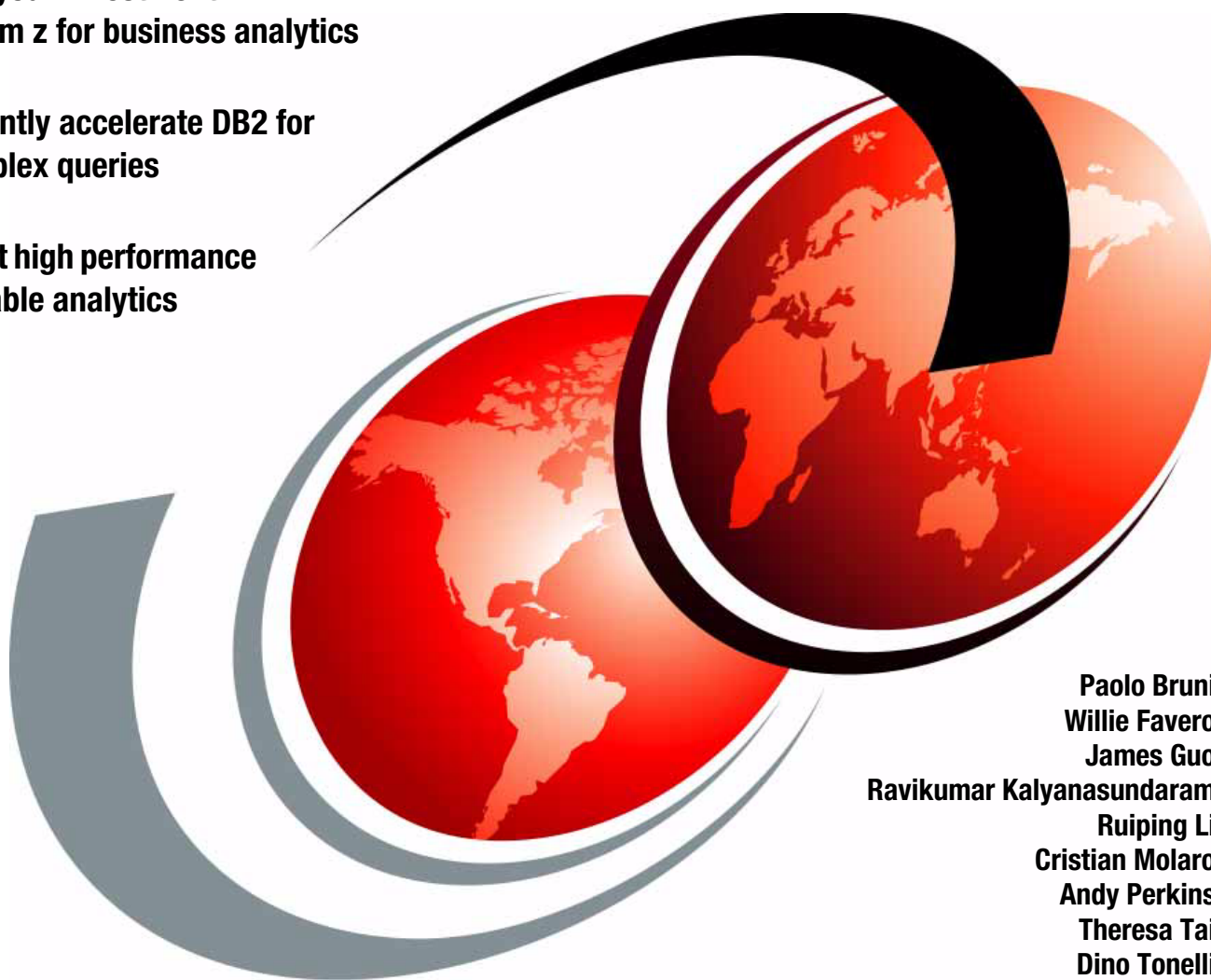


Hybrid Analytics Solution using IBM DB2 Analytics Accelerator for z/OS V3.1

Leverage your investment in
IBM System z for business analytics

Transparently accelerate DB2 for
z/OS complex queries

Implement high performance
and available analytics



Paolo Bruni
Willie Favero
James Guo
Ravikumar Kalyanasundaram
Ruiping Li
Cristian Molaro
Andy Perkins
Theresa Tai
Dino Tonelli

Redbooks



International Technical Support Organization

**Hybrid Analytics Solution using IBM DB2 Analytics
Accelerator for z/OS V3.1**

October 2013

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

First Edition (October 2013)

This edition applies to Version 3, Release 1 of IBM DB2 Analytics Accelerator for z/OS (program number 5697-DAA).

© Copyright International Business Machines Corporation 2013. 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	ix
Tables	xiii
Examples	xv
Notices	xvii
Trademarks	xviii
Preface	xix
Authors	xix
Now you can become a published author, too!	xxi
Comments welcome	xxii
Stay connected to IBM Redbooks	xxii
Chapter 1. The analytics lifecycle	1
1.1 Business analytics lifecycle	2
1.2 Lifecycle simplification	4
1.3 Overview of the solutions	8
Chapter 2. Using the Studio client to manage the environment	11
2.1 DB2 Analytics Accelerator Studio V3.1 overview	12
2.1.1 New features	12
2.1.2 Hybrid solution overview	12
2.1.3 Data Studio client and web console architecture	14
2.2 Installing IBM DB2 Analytics Accelerator Studio	15
2.2.1 Installing Accelerator Studio using the product DVD	16
2.2.2 Data Studio full client download and installation	17
2.2.3 Accelerator plug-ins only installation	23
2.2.4 Dump file on Linux	23
2.2.5 Accelerator Studio Help menu	23
2.3 Configure Studio for tuning	25
2.4 Accelerator Studio tasks	26
2.4.1 Summary of common tasks and the overall process flow	26
2.4.2 Sample screen captures on various activities from the Studio	28
2.4.3 How to set the automatic refresh frequency on the Studio	43
2.4.4 Tracing	44
2.4.5 Tasks new to Accelerator Version 3.1	46
2.5 Stored procedures used as administrative interface	46
2.5.1 Components used by stored procedures	47
2.5.2 DB2 supplied stored procedures used by Studio	48
2.5.3 DB2 Accelerator stored procedures used by Studio	48
2.6 Stored procedure security	51
2.7 Data Studio web console	53
2.7.1 Install and configure the Data Studio web console component	53
Chapter 3. Data latency management	55
3.1 Methods of copying data from DB2 to the Accelerator	56
3.2 Understanding the load process	58
3.2.1 Load streams	59

3.2.2 Loading partitions in parallel	60
3.2.3 Combining non-partitioned and partitioned table loads.	61
3.2.4 Data refresh	62
3.3 Load performance considerations.	66
3.3.1 Network considerations.	66
3.3.2 Source considerations	67
3.3.3 Target considerations	79
3.4 Automating the load process.	79
3.5 Summary.	86
Chapter 4. Accelerator resource management.	87
4.1 Accelerator resource management	88
4.2 Accelerator prioritization within a DB2 subsystem	89
4.2.1 Query prioritization	89
4.2.2 Accelerator data maintenance task prioritization	90
4.2.3 Accelerator prioritization summary table.	93
4.3 Experiments: Prioritization within a DB2 subsystem	94
4.3.1 Test environment and measurement preparation	94
4.3.2 Test scenario A: Prioritizing one user (query) higher than the others.	100
4.3.3 Test scenario B: Query prioritization as concurrency increases	102
4.3.4 Test scenario C: Mixed workload query and Accelerator table loads	105
4.4 Allocating resources with workload isolation for a shared Accelerator	106
4.4.1 Workload isolation.	107
4.4.2 Procedure for setting the workload isolation resource limits.	107
4.5 Workload isolation experiments	109
4.5.1 Test environment and data collection procedures	110
4.5.2 Key observations	111
4.6 Accelerator resource management checklist	112
4.6.1 Resource management checklist	112
4.6.2 Details referenced in resource management checklist	114
Chapter 5. Query acceleration management	121
5.1 Query acceleration processing flow	122
5.2 Query acceleration settings and enablement	123
5.2.1 DSNZPARM settings for accelerated queries.	123
5.2.2 CURRENT QUERY ACCELERATION special register.	126
5.2.3 ODBC and JDBC application query acceleration	127
5.3 Query acceleration criteria	127
5.3.1 The query types that are supported by the Accelerator	129
5.3.2 The query functionality limitations.	129
5.4 How IBM DB2 Analytics Accelerator handles correlated subqueries	133
5.4.1 Background of the correlated subqueries	133
5.4.2 Correlated subqueries processing in the Accelerator	134
5.4.3 Circumventions	135
5.5 INSERT from subselect query acceleration processing	136
5.6 Profile tables	138
5.7 Accelerator EXPLAIN support.	139
5.8 DB2 Analytics Accelerator hardware considerations.	141
Chapter 6. High-Performance Storage Saver	143
6.1 High-Performance Storage Saver design principles	144
6.1.1 HPSS overview	145
6.1.2 Current restrictions	149
6.2 Archive process and operations	149

6.2.1 Prepare for archiving partitions and tables	150
6.2.2 Batch stored procedure invocation	156
6.3 Query processing	157
6.3.1 Query execution on the Accelerator with HPSS online archive	157
6.3.2 Query execution on Accelerator without HPSS online archive	158
6.3.3 Query execution on DB2 without HPSS online archive.	158
6.3.4 Query acceleration controls	159
6.4 Monitoring and instrumentation.	159
6.5 Restore archived partition from the Accelerator	160
Chapter 7. Incremental update.	161
7.1 Overview of incremental update	162
7.2 Incremental update architecture	163
7.3 Installation and configuration	165
7.3.1 Installing the System z components for incremental update.	165
7.3.2 Installing the Accelerator components for incremental update	166
7.4 Defining incremental update between a DB2 subsystem and the Accelerator	166
7.5 Defining tables to incremental update.	168
7.6 Access server	169
7.7 z/OS capture agent.	170
7.7.1 Reading the DB2 log.	170
7.7.2 Staging changes	171
7.7.3 Capture agent considerations	171
7.8 Accelerator target agent	172
7.8.1 Staging changes	172
7.8.2 Applying changes	172
7.8.3 About delete processing	173
7.8.4 About latency	174
7.9 Incremental update scenarios	177
7.9.1 Insert only replication	177
7.9.2 Mass insert replication	178
7.9.3 Delete only replication.	179
7.9.4 Mass delete replication	180
7.9.5 Update only replication	181
7.9.6 Multi-table replication	182
7.9.7 Concurrent query with replication	183
7.10 Summary.	186
Chapter 8. Impact of new Netezza hardware and software components.	187
8.1 Hardware and software evolution	188
8.2 PureData System for Analytics N2001	188
8.2.1 PureData for Analytics benefits.	189
8.2.2 New hardware for PureData System for Analytics	191
8.2.3 Scan speed	192
8.2.4 The S-Blade components	193
8.2.5 Available PureData Systems for Analytics Models	194
8.3 Netezza Performance System 7.0.2	195
8.3.1 Directed data processing	195
8.3.2 Page granular zone maps.	197
8.4 System 9700 and 9710	198
8.5 Query performance measurements	198
8.5.1 Scenario #1: I/O bound query	199
8.5.2 Scenario #2: CPU bound query	201

8.5.3 Scenario #3: Mixed: I/O and CPU bound query	202
8.5.4 Scenario #4: Selective query with equal predicates benefitting from zone maps	204
8.5.5 Scenario #5: Directed data processing	205
8.5.6 Query performance summary	206
Chapter 9. Monitoring DB2 Analytics Accelerator environments	207
9.1 Monitoring the environment	208
9.1.1 The DB2 message DSNX881I	219
9.1.2 DB2 commands	226
9.1.3 Monitoring the Accelerator using SYSPROC.ACCEL_GET_QUERY_DETAILS stored procedure	229
9.2 Monitoring low latency updates	234
9.3 Reporting and monitoring using OMPE	238
9.3.1 Batch reports	239
9.3.2 OMPE classic interface	240
9.3.3 Online monitoring using OMPE GUI	244
9.3.4 The Accelerator support in the OMPE Performance Database	249
9.3.5 The Accelerator support using the new OMPE Spreadsheet Data Generator	259
9.4 Cataloging a DB2 for z/OS database as an ODBC data source	260
Chapter 10. IBM zEnterprise Analytics System 9700	267
10.1 Introduction	268
10.1.1 zEnterprise Analytics System 9700 configuration	268
10.1.2 System 9710 configuration	273
10.2 Architectural overview	274
10.2.1 zEnterprise Analytics System 9700	274
10.2.2 zEnterprise Analytics System 9710	276
10.2.3 IBM System Storage DS8870	276
10.3 Hardware specification	277
10.4 Software overview	278
10.4.1 System z software for 9700	278
10.5 Network specification	278
10.6 Optional software components overview	279
10.6.1 Data Analytics Pack	279
10.6.2 Data Integration Pack	281
10.6.3 Fast Start Services Pack	283
10.6.4 DB2 Connect	283
Chapter 11. High availability considerations	285
11.1 Loading data on multiple Accelerators for high availability	286
11.1.1 Shared Accelerator considerations	286
11.1.2 Disaster recovery considerations	286
11.2 High availability configuration with a hot standby Accelerator	286
11.2.1 Option 1: Both the Accelerators in the same physical location	288
11.2.2 Option 2: Accelerators not in the same physical location	289
11.2.3 Observations from a simulated failover	290
11.3 CDC considerations for high availability	292
11.3.1 High availability setup for the incremental update capture engine	292
11.3.2 Accelerator coordinator node failover considerations	296
Appendix A. Accelerated queries	299
A.1 Query CPU bound	300
A.2 Selective query	302

A.3 Other CPU bound queries	304
Appendix B. Notes on maintenance	307
B.1 Preventive maintenance	308
B.1.1 Studio	308
B.1.2 DB2	308
B.1.3 IBM DB2 Analytics Accelerator for z/OS	308
B.1.4 Change Data Capture	309
B.1.5 OMEGAMON for DB2 Performance Expert	309
B.2 What to do if you encounter an Accelerator problem	310
B.2.1 Accelerator trace	310
B.2.2 FTPing the Accelerator trace to IBM support	312
B.3 Recent fixes	313
Appendix C. DSNZPARM parameters setting for data warehouse	315
C.1 Suggested DSNZPARM parameters setting	316
C.1.1 Parallelism	317
C.1.2 Query processing and diagnostics	320
C.1.3 Disk storage allocation	322
C.1.4 Storage allocation	323
C.1.5 Star join processing	327
C.1.6 Utility sort processing	329
C.1.7 Query Accelerator	330
C.1.8 LOB specific keywords	332
C.1.9 XML specific keywords	333
C.1.10 Miscellaneous system parameters	334
C.1.11 Deprecated in DB2 10	336
C.1.12 Pre DB2 9 DSNZPARM parameters removed, but still documented	337
Appendix D. Tools for Accelerator	339
D.1 Tools	340
D.2 OMEGAMON XE for DB2 Performance Expert on z/OS	341
D.3 DB2 Query Monitor for z/OS	341
D.4 InfoSphere Optim Query Workload Tuner for DB2 for z/OS	341
D.5 InfoSphere Optim Configuration Manager	342
Appendix E. Additional material	343
Locating the Web material	343
Using the Web material	343
System requirements for downloading the Web material	344
Downloading and extracting the Web material	344
Related publications	345
IBM Redbooks	345
Other publications	345
Online resources	346
Help from IBM	347
Index	349

Figures

1-1 Data lifecycle.	2
1-2 Platform direction	5
2-1 Benefits of the hybrid analytics solution	13
2-2 IBM DB2 Analytics Accelerator Product Components Overview	14
2-3 Data Studio and web console installation scenario: Architecture	15
2-4 Studio installation: Initial product component selection window	18
2-5 Studio installation: License agreement window	19
2-6 Studio installation: Location for Installation Manager and shared resources	20
2-7 Studio installation directory name entry window	20
2-8 Studio installation: Language selection.	21
2-9 Studio installation progress screen capture	21
2-10 Studio installation: Completion window.	22
2-11 Studio Help menu for beginners	24
2-12 Navigating the Help menu to Studio documentation and demonstration videos.	24
2-13 Studio: Create missing EXPLAIN tables including DSN_QUERYINFO_TABLE.	25
2-14 Typical process flow of tasks performed on Accelerator Studio	28
2-15 Creating a new connection profile from a Data Source Explorer	29
2-16 Connection parameter entry panel	29
2-17 Creating a new connection profile from Administration Explorer	30
2-18 Start Accelerator from Studio: Administration Explorer.	34
2-19 Starting an Accelerator: Confirmation window	34
2-20 Accelerator status after starting it from Studio	35
2-21 Stopping an Accelerator: Confirmation window	35
2-22 Stopping an Accelerator: Force option from the Studio	35
2-23 Studio Accelerator view to perform Add Tables task	36
2-24 Defining tables to an Accelerator	37
2-25 Studio: Load button on the Accelerator folder of the Administration Explorer.	38
2-26 Load activity: Single table with multiple partitions selected.	38
2-27 Load activity: Multiple tables selected in one shot	39
2-28 Load operation: CPU utilization on coordinator versus worker nodes	40
2-29 SDSF active queue during load activity of a single table with 10 parallel unloads	40
2-30 Enabling or disabling table level query acceleration	41
2-31 Studio: Display Accelerator Status panel	42
2-32 Preferences selection from the main menu	43
2-33 Accelerator Studio preferences window to set refresh interval and skew threshold	44
2-34 Studio - Accelerator view: Tracing activity	44
2-35 Configuring Accelerator trace: Selecting default profile	45
2-36 Configuring trace profile: Stored procedure tracing profile	46
2-37 System overview diagram.	47
2-38 Components used by Accelerator stored procedures.	48
3-1 Accelerator method decision tree	58
3-2 Loading a partitioned table containing five partitions	60
3-3 All partitions must be loaded for the initial load.	61
3-4 Last Load status	62
3-5 Detected changes are shown in the load dialog	64
3-6 Example of change detection not being able to determine if the table changed.	65
3-7 Four source tables	68
3-8 Running two parallel load streams	69

3-9	Running six parallel load streams	70
3-10	Creating parallel load streams with multiple ACCEL_LOAD_TABLES procedure calls	71
3-11	Load throughput varying number of System z processors and parallel streams	73
3-12	Number of parallel load streams versus available resources	74
3-13	Comparing load throughput for N1001-101 and N2001-010	75
3-14	Load study workloads	75
4-1	DB2 Analytics Accelerator Configuration Console main menu	91
4-2	Run Accelerator Functions menu	91
4-3	Select the DB2 subsystem	92
4-4	Menu for setting the priority of the Accelerator maintenance tasks	92
4-5	Accelerator resource management controls within a single DB2 subsystem	94
4-6	WLM service classes used for query priority experiments	97
4-7	Classification rules for query prioritization experiments	98
4-8	Display thread results utilizing WLM_SET_CLIENT_INFO	99
4-9	RMF Monitor III display of end user BUSRH001's enclave	100
4-10	Test case QPusr5A DB2 Class 1 elapsed times by CLIENT_ENDUSER	101
4-11	Query prioritization: One user is favored over the rest	102
4-12	Test Scenario B: The impact of concurrency on query prioritization	104
4-13	Test Scenario C: Mixed workload results	106
4-14	DB2 Analytics Accelerator Configuration Console main menu	108
4-15	DB2 Analytics Accelerator: Run Accelerator Functions menu	108
4-16	Minimum resources menu	109
4-17	Example of changing DZA1DDF's minimum resource allocation	109
4-18	Workload isolation test scenario plot of Accelerator CPU time consumption	111
4-19	Accelerator stored procedures WLM service class	115
4-20	Classifying batch Accelerator load jobs	116
4-21	Classifying remote (DDF) calls of the Accelerator stored procedures	116
4-22	Classification of the InfoSphere CDC agent	118
4-23	Residency WLM service classes serving DDF queries	119
4-24	Residency WLM service classes serving local (batch reporting) queries	120
5-1	Query execution flow	122
5-2	DB2 Analytics Accelerator accelerate criteria	128
5-3	Access plan diagram of the query that is not accelerated	137
5-4	Access plan diagram of the query that is accelerated	137
5-5	Linear relationship between response time and N1000 appliance models	141
5-6	Systems and sizes for N1001	142
6-1	95% of host disk space are used for historical data	144
6-2	Archived partitions reside in the Accelerator and no longer in DB2	145
6-3	Move partitions to Accelerator	146
6-4	CALL stored procedure ACCEL_ARCHIVE_TABLES to trigger archive processing	146
6-5	HPSS creates new image copies	147
6-6	Pruning archived partitions in DB2	148
6-7	HPSS notices: Moved partitions must NOT receive any inserts, updates, or deletes in DB2 after moving	150
6-8	Storage Saver: Move Partitions to Accelerator function	151
6-9	New Storage Saver tag enables Move Partitions to Accelerator operation	152
6-10	Partitions 8 - 16 have been selected for archive into the Accelerator	153
6-11	Archive: Move partition in progress	154
6-12	Partitions 8 - 16 have been loaded into the Accelerator	155
6-13	Archive store procedure: Partitions 8 - 16	156
6-14	Archiving the additional partition 17 to the Accelerator	156
6-15	Archiving selected groups of partitions	157

6-16	Query execution with and without archive data	158
7-1	Incremental update architecture	163
7-2	Capture point reflected in the Replication Since column of the Accelerator Studio . .	169
7-3	Graphic representing pattern of source change captures and target apply changes .	173
7-4	Chart of average latency	176
7-5	Sample data from a table that is populated by the LATMONRX REXX program . . .	177
7-6	Operations per second for source and target agents: inserts only	178
7-7	Operations per second for source and target agents - mass inserts	179
7-8	Operations per second for source and target agents: Deletes only	180
7-9	Operations per second for source and target agents: mass deletes	181
7-10	Operations per second for source and target agents: updates only	182
7-11	Operations per second for source and target agents: two table replication	183
7-12	Replication result with concurrent query to different table	185
7-13	Impact on latency with concurrent query to a different table	185
8-1	Benefits of the new PureData for Analytics N2001	188
8-2	The N2001 architecture	189
8-3	The major contributors	191
8-4	N2001 hardware	192
8-5	Hardware speed	193
8-6	S-Blade components	193
8-7	Models of IBM PureSystems for Analytics	194
8-8	N2001 systems	195
8-9	NPS7 enhancement: Directed data processing	196
8-10	NPS enhancement: Page granular zone maps	197
8-11	IBM zEnterprise Analytics System 9700	198
8-12	Concurrent I/O bound queries on N2001	199
8-13	Concurrency tests for I/O bound queries on N2001 versus N1001	200
8-14	Comparing N2001 versus N1001 with concurrency test	201
8-15	Scalability of concurrent queries on N2001	202
8-16	Mixed: I/O and CPU bound query: N2001 versus N1001	203
8-17	Selective query with equal predicates: N2001 versus N1001	204
8-18	Directed data processing scenario	205
9-1	The DB2 statistics traces and the Accelerator	209
9-2	The DB2 Accounting trace and the Accelerator	213
9-3	IBM Data Studio monitoring queries	226
9-4	Selecting ACCEL_CONTROL_ACCELERATOR stored procedure in DataStudio . .	234
9-5	Run ACCEL_CONTROL_ACCELERATOR stored procedure windows	235
9-6	Browsing the command XML tree	235
9-7	Browsing the ACCEL_CONTROL_ACCELERATOR XML message file	236
9-8	Monitoring the execution of ACCEL_CONTROL_ACCELERATOR in Data Studio . .	236
9-9	ACCEL_CONTROL_ACCELERATOR RESULT example	237
9-10	OMPE Client product version information	245
9-11	OMPE GUI Statistics Details option	246
9-12	OMPE GUI Accelerators section of Statistics Details panel	247
9-13	OMPE GUI Accelerator Details panel	248
9-14	OMPE GUI Accelerator Status panel	249
9-15	How performance data is formatted and loaded into the OMPE database	250
9-16	The OMPE Accounting tables	250
9-17	The OMPE Statistics tables	252
9-18	Selecting a DB2 ODBC data source	256
9-19	Connect to DB2 for z/OS ODBC data source	256
9-20	Select source table dialog	257
9-21	Reporting accounting data using a pivot table	258

9-22	Drop, load, and refresh an accelerated table scenario	259
9-23	Reported CPU by hour	260
9-24	Opening the DB2 Command Line Processor in Windows.	261
9-25	DB2 CLP example	262
10-1	9700 core configuration	272
10-2	9710 core configuration	273
11-1	High availability configuration using two Accelerators in 2-way data sharing group.	287
11-2	High availability configuration with both the Accelerators in the same location	288
11-3	High availability/disaster recovery configuration with the Accelerators in two different locations	289
11-4	Failover scenario with primary Accelerator offline and backup Accelerator started	290
11-5	-DIS ACCEL DET output with Accelerator offline, but connection is healthy	291
11-6	High availability setup for the incremental update capture engine	293
B-1	Studio: Save trace options	312
D-1	Available DB2 for z/OS tools for IBM Analytics Accelerator.	340

Tables

2-1	DB2 stored procedures to administer the Accelerator function.	48
2-2	Accelerator stored procedures to administer Accelerator function	50
2-3	Stored procedure access privileges	51
3-1	Comparison of Accelerator refresh options.	56
4-1	Mapping of WLM Importance to Accelerator priorities	89
4-2	Relating the maintenance operation options to WLM Importance	93
4-3	DEFAULT priorities of maintenance tasks	93
4-4	9700 test configuration for resource management experiments.	95
4-5	Measurement data collection methodology	95
4-6	WLM_SET_CLIENT_INFO stored procedure mapping	98
4-7	Test scenario: Query prioritization test cases.	101
4-8	Test scenario B: Query prioritization as concurrency increases.	103
4-9	Test scenario C: Mixed workload: Query versus Accelerator table loads.	105
4-10	Workload isolation test scenario sequence.	110
4-11	Accelerator resource management checklist	112
5-1	List of DB2 scalar functions supported in DB2 Analytics Accelerator.	132
5-2	PLAN_TABLE	138
5-3	DSN_QUERYINFO_TABLE	138
5-4	DSN_PROFILE_ATTRIBUTES table	139
7-1	Query performance with no replication	184
7-2	Comparison of query metrics without and with concurrent replication	186
9-1	hlq.TKO2SAMP PDS members used to create OMPE database tables.	250
9-2	hlq.TKO2SAMP PDS members used to create OMPE Performance Database Statistics Tables.	252

Examples

2-1	Command used to Telnet to the DB2 Analytics Accelerator console	31
2-2	Password entry panel to enter the DB2 Analytics Accelerator configuration console. . .	31
2-3	DB2 Analytics Accelerator Configuration console: Welcome panel	31
2-4	DB2 Analytics Accelerator configuration console: Accelerator functions panel	32
2-5	SQL to find the Accelerator version: compatibilityLevel attribute	49
2-6	Sample output from the start Data Studio web console command: Linux.	53
2-7	Data Studio web console server log: Partial listing showing the URL.	54
3-1	Setting the MNSPAS parameter for the DSNUTILU stored procedure.	72
3-2	SQL to determine the unload size of tables	76
3-3	JCL to remove, add, load, and enable a table for acceleration.	81
3-4	Calling an Accelerator stored procedure.	84
3-5	REXX code snippet calling ACCEL_REMOVE_TABLES	85
4-1	Setting up the DB2 and DB2 Analytics Accelerator environment with commands	96
4-2	Syntax for invoking the WLM_SET_CLIENT_INFO stored procedure	98
4-3	Sample WLM_SET_CLIENT_INFO call in our test script	98
4-4	Error in WLM address space: -471	117
4-5	Failure as reported in the DB2 Analytics Accelerator GUI	117
5-1	Simple INSERT statement that is eligible for DB2 Analytics Accelerator offload	136
7-1	Suggested configuration values	165
7-2	Configuration Console dialog for enabling incremental update	167
7-3	<replicationInfo> section of result message from the getAcceleratorInfo function . . .	175
9-1	OMPE statistics report layout long command example.	209
9-2	OMPE Statistics Long report showing the Accelerator section.	210
9-3	OMPE Statistics report showing DB2 Analytics Accelerator command counters	211
9-4	OMPE Record Trace report command	211
9-5	OMPE Record Trace report output	212
9-6	OMPE Accounting LAYOUT(LONG) command	213
9-7	OMPE Accounting Report Long Accelerator section	214
9-8	OMPE Accounting LAYOUT(ACCEL) command	214
9-9	OMPE Accounting Layout Accel.	214
9-10	OMPE Accounting report, distributed activity section	215
9-11	OMPE SQL DCL block	216
9-12	High DB2 Not Accounted time when off-loaded to DB2 Analytics Accelerator	217
9-13	OMPE Accounting Trace Distributed activity section	217
9-14	OMPE Accounting Trace Accelerator activity section.	218
9-15	DSNX881I message structure	219
9-16	DSNX881I message in system log	220
9-17	DSNX881I in DB2 syslog	220
9-18	DSNX881I in DB2 syslog: NPS initialization and status change.	221
9-19	SYSACCEL.SYSACCELERATORS	222
9-20	SYSIBM.LOCATIONS.	222
9-21	SYSIBM.IPNAMES	222
9-22	-DIS ACCEL DETAIL command output example	222
9-23	TCP/IP CONVERSATION FAILED reported in system log.	223
9-24	TCP/IP CONVERSATION FAILED reported in DB2 MSTR log	223
9-25	DIS ACCEL output example	224
9-26	DIS ACCEL command showing 100 active requests	224
9-27	Job output	225

9-28	DB2 MSTR address space	225
9-29	Displaying Accelerators status using commands	226
9-30	DISPLAY ACCEL(*) output example	227
9-31	DISPLAY ACCEL DETAIL output example.	227
9-32	DISPLAY ACCEL DETAIL output sample showing activity.	228
9-33	Syntax for calling SYSPROC.ACCEL_GET_QUERY_DETAILS procedure.	229
9-34	Calling SYSPROC.ACCEL_GET_QUERIES from REXX.	230
9-35	ACCEL_QUERIES	233
9-36	ACCEL_CONTROL_ACCELERATOR command file	235
9-37	ACCEL_CONTROL_ACCELERATOR message file	235
9-38	ACCEL_CONTROL_ACCELERATOR RESULT showing latencyInSeconds.	237
9-39	ACCEL_CONTROL_ACCELERATOR RESULT replicationInfo state="ERROR"	237
9-40	Communication errors during replication failure	238
9-41	Replication failures in DB2 MSTR log.	238
9-42	OMPE SYSPARMS command example	239
9-43	OMPE SYSPARMS report	239
9-44	OMPE classic interface, real-time menu.	240
9-45	OMPE Classic interface, Thread activity panel.	241
9-46	OMPE Classic Thread detail.	241
9-47	OMPE Classic Thread Accelerator Detail panel	242
9-48	OMPE Classic resource managers main menu	242
9-49	OMPE Classic Accelerator Summary panel	243
9-50	OMPE Classic Accelerator Detail panel	243
9-51	OMPE Classic Accelerator Detail panel, part 2.	244
9-52	OMPE Classic Accelerator Detail panel, part 3.	244
9-53	-DIS GROUP output example	245
9-54	-DIS ACCEL DETAIL command	248
9-55	DDL in DGOACFXC, Accounting Table for data of category "ACCELERATOR"	251
9-56	DDL in DGOSCXCL, Statistics Accelerator Table	252
9-57	Loading the OMPE Performance Warehousing tables	253
9-58	Querying DB2PMFACCT_GENERAL and DB2PMFACCT_ACCEL tables	254
9-59	Creating a view with General and Accelerator accounting information.	255
9-60	-DIS DDF output example	261
9-61	DB2 catalog TCP/IP node example	262
9-62	DB2 catalog TCP/IP node output example	262
9-63	DB2 Terminate output example	262
9-64	DB2 list node directory command	263
9-65	DB2 catalog database command example	263
9-66	DB2 catalog database command output example	263
9-67	DB2 list database directory command output example.	263
9-68	Connect to a DB2 for z/OS database using the CLP	264
9-69	DB2 catalog ODBC data source command example	264
9-70	DB2 catalog ODBC data source command output example.	264
9-71	DB2 LIST ODBC DATA SOURCES command example	265
11-1	Sample DSNX881I message from the MSTR address space during failure.	291
11-2	System messages for a failover from Accelerator host 1 to host 2.	296
A-1	CPU bound query	300
A-2	Selective query with equal predicates	302
A-3	CPU bound query Q1B used for workload isolation	304
A-4	CPU bound query Q8B used for workload isolation	306

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 grant 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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites 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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

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

BigInsights™	InfoSphere®	Resource Measurement Facility™
CICS®	OMEGAMON®	RETAIN®
Cognos®	Optim™	RMF™
CPLEX®	OS/390®	SPSS®
DataStage®	Parallel Sysplex®	System Storage®
DB2 Connect™	POWER7®	System z®
DB2®	POWER®	Tivoli®
Distributed Relational Database Architecture™	PureData™	Velocity™
DRDA®	PureSystems™	WebSphere®
HiperSockets™	pureXML®	z/OS®
IBM PureData™	QMF™	z/VM®
IBM®	QualityStage®	z9®
ILOG®	RACF®	zEnterprise®
IMS™	Redbooks®	
	Redbooks (logo)  ®	

The following terms are trademarks of other companies:

Netezza, NPS, TwinFin, and N logo are trademarks or registered trademarks of IBM International Group B.V., an IBM Company.

