263, 265–266, 355–356, 358, 422, 425–426, 430, 433, 523, 525–526
XML column 204, 206, 271–272
XML columns 202–204, 274, 280, 300
XML data model 267, 269
XML data type 4, 265, 270, 425
XML documents 4, 265, 268, 523
XML Extender 4, 263, 266–267, 425
XML index 283
XML publishing functions 268, 275
XML schema 283, 321, 443
XML schema repository 321
XML support 2, 263, 267, 356
XML System Services 12, 420
XMLAGG 275, 279
XMLATTRIBUTES 275
XMLCOMMENT 276–277
XMLCONCAT 275, 279
XMLDOCUMENT 276–277
XMLELEMENT 275, 277, 279
XMLEXISTS 276, 283, 303, 305
XMLFOREST 275
xmlns 282
XMLPARSE 271, 276, 280
XMLQUERY 276, 283, 287
XMLSERIALIZE 271, 276–277
XMLTEXT 276, 278
XPATH 267, 275–276
XPath 4, 284–286
XPath expression 285, 287–288
XQUERY 268
XQuery 4, 275
XSR 321

Z

z/Architecture 17, 3, 44
z/OS 43–45, 73, 106–107, 123, 131, 188, 275, 335–336, 338, 376, 391, 395, 417, 419–420, 423, 493, 498
z/OS 1.7 2, 16, 60
z/OS Application Connectivity 426, 429
z/OS Enablement 427
z800 8
z890 8, 420
z900 8, 371
z990 8, 371, 420, 495
zIIP 7, 3–4, 12, 68
zIIP/K 16
zSeries 12, 353

Archived

Redbooks

DB2 9 for z/OS Technical Overview

IBM.COM/REDBOOKS

Archived

Archived

DB2 9 for z/OS
Technical Overview

Redbooks

Read about the new features and functions of DB2 Version 9.1 for z/OS

Understand prerequisites and migration process

Evaluate applicability and major returns

IBM DATABASE 2 Version 9.1 for z/OS (DB2 9 for z/OS throughout this publication) is the thirteenth release of DB2 for MVS. It brings improved synergy with the zSeries hardware and more opportunities to drive business value in the following areas:

- Business insight innovations
  Rich hybrid support for relational and pure XML data.
  Enhanced support for key business partners that allow you to get more from your data in critical business disciplines like ERP.
  A brand new user interface for the Query Management Facility to view and analyze data from either a workstation client or an ordinary Web browser.
- Cost savings through optimized innovations
  Streamlined security and regulatory compliance through the implementation of roles, network-trusted contexts, and enhanced auditing.
  Performance-boosting innovations such as load and reorg CPU reductions, improved varying length data, logging, and insert performance.
  Query management enhancements to make accessing data even faster and more accurate with indexing improvements.
- Business resiliency innovations
  Database on demand capabilities to ensure that information design can be changed dynamically, often without database outages.
  DB2 restart improvements enhancing performance usability and availability in data sharing environment.

This book introduces the enhancements made available with DB2 9 for z/OS. Its helps you understand the new functions, justify investment in migrating, and plan for exploiting the key new capabilities.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7330-00 ISBN 0738489131