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

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

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

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

Preface

The IBM® DB2® Analytics Accelerator Version 3.1 for IBM z/OS® (simply called *Accelerator* in this book) is a union of the IBM System z® quality of service and IBM Netezza® technology to accelerate complex queries in a DB2 for z/OS highly secure and available environment. Superior performance and scalability with rapid appliance deployment provide an ideal solution for complex analysis.

In this IBM Redbooks® publication, we provide technical decision-makers with a broad understanding of the benefits of Version 3.1 of the Accelerator's major new functions. We describe their installation and the advantages to existing analytical processes as measured in our test environment. We also describe the IBM zEnterprise® Analytics System 9700, a hybrid System z solution offering that is surrounded by a complete set of optional packs to enable customers to custom tailor the system to their unique needs.

Authors

This book was produced by a team of specialists from around the world working at the IBM International Technical Support Organization (ITSO), Poughkeepsie Center.

Paolo Bruni is a DB2 Information Management Project Leader at the International Technical Support Organization, which is based in the Silicon Valley Lab. He has authored several IBM Redbooks publications about DB2 for z/OS and related tools, and has conducted workshops and seminars worldwide. During his years with IBM, in development and in the field, Paolo has worked mostly on database systems.

Willie Favero is an IBM Senior Certified IT Software Specialist and the DB2 subject matter expert (SME) with the IBM Silicon Valley Lab Data Warehouse on System z Swat Team. He has 40 years of experience working with databases, including 30 years working with DB2. Willie is a sought-after speaker for international conferences and user groups. He also publishes articles and white papers, contributes to numerous IBM Redbooks publications, and has a top technical blog on the Internet.

James Guo is a Senior Software Engineer in the Silicon Valley Lab, San Jose, California. James has experience with the DB2 z system, SQL, and Data Warehousing on System z performance analysis and tuning. He has evaluated DB2 performance using several TPC as well as customer benchmarks. James helps DB2 customers migrate their applications to new DB2 releases and is a frequent speaker at IDUG and DB2 conferences.

Ravikumar Kalyanasundaram is an IBM Certified Thought Leader in the IT Specialist profession and a Distinguished IT Specialist (Certified by The Open Group). He has more than 21 years of experience and is currently working as a Senior Managing Consultant at IBM Software Group - IM Lab Services. He provides technical consulting services to clients worldwide, utilizing specialized knowledge and skills in the DB2 for z/OS database, Analytics Accelerator, and Information Management tools. He provides a truly integrated set of high-quality services with a focus on database performance management. He plays a direct role in increasing the long-term strength and enhancing the market position and competitive posture of IBM database products and tools. He holds a Bachelors degree in Electrical and Electronics Engineering and a Masters degree in Business Administration (MBA). He is a detail-oriented person, with outstanding project management, problem-solving, team-building, and decision-making skills.

Ravi is a co-author of four IBM Redbooks publications: *Optimizing Restore and Recovery Solutions with DB2 Recovery Expert for z/OS*, SG24-7606; *DB2 9 for z/OS Resource Serialization and Concurrency Control*, SG24-4725-01; *DB210 for z/OS Performance Topics*, SG24-7942; *Optimizing DB2 Queries with IBM DB2 Analytics Accelerator for z/OS V2.1*, SG24-8005. His public profile can be viewed at this site:

<http://www.linkedin.com/in/ravikalyanasundaram>

Ruiping Li is a Senior Software Engineer in DB2 for z/OS Development at IBM Silicon Valley Lab. She is currently the lead designer for DB2 query acceleration support for the IBM DB2 Analytics Accelerator. Over the past decade, Ruiping has led the development for the DB2 9 new feature, *optimistic locking support*, and DB2 10 new feature, *timestamp*, with time zone data-type support. She also designed and implemented several key enhancements in DB2 for z/OS, such as IBM pureXML® index exploitation, index on expression, materialized query tables (MQTs), and multiple coded character set identifier (CCSID) enhancements. Ruiping is also a co-author of the book *The Business Value of DB2 for z/OS: IBM DB2 Analytics Accelerator and Optimizer*, ISBN: 978-1-58347-381-8.

Cristian Molaro is an IBM Gold Consultant, an Independent DB2 Specialist, and an Instructor who is based in Belgium. He has been recognized by IBM as an IBM Champion for Information Management in 2009, 2010, 2011, 2012, and 2013. His main activity is linked to DB2 for z/OS administration and performance. Cristian is co-author of seven IBM Redbooks publications that are related to DB2. He holds a Chemical Engineering degree and a Masters degree in Management Sciences. Cristian was recognized by IBM as “TOP” EMEA Consultant at the IDUG EMEA DB2 Tech Conference Prague 2011. He can be reached at cristian@molaro.be.

Andy Perkins is a Data Warehouse and Business Intelligence SME with the IBM Silicon Valley Lab Data Warehouse on System z Swat Team. He has 30 plus years of experience in the information management field. He has worked across platforms, starting his IBM career as an IBM IMS™ database administrator (DBA) on the NASA Space Shuttle project. He spent many years as a technical pre-sales specialist focusing on Data Warehousing and Business Intelligence, primarily on distributed platforms and has trained IBMers worldwide on these solutions. He finally came back home to the System z in 2008 with his current position. He has co-authored numerous IBM Redbooks publications on Data Warehousing and Business Intelligence topics.

Theresa Tai is an IBM Senior Certified Executive IT Specialist in Systems Technology Group where her current focus areas are zEnterprise IT optimization and workload placement, Advanced Business Analytics, and IBM Mobile First framework. Her background includes JES2/JES3 Design, Development, and Service. At end of 1998, Theresa took on a client-facing role and her extended expertise includes IBM JVM, IBM WebSphere® Application Server family, and IBM Specialty Engines zAAP and zIIP workloads. During this time, Theresa earned the IBM Redbooks publication author platinum level merit badge in March 2009.

Dino Tonelli is a Senior Software Performance Analyst working in IBM Poughkeepsie. He has over 15 years of experience with System z performance, working on WLM, DB2, data warehousing, and IBM Parallel Sysplex®. Dino currently leads an IBM STG/SWG z Business Analytics Performance Integration (zBAPI) team, which conducts performance evaluations of the IBM zEnterprise Analytics System 9700/9710 offerings, including the DB2 Analytics Accelerator. His main interest is determining best practices and driving optimizations for resource management of System z data warehousing and business analytics best practices. He holds an MS degree in Computer Science from Polytechnic University.

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

Rich Conway
Mike Ebbers
Emma Jacobs
IBM International Technical Support Organization

Chih-jieh Chang
Maggie Jin
Maggie Lin
IBM Silicon Valley Lab

Patric Becker
Peter Bendel
Oliver Harm
Norbert Heck
Namik Hrle
Norbert Jenninger
Claus Kempfert
Daniel Martin
Joachim Rese
Johannes Schuetzner
Knut Stolze
IBM Boeblingen Lab

Jeffrey Feinsmith
IBM Atlanta

Larry Lutz
IBM Marlborough

Ernest Balloni
Michael Buechele
Jane Forrest
Mark Laman
Bruce Smilowitz
Joseph Stein
IBM Poughkeepsie

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more 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 books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks 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
2455 South Road
Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

<http://www.facebook.com/IBMRedbooks>

- Follow us on Twitter:

<http://twitter.com/ibmredbooks>

- Look for us on LinkedIn:

<http://www.linkedin.com/groups?home=&gid=2130806>

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>

- Stay current on recent Redbooks publications with RSS Feeds:

<http://www.redbooks.ibm.com/rss.html>



The analytics lifecycle

Business analytics has become a crucial competitive advantage when placed in the hands of the decision-makers. It has also become a business imperative to move beyond reports and dashboards to embedding real-time data into the very fabric of operations. It must be at the point of customer interaction. When quickly and properly implemented, it can enable organizations to identify new opportunities, respond to business situations faster and more accurately, allowing them to make smarter, more informed business decisions thus ultimately increasing their competitive advantage and improving their business performance. Waiting days, weeks, or even months for the necessary business information is no longer acceptable and no longer practical. Time is money and the time to make a decision is translatable to organizational cost of unattained business advantages.

The requirement for quick deployment and expansion of business analytics and data warehousing infrastructure is now a necessity to meet the growing demand of ever increasing data volumes and the user adoption of the most current applications and functionalities. Analytics and warehousing are expanding enterprise wide.

1.1 Business analytics lifecycle

Today's IT implementations are characterized by different platforms, data movements and transformations, inconsistent copies, multiple failure points, multiple security domains and the risk in meeting compliance, as well as risk and governance requirements. Figure 1-1 illustrates this complexity recognizing that the best data warehouse from a price and performance point of view is just one part of the collection of unrelated processes that are frequently out of step with the business. They consume resources by moving and validating data, which often results in missed business opportunities, missed service level agreements (SLAs), and high IT costs.

The lifecycle of data has to originate someplace. The data lifecycle architecture diagram demonstrates that for many data warehouses, its data first enters from an online transaction processing (OLTP) system. After the data has been inputted or updated, the data needs to be positioned for some type of reporting. This is the point when the data moves to the cleanse and transform stage. Often that transformation happens via the extract, transform, and load (ETL) or extract, load, and transform (ELT) process. With the data in the warehouse, reports can be produced. Eventually, the data is analyzed with the results driving additional OLTP activity. Numerous tools can become involved in this circular process. The data can also end up being moved to numerous locations, sometimes locations across multiple different platforms.

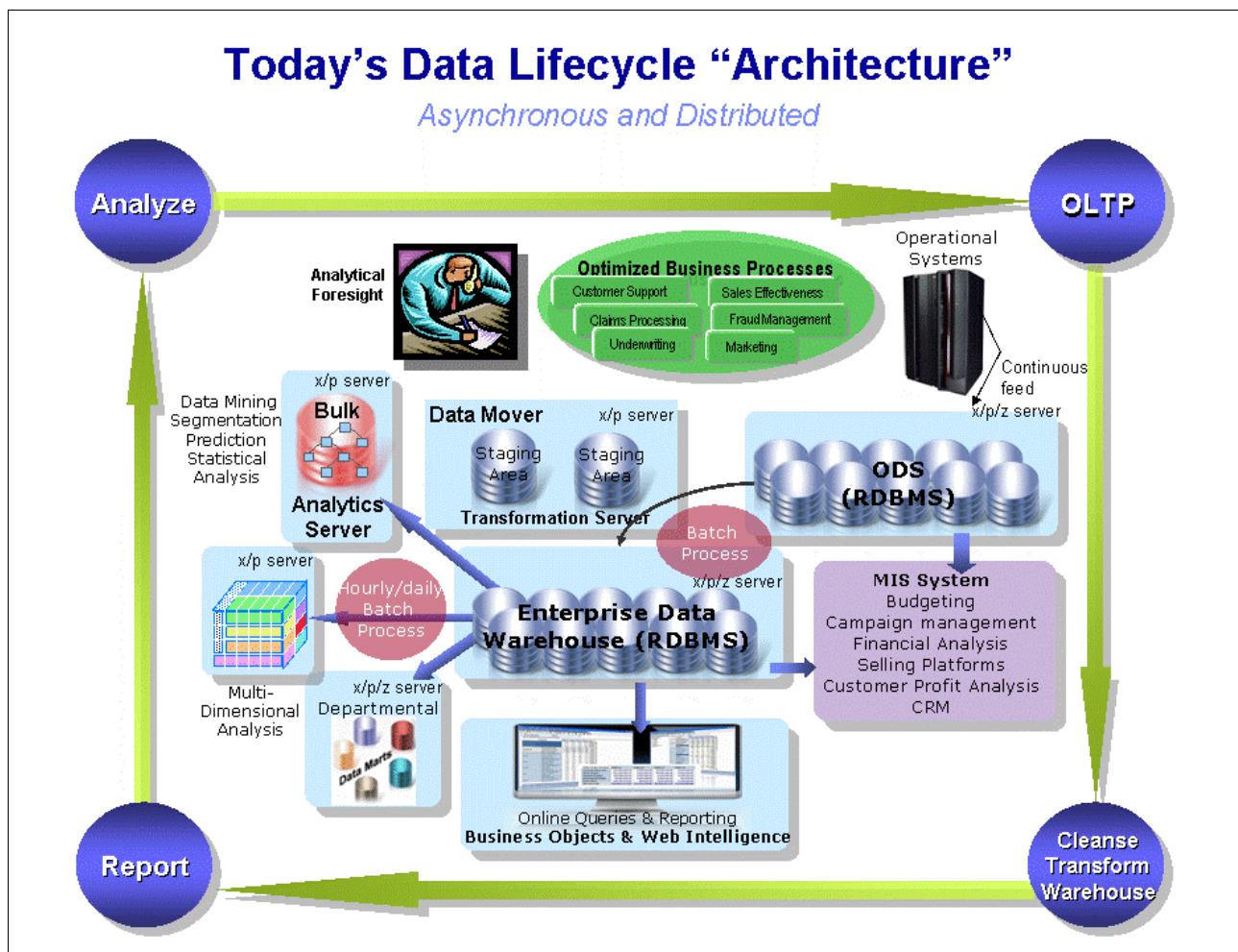


Figure 1-1 Data lifecycle

Let us now take a closer look by examining it from the data lifecycle perspective.

OLTP is where the data is generated based on business activity or it is the reason to generate data. OLTP, usually associated with high volume transaction processing applications such as IBM CICS®, IMS, and WebSphere, is a necessity for running a business because that is where the customer interactions, the customer transactions, are taking place. Data is gathered (OLTP), cleansed, and transformed for the data warehouse, reported on, and analyzed. The results drive more data to be gathered and the cycle starts all over again.

Analytics is anything that uses that operational data to help better drive the business. The goal of analytics is to harvest value from the OLTP data and use that harvested data to derive insights, trends, exceptions, and any other number of business characteristics. Part of the value of the harvested data is the ability of the analyst to predict the future and define and make actionable decisions based on that data.

This connection between business transactions and business analytics has an interesting affect on a corporation's data. Having the ability to gain insights from all of that available data to make predictions of the future, and to assist making corporate decisions, causes the attraction of more data. The more data that is available to the analytics application, the more and better information is yielded, which in turn causes the need for even more data. If a significant amount of data is available and used to produce a lot of good information, it seems only logical that you will want even more data that could cause even more information to be available. Data attracts data; it is a self-feeding cycle.

When discussing OLTP, what platform is it most often associated with? In most cases, the data associated with OLTP is maintained on z/OS. That makes z/OS the engine that is responsible for generating most of the OLTP data that could eventually be used with analytics. If the data used for analytics is finding its beginning on z/OS it only makes sense, in an effort to keep the analytics as close to the point where the data originated, that the analytics should also be on System z.

One example scenario that attempts to explain this phenomenon is the company that would like to get more insight from their data, in this case, all that data arriving by the second in an OLTP environment. However, the conception that often arises is the concern about using that production data that drives the business. Touching the actual data used by OLTP could interfere with, and possibly have a negative performance impact, on the SLAs in place for the OLTP environment. It is then believed that the solution, the work-around, is to move the data off of System z, that is, move all of that data off to another place where that introspection and supposed analytics can be performed.

So, our example company moves off the data to another platform to perform the transformations, often using a tool such as IBM DataStage®. That transformation (represented in the circle in the bottom right corner of the chart in Figure 1-1 on page 2) can result in a data warehouse that is now on another platform, away from the originating data used to create it. Often, the next step is to create data marts for the various lines of the business; again, we do more extractions, transformations, and loads. If there are plans to also use multidimensional cubes, then there is a strong possibility that the data could get moved again to yet another platform.

If there is a desire to also use predictive analytics, our example company will want to start doing modeling, using a tool such as IBM SPSS®, often pulling the necessary data from those data marts, and moving that data to yet another platform where it can be analyzed. Now that you have modeled your data, the question is, or maybe better, the challenge is to get that data back into that OLTP environment where it can be used to make better decisions at the point of interaction.

The lifecycle perspective just described might sound very familiar. The solution involves multiple different platforms, which in turn could potentially require a great deal of data movement and latency. For a company with their OLTP resident on System z, it is not unusual to see that company spending 30% of their CPU cycles just moving data off the platform all on the assumption that their analytics cannot be handled on System z. Not only can CPU cycles be wasted, there is the challenge of dealing with multiple security environments, resolving availability issue that can arise through the use of the variety of platforms, the personnel required to keep it all moving forward successfully, and the attempts to make it all perform.

Even when a company believes that they have the best warehouse that money can buy when it comes to queries per dollar or queries per second, the real problem (or issue) that will often arise is how to manage the complexity of the data flow, as well as the timeliness of the data flow, and in many cases the potential data inconsistencies that can occur as that data is moved around all the various different environments. In some cases, it can take months before the data consistency issues start to show up. And when the issues are discovered, it forces large amounts of time to be spent just validating data before presenting it to “C” level management at the cost of both time and money.

1.2 Lifecycle simplification

What is being focused on in the chart in Figure 1-2 on page 5 is the simplification of this life cycle. It attempts to demonstrate how to minimize the number of times that data has to be moved between different platforms, reduce the complexity of the overall configuration, yet promoting the accuracy, security, and availability that is necessary for a company to be successful today. In the process, lifecycle simplification provides a better business response from a line of business prospective.

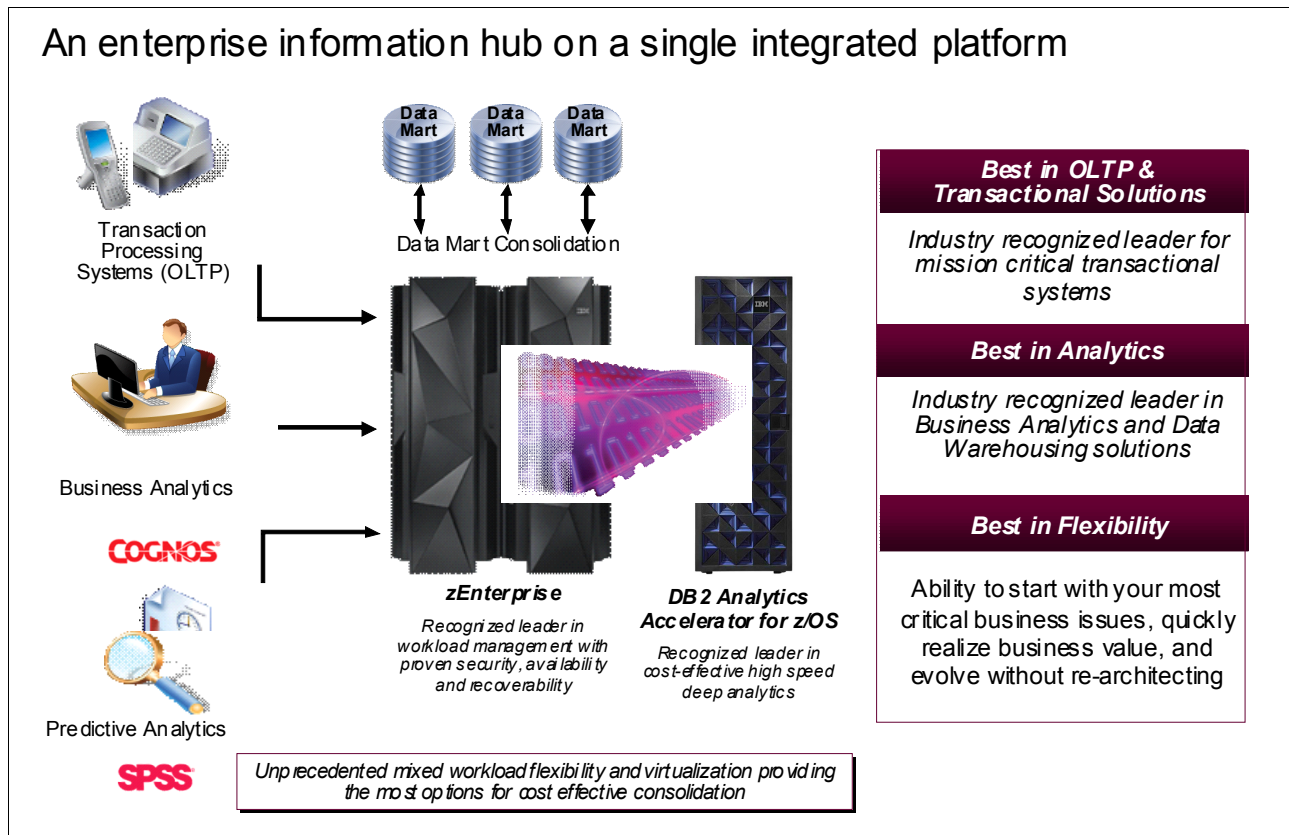


Figure 1-2 Platform direction

After the data has been gathered (OLTP), the next step is to transform and cleanse the data for use in a data warehouse. Data transformation is the process by which source data is selected, converted, and mapped to the format that is required by the targeted system. The process might, and in many cases does, manipulate the data to bring it into compliance with business, domain, and integrity rules and with other data in the target environment.

Transformation can take on many forms. For example:

- ▶ It could aggregate data by consolidating or summarizing data values into a single value.
- ▶ Basic conversion could be performed to ensure that data types are correctly converted and mapped from source to target columns.
- ▶ Data cleansing to resolve inconsistencies and anomalies in the source data is often required. Data that drives a business system today comes from various sources and disparate data structures. In addition, through organizational growth, older data systems have to work with newer improved systems, making data sometimes more difficult to manage and use.
- ▶ Data can be combined from internal or external sources to provide additional meaning to the warehouse data.
- ▶ Tables can be normalized to reduce redundant and potentially duplicated data. Often, data is denormalized to improve performance in the OLTP system.

Not only is almost all OLTP on z/OS, but in many cases, the operation data stores (ODSs) are also located on z/OS. In addition to all the usual reasons that you would want to use System z (security, reliability, scalability, and so on), there is an extra reason for having an ODS on

z/OS. If OLTP is driving data gathering, what better place for an ODS than someplace near to the data source. That is, someplace that makes it easy to get to the data because of proximity.

Note: We define an ODS as a copy of the OLTP data that still has a very normalized schema. The ODS has not been demoralized like a data warehousing schema might have been. An ODS is also an environment where data from different operational databases is integrated. The purpose is to provide the end user community with an integrated view of enterprise data. It enables the user to address operational challenges that span over more than one business function. Its principal key differentiators are the update frequency and the direct update paths from applications, compared to the controlled update path of data warehouse. The ODS typically contains current data, which is dynamically updated in a real-time (or near-real-time) manner. In short, an ODS is an architectural construct containing subject-oriented, integrated, volatile, current (or near-current), and detailed operational information.

An ODS becomes the first staging area, an area that is easy to consolidate around, because the data in the ODS might already be somewhat cleaned up. The ODS is also close to the central feed system (DB2 for z/OS), which allows for easy access to the needed data. This is one of the primary reasons why the ODS is there. Our challenge is to change the usage of the data from just being a central unload point to a data processing point. It can no longer just be used as a place to obtain the data needed. Get there, and you know that you can get all the data from there that you need.

There is a drive today to perform real-time scoring within an OLTP transaction.

An example of moving predictive decision making analytics closer to the data's source is predictive-scoring processes run within the transactional scope. For example, fraud-detection scoring might indicate a 90% probability of credit card fraud, and business rules could reject transactions with scores greater than 80%. This combining of rules and scoring into OLTP, which allows for the incorporation of the most recent customer activity, is a growing trend, both intensifying and exhibiting the need to move analytics computations closer to the data.

At one time, it would be expected that a transaction, although going against the database, would have to “ETL” the data out of the database to a data warehouse or data mart, where a batch scoring process would use that extracted data to perform the scoring (usually on another platform). The scores are then deployed back to System z and inserted into the OLTP database. When a request is made, when a transaction is executed, scoring might be quickly performed because it is in the OLTP database. However, that score is based on old or stale data, depending on how frequently the ETL is performed in an attempt to keep the data current.

Over time, this process has been improved. Today, when a request arrives, it gets the data in DB2, the SPSS scoring infrastructure. The scoring engine is a user-defined function (UDF) in DB2 for z/OS, which allows the scoring to be performed immediately in real time. Network calls and accesses to other platforms can be completely avoided while scoring can be performed using the most current data and performing the action directly in the database. Measured improvements have been as high as four to seven times depending on the originating environment.

Complex queries, batch scoring, and modeling require access to large amounts of near real-time and historical data. In this case, alternate technologies are ideal. For this analytic work, DB2 has integrated a Netezza technology appliance (now made available as IBM PureData™ System for Analytics) as a component of DB2 for z/OS. This appliance combines a massively parallel processing (MPP) structure using hardware- and software-optimization techniques for industry-leading query acceleration with the broad mixed workload capabilities

of DB2. This offering is called the *IBM DB2 Analytics Accelerator for z/OS V3.1*, which is also the primary focus of the remainder of this book.

There is an industry trend, a trend not specific to any one vendor, for consolidation of workloads with the movement of complex analytics closer to the OLTP data. One of the reasons it is happening today is that companies no longer have time. It takes time to move data to the environment in order to move the data to the process that will handle the complex analytics. It is in many cases simpler, more convenient, and less time consuming to move to the analytic processing to the data, to move the processing to where the OLTP environment is located, to move the analytics on the real data, or to a copy of the real data as close to the real data as possible. The reason why this has to happen today is that people no longer have time. It takes time to feed this complex analytics environment. It is more convenient to move the processing to OLTP and perform on the same platform.

The System z platform, along with the DB2 Analytics Accelerator, challenges the myth that transactional, decision support, and analytics workloads cannot reside on the same platforms. With the IBM approach, an approach unique in the industry, that myth can be dispelled. System z has the software portfolio, the outstanding System z hardware, and the integration with Netezza technology that provides a better business response created by decreased data movement and reduced complexity with superior security and availability.

The changing landscape for business analytics on System z has been accomplished through well established System z core values and its software strengths. For example, the strength of the System z business analytics software portfolio currently consists of IBM Cognos® on Linux for System z and z/OS, SPSS on Linux for System z, SPSS Real Time Analytics Transactional Scoring built into DB2 10 for z/OS through its UDFs and delivered via the DB2 10 maintenance stream, IBM ILOG® BRMS, IBM CPLEX® Optimization on z/OS, and DB2 10 for z/OS along with IBM QMF™. In fact, DB2 for z/OS has delivered more business analytic functions in the last couple of versions than at any time in the product's history. For example, DB2 10 provides bi-temporal data capabilities that can be enabled by a customer to implement time-aware applications and queries that analyze and manage past, current, and future events with minimal effort by using point-in-time and end-of-business query processing. This ensures the integrity of data over time while satisfying audit and compliance initiatives.

Note: Linux for System z runs on top of System z on virtualization software called IBM z/VM®. z/VM virtualization technology is designed to allow the capability for clients to run hundreds to thousands of Linux servers on a single mainframe running along side other System z operating systems, such as z/OS, or as a large-scale Linux only enterprise server solution. This technology provides exceptional levels of availability, security, and operational ease that System z is famous for.

A critical aspect of “outstanding System z hardware” is the ability of the System z to do compute-intensive performance. In the last few years, System z has been able to improve performance by 12 times starting with the IBM z9® platform through to the current zEC12 that is available today. Binary and hexadecimal floating-point instructions are implemented in zEC12. They incorporate IEEE Standards into the system. The decimal floating point (DFP) architecture on zEC12 has been improved to facilitate better performance on traditional zoned-decimal operations for COBOL programs. Additional instructions are provided to convert zoned-decimal data into DFP format in floating-point register (FPR) registers.

Data sharing has long been used to implement ad-hoc queries without impacting CICS/IMS workloads. It can be used to create affinities to a subset of members for certain types of processing, allowing for performance tuning based on the type of processing, all while allowing OLTP and data warehousing to coexist in the same single system.

A company can also enhance its analytics in a transaction environment by taking advantage of the data sharing capabilities of DB2, which prioritize the use of resources by workload.

This is a solution that not only meets and exceeds the needs of yesterday, this solution provides a unique analytics infrastructure that is required today and is necessary to take us into the future.

1.3 Overview of the solutions

Efficient and effective information management is key to the success and competitiveness of an enterprise. Having the right information available at the right moment for the right people is critical for any business process. Simply designing and implementing smart application programs is always not enough to achieve this goal. Instead, solutions are needed that organize information in such a way that it can be accessed when and where it is needed.

An additional upcoming issue is that IT infrastructure and IT resources can inhibit you from achieving this goal. This can sometimes occur because accessing information is so resource-intensive that even on the largest systems, information cannot be returned in a timely manner. Also, accessing the same data from many programs at the same time can cause serious performance problems, especially when updating data.

Having a powerful infrastructure is critical to implementing an effective information management architecture. That is today's information challenge.

Today, information is one of the most valuable assets that a company possesses. Information consists of data, and data is everywhere, and sometimes is in extremely large quantities. Unfortunately, the reality in many organizations is that data is spread across the organization, duplicated, conflicting, in silos, hidden, buried in an application, inaccessible, incomprehensible, compromised, unsecured, inaccurate, meaningless, out of context, and out of place.

Data is meant to provide insight into business, supplying a source for performance metrics and offering a competitive edge. But without a well-designed structure and a well-planned architecture, data becomes a huge challenge to contain, manage, and understand. It is almost impossible to derive useful information from meaningless data. But you will probably not obtain useful information from useful data either, without implementing a well-designed information architecture.

Companies are investing more in optimizing business than in simplifying automation processes. They are investing to better use information, including applying better analytics to more information to drive better decisions and smarter business outcomes, and this is true across all industries.

Those who have finally reached the goal of structuring information in an acceptable manner are faced with the next challenge, which is managing large quantities of data and information.

Focus is shifting from building a functional information management architecture to providing an infrastructure that is powerful enough to run it.

Business intelligence and business analytics rely on the availability of massive amounts of data. This trend is continuing based on the growing importance of the following aspects:

- The need for reporting is increasing.

Reports need to provide useful information, and they are increasingly based on the complex integration and mining of multiple data sources across the enterprise. The

increasing need for reporting is a result of increasing needs from company executives to continuously be able to assess the state of the business. But there are also growing reporting needs as a result of regulatory requirements.

- The amount of data is simply increasing year over year.

Enterprises tend to collect more and more data. Consider all of the data that is collected when people visit web sites, let alone do business on the Internet. The way companies do business these days, especially using the Internet channel, warrants a steady growth of data, and the end of this growth is nowhere in sight.

- Being able to predict the future is becoming a significant issue for most companies.

Predictive analytics is the capability to make decisions based on future trends. These future trends are derived from current trends and future parameters, within and outside of the company. To be able to detect current trends, massive amounts of data are again needed, and this data needs to be organized and analyzed. On a computer system, this is an I/O- and CPU-intensive process.

The System z warehousing and analytics software portfolio is continually evolving and improving. Over the past several years, it has grown with certain aspects taking on new roles. Software products that are available in the portfolio can exist natively on System z, run under Linux for System z, and in some cases are available for both. In this chapter, we introduce, provide, describe, or discuss some of those software products.

The IBM zEnterprise Analytics System 9700 is a hybrid system built from the MPP of the DB2 Analytics Accelerator and symmetric multiprocessing (SMP) of System z technologies. This merges fit-for-purpose and mixed workload capabilities into a single system for both transaction and high speed analytical applications on a single platform for operational business analytics:

- It is able to run transaction processing applications and analytics applications on one machine, enabling users to analyze data and utilize it at the time of decision.
- It simplifies the process of deploying operational analytics while cutting the cost of running analytics on System z.
- It is routed on a platform of unequalled security (ELA5+) with unmatched availability and recoverability while continuing to be the most reliable system platform with highly scalable servers and storage.
- A trusted information platform offering high performance data warehouse management and storage optimization.
- An analytics platform that can support operational analytics, deep mining, and analytical reporting with minimal data movement.
- The ability to perform operational analytics with the same quality of services as your OLTP environment on System z.
- The base offering is surrounded by a complete set of optional packs to enable customers to custom tailor the system to their unique needs.
- An Information Analytics Pack that includes both Cognos BI and SPSS.
- Embedded analytics that can be deployed within the application or the database, providing maximum flexibility in operational analytics.
- Data Integration Pack, which provides data movement and transformation, data discovery, real-time delivery with complete metadata management, and lineage.
- FastStart set of services to enable the quickest time to value with the least amount of staff impact.

The 9710 is a cost-effective critical data decision system for analytics, offering high performance data warehouse management with System z availability and recoverability. It can be optionally combined with the DB2 Analytics Accelerator for the same high-speed analytical capabilities as the 9700.

- ▶ Entry-level pricing with robust capabilities.
- ▶ Built upon the zEnterprise BC12 (zBC12) platforms, setting the standard for scalability and availability.
- ▶ Able to run transaction processing applications and analytics applications on one machine, enabling users to analyze data and utilize it at the time of decision.
- ▶ Simplifies the process of deploying operational analytics while cutting the cost of running analytics on System z.
- ▶ An analytics platform that can support operational analytics, deep mining, and analytical reporting with minimal data movement.
- ▶ The ability to perform operational analytics with the same quality of services as your OLTP environment on System z.
- ▶ The base offering is surrounded by a complete set of optional packs to enable customers to custom tailor the system to their unique needs.
- ▶ An Information Analytics Pack that includes both Cognos BI and SPSS.
- ▶ Embedded analytics that can be deployed within the application or the database, providing maximum flexibility in operational analytics.
- ▶ Data Integration Pack, which provides data movement and transformation, data discovery, real-time delivery with complete metadata management, and lineage.
- ▶ FastStart Pack set of services to enable the quickest time to value with the least amount of staff impact.



Using the Studio client to manage the environment

IBM Data Studio, along with the IBM DB2 Analytics Accelerator plug-ins, is a fully licensed product that is available at no charge and has no time restrictions.

This chapter contains information for database administrators who need to install, maintain, and run IBM DB2 Analytics Accelerator Studio to complete administration tasks for IBM DB2 Analytics Accelerator for IBM z/OS V3.1. Some of the most important tasks in this context are the definition and deployment of data for accelerated database queries. Other important tasks include the adding, removal, activation, and deactivation of individual Accelerators within the system configuration.

The following topics are described in this chapter:

- ▶ DB2 Analytics Accelerator Studio V3.1 overview
- ▶ Installing IBM DB2 Analytics Accelerator Studio
- ▶ Configure Studio for tuning
- ▶ Accelerator Studio tasks
- ▶ Stored procedures used as administrative interface
- ▶ DB2 Accelerator stored procedures used by Studio
- ▶ Data Studio web console

2.1 DB2 Analytics Accelerator Studio V3.1 overview

IBM DB2 Analytics Accelerator Studio is a component of the DB2 Analytics Accelerator for z/OS solution that provides the graphical user interface (GUI) for the administration of IBM DB2 Analytics Accelerator for z/OS V3.1.

Launching an IBM DB2 Analytics Accelerator Studio allows users to connect to the DB2 for z/OS data server, which calls various stored procedures to execute corresponding commands on the Accelerator. The Accelerator is thus always controlled and maintained indirectly via the data server.

IBM DB2 Analytics Accelerator Studio V3.1 is installed as an extension of IBM Data Studio V3.2. The Accelerator plug-ins can also be installed with the latest Data Studio Version 4.1. However, Data Studio Version 3.2 has been used to capture various screen captures that are depicted in this chapter (and throughout most of this book).

For installation steps, read 2.2, “Installing IBM DB2 Analytics Accelerator Studio” on page 15.

If needed, the stored procedures that come with the product could be run outside the IBM DB2 Analytics Accelerator Studio to yield the same results, even without the Studio interface. See “Calling the Accelerator stored procedures using JCL” on page 229.

2.1.1 New features

The Accelerator Studio V3.1, which is based on IBM Data Studio 3.2, provides new functions that are relevant to IBM DB2 Analytics Accelerator for z/OS V3.1 in the following areas:

- ▶ Syntax validation of the DB2 supplied special registers CURRENT QUERY ACCELERATION and CURRENT GET_ACCEL_ARCHIVE in the SQL Script Editor.
- ▶ Support for the new values ALL and ELIGIBLE of the CURRENT QUERY ACCELERATION special register in Visual Explain.
- ▶ Support for the CURRENT GET_ACCEL_ARCHIVE special register in Visual Explain, and Enable and Configure HPSS partitions.
- ▶ Automatic creation of the DSN_QUERYINFO_TABLE in the “Configure for Tuning” step. With DSN_QUERYINFO_TABLE, additional DB2 EXPLAIN information about accelerated queries can be extracted and visualized in an access plan graph.
- ▶ Start and stop incremental update function (sometimes called *change data capture based replication* or *trickle feed*).

2.1.2 Hybrid solution overview

This enhanced hybrid solution provides faster, more efficient performance of analytics workload without any application code change transparent to the users. The benefits of the hybrid analytic solutions are depicted in Figure 2-1 on page 13.

Contrary to what many believe, no architecture can handle all analytics workloads well. There are strengths and weaknesses of each architecture and technology. Although in-memory databases execute queries extremely fast, they limit the usage to small data warehouses due to the higher cost of RAM. Column-based databases favor queries at the expense of updates. Index-based databases deliver quick results to queries accessing a few records, but they require experienced DBAs to be familiar with the queries to build the proper indexes.

Although Hadoop MapReduce makes it feasible to support large-scale distributed parallel processing using commodity servers, it does not support joins and it requires a programming language interface rather than simple SQL.

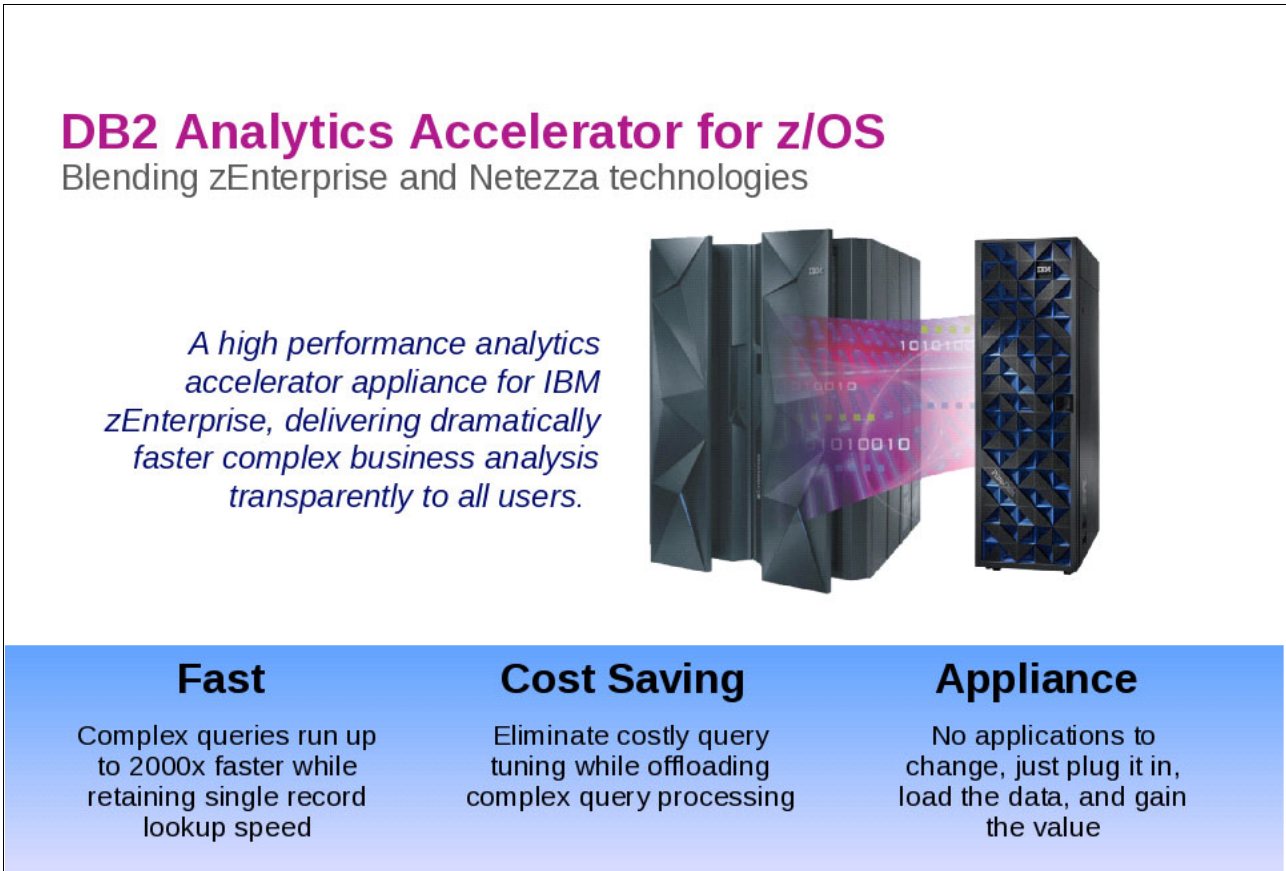


Figure 2-1 Benefits of the hybrid analytics solution

IBM DB2 Analytics Accelerator product components are depicted in Figure 2-2 on page 14. Technically, the DB2 Analytics Accelerator is an appliance that comes as additional hardware and software to be connected to System z hardware and software. The DB2 Analytics Accelerator connects to the zEnterprise through a high speed 10-Gigabit interface. While it uses the same technology as a free standing PureData System for Analytics N1001 or N2001, here it becomes an Accelerator that is only connected to the zEnterprise. All users and applications come through the zEnterprise to gain access to the Accelerator, which are all controlled by DB2 for z/OS.

There are several connection options using switches to increase redundancy, as discussed in Chapter 11, “High availability considerations” on page 285.

As depicted in Figure 2-2 on page 14, all DB2 Analytics Accelerator administration tasks are performed via DB2 for z/OS, thus there is no need to have direct access to the DB2 Analytics Accelerator box from outside this secured network. The Accelerator Studio and DB2 stored procedures play a pivotal role in these administrative tasks, as discussed in section 2.5, “Stored procedures used as administrative interface” on page 46.

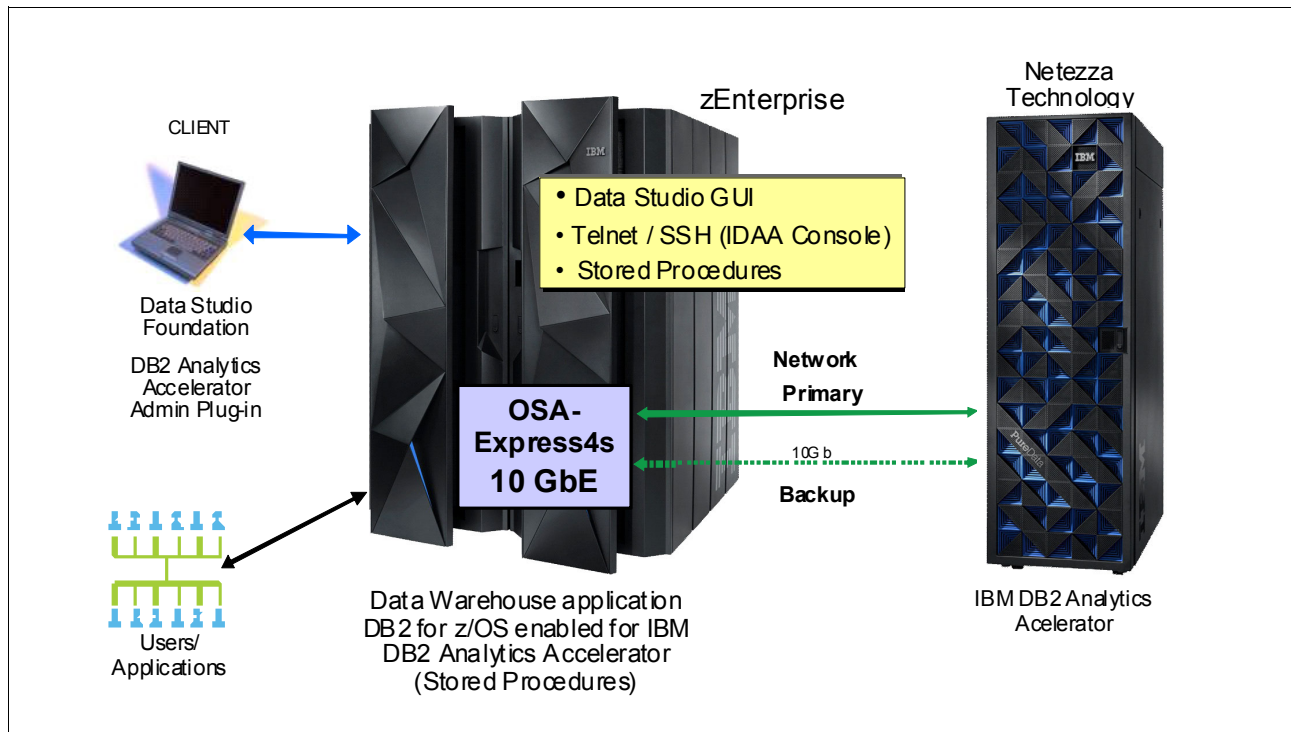


Figure 2-2 IBM DB2 Analytics Accelerator Product Components Overview

The Accelerator Configuration console is used to pair a DB2 subsystem with an Accelerator, as discussed in “Obtaining the pairing code for authentication (done outside the Studio)” on page 30.

2.1.3 Data Studio client and web console architecture

Figure 2-3 on page 15 shows the installation scenario for a single Data Studio client, a single database, and an instance of Data Studio web console. You can also connect to multiple databases from a single Data Studio client (or web console).

You can also install multiple instances of the Data Studio client (web console) to access a single or multiple databases.

Data Studio web console is discussed in section 2.7, “Data Studio web console” on page 53.

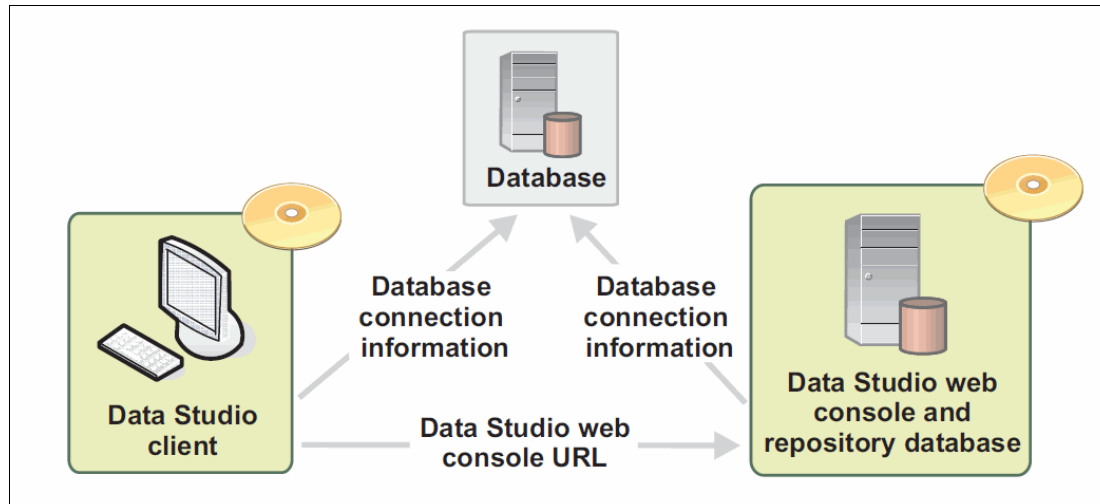


Figure 2-3 Data Studio and web console installation scenario: Architecture

2.2 Installing IBM DB2 Analytics Accelerator Studio

The IBM DB2 Analytics Accelerator Studio client can be installed without any license key.

On the product DVD of IBM DB2 Analytics Accelerator for z/OS V3.1, IBM DB2 Analytics Accelerator Studio V3.1 and IBM Data Studio Administration client V3.1.1 are bundled up in a single installation package. Extracting the code from the package installs both components.

IBM DB2 Analytics Accelerator Studio is installed on a remote computer, typically a workstation or a personal computer. IBM DB2 Analytics Accelerator Studio V3.1 is available for Red Hat Linux, SUSE Linux, and Windows operating systems.

For Linux operating systems, you cannot install or upgrade to the 64-bit version of IBM Installation Manager if existing 32-bit products are installed on your computer. The 32-bit products that you previously installed with Installation Manager do not support the 64-bit version of Installation Manager. You must download the 32-bit version of the Linux package, which includes a 32-bit version of Installation Manager. You can use the 32-bit version of Installation Manager to install either a 32-bit or 64-bit version of the product.

Installation scenarios

For IBM DB2 Analytics Accelerator for z/OS V3.1, the Studio client consists of a set of Eclipse plug-ins that are added to IBM Data Studio Administration Client V3.1.1 or Data Studio V3.2 (or later releases).

Tip: IBM suggests that users running Accelerator Studio on older versions of Data Studio should move to at least Data Studio Client 3.2. For details about Data Studio V3.2 installation, see the following site:

<http://www.ibm.com/support/docview.wss?uid=swg24033663>

Depending on whether or not you already have one of the versions of the Data Studio client on your workstation, there are a few installation scenarios to consider:

- ▶ Install Accelerator Studio from the product DVD and then perform software updates
- ▶ Upgrade Accelerator Studio client from V2.1 to V3.1

Note: To upgrade the plug-ins pertaining to Accelerator Studio V2.1 to V3.1, refer to the *IBM DB2 Analytics Accelerator for z/OS Version 3.1.0 Installation Guide*, SH12-6983, under the “Migrating from version 2.1.x to 3.1.0” and “Installing updates” chapters.

- Download and install the latest version of Data Studio, along with Accelerator components
- Apply the Accelerator plug-ins on an already installed Data Studio client

This scenario also applies to installation of the Accelerator plug-ins on an already installed Data Studio client V3.2 (or Version 4.1), which could be a part of another product, such as IBM Optim™ Query (Workload) Tuner client, and Optim Database Administrator.

2.2.1 Installing Accelerator Studio using the product DVD

Refer to “Installing IBM DB2 Analytics Accelerator Studio” in *IBM DB2 Analytics Accelerator for z/OS Version 3.1 Installation Guide*, SH12-6983, for detailed information about how to install Accelerator Studio from the DVD.

Prerequisites for Analytics Accelerator Studio V3.1

Before you start the product download, see the following product website to ensure that your environment meets the minimum hardware and software requirements:

<http://www.ibm.com/support/docview.wss?uid=swg27035960>

IBM DB2 Analytics Accelerator Studio (client) must be at the same or newer level than the DB2 Analytics Accelerator Software (server). Upgrade the client first.

Prerequisites for running DB2 Analytics Accelerator Studio on Linux

IBM DB2 Analytics Accelerator Studio uses the Eclipse SWT¹ Browser Control for some Software Update dialogs. To run IBM DB2 Analytics Accelerator Studio on Linux, you must therefore install a compatible Mozilla/XulRunner. IBM DB2 Analytics Accelerator Studio 3.1 supports Mozilla 1.4 GTK2 - 1.7.x GTK2, XULRunner 1.8.x - 1.9.x and 3.6.x (but not 2.x), and WebKitGTK+ 1.2.x

More information about prerequisites is available on the Eclipse FAQ Entry for Eclipse 3.6, at the following location:

<http://www.eclipse.org/swt/faq.php#browserlinux>

To install or upgrade a Linux package, you can use the *yum* package manager tool, which installs all dependent packages automatically. For instance, if you do not have the appropriate version of WebKitGTK (which could be easily verified using the *Add/Remove Software* application on Linux by searching for the package name installed on your machine), you can issue the following command from an *Open in Terminal*:

```
sudo yum install webkitgtk-devel
```

If your Linux workstation is connected to the Internet, this command would get the current level of the WebKitGTK, and after downloading, it would prompt for your confirmation. If you select “y” (yes), the package will be installed along with all the dependent packages. The other option is to use the **rpm** command, which is more tedious if you have many dependent packages.

¹ Standard Widget Toolkit

2.2.2 Data Studio full client download and installation

To install IBM Data Studio Version 4.1 (or Data Studio Version 3.2) and IBM DB2 Analytics Accelerator Studio V3.1 (plug-ins), you should first download, install, and start a pre-configured version of IBM Installation Manager from the web. IBM Installation Manager handles the download and installation of the Studio components. Take the following steps:

1. Open the IBM Data Studio download page at the following website:
<http://www.ibm.com/developerworks/downloads/im/data>
2. In the “Download” section, click the **Download** button for Data Studio client and 3rd-party product extensions.
3. On the password page, enter your IBM ID and password.
4. On the license confirmation page, select **I agree to accept the Installation Manager license** and click **I confirm**.
5. On the download page, select the proper IBM Installation Manager program for your operating system and click **Download now**.
Optional: If your browser does not support Java plug-ins, click the **Download using http** tab and click **Download now** after selecting the IBM Installation Manager program of your choice.
6. After the download is complete, install the Studio on the appropriate operating system.

Windows installation

For Windows installation, take the following steps:

- ▶ When the download is complete, confirm the message, **Would you like to launch this file?** or double-click the downloaded ***.exe file**.
- ▶ IBM Installation Manager starts. Follow the instructions in the IBM Installation Manager wizard and select the IBM Data Studio and IBM DB2 Analytics Accelerator Studio features to be installed:
 - Enter Administration password when prompted and confirm the firewall message.
 - Enter your IBM ID and password when prompted.

Linux installation

Take the following steps for a Linux installation, for instance, on a Red Hat Enterprise Linux Open Client environment:

- ▶ After the download is complete, go to the location where the file was downloaded.
- ▶ Extract the files to a directory of your choice.
- ▶ Open the folder where you extracted the files and right-click and pick *Open in Terminal*.
- ▶ Sign in as root by typing *su* from the Terminal window and enter the root password.
- ▶ Type *./install*
Enter your IBM ID and password when prompted.
- ▶ IBM Installation Manager starts. Follow the instructions in the IBM Installation Manager wizard. Figure 2-4 on page 18 shows a sample installation process on an RHEL Open Client V6.3 workstation. These windows are similar to that of installing the Studio on other platforms.

a. Figure 2-4 shows the initial window to select the product components.

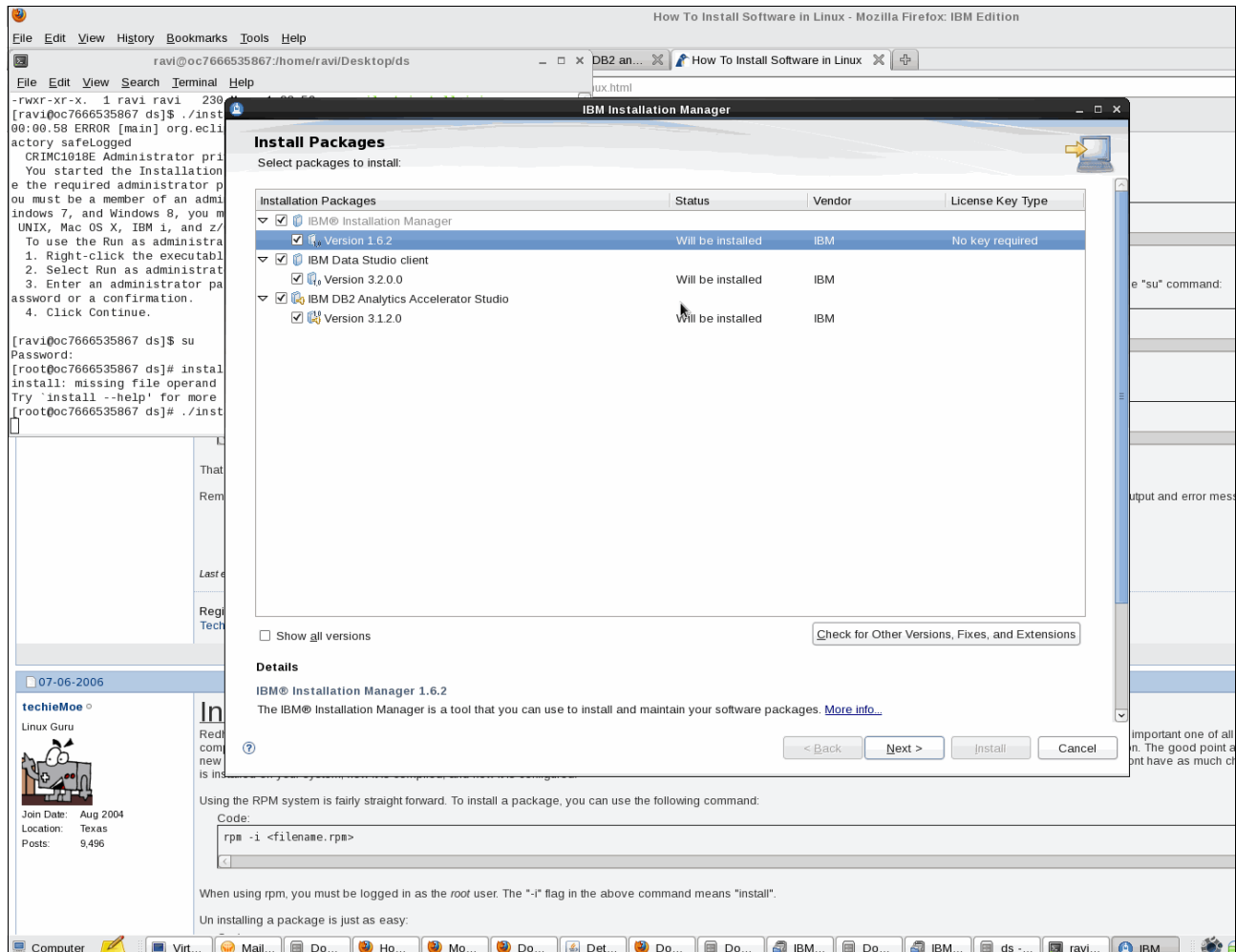


Figure 2-4 Studio installation: Initial product component selection window

b. Figure 2-5 shows the terms of the license agreement.

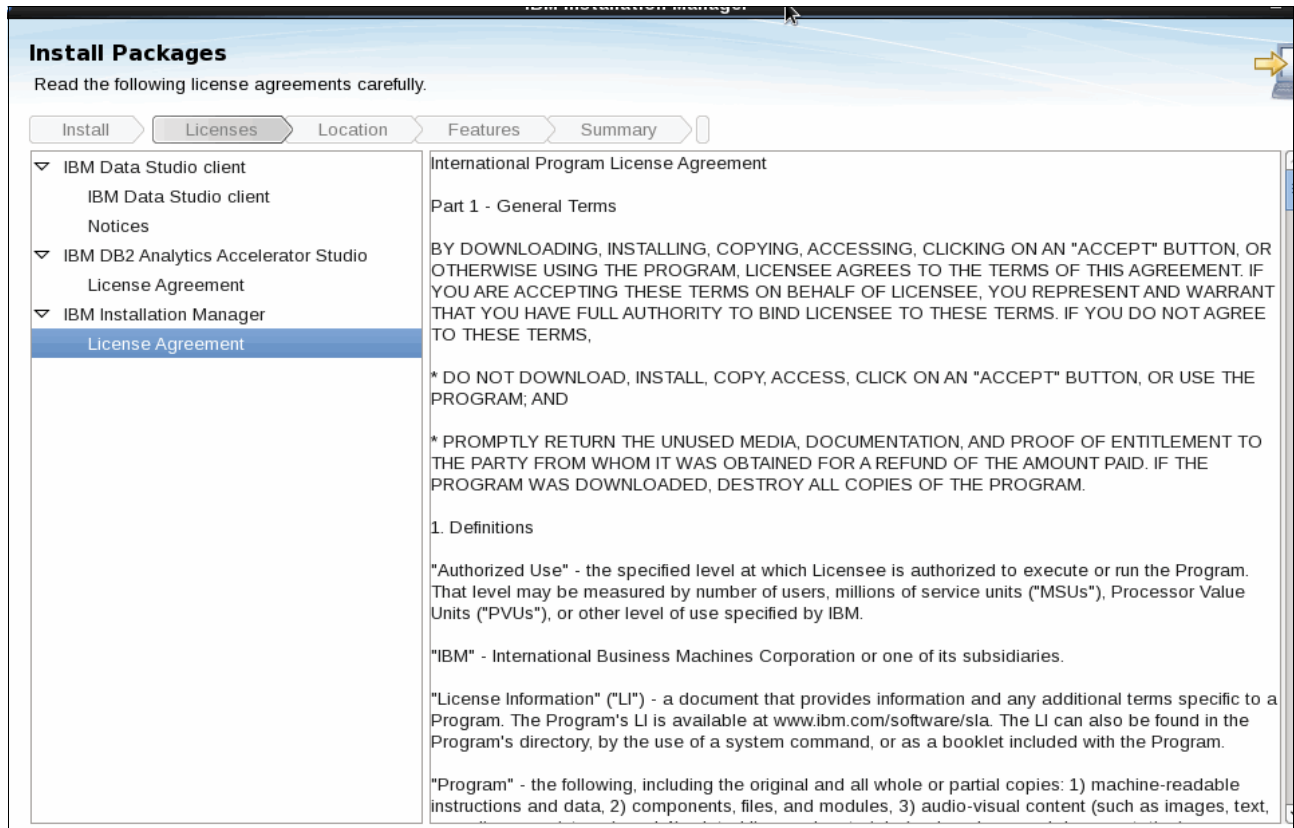


Figure 2-5 Studio installation: License agreement window

- c. Figure 2-6 shows the window where you enter the path name for Installation Manager and shared resources.

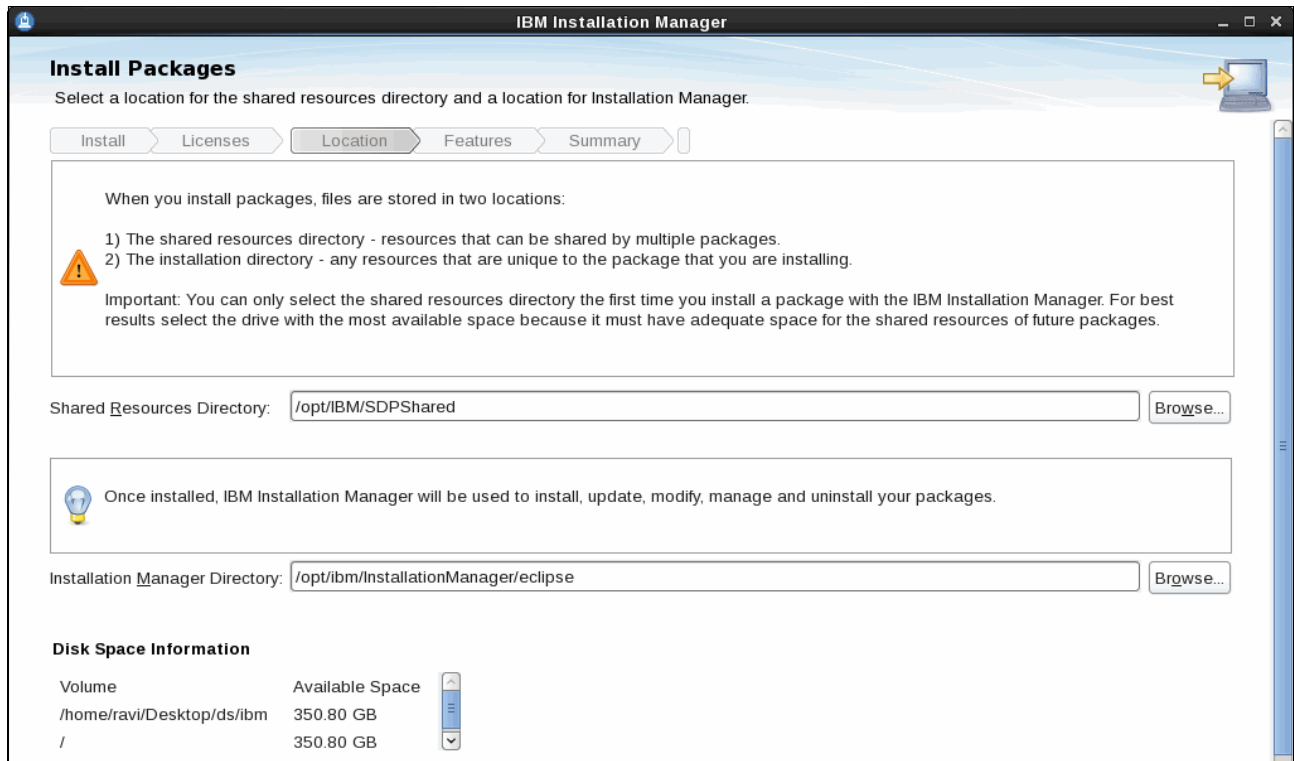


Figure 2-6 Studio installation: Location for Installation Manager and shared resources

- d. Figure 2-7 shows where you enter the location for the Data Studio installation files.

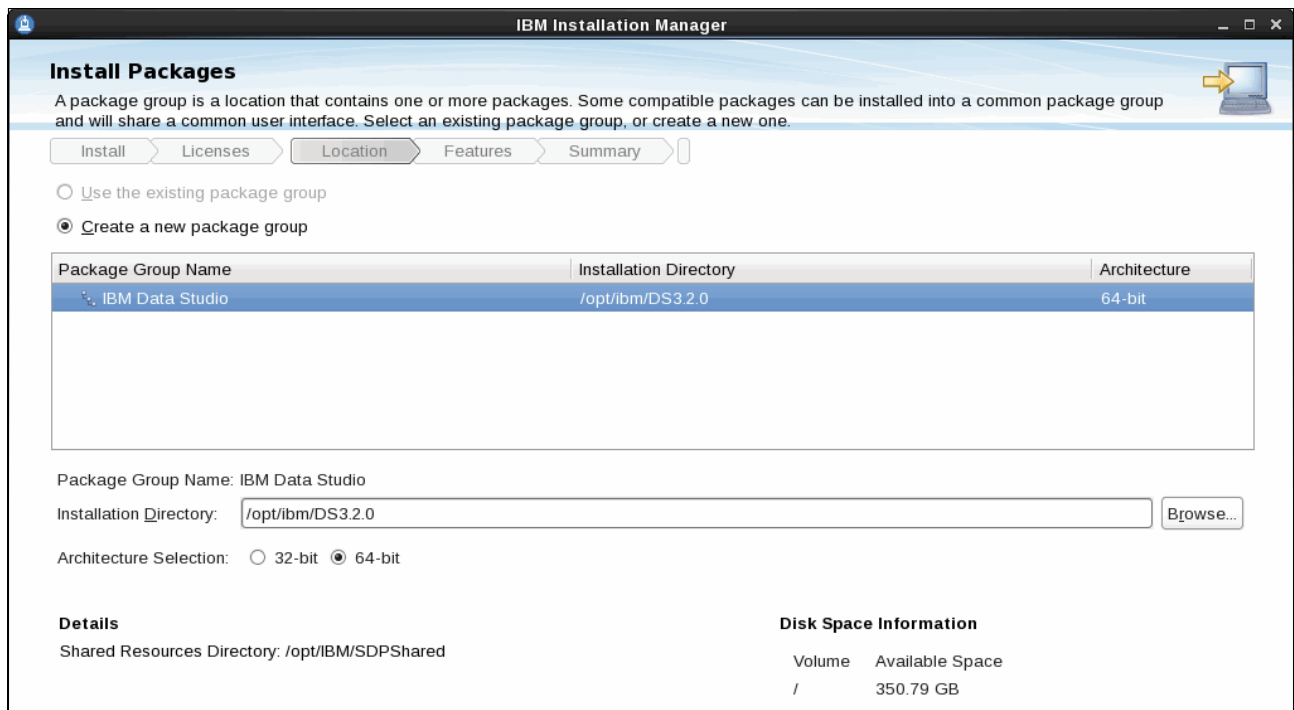


Figure 2-7 Studio installation directory name entry window

- e. Figure 2-8 shows the window where you can select the language. From here, click **Install** to start the installation.

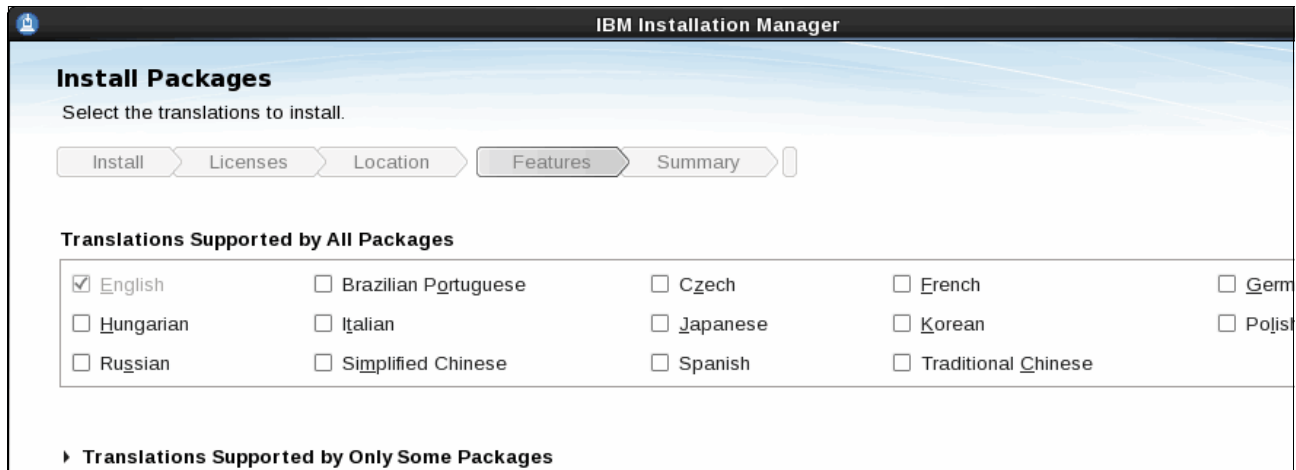


Figure 2-8 Studio installation: Language selection

- f. Figure 2-9 shows the progress of the installation.

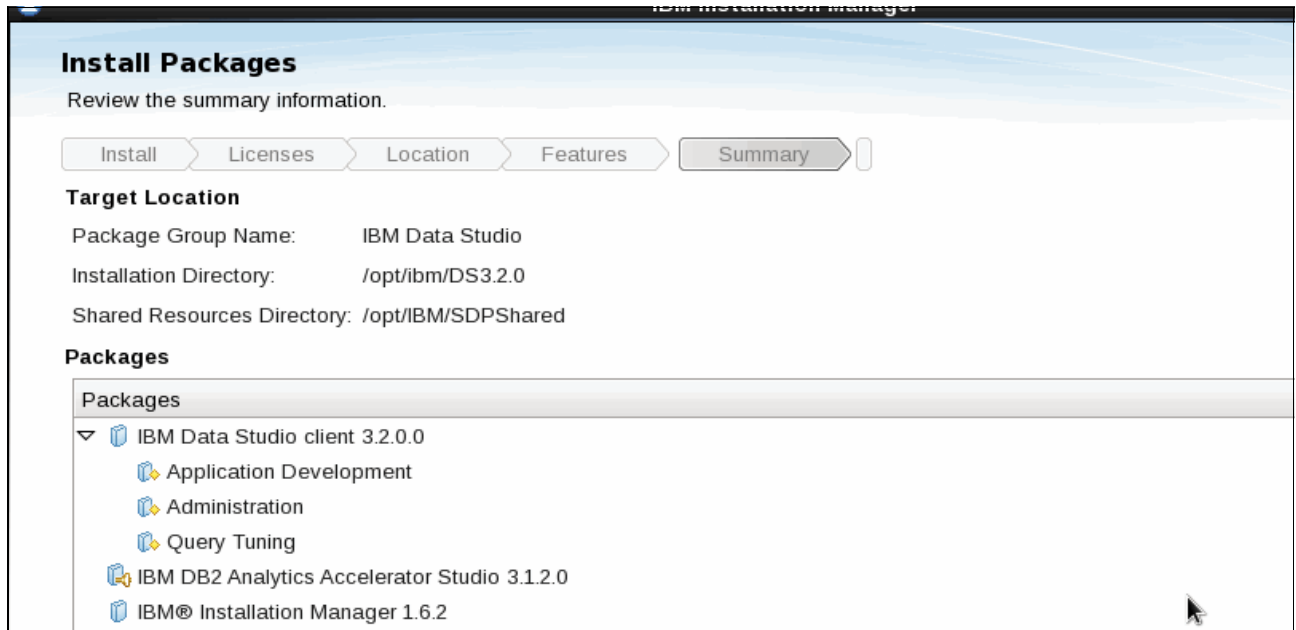


Figure 2-9 Studio installation progress screen capture

g. Figure 2-10 shows the final window after successful installation of the product.

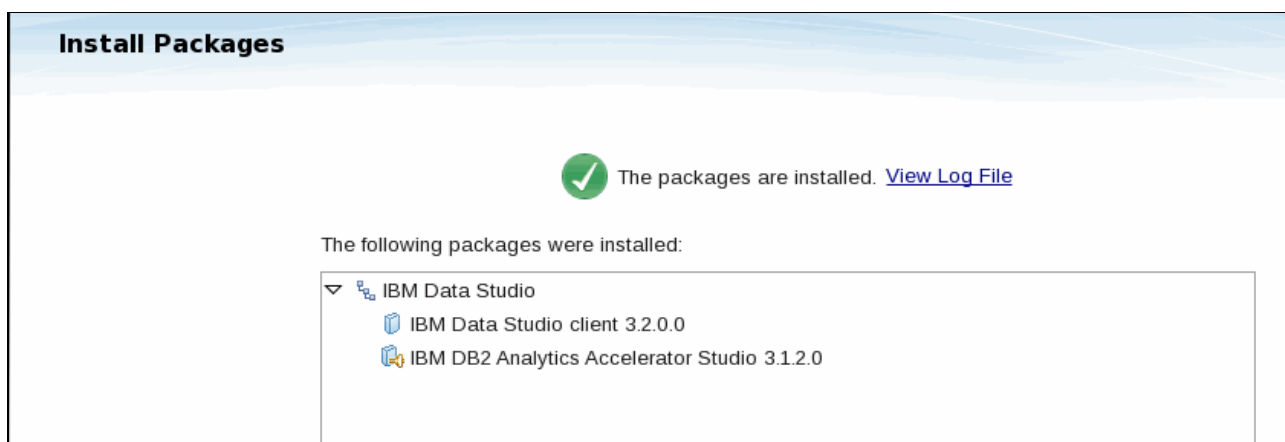


Figure 2-10 Studio installation: Completion window

Alternative installation options

In the following cases, you might want to download a compressed archive of the IBM DB2 Analytics Accelerator Studio repository for IBM Installation Manager rather than following the previous instructions:

- ▶ If you do not have a direct Internet connection or your intranet uses a proxy server to connect to the Internet
- ▶ If you want to install the product on many computers without an Internet connection

You can download the repository from the following location:

<http://download4.boulder.ibm.com/sar/CMA/IMA/03quc/0/idaav3ptf2.zip>

Then point IBM Installation Manager to the downloaded, extracted repository.

See the IBM Installation Manager support page for more information:

http://www.ibm.com/support/entry/portal/overview/software/rational/ibm_installation_manager

If you encounter problems with IBM Installation Manager, you can continue to use the one-step installation program on the IBM DB2 Analytics Accelerator for z/OS V3.1 product DVD. The installation program on the DVD is based on InstallAnywhere technology and will install an earlier release of IBM Data Studio (such as V3.1.1). However, if you choose this option, you will not be able to use the new features that were introduced with IBM Data Studio V3.2. In this case, you can upgrade the Studio software to the most recent level using the processes documented in the product online help (as applicable, enter one of the following topics in the Search box and follow the instructions):

- ▶ Updating IBM DB2 Analytics Accelerator Studio from a workstation without an Internet connection
- ▶ Updating IBM DB2 Analytics Accelerator Studio from a workstation with an Internet connection
- ▶ Updating IBM DB2 Analytics Accelerator Studio from another product

Note: If possible, install IBM DB2 Analytics Accelerator Studio on a local disk. Installing IBM DB2 Analytics Accelerator Studio on a shared network drive has the potential risk of damaging or even destroying the Eclipse workspace.

2.2.3 Accelerator plug-ins only installation

You can also add the plug-ins to an existing installation of earlier or current versions of Data Studio or products based on Data Studio.

The following combinations are supported:

- ▶ Data Studio V3.1.1 Administration Client
- ▶ Data Studio V3.1.1 Full Client

The plug-ins and fixes to IBM DB2 Analytics Accelerator Studio are available from the following download site:

<http://public.dhe.ibm.com/ibmdl/export/pub/software/data/db2/analytics-accelerator-studio>

The procedure to upgrade these products with IBM DB2 Analytics Accelerator Studio are described in Chapter 10, “Installing Updates” under the section “Updating IBM DB2 Analytics Accelerator Studio from another product” of *IBM DB2 Analytics Accelerator for z/OS Version 3.1.0 Installation Guide*, SH12-6983.

For silent installations, refer to the Data Studio V3.2 documentation that is available at the following site:

http://pic.dhe.ibm.com/infocenter/dstudio/v3r2/topic/com.ibm.datatools.base.install.doc/topics/t_installing_silently_overview11.html

2.2.4 Dump file on Linux

If the Accelerator Studio crashes while using it, check the dump file for additional diagnostics information. Most often, this can be due to your installation not meeting the prerequisites. Read “Prerequisites for running DB2 Analytics Accelerator Studio on Linux” on page 16 and ensure that you have implemented all the prerequisites.

On Linux, you can typically locate the dump file named `javacore.20130507.072349.29308.0001.txt` that is written in the current working directory where you started Data Studio executable, or in the `/tmp` directory.

2.2.5 Accelerator Studio Help menu

There is a wealth of information that is available from the Accelerator Studio product *Help* documentation. This online help is searchable and can be accessed from the product *Help* menu via **Help** → **Search**.

Context-sensitive help is a mechanism that enables linking a user interface part with a specific help topic. When a user activates the associated user interface part, the help page is displayed in the dynamic help area (usually on the right hand side of the window). Context-sensitive help is available via **Help** → **Dynamic Help**.

Novice users can also get an overview and basic introduction to Studio tasks by selecting the **Welcome** option item on the Help menu, as depicted in Figure 2-11 on page 24.

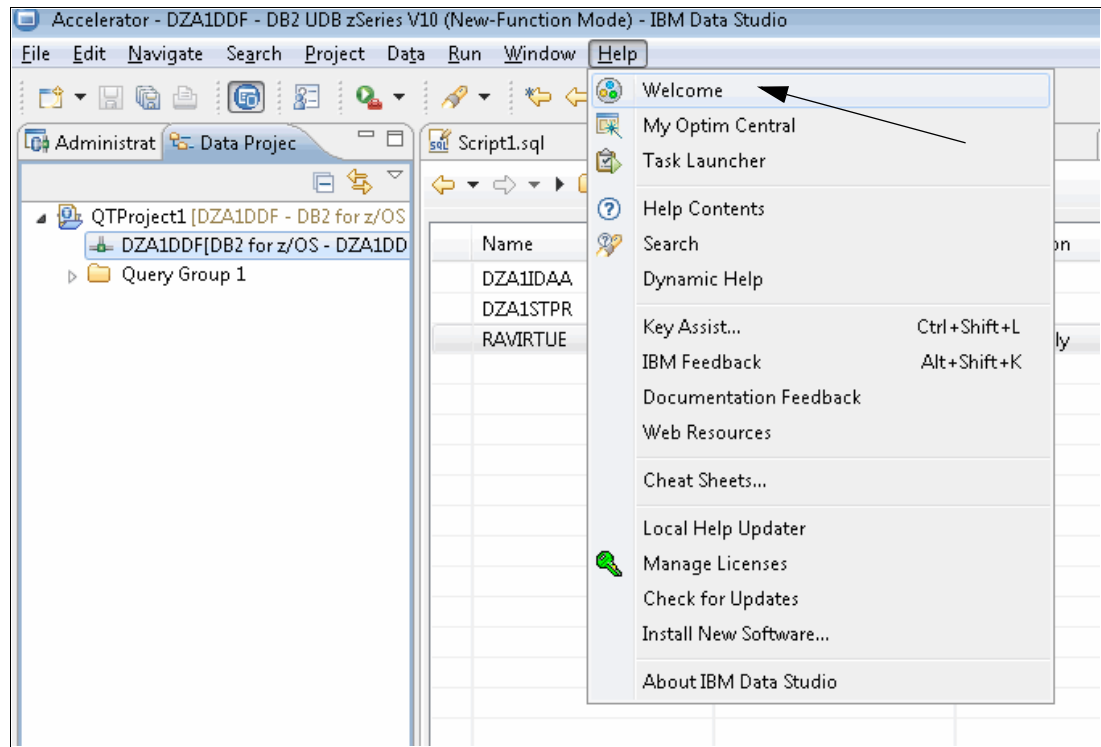


Figure 2-11 Studio Help menu for beginners

The Welcome window provides useful links to Accelerator Studio documentation. It also provides a link to demonstration video playbacks (without voice) for common tasks that are performed from the Studio. Figure 2-12 shows the links to the demonstration videos (under the *First Steps* area on the right side) and the *Overview* documentation (on the left side).

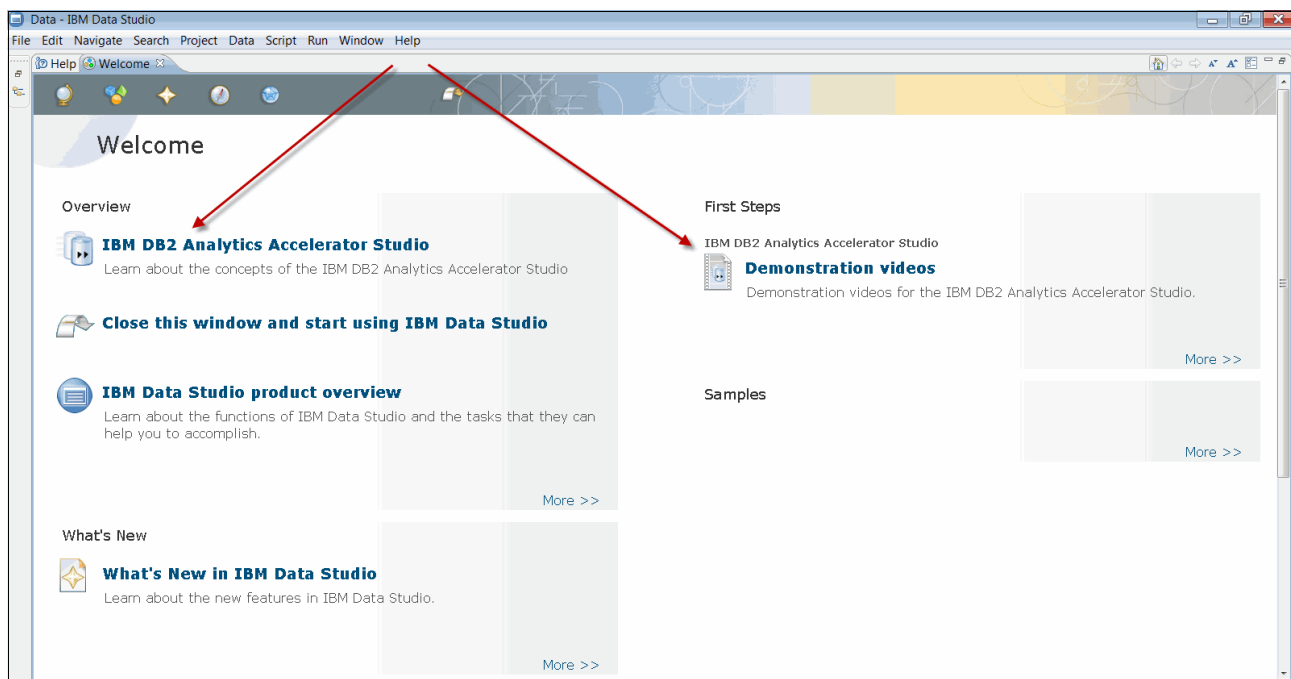


Figure 2-12 Navigating the Help menu to Studio documentation and demonstration videos

2.3 Configure Studio for tuning

This task can only be performed after completing the task described in the section “Creating a connection profile to a DB2 for z/OS subsystem” on page 28.

Before you start using Visual Explain from the Accelerator Studio or the EXPLAIN statement with `CURRENT QUERY ACCELERATION = ENABLE` setting, you should create all the standard EXPLAIN tables plus the additional `DSN_QUERYINFO_TABLE`. One option to create all the EXPLAIN tables is shown in Figure 2-13. From the Data Source Explorer perspective, select your DB2 subsystem location by right-clicking the location name (DZA1DDF on the sample screen shot in Figure 2-13), followed by selecting **Analyze and Tune** → **Configure for Tuning** → **Guided Configuration**.

Follow the instructions in the wizard and enter the required information to complete the task. If you do not have any of the EXPLAIN tables installed on the connected DB2 subsystem, this would create all the EXPLAIN tables including `DSN_QUERYINFO_TABLE`.

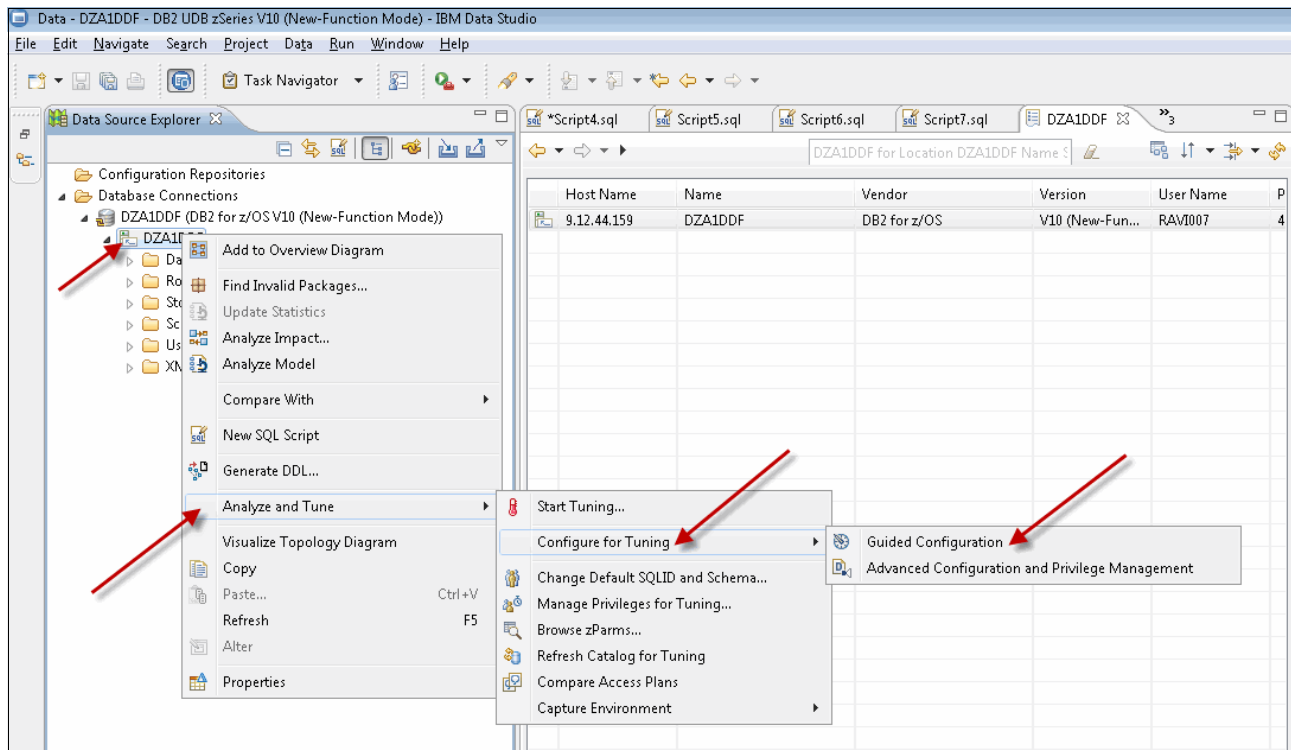


Figure 2-13 Studio: Create missing EXPLAIN tables including `DSN_QUERYINFO_TABLE`

If only the `DSN_QUERYINFO_TABLE` is missing in your DB2 for z/OS environment, you can still use the process outlined above to automatically create the missing `DSN_QUERYINFO_TABLE` under your schema. For more information about how to configure your DB2 for z/OS subsystems for tuning SQL statements, refer to the Data Studio V3.2 documentation available at the following URL:

<http://pic.dhe.ibm.com/infocenter/dstudio/v3r2/nav/4>

When all the tuning options are exhausted and if your dynamic queries are still not meeting the expected performance levels (or service level agreements), it is time to use the IBM DB2 Analytics Accelerator for z/OS appliance. A summary of the minimum tasks that are required to accelerate an SQL statement is provided in Figure 2-14 on page 28.

Unlike the earlier versions of the Accelerator Studio, you no longer need to create the DSN_QUERYINFO_TABLE manually using the Data Definition Language (DDL) from the DSNTESC job in the DB2 SDSNSAMP library, but it is still an option.

2.4 Accelerator Studio tasks

In this section, we provide some insights on the tasks that can be performed from the Accelerator Studio. These tasks are discussed in detail in the *IBM DB2 Analytics Accelerator for z/OS Version 3.1.0 User's Guide*, SH12-6985.

Prerequisites

Before you begin any of the tasks described in this section, check whether you meet the requirements for running IBM DB2 Analytics Accelerator Studio:

- ▶ Network connections must exist between the database server and the machine on which IBM DB2 Analytics Accelerator Studio is installed.
- ▶ Network connections must exist between the DB2 for z/OS database server and the Accelerators (hardware).
- ▶ At least one Accelerator (hardware) must be installed and connected to the DB2 for z/OS database server.
- ▶ Appropriate program temporary fixes (PTFs) must be installed so that the IBM DB2 Analytics Accelerator for z/OS can connect to your DB2.
- ▶ IBM DB2 Analytics Accelerator for z/OS stored procedures must exist on the DB2 for z/OS database server.
- ▶ IBM DB2 provided stored procedures must exist on the DB2 for z/OS server.
- ▶ IBM DB2 Analytics Accelerator for z/OS jobs must have been run to configure and integrate the Accelerator hardware and software into your DB2 for z/OS environment.

2.4.1 Summary of common tasks and the overall process flow

The following topics were described in *Optimizing DB2 Queries with IBM DB2 Analytics Accelerator for z/OS*, SG24-8005 in detail with the earlier version of the Accelerator Studio. Because the steps are the same as Version 3.1 of the Accelerator Studio, these tasks are not discussed in detail in this book:

- ▶ Creating a connection profile to a DB2 subsystem
- ▶ Adding an Accelerator to a DB2 subsystem
- ▶ Adding tables to an Accelerator
- ▶ Loading tables into an Accelerator
- ▶ Enabling and disabling a table for query acceleration

In addition, the following topics are also available on the Accelerator Studio online help documentation, as shown in Figure 2-12 on page 24 (via the *Overview* link):

- ▶ Connecting to a database server
- ▶ Exploring a database server
- ▶ Obtaining the pairing code for authentication and completing the authentication
- ▶ Adding virtual accelerators
- ▶ Displaying Accelerator networking details
- ▶ How to select tables for query acceleration

- ▶ Determining the status of an Accelerator
- ▶ Enabling an Accelerator (**-START ACCEL DB2** command)
- ▶ Disabling an Accelerator (**-STOP ACCEL DB2** command)
- ▶ Enabling or disabling an Accelerator for members of a data sharing group
- ▶ Specifying or changing a distribution key or organizing keys
- ▶ Running an SQL script from IBM DB2 Analytics Accelerator Studio
- ▶ EXPLAIN information (based on DSN_QUERYINFO_TABLE)
- ▶ Displaying an access plan graph (for different values of CURRENT QUERY ACCELERATION and GET_ACCEL_ARCHIVE special registers)
- ▶ Nodes in access plan graphs
- ▶ Selecting and ordering query monitoring columns
- ▶ Canceling tasks (queries or load operations)
- ▶ Removing tables from an Accelerator
- ▶ Exporting a table specification
- ▶ Importing a table specification
- ▶ Removing Accelerators

Typical process flow of basic tasks performed on Studio

Figure 2-14 on page 28 depicts a typical process flow of all the key tasks that can be performed from the Accelerator Studio.

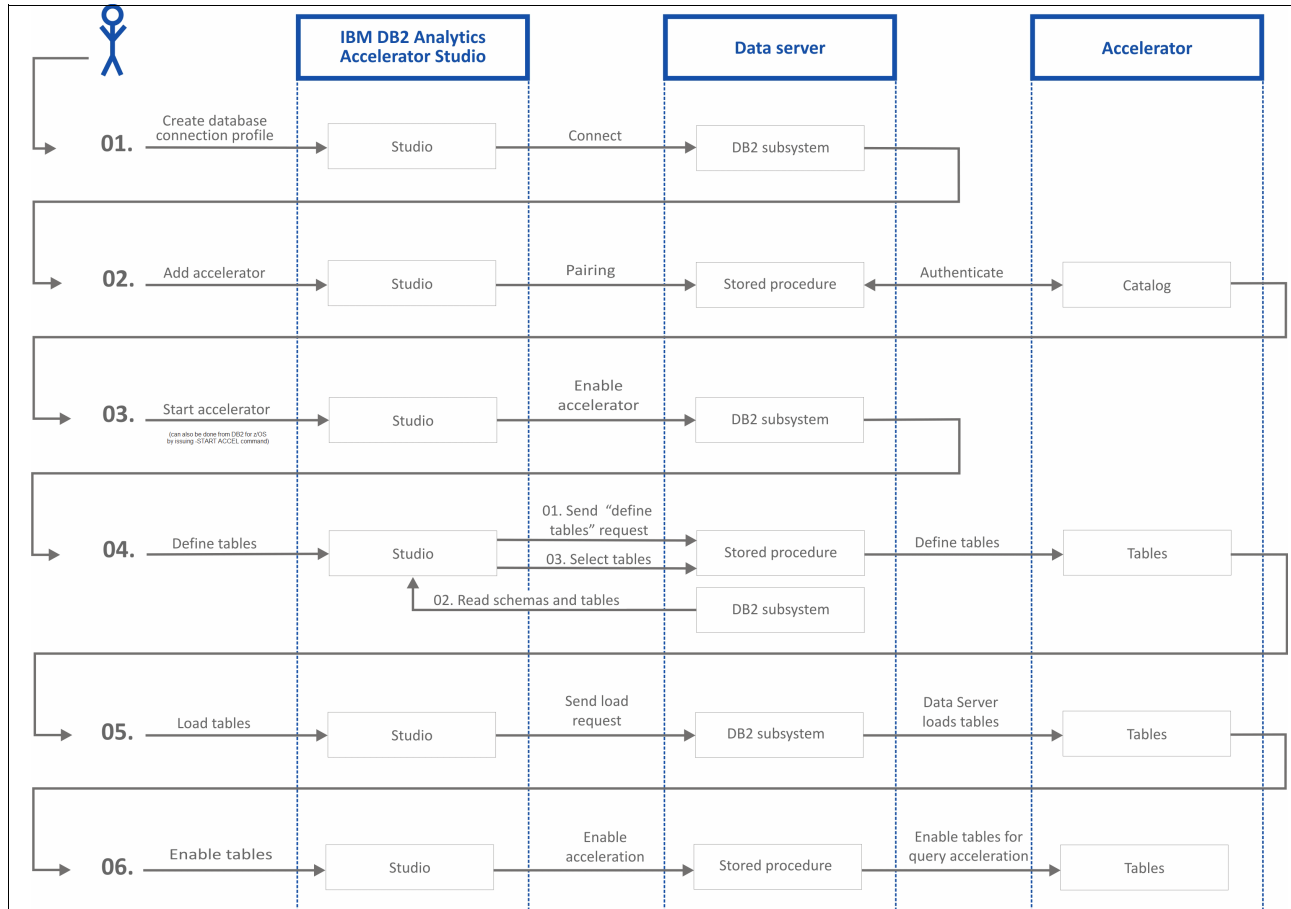


Figure 2-14 Typical process flow of tasks performed on Accelerator Studio

2.4.2 Sample screen captures on various activities from the Studio

In this section, some sample screen captures from Accelerator Studio (based on Data Studio V3.2) are provided along with some useful tips.

Creating a connection profile to a DB2 for z/OS subsystem

This task can be performed even before the Accelerator is installed.

You can use the pages in the New Connection wizard to create a connection profile so that you can connect to and browse existing data objects. This task is the same as what is available from other versions of Data Studio (that is, it is not unique to the Accelerator Studio). You can perform any one of the following tasks to create a connection profile:

- Right-click in the Data Source Explorer's **Database Connections** folder, and select **New Connection** from the pop-up menu or click the symbol (hand-in-hand with a plus sign) that is shown in Figure 2-15 on page 29.

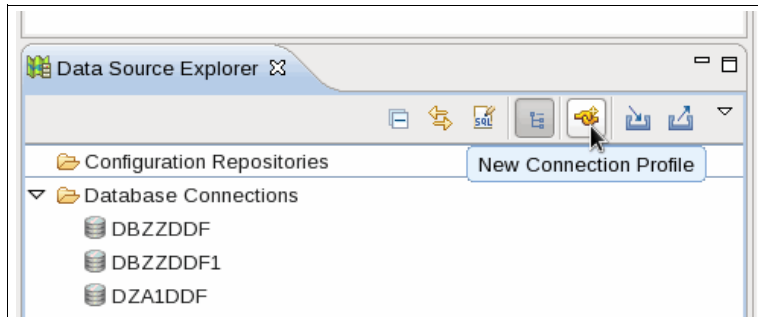


Figure 2-15 Creating a new connection profile from a Data Source Explorer

Enter the profile details that are shown in Figure 2-16, which shows the common panel that is used to enter the connection profile parameters.

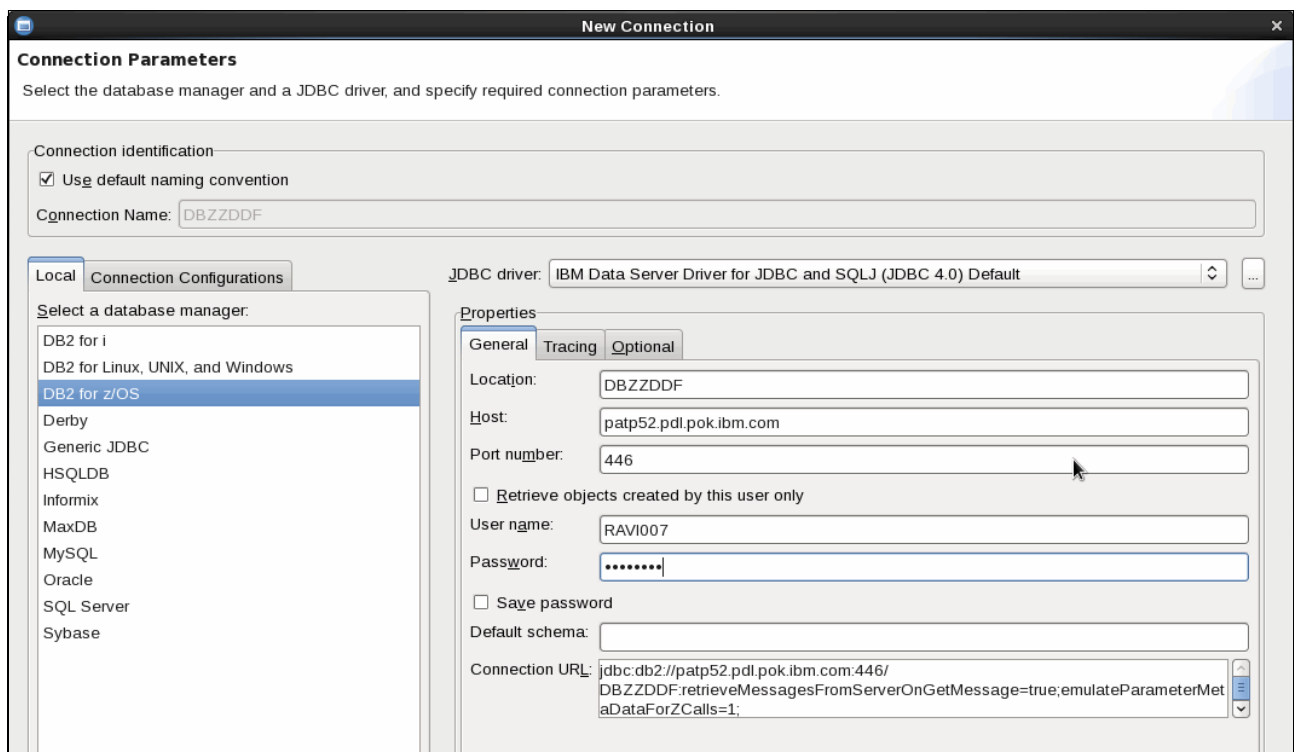


Figure 2-16 Connection parameter entry panel

- For products that contain the Administration Explorer, you can also click the down arrow next to New and select **New Connection to a database**, as shown in Figure 2-17.

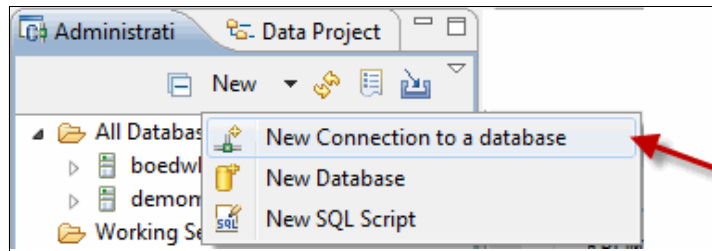


Figure 2-17 Creating a new connection profile from Administration Explorer

- You can also create a new connection by creating a duplicate of an existing connection. This method is useful if you want to create a connection that is similar to an existing connection but with different properties. To create a duplicate of an existing connection, right-click a connection and select Duplicate. You can then edit the properties of the duplicate connection as needed.

Adding an Accelerator to a DB2 system by pairing

This is a mandatory task that has to be performed prior to using any Accelerator functions. You need to create a location name for IBM Distributed Relational Database Architecture™ (IBM DRDA®) connectivity. It is a unique name that helps you identify the target accelerator and shows up in display and error messages. This is once per DB2 subsystem or data sharing group, but then each member will have to start individually.

Pairing also involves obtaining the pairing code for authentication and completing the authentication. This is performed once for a given pair of a DB2 subsystem and an Accelerator. Therefore, if you have two Accelerators connected to the same DB2 subsystem, you need to perform this task twice. Similarly, if two Accelerators are connected to two DB2 subsystems, you need to perform this task four times.

Obtaining the pairing code for authentication (done outside the Studio)

The pairing is performed via the IBM DB2 Analytics Accelerator Configuration console. Other tasks that can be carried out through the IBM DB2 Analytics Accelerator Configuration console are described “IBM DB2 Analytics Accelerator Configuration console” on page 31.

Communication between an Accelerator and a DB2 subsystem requires both components to share credentials. These credentials are generated after you submit a temporarily valid pairing code.

Note: This step is required each time that you add a new Accelerator. This includes performing a high availability configuration

Here are the steps for pairing the two systems:

1. Obtain the IP address of the Accelerator from your network administrator.
2. Start a 3270 emulator and log on to the Time Sharing Option/Interactive System Productivity Facility (TSO/ISPF).

3. Enter the following command as shown in Example 2-1:

```
tso telnet <hostname> 1600
```

Where:

<hostname>	This is the IP address of the Accelerator that is connected to the DB2 for z/OS data server.
1600	This is the number of the port that is configured for accessing the IBM DB2 Analytics Accelerator Console using a Telnet connection between the DB2 for z/OS data server and the Accelerator.

Example 2-1 Command used to Telnet to the DB2 Analytics Accelerator console

```
tso telnet 10.199.9.199 1600
```

4. Press Enter until you receive a prompt to enter the console password. Enter your console password (Example 2-2) and press Enter. The initial DB2 Analytics Accelerator console password is **dwa-1234**, and it is case-sensitive. After you log in for the first time, you will be prompted to change the console password.

Forgotten password: If you forgot the password, you need to open an IBM Service ticket to get a new password. But, if you simply need to change the password, you can use the “IBM DB2 Analytics Accelerator Configuration console” to set a new password.

During normal login, you are prompted to enter the password as shown in Example 2-2.

Example 2-2 Password entry panel to enter the DB2 Analytics Accelerator configuration console

Using Line Mode...

Notes on using Telnet when in Line Mode:

- To hide Password, Hit PF3 or PF15
 - To enter Telnet Command, Hit PF4-12, or PF16-24
- ***

Enter password (in TS0, use PF3 to hide input):

You are then presented with the IBM DB2 Analytics Accelerator Configuration console panel, as shown in Example 2-3.

IBM DB2 Analytics Accelerator Configuration console

The following panels are samples with IBM PureData System for Analytics N2001 networked with a DB2 10 for z/OS subsystem on a zEC12 mainframe box.

Configuration console: From the Configuration console, you can perform other activities, such as changing the configuration console password and managing incremental updates.

Example 2-3 DB2 Analytics Accelerator Configuration console: Welcome panel

Licensed Materials - Property of IBM

5697-DAA

(C) Copyright IBM Corp. 2009, 2013.

US Government Users Restricted Rights -

Use, duplication or disclosure restricted by GSA ADP Schedule

Contract with IBM Corporation

```
*****
* Welcome to the IBM DB2 Analytics Accelerator Configuration Console
*****
```

You have the following options:

- (1) - Change the Configuration Console Password
- (2) - (Menu) Run Netezza Commands
- (3) - (Menu) Run Accelerator Functions
- (4) - (Menu) Manage Incremental Updates

(x) - Exit the Configuration Console

3

-
5. From the Welcome panel, type 3 and press Enter to display the submenu for **Run Accelerator Functions**.

After you press the *clear screen* key, you are presented with the Accelerator functions panel, as shown in Example 2-4.

Example 2-4 DB2 Analytics Accelerator configuration console: Accelerator functions panel

```
main -> 3
```

You have the following options:

- (0) - Go back one level
- (1) - Obtain pairing code, IP address, and port
- (2) - List paired DB2 subsystems
- (3) - Set resource limits for DB2 subsystems
- (4) - Clear query history
- (5) - Specify the priority of maintenance tasks
- (6) - Set the DB2 subsystem for time synchronization

(Default 0) > 1

Specify for how long you want the pairing code to be valid. Enter a value between 5 and 1440 minutes. Press <return> to accept the default of 30 minutes. Cancel the process by entering 0.

(

-
6. Type 1 and press Enter to select **Obtain pairing code, IP address, and port**.

Configuration console: From the Acceleration functions menu, you can perform other functions such as clearing the query history, specifying the priority of maintenance tasks, setting resource limits for DB2 subsystems, as shown in Example 2-4.

7. When the message *Specify for how long you want the pairing code to be valid.* is displayed, enter an appropriate integer to specify the validity period in minutes. The time that you choose must be sufficient for you or a coworker to go to the workstation that runs IBM DB2 Analytics Accelerator Studio, start the Add New Accelerator wizard, and enter the information that is returned by the console. Values 5 - 1440 are allowed. If you just press Enter, you accept the default of 30 minutes.

Important: A pairing code is valid for a single try only. Furthermore, the code is bound to the IP address that is displayed on the console.

8. Make a note of the following information about the console:
 - Pairing code
 - IP address
 - Port
9. Press Enter to return to the main menu of the console.
10. Type x and press Enter to exit the console and close the Telnet session.

Adding Accelerators

Consider the following factors before adding an Accelerator:

- You need privileges to run DB2 administration commands and stored procedures on z/OS.
- You need to create the connection profile of the DB2 subsystem that you are trying to pair your Accelerator with by completing the steps described under “Creating a connection profile to a DB2 for z/OS subsystem” on page 28.
- Do not give ordinary users SELECT authorization on the SYSIBM.USERNAMES table because this allows the users to see the authentication information in clear text in the SYSIBM.USERNAMES.NEWAUTHID column.
- Making a new backup of your DB2 catalog tables is strongly recommended after each authentication update because restoration processes in your DB2 subsystem can make an Accelerator unusable. This happens if you must restore your DB2 catalog and the backup of the catalog was made before the last update of the Accelerator credentials. In this case, the latest authentication information will not be in the catalog tables of the backup, and so the Accelerator can no longer be used.

By using the pairing code obtained in step 8 above, you can add the Accelerator via the Administration Explorer view from the Studio client using the following steps:

11. Select the Accelerators folder in the Administration Explorer.
12. On the menu bar of the Object List Editor, click the downward-pointing arrow next to the green plus sign.
13. From the drop-down menu, select **Add Accelerator**.
14. In the Name field, type a name for the Accelerator. This name is automatically copied to the Location field. The location name is the unique name of the Accelerator in the SYSIBM.LOCATIONS table. Mostly, this is the same name as the Accelerator name.

Restriction: An Accelerator cannot be shared between two or more DB2 subsystems if the subsystems use the same location name. If you copy an entire subsystem, ensure that you change the location name of the copy afterwards.

15. In the Pairing code field, type the pairing code.
16. In the IP address field, type the IP address of the Accelerator.
17. In the Port field, type 1400. This is the fixed port for network communication between the z/OS data server and the Accelerator.
18. Click **Test Connection** to check whether the Accelerator with the given address can be connected to.
19. Click **OK** to complete this task.

Starting an Accelerator (enabling)

This task is described under “Enabling an Accelerator” in Chapter 6, Tasks Overview of the *DB2 Analytics Accelerator for z/OS Version 3.1.0 User's Guide*, SH12-6985. This task can also be performed from the DB2 subsystem using the **-START ACCEL** command. This task can be performed only when the Accelerator is in the *stopped* state.

From the Accelerator Studio, you can click the **Start** link (next to *Stopped*) in the top left part of the Accelerator view, as shown in Figure 2-18.

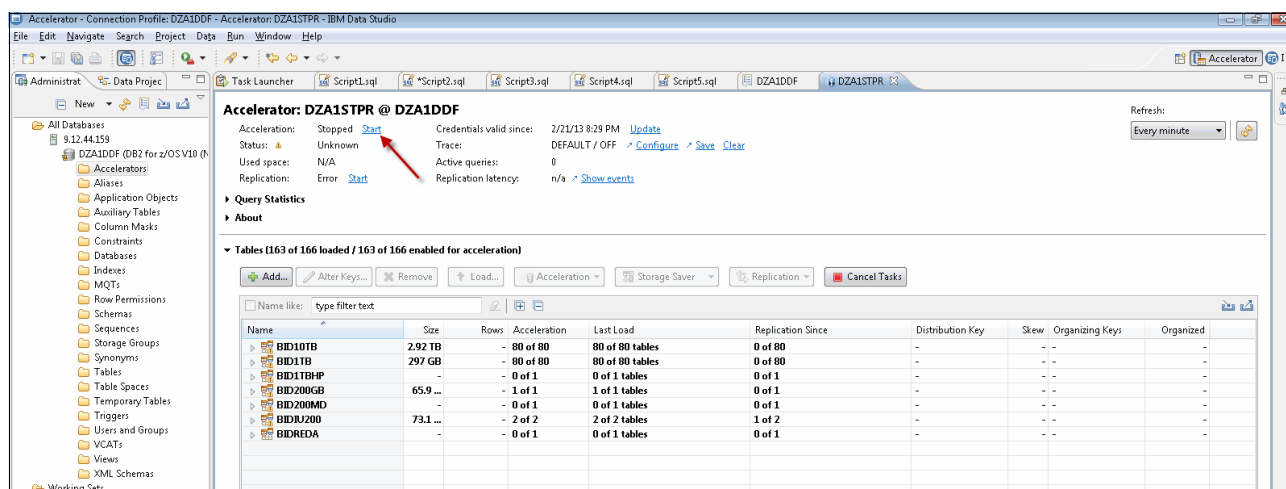


Figure 2-18 Start Accelerator from Studio: Administration Explorer

You are then presented with a confirmation window, as shown in Figure 2-19. Click **Yes** to proceed.

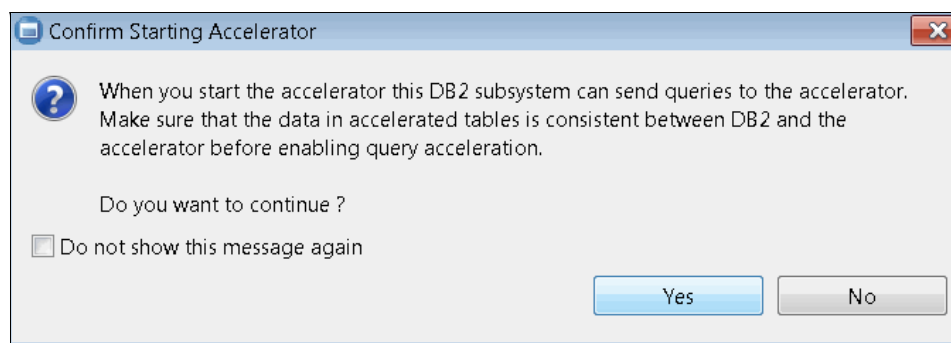


Figure 2-19 Starting an Accelerator: Confirmation window

Tip: Because the tables involved can be loaded even when the Accelerator is in the Stopped state, it is the responsibility of the database administrators to ensure that the data is consistent between DB2 and the Accelerator before starting the Accelerators, or at least before the queries are expected to be routed to this Accelerator.

Figure 2-20 on page 35 depicts the change in the state of the Accelerator from *Stopped* to *Started* (and the status is changed from *Unknown* to *Online*) upon successful completion of this task.

Accelerator: DZA1STPR @ DZA1DDF			
Acceleration:	Started Stop	Credentials valid since:	2/21/13 8:29 PM Update
Status:	Online	Trace:	REPLICATION / SPPROFILING Configure Save Clear
Used space:	3.72 TB of 45.8 TB	Active queries:	0
Replication:	Error Start	Replication latency:	n/a Show events

Figure 2-20 Accelerator status after starting it from Studio

Stopping an Accelerator (disabling)

This task is described under “Disabling an Accelerator” in the *DB2 Analytics Accelerator for z/OS Version 3.1.0 User's Guide*, SH12-6985. This task can also be performed from the DB2 subsystem using the **-STOP ACCEL** command. From the Accelerator Studio, you can click the *Stop* link in the top left part of the Accelerator view (next to *Started*), which is also showing in Figure 2-20 for your reference.

You are then presented with a confirmation window, as shown in Figure 2-21. Click **Yes** to proceed.

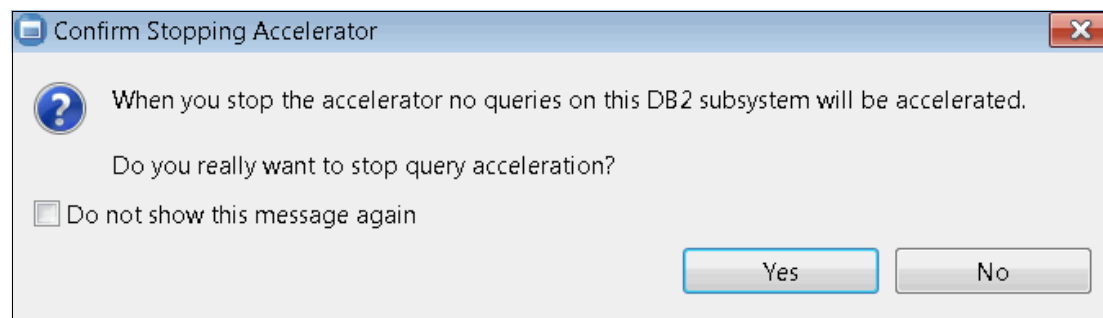


Figure 2-21 Stopping an Accelerator: Confirmation window

The Accelerator status first changes to Stopping, then to Stopped. During the Stopping phase, running queries are completed. You are also given a choice to select *Force* from the menu to disable the selected Accelerator and cancel all running queries, as shown in Figure 2-22. In this case, the status of the Accelerator changes to Stopped immediately.

Accelerator: DZA1STPR @ DZA1DDF			
Acceleration:	Stopping. Force	Credentials valid since:	2/21/13 8:29 PM Update
Status:	Online	Trace:	REPLICATION / SPPROFILING Configure Save Clear
Used space:	3.72 TB of 45.8 TB	Active queries:	0
Replication:	Error Start	Replication latency:	n/a Show events

Figure 2-22 Stopping an Accelerator: Force option from the Studio

Upon successful completion of this task, as shown in Figure 2-18 on page 34, the state of the Accelerator is changed from *Started* to *Stopped* (and the status is changed from *Online* to *Unknown*).

Additional information about Data Sharing considerations is discussed in the “Enabling or disabling an Accelerator for data sharing group” section of the *DB2 Analytics Accelerator for z/OS Version 3.1.0 User's Guide*, SH12-6985.

Adding virtual accelerators

Virtual accelerators are simulators used for query evaluation. Using a virtual Accelerator, you can check whether enough queries can be accelerated to justify acceleration.

1. Select the **Accelerators** folder in the Administration Explorer.
2. On the menu bar of the Object List Editor, click the downward-pointing arrow next to the green plus sign.
3. From the drop-down menu, select **Add Virtual Accelerator**.
4. In the Add Virtual Accelerator window, in the Name field, type a name for the virtual accelerator.
5. Click **OK**.

You cannot load data into a virtual accelerator. But, you can complete the following steps in the same way as for a regular Accelerator:

6. Define tables on the virtual accelerator.
7. Enable acceleration for these tables as necessary.
8. Submit queries to the virtual accelerator.

Virtual Accelerator's Explain versus real Accelerator's Visual Explain

With the virtual accelerators, you do not get any access path information for tuning. It would simply indicate whether or not your SQL statement is eligible for routing to the Accelerator. The real Accelerator's Visual Explain provides detailed access path information.

Adding tables to an Accelerator

Figure 2-23 shows a sample Accelerator view screen capture from where you can add tables to an Accelerator.

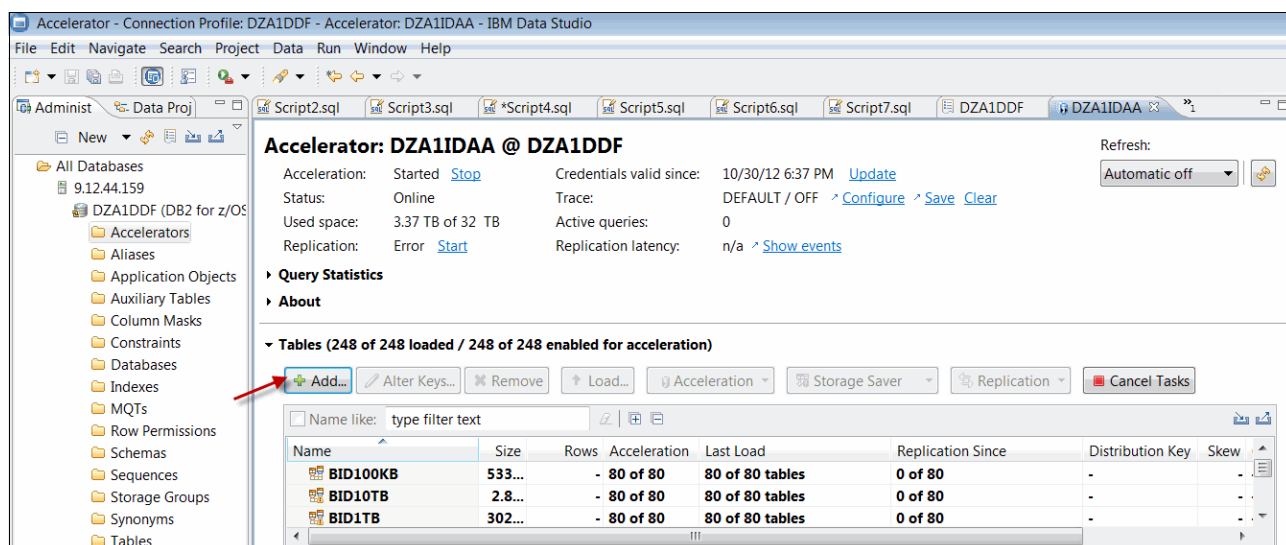


Figure 2-23 Studio Accelerator view to perform Add Tables task

Figure 2-24 shows the Add Tables window, where you can optionally specify the filter text (either the schema name or the table name) to short list the tables and then select the tables to add to the Accelerator.

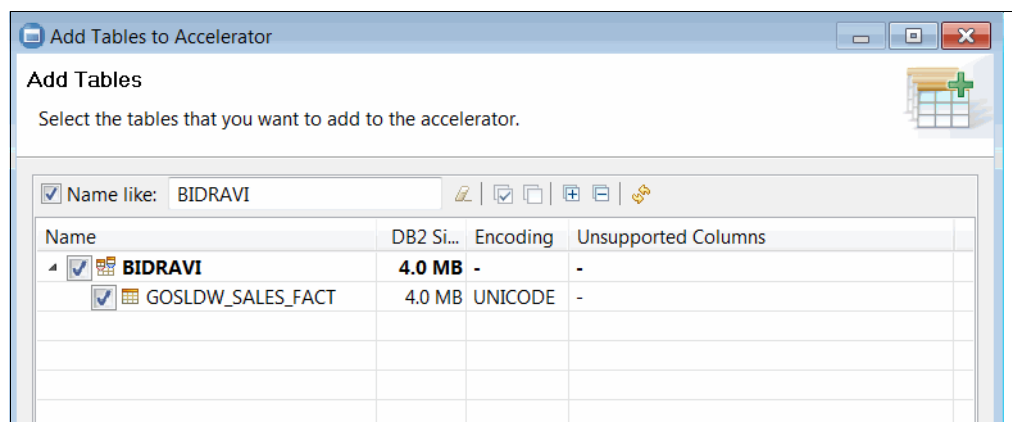


Figure 2-24 Defining tables to an Accelerator

This task is carried out by the SYSPROC.ACCEL_ADD_TABLES stored procedure. For information about the privileges that are required to run this procedure and further details, see the appropriate section in the *IBM DB2 Analytics Accelerator for z/OS Version 3.1.0 Stored Procedures Reference*, SH12-6984. The names of tables that have already been added to this accelerator are grayed out. The Unsupported Columns column lists the unsupported columns in each table, if any. If possible, do not select tables with unsupported columns for query acceleration because it is very likely that these will cause errors. In most cases, a column is not supported because its data type cannot be handled by IBM DB2 Analytics Accelerator for z/OS.

This task updates the SYSACCEL.ACCELERATEDTABLES table on the DB2 for z/OS side, and hence there is a potential for a locking/concurrency issue here. Therefore, do not grant update access to this table to users who do not need access to this table to avoid locks.

Loading tables into an Accelerator

Loading is a mandatory operation that must be carried out after defining a table to a given Accelerator. Apart from the volume of data, following are some of the key factors that can influence the load performance:

- ▶ Partitioning of tables in your DB2 subsystem. In DB2 for z/OS, this determines the degree of parallelism that can be used internally by the DB2 **UNLOAD** utility.
- ▶ Number of DATE, TIME, and TIMESTAMP columns in your table.
- ▶ Compression of data in DB2 for z/OS.
- ▶ Workload Manager (WLM) configuration.
- ▶ Workload on the zEnterprise server during the UNLOAD phase.
- ▶ Concurrent workload on the Accelerators.

Chapter 3, “Data latency management” on page 55 provides details about load performance.

The load activity can be performed via simple point and click from the Accelerator folder on the Administration Explorer by selecting the table to be loaded and clicking **Load**, as shown in Figure 2-25.

Accelerator: DZA1IDAA @ DZA1DDF

Acceleration: Started [Stop](#) Credentials valid since: 10/30/12 6:37 PM [Update](#)
 Status: Online Trace: [Trace](#) DEFAULT / SPPROFILING [Configure](#) [Save](#) [Clear](#)
 Used space: 3.32 TB of 32 TB Active queries: 0
 Replication: Error [Start](#) Replication latency: n/a [Show events](#)

Refresh: Automatic off

► Query Statistics
 ► About

► Tables (248 of 249 loaded / 248 of 249 enabled for acceleration)

[Add...](#) [Alter Keys...](#) [Remove](#) [Load...](#) [Acceleration](#) [Storage Saver](#) [Replication](#) [Cancel Tasks](#)

Name like: type filter text

Name	Size	Rows	Acceleration	Last Load	Replication Since	Distribution Key	Skew	Organizing Keys	Organiz...
BID100KB	533...	- 80 of 80	80 of 80 tables	0 of 80	-	-	-	-	-
BID10TB	2.8...	- 80 of 80	80 of 80 tables	0 of 80	-	-	-	-	-
BID1TB	302...	- 80 of 80	80 of 80 tables	0 of 80	-	-	-	-	-
BID200GB	65....	- 1 of 1	1 of 1 tables	0 of 1	-	-	-	-	-
BID200MD	-	- 0 of 1	0 of 1 tables	0 of 1	-	-	-	-	-
GOSLDW_SALES_FAC	-	- Disabled	Initial load pending	Disabled	Random	0.000	-	-	-
BIDIU200	71....	- 6 of 6	6 of 6 tables	0 of 6	-	-	-	-	-

Figure 2-25 Studio: Load button on the Accelerator folder of the Administration Explorer

Clicking **Load** opens another window as shown in Figure 2-26 if a partitioned table is selected, or Figure 2-27 on page 39 if multiple tables are selected.

Load Tables

Load Tables / Partitions

Select the whole table or partitions to be loaded from DB2 for z/OS to the accelerator.

Name	Partitioned By	Load Recommended	DB2 Size	Schema
<input checked="" type="checkbox"/> GOSLDW_SALES_F4 ORDER_DAY_KEY		Yes	29 MB	BIDRAVI
<input checked="" type="checkbox"/> Partition 1	Ending at '20040101'	Yes - load replace	0.0 B	
<input checked="" type="checkbox"/> Partition 2	Ending at '20040201'	Yes - load replace	0.0 B	
<input checked="" type="checkbox"/> Partition 3	Ending at '20040301'	Yes - load replace	0.0 B	
<input checked="" type="checkbox"/> Partition 4	Ending at '20040401'	Yes - load replace	0.0 B	
<input checked="" type="checkbox"/> Partition 5	Ending at '20040501'	Yes - load replace	0.0 B	
<input checked="" type="checkbox"/> Partition 6	Ending at '20040601'	Yes - load replace	0.0 B	
<input checked="" type="checkbox"/> Partition 7	Ending at '20040701'	Yes - load replace	0.0 B	
<input checked="" type="checkbox"/> Partition 8	Ending at '20040801'	Yes - load replace	0.0 B	
<input checked="" type="checkbox"/> Partition 9	Ending at '20040815'	Yes - load replace	0.0 B	
<input checked="" type="checkbox"/> Partition 10	Ending at '20040901'	Yes - load replace	0.0 B	
<input checked="" type="checkbox"/> Partition 11	Ending at '20040906'	Yes - load replace	0.0 B	
<input checked="" type="checkbox"/> Partition 12	Ending at '20040911'	Yes - load replace	0.0 B	
<input checked="" type="checkbox"/> Partition 13	Ending at '20040916'	Yes - load replace	0.0 B	
<input checked="" type="checkbox"/> Partition 14	Ending at '20040921'	Yes - load replace	0.0 B	
<input checked="" type="checkbox"/> Partition 15	Ending at '20040926'	Yes - load replace	0.0 B	
<input checked="" type="checkbox"/> Partition 16	Ending at '20041001'	Yes - load replace	0.0 B	
<input checked="" type="checkbox"/> Partition 17	Ending at '20041006'	Yes - load replace	0.0 B	

☐ Lock DB2 tables while loading: All Tables

☒ After the load, enable acceleration for disabled tables.

Amount of data to load: 29 MB

Note: This accelerator client must stay connected until the tables are fully loaded. This can take a long time depending on the amount of data to load, for example several minutes or hours.

Figure 2-26 Load activity: Single table with multiple partitions selected

Figure 2-26 and Figure 2-27 on page 39 also show the locking options at the bottom of the window.

When you initiate a load activity, ensure that the corresponding DB2 table is not locked exclusively at the table or table space, or partition level. Also, in the current implementation, the **UNLOAD** utility skips exclusively locked rows (that is, X and U locks on the rows or pages) on the DB2 side. Therefore, it is better to schedule the Accelerator load activity when there is minimal or no update on the corresponding DB2 tables.

Additional useful information about Figure 2-26 on page 38 and Figure 2-27 is the column labeled *Load Recommended*. The information in this column helps you with the selection. Because table loads are long-running processes, you should only load or reload a table or partition when necessary.

Load Tables

Load Tables / Partitions

Select the whole table or partitions to be loaded from DB2 for z/OS to the accelerator.

Name	Partitioned By	Load Recommended	DB2 Size	Schema	
<input checked="" type="checkbox"/> GOHR_EMPLOYEE	Growth	Yes, enforced	1.0 MB	BID1TB	
<input checked="" type="checkbox"/> Partition 1		Yes, enforced - created bef...	1.0 MB		
<input checked="" type="checkbox"/> GOHR_EMPLOYEE_I	Growth	Yes, enforced	1.0 MB	BID1TB	
<input checked="" type="checkbox"/> Partition 1		Yes, enforced - created bef...	1.0 MB		
<input checked="" type="checkbox"/> GOHR_GENDER_LO	Growth	Yes, enforced	1.0 MB	BID1TB	
<input checked="" type="checkbox"/> Partition 1		Yes, enforced - created bef...	1.0 MB		
<input checked="" type="checkbox"/> GOHR_RANKING_L	Growth	Yes, enforced	1.0 MB	BID1TB	
<input checked="" type="checkbox"/> Partition 1		Yes, enforced - created bef...	1.0 MB		
<input checked="" type="checkbox"/> GOHR_RANKING_R	Growth	Yes, enforced	1.0 MB	BID1TB	
<input checked="" type="checkbox"/> Partition 1		Yes, enforced - created bef...	1.0 MB		
<input checked="" type="checkbox"/> GOHR_SATISFACTI	Growth	Yes, enforced	1.0 MB	BID1TB	
<input checked="" type="checkbox"/> Partition 1		Yes, enforced - created bef...	1.0 MB		

☐ Lock DB2 tables while loading: All Tables

☒ After the load, enable acceleration for disabled tables.

Figure 2-27 Load activity: Multiple tables selected in one shot

Figure 2-28 shows typical CPU utilization on the Accelerator side. As shown in this sample scenario, the coordinator nodes are 100% busy while the worker nodes are only 20% busy during a typical load operation. The Load processing gets the data via the DRDA connection, parses it, converts it to CSV format, and writes it to the named pipe. Data is read from the pipe, converted into internal Accelerator format, and distributed across the worker nodes. This process is CPU intensive. Therefore, a parallelism not higher than 10 is suggested.

```

DSNX810I  -DZA1 DSNX8CMD DISPLAY ACCEL FOLLOWS -
DSNX830I  -DZA1 DSNX8CDA
ACCELERATOR                                MEMB  STATUS  REQUESTS  ACTV  QUED  MAXQ
-----
DZA1STPR                                DZA1  STARTED      1871    10     0       7
LOCATION=DZA1STPR HEALTHY
DETAIL STATISTICS
  LEVEL  = AQT03010
  STATUS = ONLINE
  FAILED QUERY REQUESTS                        =         5
  AVERAGE QUEUE WAIT                          =        27 MS
  MAXIMUM QUEUE WAIT                          =       404 MS
  TOTAL NUMBER OF PROCESSORS                   =        224
  AVERAGE CPU UTILIZATION ON COORDINATOR NODES =    100.00%
  AVERAGE CPU UTILIZATION ON WORKER NODES     =     20.00%
  NUMBER OF ACTIVE WORKER NODES                =         7
  TOTAL DISK STORAGE AVAILABLE                 =  48000960 MB
  TOTAL DISK STORAGE IN USE                    =         8.34%
  DISK STORAGE IN USE FOR DATABASE             =   3514619 MB
DISPLAY ACCEL REPORT COMPLETE
DSNX9022I  -DZA1 DSNX8CMD '-DISPLAY ACCEL' NORMAL COMPLETION
***

```

Figure 2-28 Load operation: CPU utilization on coordinator versus worker nodes

Figure 2-29 shows the number of WLM tasks that were running during a load operation for a sample setting of the environment variable AQT_MAX_UNLOAD_IN_PARALLEL=10.

See “Running parallel load streams” on page 67 for more details.

Display Filter View Print Options Search Help											

SDSF	DA	P59	P59	PAG	0	CPU/L	18/ 18	LINE 1-11 (11)			
NP	JOBNAME	StepName	ProcStep	JobID	Owner	C	Pos	DP	Real	Paging	SIO
	DZA1WLMU	DZA1WLMU	IEFPROC	STC04759	IBM	NS	F2	2075	0.00	8546.2	
	DZA1WLMU	DZA1WLMU	IEFPROC	STC04757	IBM	NS	F2	2061	0.00	9134.6	
	DZA1WLMU	DZA1WLMU	IEFPROC	STC04754	IBM	NS	F2	2066	0.00	8163.3	
	DZA1WLMU	DZA1WLMU	IEFPROC	STC04758	IBM	NS	F2	2060	0.00	9324.2	
	DZA1WLMU	DZA1WLMU	IEFPROC	STC04751	IBM	NS	F2	2096	0.00	9210.9	
	DZA1WLMU	DZA1WLMU	IEFPROC	STC04843	IBM	LO	FF	975	0.00	0.00	
	DZA1WLMU	DZA1WLMU	IEFPROC	STC04753	IBM	NS	F2	2064	0.00	9233.0	
	DZA1WLMU	DZA1WLMU	IEFPROC	STC04755	IBM	NS	F2	2068	0.00	8802.1	
	DZA1WLMU	DZA1WLMU	IEFPROC	STC04752	IBM	NS	F2	2077	0.00	9232.1	
	DZA1WLMU	DZA1WLMU	IEFPROC	STC04756	IBM	NS	F2	2076	0.00	8638.1	
	DZA1WLMU	DZA1WLMU	IEFPROC	STC04750	IBM	NS	F2	2238	0.00	9195.1	

Figure 2-29 SDSF active queue during load activity of a single table with 10 parallel unloads

More information about the load activity is available in the *DB2 Analytics Accelerator for z/OS Version 3.1.0 User's Guide*, SH12-6985.

After the definition and initial load and testing of the tables, it is likely that this activity would be included in batch cycles, possibly extending existing extract, transform, and load (ETL) procedures. For details, refer to Chapter 11, “Latency management” of *Optimizing DB2 Queries with IBM DB2 Analytics Accelerator for z/OS*, SG24-8005.

Enabling and disabling a table for query acceleration

You can permit or prevent the sending of queries to an Accelerator by enabling or disabling the corresponding tables. The task of disabling and enabling tables is carried out by the SYSPROC.ACCEL_SET_TABLES_ACCELERATION stored procedure on your data server. For information about the privileges that are required to run this procedure and further details, see the appropriate section in the *IBM DB2 Analytics Accelerator for z/OS: Stored Procedures Reference*.

Figure 2-26 on page 38 and Figure 2-27 on page 39 also show you a check box at the bottom of the window that would allow you to enable the tables for query acceleration automatically upon successful completion of the load operation. If you do not select the check box, the tables should be enabled separately by selecting the tables and clicking **Acceleration** → **Enable Acceleration** on the toolbar, as shown in Figure 2-30.

You can disable accelerated queries against selected tables without entirely disabling the associated Accelerator by clicking **Acceleration** → **Disable Acceleration** on the toolbar, as shown in Figure 2-30.

Figure 2-30 also highlights the placement of the twistie on the Acceleration button on the toolbar, which pulls down two mutually exclusive radio buttons to either enable a disabled table, or to disable an enabled table.

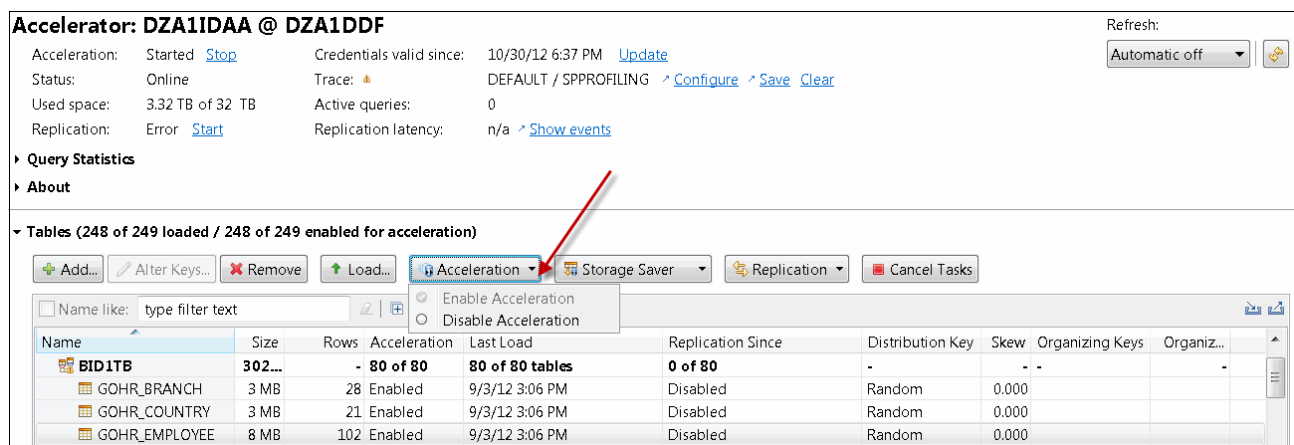


Figure 2-30 Enabling or disabling table level query acceleration

Tip: If you want to disable all tables that are associated with an Accelerator, it is better to stop the entire Accelerator.

Query acceleration can be always disabled regardless of the state a table is in. This is useful if you need to recover from an error situation in which an Accelerator is down or unreachable.

Determining the status of an Accelerator

DB2 queries can only be routed to an Accelerator if the Accelerator has been enabled in DB2 for z/OS. To determine the status of an Accelerator that is connected to a particular DB2

subsystem, you can issue a **-DIS ACCEL(*) DETAIL** command from z/OS, or you must first connect to that DB2 server from the Data Studio and then perform the following steps:

1. In the Administration Explorer, select the **Accelerators** folder.
2. In the Object List Editor, double-click an Accelerator to open the Accelerator view.

The Accelerator status is displayed on top of the resulting panel, as shown in Figure 2-31 (bottom half).

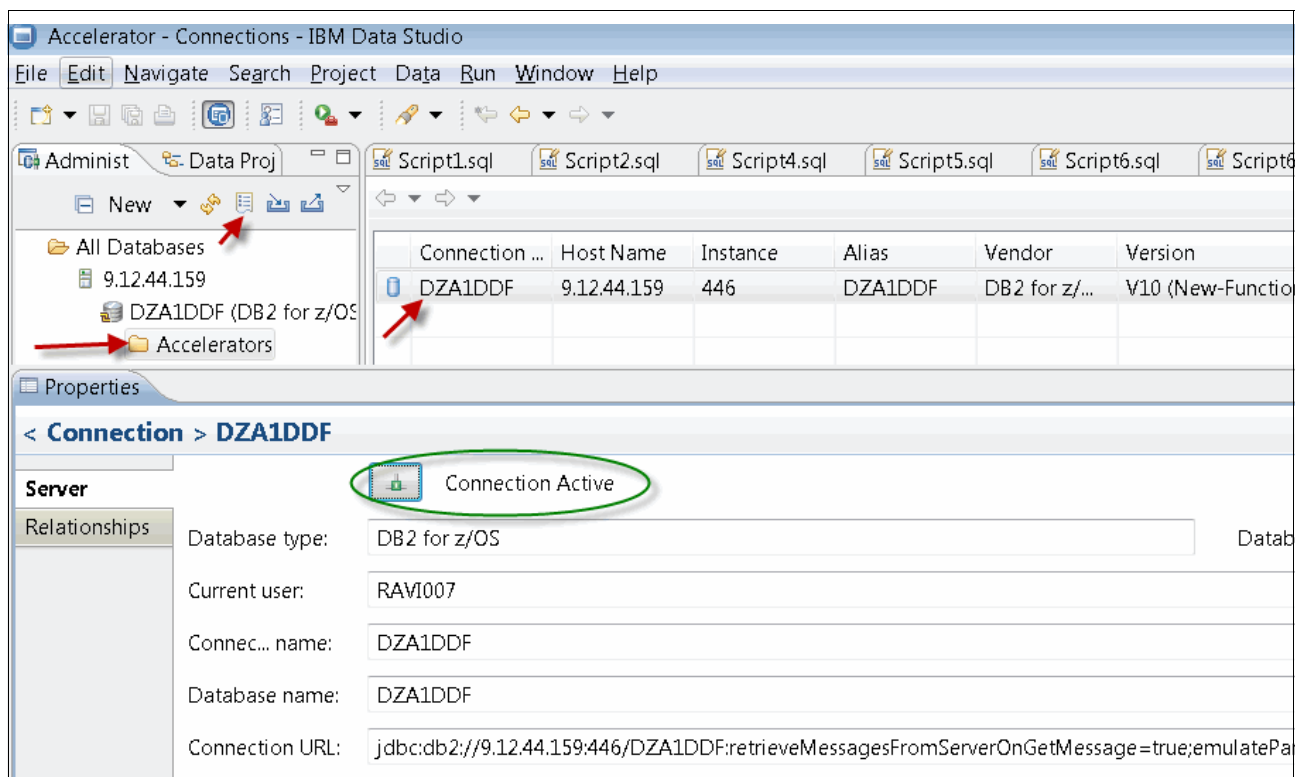


Figure 2-31 Studio: Display Accelerator Status panel

The following tasks are described in the *IBM DB2 Analytics Accelerator for z/OS Version 3.1.0 User's Guide*, SH12-6985:

- ▶ Selecting and Ordering Query monitoring columns
- ▶ Removing tables from an Accelerator
- ▶ Exporting a table specification
- ▶ Importing a table specification
- ▶ Removing Accelerators

The following Studio tasks are explained in Chapter 5, "Query acceleration management" on page 121:

- ▶ EXPLAIN information (based on DSN_QUERYINFO_TABLE)
- ▶ Displaying an access plan graph (for different values of CURRENT QUERY ACCELERATION and GET_ACCEL_ARCHIVE special registers)
- ▶ Specifying or changing a distribution key or organizing key

2.4.3 How to set the automatic refresh frequency on the Studio

From the Main menu, select **Window** → **Preferences**, as shown in Figure 2-32. Then, select **IBM DB2 Analytics Accelerator** → **Accelerator View**, as shown in Figure 2-33 on page 44, to set the refresh interval for the Accelerator view.

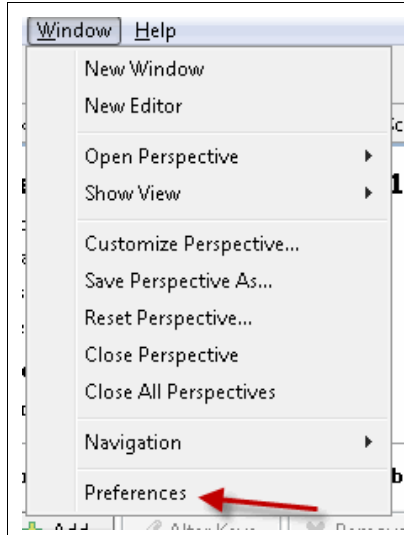


Figure 2-32 Preferences selection from the main menu

The same window that is shown in Figure 2-33 on page 44 can be used to set the warning threshold for table skew.

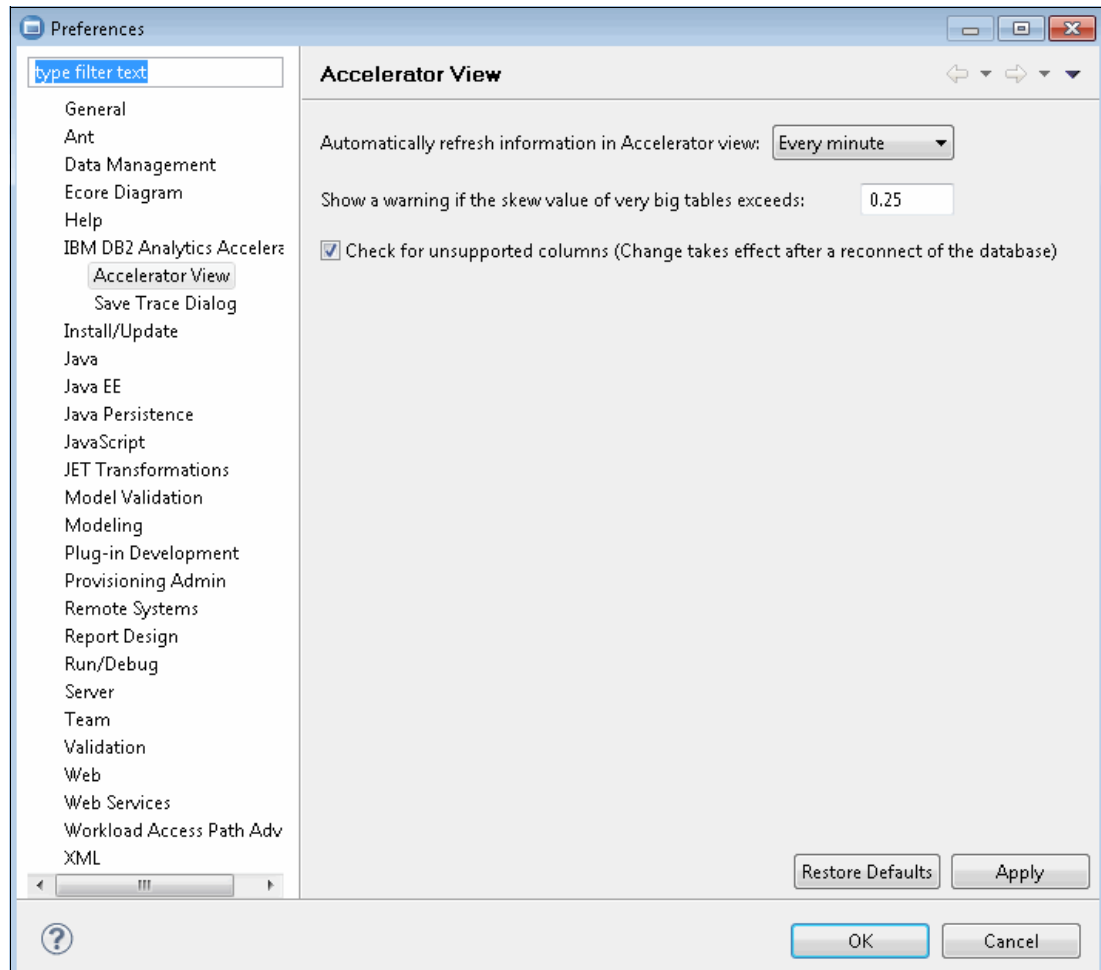


Figure 2-33 Accelerator Studio preferences window to set refresh interval and skew threshold

2.4.4 Tracing

IBM DB2 Analytics Accelerator for z/OS offers various predefined trace profiles in case you need diagnostic information. These profiles determine the trace detail level and the components or events that will generate trace information. Figure 2-34 identifies the trace section on the DB2 Analytics Accelerator Studio's "Accelerator View".

Accelerator: DZA1STPR @ DZA1DDF			
Acceleration:	Started Stop	Credentials valid since:	2/21/13 8:29 PM Update
Status:	Online	Trace:	DEFAULT / OFF Configure Save Clear

Figure 2-34 Studio - Accelerator view: Tracing activity

Note: Trace information is always collected but is externalized to disk only when it is saved.

When you click the *Configure* link from the Accelerator view (the links are shown on the bottom right side of Figure 2-34), a Configure Trace window opens.

You can select appropriate Accelerator trace profile names and stored procedure profiles, as depicted in Figure 2-35 and Figure 2-36 on page 46.

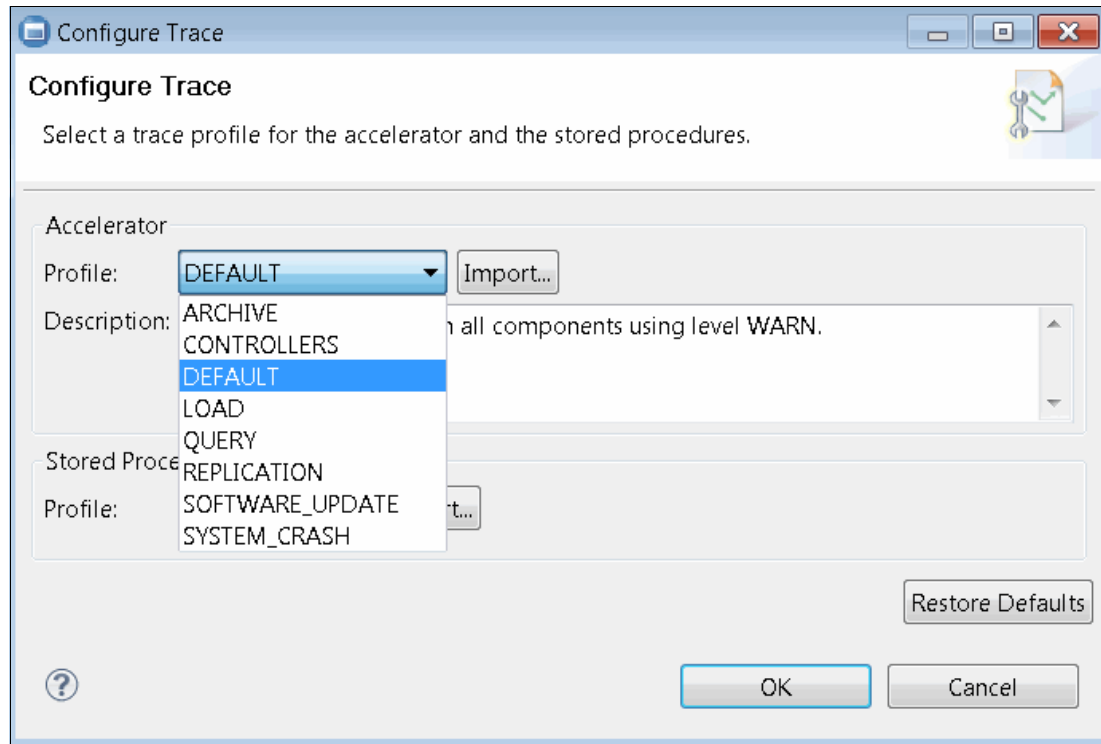


Figure 2-35 Configuring Accelerator trace: Selecting default profile

Note: Run the Accelerator with the profile, DEFAULT. The other profiles are to be used upon advice by IBM support.

To discard your individual settings and return to the default profiles for both Accelerator tracing and stored-procedure tracing (DEFAULT and Off), click **Restore Defaults**, which is shown in Figure 2-35 on page 45.

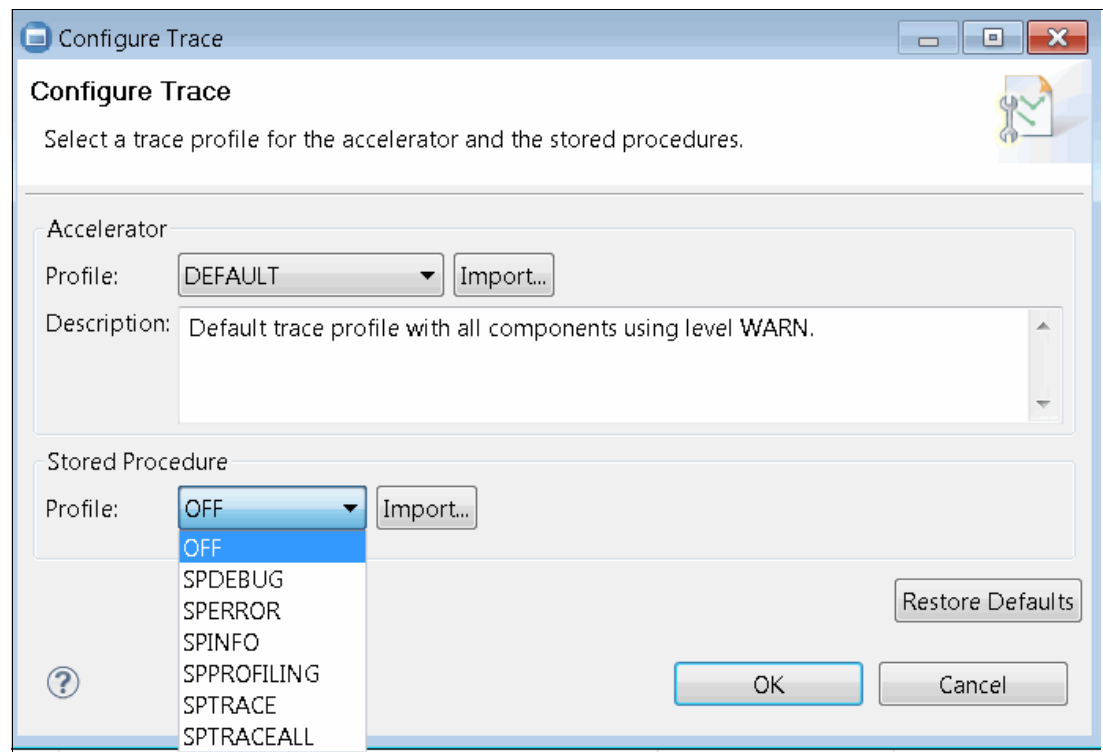


Figure 2-36 Configuring trace profile: Stored procedure tracing profile

More information about configuring, saving, and clearing traces can be found in the *IBM DB2 Analytics Accelerator for z/OS Version 3.1.0 User's Guide*, SH12-6985, under the “Tracing” section of the “Tasks Overview” chapter.

Note: Configuring stored procedure profiles to collect traces is only applicable for stored procedures that are executed from within the GUI, for instance, the trace configuration does not apply when a stored procedure is started from batch.

See Appendix B, “Notes on maintenance” on page 307 for Accelerator support highlights.

2.4.5 Tasks new to Accelerator Version 3.1

The Studio tasks associated with freeing up disk storage, the High-Performance Storage Saver option, is discussed in Chapter 6, “High-Performance Storage Saver” on page 143.

The Studio tasks associated with updating the Accelerator tables continuously, the incremental update option, is discussed in Chapter 7, “Incremental update” on page 161.

2.5 Stored procedures used as administrative interface

For enabling and disabling replication on the Accelerator, stored procedures are used. These stored procedures are the administration interface to the Accelerators. For each activity

performed by the IBM DB2 Analytics Accelerator Studio, the corresponding DB2 for z/OS stored procedure is called.

Note: The incremental update process that is based on replication does not use stored procedures.

The stored procedures provide most of the functions that are related to tables and accelerators. The stored procedures can also be invoked from customer tools to control system automation (in application programs). The system overview diagram in Figure 2-37 shows the flow of data and administrative information between DB2 and the DB2 Analytics Accelerator appliance.

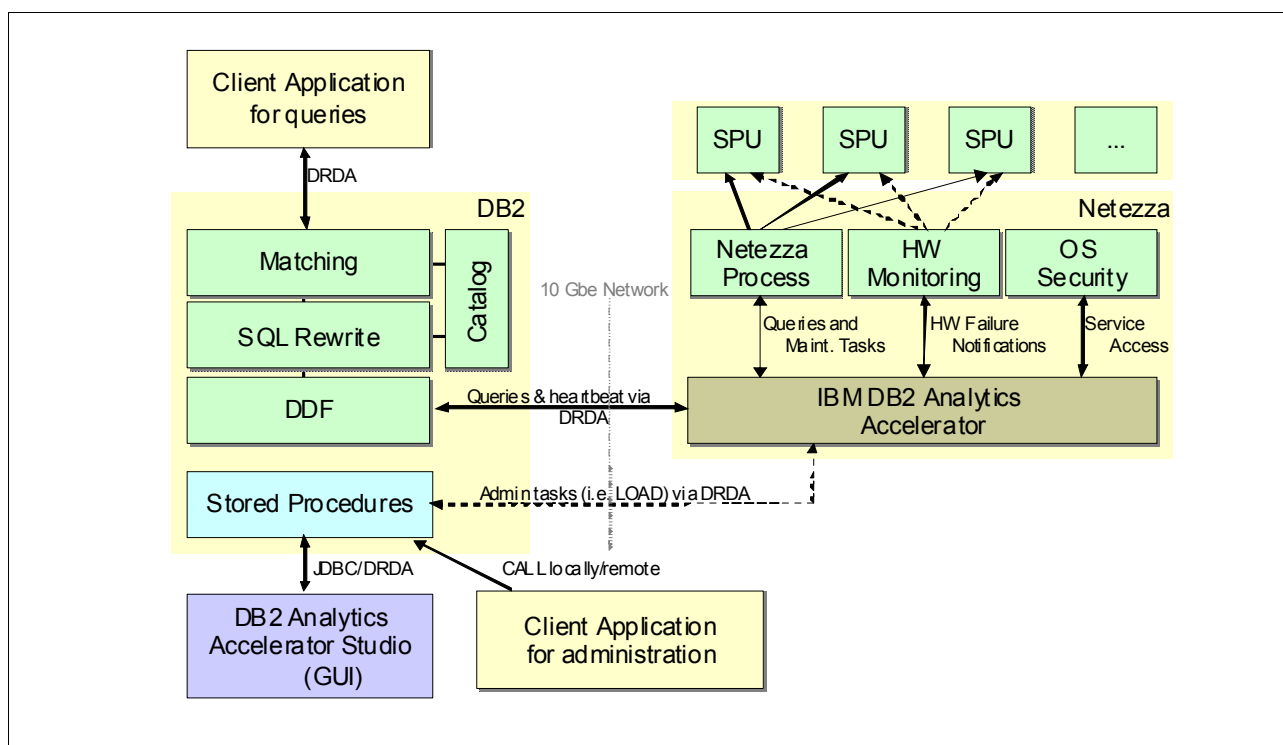


Figure 2-37 System overview diagram

2.5.1 Components used by stored procedures

Figure 2-38 on page 48 shows the components used by the Accelerator Stored Procedures regarding DB2 Services and z/OS Services.

All the required DB2 objects, such as user-defined functions, global temporary tables, views, and sequences, are created along with the DB2 Analytics Accelerator stored procedure as part of the installation.

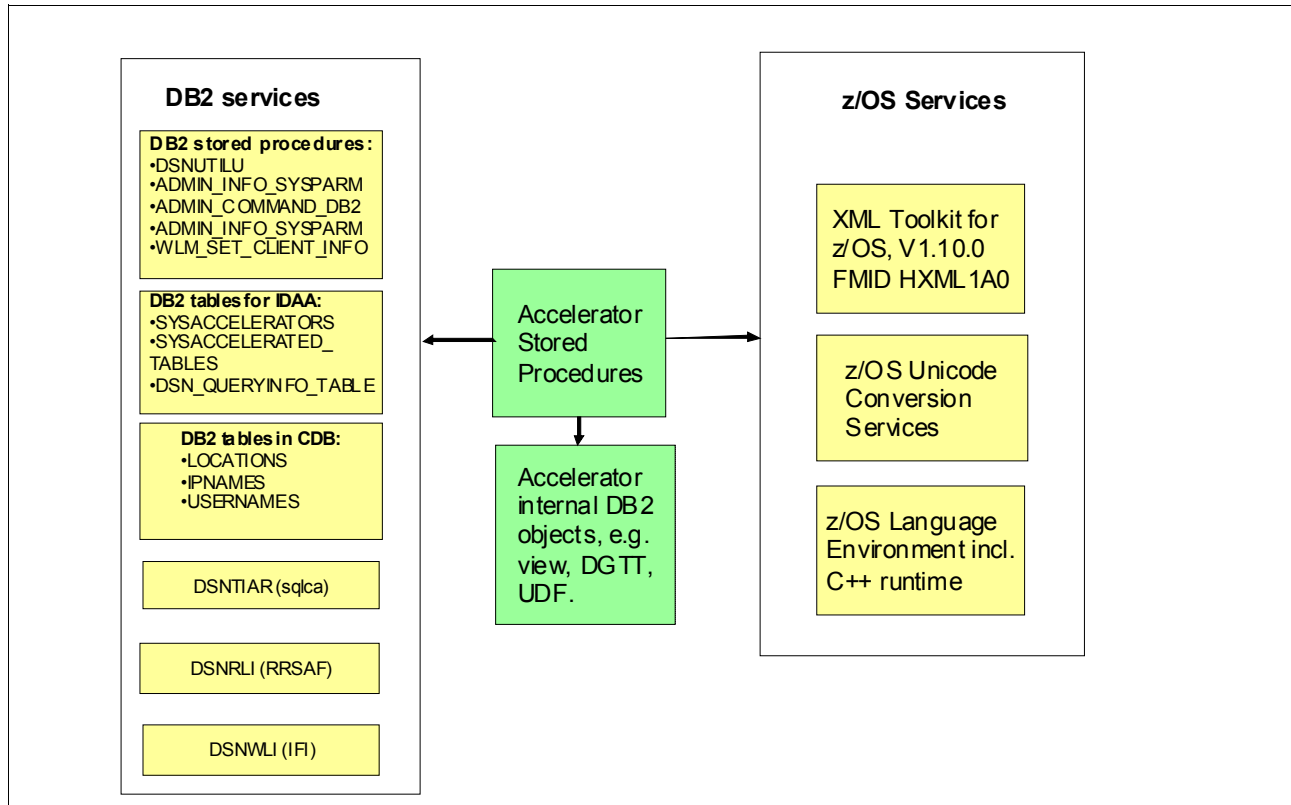


Figure 2-38 Components used by Accelerator stored procedures

2.5.2 DB2 supplied stored procedures used by Studio

The list of DB2 standard stored procedures used in the administration of the Accelerator is provided in Table 2-1 along with the corresponding short descriptions.

Table 2-1 DB2 stored procedures to administer the Accelerator function

DB2 supplied stored procedure name	Description
DSNUTILU	Utility processing
ADMIN_COMMAND_DB2	Command processing
ADMIN_INFO_SYSLOG	Syslog information
ADMIN_INFO_SYSPARM	DSNZPARM information
WLM_SET_CLIENT_INFO	WLM information

2.5.3 DB2 Accelerator stored procedures used by Studio

The complete list of DB2 stored procedures that are part of the Accelerator, which is available to administer the Accelerator, is provided in Table 2-2 on page 50 along with the corresponding short descriptions. These stored procedures can be run from the command line or embedded in custom applications. For instance, the *DB2 administrative task scheduler* can be used to execute these stored procedures as a one-time, periodical, or as a triggered task (DB2 triggered or task triggered) to perform a full refresh of the tables being accelerated using the SYSPROC.ACCEL_LOAD_TABLES stored procedure.

The stored procedures provide functions that are related to the Accelerators or the tables being accelerated. You might be able to use some of these stored procedures, such as ACCEL_TEST_CONNECTION and ACCEL_CONTROL_ACCELERATOR to monitor the general health of the Accelerator. Additional guidelines are provided in Chapter 9, “Monitoring DB2 Analytics Accelerator environments” on page 207.

If you are not sure whether a new custom application will tolerate extensions of a message parameter in the future (for example, additional attributes), follow these steps:

1. Determine the current interface level of the stored procedures by running the query shown in Example 2-5.

Example 2-5 SQL to find the Accelerator version: compatibilityLevel attribute

```
SELECT DSNAPT.ACCEL_GETVERSION()  
FROM SYSIBM.SYSDUMMY1
```

2. Make a note of the number that is returned as the query result.
3. Use this number as the value of the compatibilityLevel attribute in each message input parameter that is used when your custom applications call IBM DB2 Analytics Accelerator for z/OS stored procedures.

Note: By setting the correct *compatibilityLevel* attribute, you do not need to change your custom applications in the future after installing a newer version of the IBM DB2 Analytics Accelerator for z/OS stored procedures.

Table 2-2 on page 50 shows the Accelerator stored procedures to administer the Accelerator function. The last three shaded rows include stored procedures made available with Version 3.1.

Table 2-2 Accelerator stored procedures to administer Accelerator function

Name	Description	Remarks
ACCEL_ADD_ACCELERATOR	Pairing an Accelerator to a DB2 subsystem or data sharing group	Mandatory
ACCEL_TEST_CONNECTION	Check connectivity between DB2 subsystem and the Accelerator	Can be used for health check
ACCEL_REMOVE_ACCELERATOR	Removing an Accelerator from a DB2 and cleanup resources on Accelerator (including the data that was loaded)	Do not grant execute access to the public
ACCEL_UPDATE_CREDENTIALS	Changes the authentication token that all the DB2 stored procedures use to communicate with the Accelerator	User ID must have OMVS segment in IBM RACF®
ACCEL_ADD_TABLES	Add a set of tables to the Accelerator	Run before loading tables
ACCEL_ALTER_TABLES	Alter distribution and organizing key definitions for a set of tables on the Accelerator	This is <i>not</i> for table DDL ^a changes
ACCEL_REMOVE_TABLES	Remove tables from the Accelerator and the SYSACCEL.SYSACCELERATEDTABLES catalog table	Data loaded on the Accelerator will be lost
ACCEL_GET_TABLES_INFO	List set of tables on the Accelerator together with XML table specifications and the table status information	User ID must have OMVS segment in RACF
ACCEL_LOAD_TABLES	Load/Reload/Update data from DB2 into a set of tables on the Accelerator	Could be done even when Accelerator is stopped
ACCEL_SET_TABLES_ACCELERATION	Enable or disable a set of tables for query routing	Mandatory
ACCEL_CONTROL_ACCELERATOR	Controlling the Accelerator tracing, collecting trace and detail of the Accelerator, as well as cancel tasks currently running on the Accelerator	Can be used for health check
ACCEL_UPDATE_SOFTWARE	Update software on the Accelerator (transfer versioned software packages or apply an already transferred package; also list software both on z/OS and Accelerator side)	Software maintenance
ACCEL_GET_QUERY_DETAILS	Retrieve statement text and query plan for a running or completed Accelerator query	Monitoring API
ACCEL_GET_QUERY_EXPLAIN	Generate and retrieve Accelerator explain output for a DB2 query	Optimization
ACCEL_GET_QUERIES	Retrieve active or history query information from Accelerator	Monitoring
ACCEL_ARCHIVE_TABLES	Move table partitions from DB2 to the Accelerator	HPSS
ACCEL_SET_TABLES_REPLICATION	Enable or disable incremental updates for one or more tables on an Accelerator	Process must be enabled at the subsystem level

ACCEL_GET_TABLES_DETAILS	Collect information about a set of tables with regard to data changes (consistency) or move operations with HPSS	Can be used to determine the need to load for synchronization
--------------------------	--	---

- a. For details about how to propagate table Data Definition Language (DDL) changes on the Accelerator when the definition changes in DB2, refer to Chapter 5, “Query acceleration management” on page 121.

2.6 Stored procedure security

Table 2-3 outlines the required DB2 security privileges at the stored procedure level to carry out the corresponding task that is pertaining to a given stored procedure.

Table 2-3 Stored procedure access privileges

Accelerator stored procedure	Required rights
ACCEL_ADD_ACCELERATOR	<ol style="list-style-type: none"> 1. EXECUTE on the stored procedure. 2. EXECUTE on the SYSACCEL.* packages.
ACCEL_ADD_TABLES	<ol style="list-style-type: none"> 1. EXECUTE on the stored procedure. 2. EXECUTE on the SYSACCEL.* packages. 3. MONITOR1 (needed for calling ADMIN_INFO_SYSPARM from the procedure).
ACCEL_ALTER_TABLES	<ol style="list-style-type: none"> 1. EXECUTE on the stored procedure. 2. EXECUTE on the SYSACCEL.* packages. 3. MONITOR1 (needed for calling ADMIN_INFO_SYSPARM from the procedure).
ACCEL_ARCHIVE_TABLES	<ol style="list-style-type: none"> 1. EXECUTE on the stored procedure. 2. EXECUTE on the SYSACCEL.* packages. 3. Privilege to connect to DB2 via RRSAF. 4. Privilege to read the original DB2 base tables of the accelerator tables that are listed in the table_archive_specification parameter. 5. Write access to the /tmp directory (USS pipes are created in this directory). 6. DISPLAY privilege. 7. Privilege to run the COPY utility. 8. Privilege to run the LOAD utility. 9. Write access in RACF for COPY data sets. 10. Privilege to create data sets with the qualifier <USERID>.AQT, where <USERID> is the user ID of the person invoking the stored procedure.
ACCEL_CONTROL_ACCELERATOR	<ol style="list-style-type: none"> 1. EXECUTE on the stored procedure. 2. EXECUTE on the SYSACCEL.* packages. 3. If an output file or data-set location is specified: Write access in RACF to the output data set for trace data or write access to the specified location in the hierarchical file system.
ACCEL_GET_QUERIES	<ol style="list-style-type: none"> 1. EXECUTE on the stored procedure. 2. EXECUTE on the SYSACCEL.* packages.
ACCEL_GET_QUERY_DETAILS	<ol style="list-style-type: none"> 1. EXECUTE on the stored procedure. 2. EXECUTE on the SYSACCEL.* packages.

Accelerator stored procedure	Required rights
ACCEL_GET_QUERY_EXPLAIN	<ol style="list-style-type: none"> 1. EXECUTE on the stored procedure. 2. EXECUTE on the SYSACCEL.* packages. 3. SELECT on the schema containing DSN_QUERYINFO_TABLE.
ACCEL_GET_TABLES_DETAILS	<ol style="list-style-type: none"> 1. EXECUTE on the stored procedure. 2. EXECUTE on the SYSACCEL.* packages.
ACCEL_GET_TABLES_INFO	<ol style="list-style-type: none"> 1. EXECUTE on the stored procedure. 2. EXECUTE on the SYSACCEL.* packages.
ACCEL_LOAD_TABLES	<ol style="list-style-type: none"> 1. EXECUTE on the stored procedure. 2. EXECUTE on the SYSACCEL.* packages. 3. Privilege to connect to DB2 for z/OS via RRSAF. 4. Privilege to read the original DB2 base tables of the accelerator tables that are listed in the table_load_specification parameter. 5. Write access to the /tmp directory (UNIX System Services pipes are created in this directory). 6. DISPLAY privilege (for calling -DIS GROUP). 7. Authorization to execute the RUNSTATS utility on the databases that contain the tables to be loaded, such as the STATS auth.
ACCEL_REMOVE_ACCELERATOR	<ol style="list-style-type: none"> 1. EXECUTE on the stored procedure. 2. EXECUTE on the SYSACCEL.* packages.
ACCEL_REMOVE_TABLES	<ol style="list-style-type: none"> 1. EXECUTE on the stored procedure. 2. EXECUTE on the SYSACCEL.* packages.
ACCEL_SET_TABLES_ACCELERATION	<ol style="list-style-type: none"> 1. EXECUTE on the stored procedure. 2. EXECUTE on the SYSACCEL.* packages.
ACCEL_SET_TABLES_REPLICATION	<ol style="list-style-type: none"> 1. EXECUTE on the stored procedure. 2. EXECUTE on the SYSACCEL.* packages. 3. ALTER TABLE for the tables in the table set.
ACCEL_TEST_CONNECTION	<ol style="list-style-type: none"> 1. EXECUTE on the stored procedure. 2. EXECUTE on the SYSACCEL.* packages.
ACCEL_UPDATE_CREDENTIALS	<ol style="list-style-type: none"> 1. EXECUTE on the stored procedure. 2. EXECUTE on the SYSACCEL.* packages.
SYSPROC.ACCEL_UPDATE_SOFTWARE	<ol style="list-style-type: none"> 1. EXECUTE on the stored procedure. 2. EXECUTE on the SYSACCEL.* packages. 3. Read access to the installation images. This is the directory structure under <prefix>/usr/lpp/aqt/packages in the hierarchical file system of the data server where <prefix> is the value of the AQT_INSTALL_PREFIX environment variable. If the variable is not set, the path starts from the root directory. 4. Read access to the directory that the AQT_HOST_PACKAGE_DIRECTORY environment variable points to.

2.7 Data Studio web console

Optionally, for job management and health monitoring, you can also install the IBM Data Studio web console component. Although there are no out-of-the-box features available specifically for the Accelerators, you should be able to configure user-defined alerts, as described in “Creating user-defined alert types” on page 54, to monitor the health of an Accelerator environment.

2.7.1 Install and configure the Data Studio web console component

The download site is the same as that of the Data Studio client component, which is provided in 2.2.2, “Data Studio full client download and installation” on page 17. The procedure to install the web console component is very similar to that of the Data Studio client component. You can accept the default port values during the installation and enter a password of your choice.

After the installation is complete, the Data Studio web console has to be configured using the steps described in the following sections.

Configuring on Linux environments

Use the following steps to configure Linux environments:

1. If the Data Studio web console is not running, start it. From the `DSWC_installation_dir/bin` directory, issue `./start.sh`

Example 2-6 shows the sample output from the start command issued from the Data Studio web console for Linux.

Example 2-6 Sample output from the start Data Studio web console command: Linux

```
...dsserver_home: /opt/ibm/DS3.2.0
port: 11080
https.port: 11081
SERVER STATUS: ACTIVE
Log: /opt/ibm/DS3.2.0/logs/dsserver.log
```

2. Log in to the web console of the Data Studio web console (using the password that you entered during installation of the web console).
3. Open the web console by entering the following web address in a web browser:

`http://IP_address:port_number/datatools/`

The following URLs are the defaults for the Data Studio web console and web console secure:

- `http://localhost:11083/datatools/`
- `https://localhost:11084/datatools/`

Diagnostics information about Linux

Issue the following commands to collect diagnostics information.

- To verify whether the web console server is up or not, issue the following command from the Linux terminal:

```
netstat -lnp
```

Check the output to see whether any process is LISTENing on your ports.

- To check the server log, issue the following command:

```
cat dsserver.log
```

You can obtain the directory path from the output of the start command that is shown in Example 2-6 on page 53.

The server log contains much information. For instance, you can get the URL for your web console, as shown in Example 2-7.

Example 2-7 Data Studio web console server log: Partial listing showing the URL

```
DS_System Logger Initialized: /opt/ibm/DS3.2.0/logs/DS_System
[AUDIT  ] CWWKT0016I: Web application available (default_host):
http://localhost.localdomain:11080/*
[AUDIT  ] CWWKZ0001I: Application dsweb started in 26.930 seconds.
[AUDIT  ] CWWKF0011I: The server dsweb is ready to run a smarter planet.
STARTED CoreServiceComponent
```

Configuring on Windows environments

If Data Studio web console is not running, start it from the Start menu. Click **All Programs** → **IBM Data Studio** → **Data Studio Web Console** → **Start Web Console Server**.

Data Studio web console can be further configured, if needed, by performing the following tasks:

- Configuring Data Studio web console for the multi-user mode
To use Data Studio web console with multiple users, you must configure it for multi-user mode by adding a repository database. Then, configure authentication for the users and groups of that database. The repository database is also used to store database connections, alerts and alert configuration settings, job manager settings, and job history.
- Adding database connections from the Data Studio client
You can automatically add a database connection from the Data Studio client to the Data Studio web console when you open a web console page from the workbench.
- Configuring Data Studio web console alerts
You can configure the alert types to display for each monitored database and the alert thresholds for each alert type. You do not need to configure any alerts to start using Data Studio web console. Monitoring is enabled by default for each database connection, and the product comes with default threshold settings for all alerts.
- Creating user-defined alert types
In addition to the predefined alert types that are provided by the web console, you can create your own alert types. For more information, see Chapter 9, “Monitoring DB2 Analytics Accelerator environments” on page 207.



Data latency management

To accelerate queries, the IBM DB2 Analytics Accelerator (*Accelerator*) must have access to the data of interest. This, naturally, requires that data must be copied from DB2 tables to corresponding tables defined to the Accelerator.

There must be an initial population of the data, which is accomplished using the batch load process. After a table has been populated initially, the data needs to be updated periodically to keep in synch with the data in the source DB2 table. There are two methods: *batch refreshing* and *incremental update*.

The first part of this chapter introduces both methods, batch and incremental update. The rest of the chapter focuses on the details of the batch method to understand the underlying tasks that take place when doing a load. This chapter also describes how to see what data has changed (change detection) to identify tables and partitions that need refreshing and how to extend the extract, transform, and load (ETL) processes to include loading data to the Accelerator. Finally, there is a description of considerations when formulating a load strategy.

The incremental update method is described in Chapter 7, “Incremental update” on page 161.

This chapter contains the following sections:

- ▶ Methods of copying data from DB2 to the Accelerator
- ▶ Understanding the load process
- ▶ Load performance considerations
- ▶ Automating the load process
- ▶ Summary

3.1 Methods of copying data from DB2 to the Accelerator

The value proposition of the IBM DB2 Analytics Accelerator is to run certain kind of queries, normally associated with data analytics, much faster than can be handled by a typical DB2 installation. Of course, to run these queries on the Accelerator, the data must also reside in the Accelerator as well as within DB2 tables. When a DB2 table is added to the Accelerator, the Accelerator copy of the table is initially empty. Therefore, a population of the Accelerator copy of the table is required to synchronize the data between the DB2 table and the Accelerator. After initial population of the Accelerator, the data in DB2 will change at some frequency depending on the business requirements of the data. When that happens, the data within the Accelerator copy must again be synchronized.

There are two complementary mechanisms available to keep the DB2 data and the Accelerator copy in synch:

- ▶ The batch load mechanism, also used in the initial population, can be used to refresh all or part of the Accelerator copy of the table.
- ▶ The other mechanism is the apply of changes from the DB2 table to the Accelerator, which is known as *incremental update*.

It is important to understand each of these mechanisms to design a solution that meets the latency needs based on the application requirements and the characteristics of the DB2 table in terms of structure, contents, and usage. The “application perceived latency” should be understood. If the application does not require data that is more recent than the previous day, a nightly batch synchronization of the data should suffice. However, if an application requires that data is near real-time and can handle data that is out of synch for a few minutes, incremental update might be needed.

The choice of which mechanism to use is made on an individual table basis. There are many factors that influence the decision to use batch load or the incremental load. The factors include requirements for the applications that drive the changes into the DB2 database and the applications that will benefit from off loading query processing to the Accelerator along with database characteristics such as the total volume of changes, distribution of changes across partitions, the frequency of changes to the DB2 table, number of changes that are updates or deletes, and so on.

Table 3-1 gives a high-level comparison of Accelerator refresh options.

Table 3-1 Comparison of Accelerator refresh options

Batch refresh with or without change detection	Incremental update
<ul style="list-style-type: none">▶ Utilizes batch streaming load method based on the DB2 UNLOAD utility▶ Snapshot of data at table or partition granularity▶ High performance for larger data volumes▶ High throughput: 1 Terabyte per hour▶ Explicitly triggered▶ Can handle not logged changes▶ Lower processor cost per MB▶ Not logged updates	<ul style="list-style-type: none">▶ Uses separate log-reading capture and apply technology▶ Single row granularity▶ High performance for frequent, small, and arbitrary changes▶ Lower throughput <= 18 GB/hour▶ Data continuously synchronized▶ Low latency▶ Does not handle not logged changes▶ Higher processor cost per MB

Figure 3-1 on page 58 depicts a generic decision tree that might be used for general guidance in selecting a possible refresh method. There are paths that lead to a clear suggestion of a bulk refresh and other paths that lead to an incremental update suggestion. Unfortunately, it is not always obvious which one to choose.

Technical characteristics might lead you to one method or the other. If, for example, you have a large partitioned table for which changes are scattered over many partitions, you might not have the time in your batch ETL window to do a refresh of those partitions. In this case, you might evaluate the incremental update method to just process the changed data. On the other extreme, one or more key tables in your data warehouse might already be updated throughout the data using some type of trickle-feed technology. This would indicate that an incremental update to the Accelerator should be considered for updating the corresponding table in the Accelerator.

Often, there is a business need that drives the requirement for low latency updates to one or more tables that would lead to choosing the incremental update method. Of course, then source DB2 tables would also need to be updated with some kind of low latency technique.

Using Table 3-1 on page 56 and Figure 3-1 on page 58 together can give some initial guidance in selecting an appropriate refresh method. Sometimes, the choice of which refresh method to use is obvious, while at other times additional analysis or measurement might be needed to determine the most appropriate method for the situation.

Incremental update and load could also be combined. For instance, overnight non-logged batch loads of DB2 data can be refreshed using IBM DB2 Analytics Accelerator (IDAA) load, which will cause incremental update to stop briefly while the tables are refreshed and then set a new capture point.

Figure 3-1 on page 58 shows an Accelerator method decision tree.

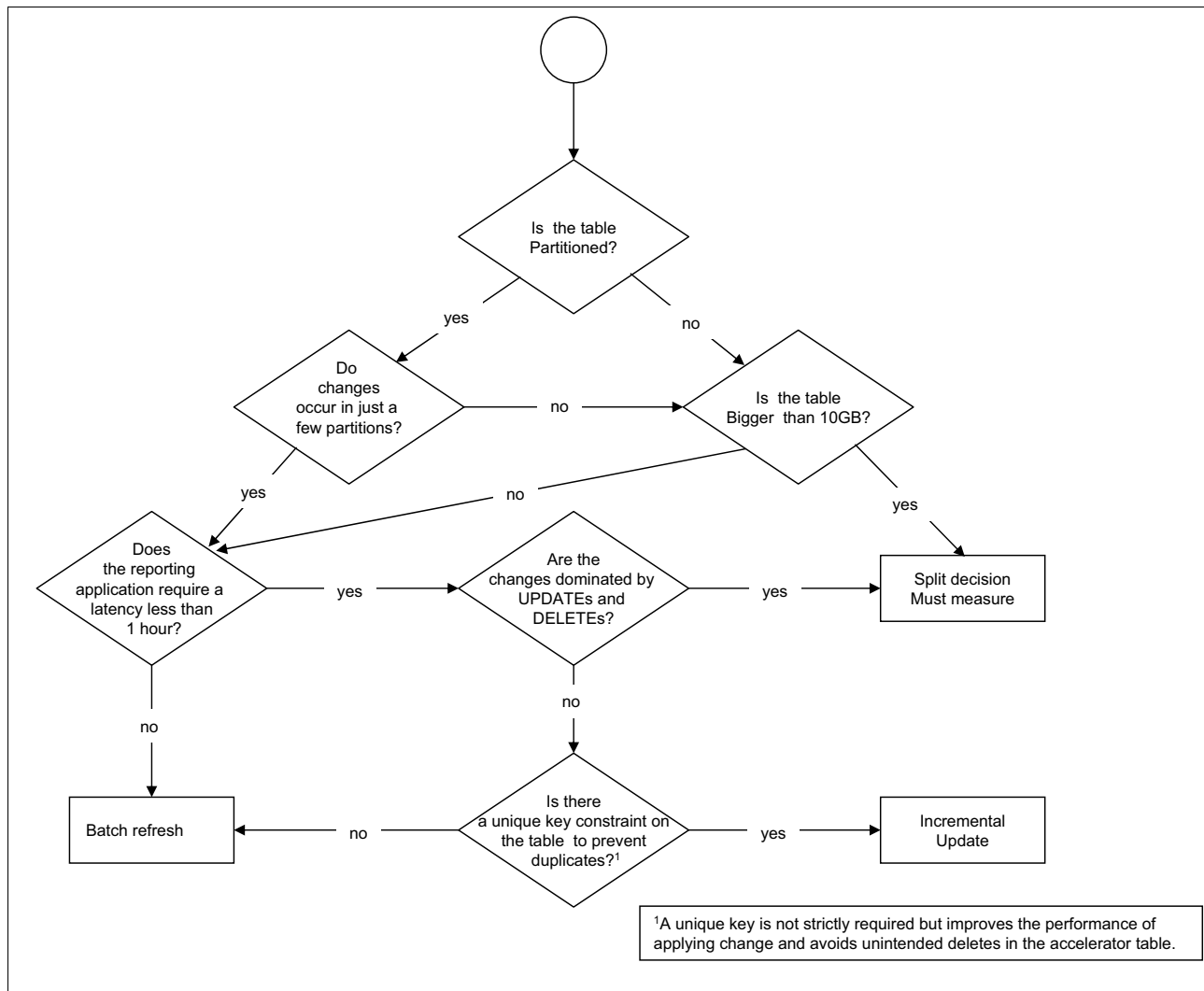


Figure 3-1 Accelerator method decision tree

An additional consideration is the requirement for concurrently executing queries while the Accelerator is being refreshed. Queries can be run concurrently during both batch refresh and incremental load. This process is described in Chapter 4, “Accelerator resource management” on page 87, and in Chapter 7, “Incremental update” on page 161.

3.2 Understanding the load process

The primary value point of the Accelerator is to improve query response time by executing the query to the Accelerator for processing. This, of course, requires the data to be in the Accelerator. This means that data has to be selected, or unloaded, from the DB2 table, sent across the network to the Accelerator, then converted appropriately and inserted into the Accelerator tables. Fortunately, there is a process that wraps all of these tasks into a simple invocation from the Accelerator Studio. See “Loading tables into an Accelerator” on page 37, or “calling the LOAD_ACCEL_TABLES stored procedure” as described in 3.4, “Automating the load process” on page 79.

3.2.1 Load streams

The load process is started in one of two ways:

- ▶ In the Accelerator Studio, by selecting one or more tables and clicking the **load** button.
- ▶ By calling the `LOAD_ACCEL_TABLES` stored procedure and providing a list of table names in an input XML document.

If you use the Accelerator Studio method, it formulates the call to the `LOAD_ACCEL_TABLES` stored procedure for you. In the rest of this chapter, the phrase “invoking the load procedure” refers to either of these two methods.

Caution: If you invoke the load procedure from the Accelerator Studio, you must maintain the connection from the studio to DB2. If that connection fails, the load will fail.

What is a load stream?

To copy data from DB2 to the Accelerator, a multi-threaded, pipelined approach is used. A single load stream consists of process on System z and the Accelerator to load one partition of data from DB2 to the Accelerator.

When using a table with only one partition, use the following steps:

1. A call to the `ACCEL_LOAD_TABLES` stored procedure is done by using the Accelerator Studio, or explicitly, for a single non-partitioned table.
2. An address space is created for running the `ACCEL_LOAD_TABLES`. The stored procedure creates and manages the entire load stream.
3. The `ACCEL_LOAD_TABLES` stored procedure extracts metadata from the DB2 catalog and calls the `DSNUTILU` stored procedure to execute the DB2 `UNLOAD` utility using the DB2 `INTERNAL FORMAT` option.
4. On the Accelerator side, the data is received from the network, converted from DB2 internal format, and sent across a named pipe to be inserted into the Accelerator database using one `INSERT` statement. The accelerator database will then spread the data across the worker nodes of the Accelerator.

If there are multiple tables that are defined in one call to `ACCEL_LOAD_TABLES`, the `ACCEL_LOAD_TABLES` stored procedure is still invoked only once. It then serially loads the specified tables, one table at a time until it has processed all the tables provided in the input list.

To run loads concurrently for non-partitioned tables requires multiple invocations of the `ACCEL_LOAD_TABLES`. For example, you have two non-partitioned tables, `TABLEA` and `TABLEB` that you would like to load concurrently and would require two independent and concurrent invocations of the `ACCEL_LOAD_TABLES` stored procedure. The `ACCEL_LOAD_TABLES` stored procedure is invoked twice. The first invocation of `ACCEL_LOAD_TABLES` includes only `TABLEA`; there is a separate invocation for `TABLEB`. This results in two instantiations of the `ACCEL_LOAD_TABLES` stored procedure and two independent parallel loads.

Note: When invoking concurrent loads using multiple invocations of `ACCEL_LOAD_TABLES`, you must ensure that all sets of tables are disjoint. In other words, any one table should appear only in the input set of only one `ACCEL_LOAD_TABLES` call.

3.2.2 Loading partitions in parallel

We introduced the concept of load streams in the context of non-partitioned tables. If a table is partitioned, ACCEL_LOAD_TABLES processes each partition with an independent load stream. In addition, it instantiates multiple independent load streams that run concurrently. The number of concurrent, or parallel, load streams that can be instantiated for one partitioned table is controlled by an environment variable that is called **AQT_MAX_UNLOAD_IN_PARALLEL**. When one parallel stream that is associated with one partition completes, ACCEL_LOAD_TABLES processes the next partition. This continues until all partitions have been loaded.

Note: The **AQT_MAX_UNLOAD_IN_PARALLEL** variable is found in the AQTENV member of the SAQTSAMP data set.

What happens when calling ACCEL_LOAD_TABLES to load a partitioned table that contains five partitions? As before, ACCEL_LOAD TABLES is called either explicitly or implicitly via the Accelerator Studio with a table list containing the table to be loaded. Using metadata from the DB2 catalog, ACCEL_LOAD_TABLES knows that the table is partitioned and will start instantiating concurrent load streams, one for each partition up to the maximum specified in the **AQT_MAX_UNLOAD_IN_PARALLEL** environment variable. In this example, **AQT_MAX_UNLOAD_IN_PARALLEL** is set to 4, which means that up to four load streams can be run at the same time for the partitions of a table. Initially, ACCEL_LOAD_TABLES creates load streams for partitions 1 - 4, as seen in Figure 3-2.

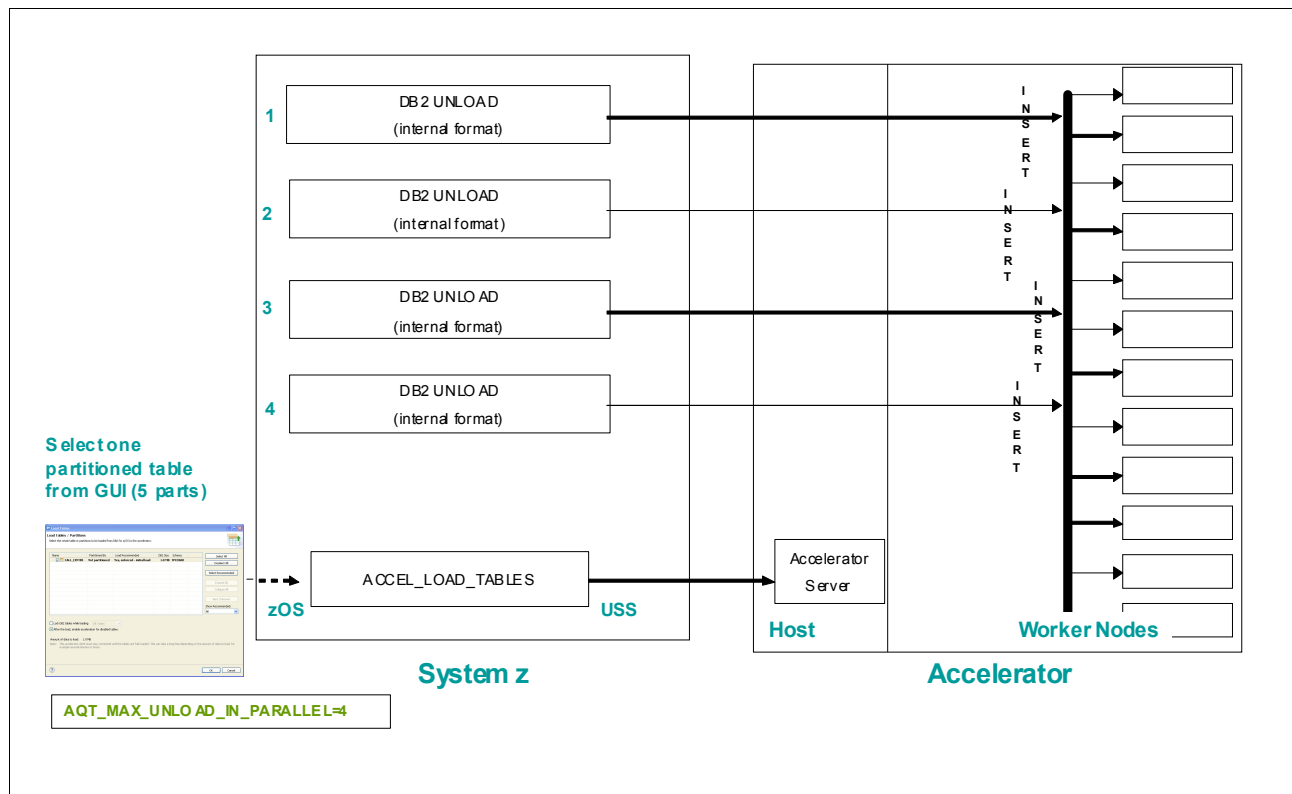


Figure 3-2 Loading a partitioned table containing five partitions

When the first load stream completes, ACCEL_LOAD_TABLES can load the next partition. Assuming that partition 3 completed first, the next partition in line, partition 5, is loaded. In this

case, that is the last partition for the table and as these streams complete, the ACCEL_LOAD_TABLES stored procedure cleans up the resources and finally terminates.

All partitions of a table must be loaded at initial load. This is enforced by the load process, as shown in Figure 3-3.

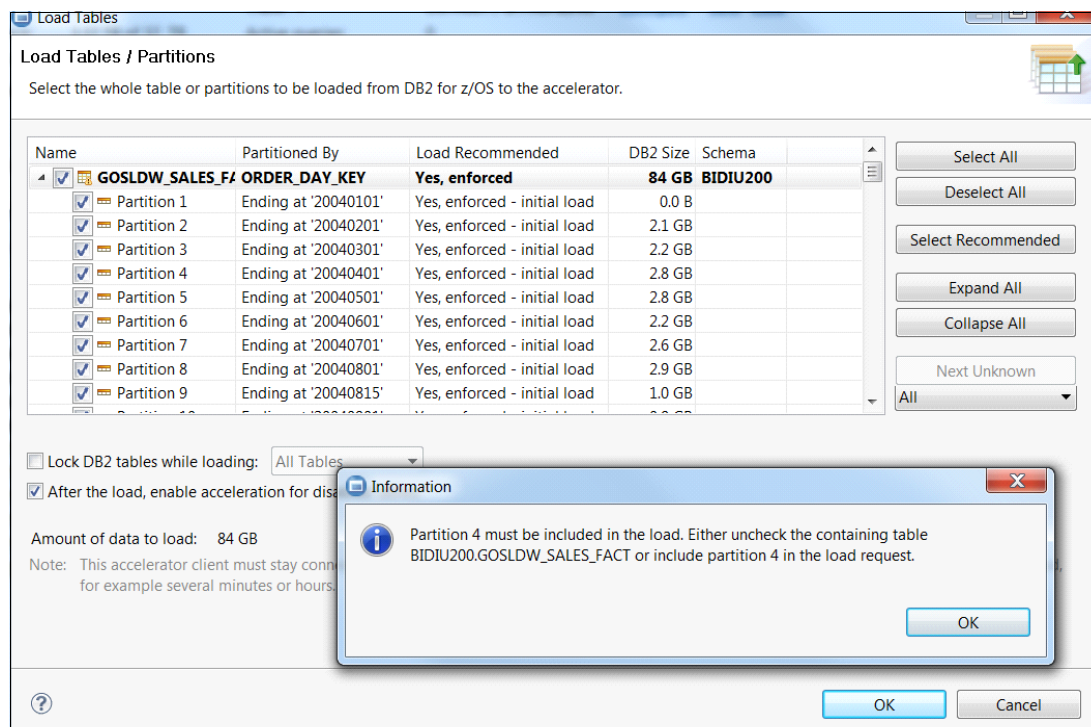


Figure 3-3 All partitions must be loaded for the initial load

3.2.3 Combining non-partitioned and partitioned table loads

Before continuing, we summarize what was learned in the previous sections:

1. A single invocation of the ACCEL_LOAD_TABLES stored procedure processes the specified tables serially, one at a time. There might be partitioned and non-partitioned tables in the list.
2. Non-partitioned tables are loaded using one load stream.
3. Partitioned tables are loaded using concurrent load streams up to the value specified in the AQT_MAX_UNLOAD_IN_PARALLEL environment variable, which applies independently to each partitioned load stream.
4. Multiple invocations of the ACCEL_LOAD_TABLES are allowed, each processing its own list of tables. Remember to use disjoint table sets for each invocation.

There has to be care taken when running loads to the Accelerator. The AQT_MAX_UNLOAD_IN_PARALLEL environment variable is not a system-wide variable. It applies to each and every call to ACCEL_LOAD_TABLES that processes partitioned tables. For example, if AQT_MAX_UNLOAD_IN_PARALLEL is set to 10, with two independent invocations of ACCEL_LOAD_TABLES each loading a partitioned table of 100 partitions, there will be:

- Up to 20 concurrent load streams with 20 DB2 DSNUTILU stored procedure address spaces executing the DB2 UNLOAD utility
- Two DB2 stored procedure addresses running ACCEL_LOAD_TABLES

- Twenty parallel processes inserting data to the Accelerator database

If there are multiple DB2 subsystems paired with the same Accelerator, there has to be care taken there as well. If there are 10 load streams running on DB2A and 10 load streams running on DB2B, both to the same Accelerator, there will be processes to support 20 loads on the Accelerator.

Understanding the concept of load streams is the basis to establishing an overall load and refresh strategy. Resources need to be shared on the Accelerator during load processing and loading is CPU intensive on the host node. There are other considerations as well and are covered throughout the rest of this chapter.

3.2.4 Data refresh

The load stored procedure, ACCEL_LOAD_TABLES, determines whether the load for a table is an initial load situation or a refresh situation.

An initial load situation occurs just after a table has been added to the Accelerator and no load has been processed. This is indicated by a status of *Initial Load Pending* on the table in the Accelerator Studio.

A refresh situation occurs when the target Accelerator table has already been populated. This is indicated by a date in the *Last Load* column for the table in the Accelerator Studio.

Figure 3-4 shows one table that has not yet been loaded and is in Initial Load Pending. The other five tables have been successfully loaded and therefore would be refreshed if selected for load.

Name	Size	Rows	Acceleration	Last Load	Replication Since	Distribution Key	Skew	Organizing Keys	Organiz...
BID100KB	533 MB	- 80 of 80	80 of 80 tables	0 of 80	-	-	-	-	-
BID10TB	2.89 TB	- 80 of 80	80 of 80 tables	0 of 80	-	-	-	-	-
BID1TB	302 GB	- 80 of 80	80 of 80 tables	0 of 80	-	-	-	-	-
BID200GB	65.2 GB	- 1 of 1	1 of 1 tables	0 of 1	-	-	-	-	-
BID200MD	56.7 GB	- 1 of 1	1 of 1 tables	0 of 1	-	-	-	-	-
BIDU200	27 MB	- 5 of 6	5 of 6 tables	1 of 6	-	-	-	-	-
GOHR_BRANCH	3 MB	28 Enabled	5/9/13 10:39 AM	Disabled	Random	0.000			
GOHR_COUNTRY	2 MB	21 Enabled	4/30/13 2:49 PM	Disabled					
GOHR_EMPLOYEE	7 MB	102 Enabled	4/30/13 2:49 PM	Disabled					
GOHR_EMPLOYEE_DETAILS	11 MB	303 Enabled	4/30/13 2:49 PM	Disabled					
GOHR_GENDER_LOOKUP	4 MB	46 Enabled	4/30/13 2:49 PM	Disabled	Random	0.000			
GOSLDW_SALES_FACT	-	- Disabled	Initial load pending	Disabled	Random	0.000			

Figure 3-4 Last Load status

For a refresh operation, the data in the Accelerator for the selected table or partitions is replaced using the following process:

1. A “soft” delete occurs for the rows associated with the table or partition being refreshed. The rows are not physically deleted and are still available for query processing while the load is running. Their physical delete occurs when the LOAD has completed and the transaction deleting the rows finally executes COMMIT. If the table is very large, this delete processing can add some time to the overall load process.
2. The table or partition is loaded as described earlier in this chapter. The new data is not available until the load successfully completes. If the load fails, the table will be returned to its original state just before the load.

3. After a successful refresh load, the new data is made available and the old data is no longer available. A low-priority background process is started that physically removes the deleted data and will not interfere with query processing.

Hint: If you are doing a full table refresh for a large table and there is no need to have queries running during the load process, it might be faster to remove the table from the Accelerator, add it back to the Accelerator, and then do an initial load. This avoids the overhead of processing the soft deletes prior to the load.

If the DB2 table being refreshed is a non-partitioned table, the entire table is refreshed. If the table is partitioned, individual partitions can be selected for refresh. However, if the partition boundaries of a range-partitioned table have changed, the entire table must be reloaded. There is a feature, *change detection*, that helps determine what tables or partitions have changed and therefore need to be refreshed.

Change detection

The change detection feature helps make data maintenance on the Accelerator easier and helps reduce resource consumption by identifying DB2 tables or partitions that have data changes since the last load. Changes are detected by utilizing DB2 real-time statistics. The following changes can be detected:

- ▶ SQL statements: INSERT, UPDATE, DELETE, and TRUNCATE TABLE
- ▶ Utility executions: LOAD RESUME and LOAD REPLACE
- ▶ Partition operations: ADD PARTITION and ROTATE PARTITION
- ▶ Partition boundary changes: ALTER TABLE and REORG REBALANCE
A boundary change will require a reload of the entire table
- ▶ Table data exchanged with a clone table: EXCHANGE DATA

Figure 3-5 on page 64 shows the changes that are detected for a partitioned table in the Accelerator. Data changes have been detected in the first two partitions and are indicated as such in the Load Recommended column. In addition, those partitions are pre-selected to be refreshed. However, refreshing data changes is technically optional, but the DB2 table and the Accelerator will be out of synch. A new partition has been added to the table and will need to have an initial load performed as indicated, with the value Enforced - Initial load in the Load Recommended column and it will be loaded when the load runs.

Figure 3-5 on page 64 depicts detected changes that are shown in the load dialog.

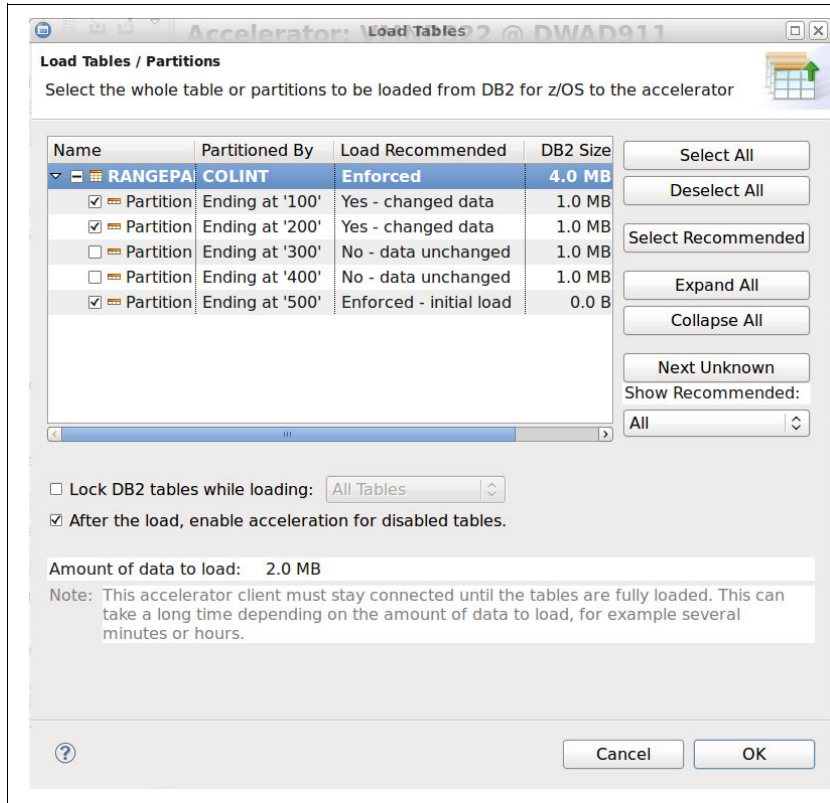


Figure 3-5 Detected changes are shown in the load dialog

There are situations where it is not possible to detect the changes. In these cases, the Load Recommended columns will have a recommendation of yes with a status of unknown, as shown in Figure 3-6 on page 65. In this case, a **REORG** utility was run on the table after the last load and the table has not had any updates since then. An unknown status might also occur for a table if another table in the same table space has had changes.

The **REORG** utility resets the real-time statistics (RTS) counters, and a refresh of the table or partition will be enforced. If there were no changes to the DB2 table since the last load to the Accelerator, this results in a false positive situation. This scenario forces the table to be refreshed even though it had no changes. This can be avoided by running the **REORG** utility before loading to the Accelerator.

The **RECOVER** utility clears the RTS statistics for the affected table or partitions that are involved in the recover. This results in no change detection being possible until the table is REORGed in DB2 and reloaded to the Accelerator.

Figure 3-6 on page 65 shows an example of change detection not being able to determine if the table changed.

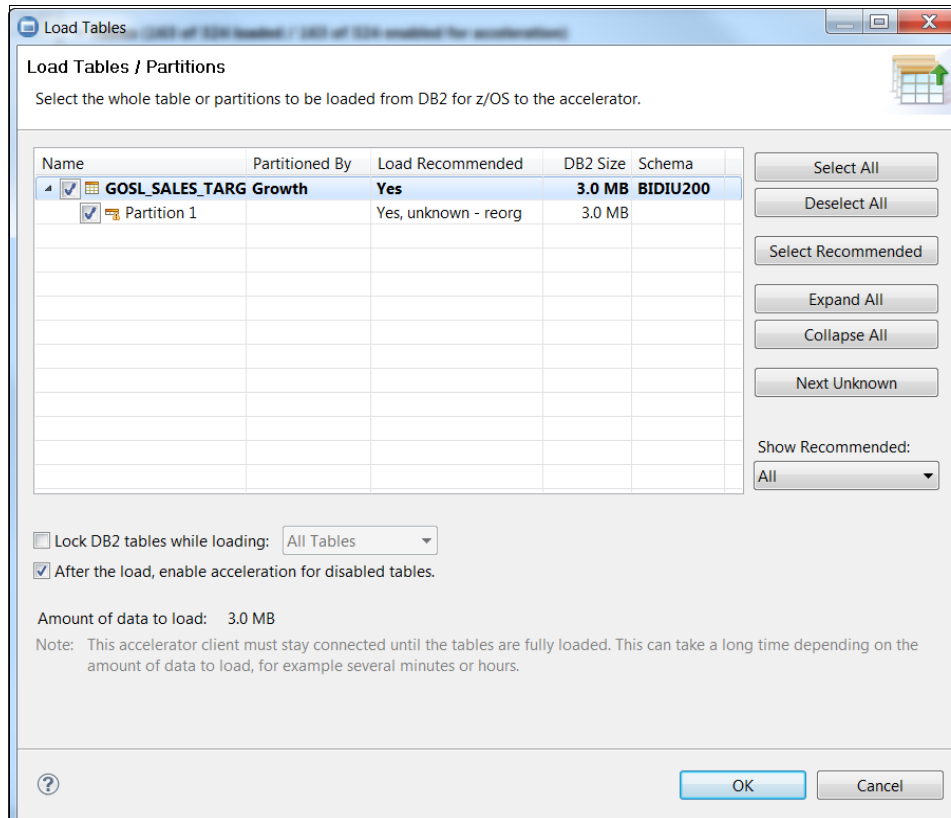


Figure 3-6 Example of change detection not being able to determine if the table changed

DB2 real-time statistics

DB2 real-time statistics (RTS) was originally designed to help determine when DB2 utilities, such as **REORG**, should be run. RTS keeps counters of how many changes in terms of inserts, updates, and deletes have been made to a table space or partition. These counters are externalized to the `SYSIBM.SYSTABLESPACESTATS` catalog table at an interval that is determined by the `STATSINT` system parameter with a default of 30 minutes. Some DB2 utilities such as **REORG**, **COPY**, and **RUNSTATS** also externalize the counters. At Accelerator load time, these counters are stored by the Accelerator for every table or partition loaded. If the counters and timestamps match from the last Accelerator load, it is assumed that the table or partition has not changed and does not need to be refreshed. Conversely, if they do not match, it is assumed that some changes have occurred.

Accurate change detection depends upon timely information in `SYSIBM.SYSTABLESPACESTATS`. False change detection might occur if changes occur to a table space since the last externalization of the RTS statistics. This can be avoided by forcing an externalization of the RTS statistics:

- ▶ Issue the DB2 command: **START DATABASE (db) SPACENAM (ts)**
- ▶ Run DB2 Runstats: **RUNSTATS TABLESPACE db.ts REPORT NO UPDATE NONE**
- ▶ Issue the DB2 command: **ACCESS DATABASE(db) SPACENAM(*) MODE(STATS)**

Note: The first two options might have side effects. The third option externalizes the RTS statistics with no side effects. This adds a new `MODE` value, `STATS`, for the **ACCESS DATABASE** command. DB2 APAR PM71744 is required to use this new mode.

3.3 Load performance considerations

As discussed in 3.2, “Understanding the load process” on page 58, the load builds a pipeline between DB2 and the Accelerator and pumps data from the source DB2 to table to the target Accelerator table. This is analogous to a water pipeline. The flow rate of water through the pipeline is limited by the flow rate of the smallest component, whether that be the speed that the water pump can push water into the pipe, how fast the target pump can take the water out of the pipe, or the size of the pipe.

The same is true for the data load pipeline between DB2 and the Accelerator. DB2 is the pump that pushes the data into the pipeline, the software running on the Accelerator taking the data out of the pipeline and inserting it into the target table on the Accelerator, and the pipeline is the USS and DRDA pipes and the networking infrastructure between DB2 and the software running on the Accelerator. If any one of these components is restricted, the flow of data, the load throughput, will be restricted. Assuming the absence of network issues, performance is primarily limited by how many parallel load streams can be supported on System z up to a certain point when the limit becomes the available resources on the Accelerator side for processing and loading the data.

Note: For optimal load performance, we suggest the following product level and authorized program analysis reports (APARs):

- ▶ IBM DB2 Analytics Accelerator V3.1 PTF3 or above (reduced overhead per load per table)
- ▶ DFSMS APAR OA41706 (better performance for BSAM/QSAM/BPAM)
- ▶ DB2 APAR PM90151 (DB2 now uses the DEPENDENT(YES) attribute when inserting the Workload Manager (WLM) request to schedule the stored procedure)

3.3.1 Network considerations

The load process can transfer hundreds of GB or even terabytes of data across the network. Good network design is necessary to obtain the maximum load throughput between DB2 and the Accelerator. Ensure that this network is designed properly and is fast enough. Make sure that maximum transmission unit (MTU) sizes match across the network. The ACCEL_TEST_CONNECTION stored procedure can be used to test the network and to view the network setting that is being used, which would be useful in troubleshooting any network issues. See *IBM DB2 Analytics Accelerator for z/OS Version 3.1.0 Stored Procedures Reference*, SH12-6984, for details for the ACCEL_TEST_CONNECTION stored procedure.

Loading data from DB2 to the Accelerator can result in a massive amount of data flowing across the network. The quality of the network between the System z and the Accelerator can affect the overall throughput of loads and can also impact query performance. The network should be a dedicated 10 GB Ethernet network, as defined in the Accelerator prerequisites document at <http://www.ibm.com/support/docview.wss?uid=swg27035960>.

Of course, network design can be very complex. Guidance for the network design between the System z and the Accelerator can be found in the following documents:

- ▶ *IBM DB2 Analytics Accelerator for z/OS Version 3.1.0 Installation Guide*, SH12-6983
<http://publibfp.dhe.ibm.com/epubs/pdf/h1269831.pdf>
- ▶ Support home for IBM DB2 Analytics Accelerator for z/OS
https://www-947.ibm.com/support/entry/myportal/Overview/Software/Information_Management/DB2_Analytics_Accelerator_for_z~OS

- ▶ *Optimizing DB2 Queries with IBM DB2 Analytics Accelerator for z/OS*, SG24-8005
<http://www.redbooks.ibm.com/abstracts/sg248005.html?Open>
- ▶ IBM Whitepaper: *Network requirements for System z*
<http://www.ibm.com/support/docview.wss?uid=swg27024236>
- ▶ IBM Whitepaper: *Network connections for IBM DB2 Analytics Accelerator*
<http://www.ibm.com/support/docview.wss?uid=swg27023654>
- ▶ IBM Whitepaper: *Network Configurations for IBM DB2 Analytics Accelerator for z/OS*
<http://www.ibm.com/support/docview.wss?uid=swg27028171>

3.3.2 Source considerations

The primary tuning knob for adjusting overall throughput is the number of parallel load streams that can be supported. From 3.2.2, “Loading partitions in parallel” on page 60, you can see that each load stream creates a DB2 **UNLOAD** utility and an Accelerator process to receive the data from the network, Accelerator process to convert the data, and a data feed process to insert the data into the target Accelerator table. This consumes resources on the System z side, the network, and on the Accelerator. This section focuses on the System z side of the equation.

Running parallel load streams

For this discussion, refer to Figure 3-7 on page 68. There are four source tables. Table1 is a range-partitioned table with nine partitions of various sizes. Table 2 is also a range-partitioned table with four partitions of roughly the same size. Table3 and Table4 are non-partitioned tables. Each table has the same column definition, the same number of columns, and the same data types. The difference is the number of rows and number of partitions.

Serial execution: no parallel load streams

Figure 3-7 on page 68 also indicates how many time units that each table takes to unload, assuming no partition parallelism. Table1 takes twelve time units; the other three tables take roughly the same amount of time at four time units.

When loading all four tables to the Accelerator with one call to `ACCEL_LOAD_TABLES` with parallelism forced off (`AQT_MAX_UNLOAD_IN_PARALLEL = 1`), the four tables are loaded to the Accelerator serially and each partition within the partitioned tables are also loaded serially. The `ACCEL_LOAD_TABLES` stored procedure is called once. It will, one at a time, create a load stream for each partition to load that partition.

That means an invocation of the DB2 **UNLOAD** utility via a call to the `DSNUTULU` stored procedure, creation of USS pipes, creation of a DRDA connection, and starting of processes on the Accelerator. This happens fifteen times: Nine for the nine partitions of Table1, four for the partitions of Table2, and one time each for Table3 and Table4. The total elapsed time is twenty-four time units, which is the sum of the total load time of the four tables.

Figure 3-7 on page 68 shows four source tables.

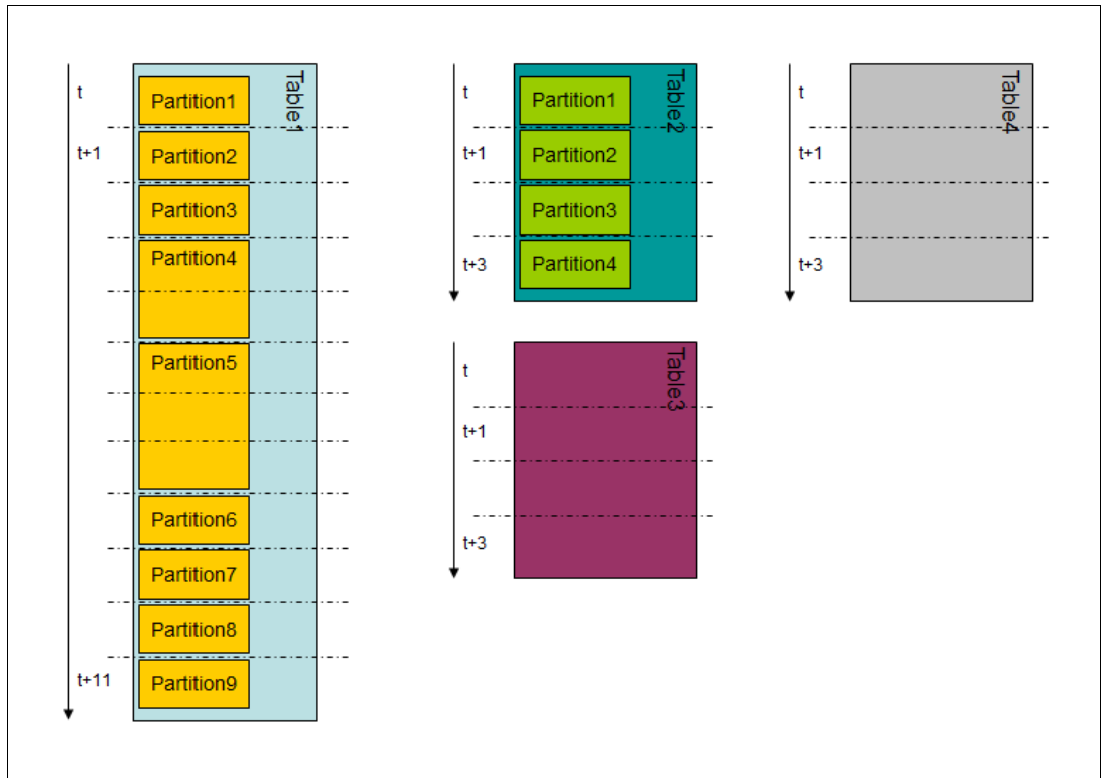


Figure 3-7 Four source tables

Two parallel load streams

Recall that **AQT_MAX_UNLOAD_IN_PARALLEL** defines the maximum number of parallel load streams that can be created for one partitioned table. This applies independently to each partitioned table and has no effect on non-partitioned tables.

Setting **AQT_MAX_UNLOAD_IN_PARALLEL** = 2 means that for each partitioned table, up to two parallel load streams will be established. Figure 3-8 on page 69 shows how the load proceeds over time. Because there is one invocation of the **ACCEL_LOAD_TABLES** stored procedure, the tables are processed serially in order: Table1, then Table2, then Table3, then Table4.

Table1 is a partitioned table. Therefore, it can establish up to two parallel load streams. **ACCEL_LOAD_TABLES** establishes a load stream for Partition1. It has not reached the **AQT_MAX_UNLOAD_IN_PARALLEL** value of 2, so it establishes another load stream for Partition2. Now the max load parallelism has been reached so it has to wait until one finishes before starting another. It continues in this manner until all partitions of the table have been loaded. The order that was selected, resulted in the two sets of tables that had the same amount of data. Therefore, the total load time was cut in half to six time units compared to the twelve for loading the partitions serially. This doubled the overall load throughput of this table, but at a cost of more resources being utilized.

This scenario requires two DSNUTILU stored procedure address spaces and two sets of processes on the Accelerator.

Hint: The DB2 **DISPLAY UTILITY** command can be used to see the number of DB2 unload utilities that are executing concurrently. The **ACCEL_LOAD_TABLES** stored procedure creates utility IDs that start with "AQT". Use: **-DIS UTIL(AQT*)** to display these unload utilities.

After the load for Table1 has completed, ACCEL_LOAD_TABLES establishes the load streams to load Table2, which is also partitioned so that up to two parallel load streams can be run in parallel. The load time for Table2 was also cut in half, from four time units to two units.

Upon completion of the load for Table2, only one load stream is established for Table3 because it is a non-partitioned table and the load time remains at four time units. When the load for Table3 has completed, Table4 is loaded. It is also a non-partitioned table and cannot utilize parallel load streams and the load time remains the same at four time units.

Running two parallel load streams is shown in Figure 3-8.

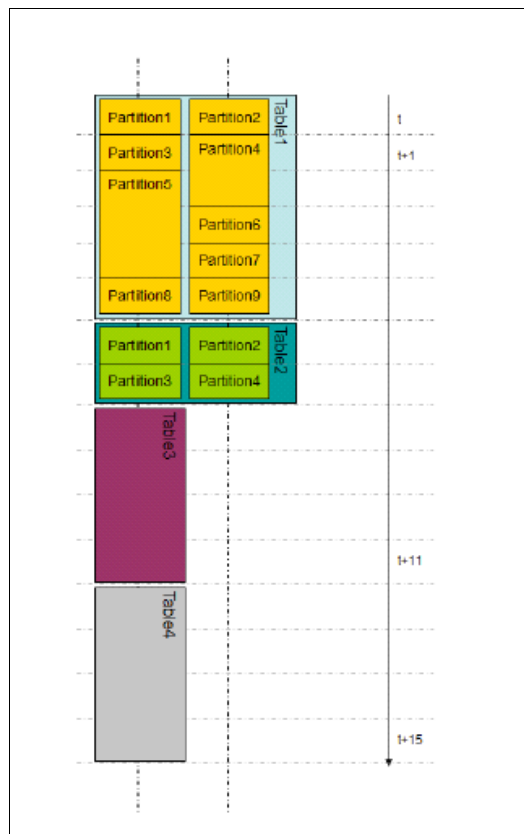


Figure 3-8 Running two parallel load streams

Taking advantage of utilizing parallel load stream capability of partitioned tables, the overall load time for this set of four tables was reduced from twenty-four time units to sixteen time units, resulting in a load time improvement of 33%. Of course, the cost is having the resources to run two concurrent load streams.

Six parallel load streams

Utilizing two parallel load streams reduced the overall load time significantly. Will increasing the parallelism to six further reduce the load time? Setting `AQT_MAX_UNLOAD_IN_PARALLEL = 6` results in the run profile that is shown in Figure 3-9 on page 70.

Notice that ACCEL_LOAD_TABLES initially establishes six load streams for Table1 for the first six partitions. When Partition1 completes, a load stream for Partition7 is established. After the completion of Partition2, Partition8 is loaded; and after Partition3, Partition9 is loaded. The next partition to complete is Partition6 but there are no more partitions to load, resulting in a parallelism of five. Partition4, Partition7, Partition8, and Partition9 all complete

around the same time, but Partition5 is not yet finished. It is the only load during the last time unit. The load time for Table1 has been reduced to three time units from six time units, with a parallelism of two. There was only one time unit where there were six parallel load streams, one time unit where there were only five parallel load streams, and one unit where there was only one load stream. Furthermore, the largest partition, Partition5, took three time units, which defines the minimum run time for loading this table, no matter how many parallel load streams are established.

Table2 is loaded only after all partitions of Table1 have completed loading. Remember, load parallelism is on an individual table basis. Table2 only has four partitions and parallel load streams are created for all four. The load time for Table2 has been reduced from two units at a parallelism of two, to one time unit at a parallelism of four.

Table3 and Table4 are loaded serially as before. Because they are both non-partitioned, they cannot take advantage of parallelism. Therefore, they run in the same four time units each as in the other scenarios.

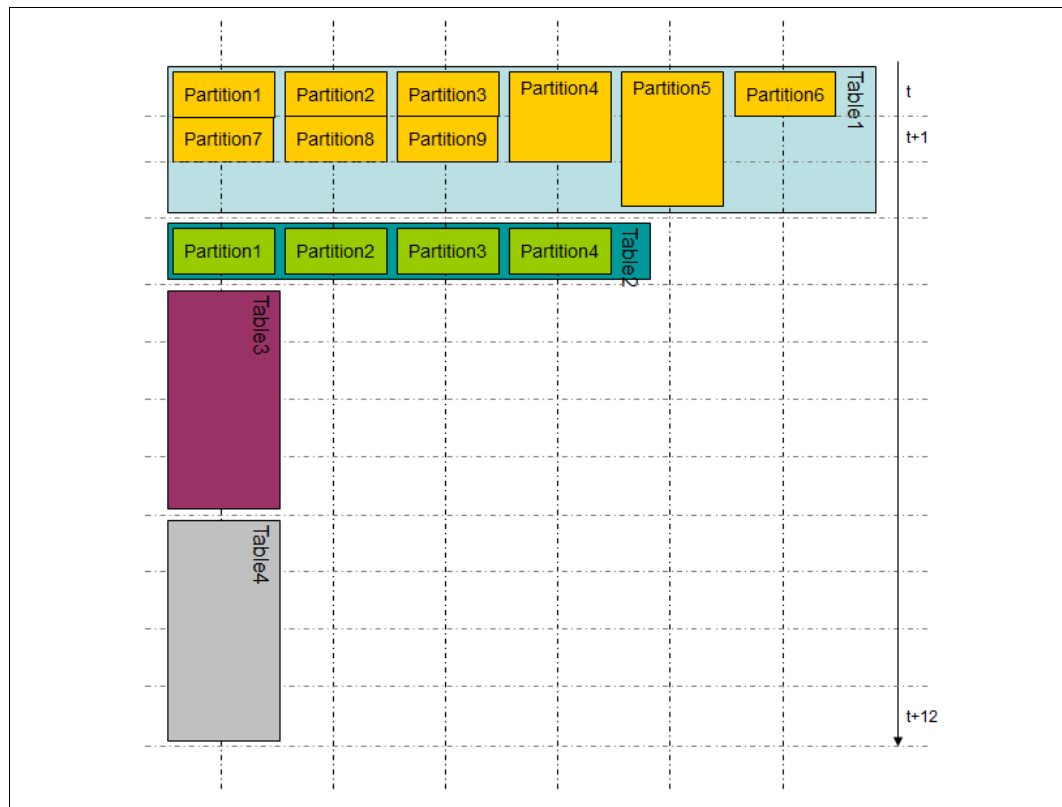


Figure 3-9 Running six parallel load streams

Changing the load parallelism to six did allow more parallel load streams in the two parallel tables. There had to be enough free resources to concurrently run up to six DB2 **UNLOAD** utilities in six DSNUTILU address spaces. This further reduced the overall load time for all four tables by another four time units, resulting in a total load time of twelve time units. Would increasing **AQT_MAX_UNLOAD_IN_PARALLEL** reduce overall load time further? It would not. The only table that would have an increase in parallelism is Table1. In theory, all nine partitions could load in parallel. However, the load time for Partition5 will always be three time units; therefore, there would be no increase in load time for Table1. **AQT_MAX_LOAD_IN_PARALLEL** could even be reduced to five without any effect on the overall load time.

Parallel load streams with multiple stored procedure calls

The previous scenarios used only one call to the ACCEL_LOAD_TABLES stored procedure, passing in the list of all four tables. Of course, the stored procedure is not limited to one invocation at a time. In the Accelerator Studio, after selecting a set of tables and clicking the **load** button, the running load can be run in the background, freeing up the interface to allow another selection of tables to be loaded. Multiple invocations can also be accomplished by having multiple invocations of batch programs, REXX execs, and so on, explicitly calling ACCEL_LOAD_TABLES with a restriction that any one table can be used only once in any concurrent load scenario.

In this scenario, shown in Figure 3-10, there are three concurrent invocations of the ACCEL_LOAD_TABLES stored procedure. The two partitioned tables are provided in the table's list of the first invocation, Proc1. **AQT_MAX_UNLOAD_IN_PARALLEL** is set to 4. ACCEL_LOAD_TABLES is called again; Proc2, for Table3; and again, Proc3, for Table4.

Proc1 loads Table1 using up to four parallel load streams followed serially by Table2 with up to four parallel load streams. The load for Table1 takes four time units and Table2 takes one time unit for a total overall load time of five time units for Proc1. There were up to five address spaces used at one time for this procedure. There was one for the ACCEL_LOAD_TABLES stored procedure and up to four for the parallel load streams.

Proc2 was submitted at the same time as Proc1 but only included Table3 in its table list. Because this was an additional invocation of the ACCEL_LOAD_TABLES stored procedure, another stored procedure address space was required for it and a new DSNUTILU address space for the associated DB2 **UNLOAD** utility. Table3 is not partitioned so it did not take advantage of parallelism and completed in four time units.

Proc3 was also submitted at the same time as Proc1 and included only Table4 in the table list. As with Proc2, two additional stored procedure address spaces were established, one for the new invocation of ACCEL_LOAD_TABLES, and one for DSNUTILU to run the DB2 **UNLOAD** utility. Table4, like Table3, was not partitioned and took four time units to complete.

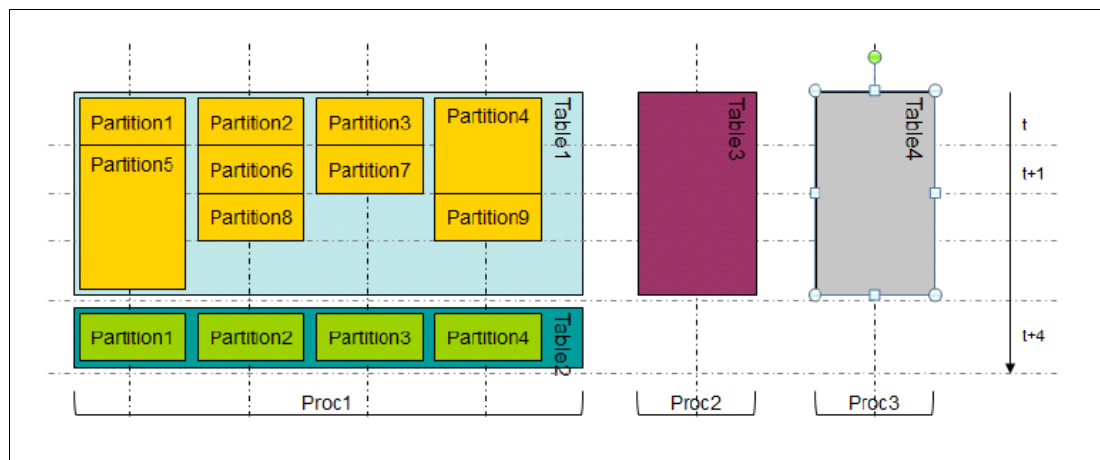


Figure 3-10 Creating parallel load streams with multiple ACCEL_LOAD_TABLES procedure calls

The overall load time is the load time of the procedure that took the longest. Proc1 took five time units. Therefore, the overall load time has been reduced to five time units while not exceeding a maximum of six parallel streams. On the other hand, this required enough resources to support up to nine concurrent DB2 stored procedure address spaces. However, the resultant load time was reduced from the original twenty-four time units to just five time units.

You might be wondering why not call a fourth procedure for loading Table2. Referring to Figure 3-10 on page 71, you see that Table3 takes one time unit. If we used a fourth invocation, Proc4, to ACCEL_LOAD_TABLES keeping AQT_MAX_UNLOAD_IN_PARALLEL = 4, Proc4 would take one time unit reducing Proc1 to four time units, which takes down our overall load time to four time units.

However, we have added an additional call to ACCEL_LOAD_TABLES resulting in an additional stored procedure address space. In addition, ACCEL_LOAD_TABLES will establish four more concurrent stored procedure address spaces for the four DB2 unload utilities. This results in a total of four stored procedure address spaces for the four ACCEL_LOAD_TABLES and ten stored procedure address spaces for the ten concurrent DB2 unload utilities. In addition, there will be ten USS pipes, ten DRDA connections, and a set of ten Accelerator processes for conversion and loading. This could result in resource contention either on the System z side or the Accelerator side, which might have the effect of elongating the total load time.

If the establishment of a stored procedure address space is very high, you can use the **MNSPAS** parameter in the startup job control language (JCL) of the WLM address space that is associated with the DB2 utilities stored procedure, DSNUTILU, as shown in Example 3-1. The **MNSPAS** parameter specifies the minimum number of address spaces that are to be started and maintained. The value can be 0 - 50. This keeps the specified number of address spaces established, thereby avoiding the overhead of starting a new address space for each invocation of the DB2 **UNLOAD** utility. If there are numerous relatively small tables or partitions, this decrease in address space startup time can help to reduce the overall elapsed time of loading data to the Accelerator.

Example 3-1 Setting the MNSPAS parameter for the DSNUTILU stored procedure

```
//DZA1WLMU PROC APPLENV='DB2WLM_UTILS',
//                DB2SSN=DZA1,RGN=OK,NUMTCB=1,MNSPAS=10
```

DB2 APAR PM90151 reduces the time that it takes when a stored procedure calls another stored procedure by a few seconds. Although this might be noise when loading a large table or partition, it can result in an increased load throughput for situations where there are many small tables or small partitions to load.

Note: If resources on the System z side are constrained, the System z Workload Manager (WLM) might slow the establishment of stored procedure address spaces or might fail the call if it cannot start another address space. The error received is: SQLCODE -471, with a REASON CODE of 00E97002.

Tip: Generally, the range of 8 - 10 parallel load streams gives a good load processing performance for the N1001; the N2001 might yield 12-16 parallel load streams. This assumes that there are no constraints on the z/OS side. Other factors might allow more or less parallel load streams.

Determining the number of parallel load streams

The number of parallel load streams that can be supported is a primary factor in increasing the overall load throughput. The maximum number of parallel streams can be calculated by the following formula:

$$\text{number of concurrent load jobs for nonpartitioned tables} + (\text{AQT_MAX_UNLOAD_IN_PARALLEL value} * \text{number of concurrent load jobs for partitioned tables})$$

The example in Figure 3-10 on page 71 shows one concurrent load job for all the partitioned tables, plus two concurrent load jobs for the non-partitioned tables with **AQT_MAX_UNLOAD_IN_PARALLEL** = 4. The formula for this example is $2 + (1 \times 4)$, which equals 6. There would need to be enough resources available on System z, the network, and the Accelerator to support six concurrent load streams, plus the three address spaces for **ACCEL_LOAD_TABLES**.

There is a practical limit to how many concurrent load streams that can run. This is primarily dependent on available system resources but other factors also need to be considered, such as the row size, number of rows, data types, and the number and size of partitions. Tests should be conducted to determine the specifics for a particular configuration of hardware and set of DB2 tables.

Figure 3-11 shows the results of a study to determine the optimum number of concurrent load streams that could be supported by various levels of available System z processor resources for a particular workload. At each System z processor configuration (1, 2, 3, 10, 16), there was a point where adding additional load streams did not result in an appreciable increase in average load throughput (top line). At a configuration of one processor, we maxed out at about three concurrent load streams. With no processor constraints on System z, we saw no benefit after 8 - 10 concurrent load streams. Another interesting fact is that the average throughput per individual load stream decreased as more load streams were added.

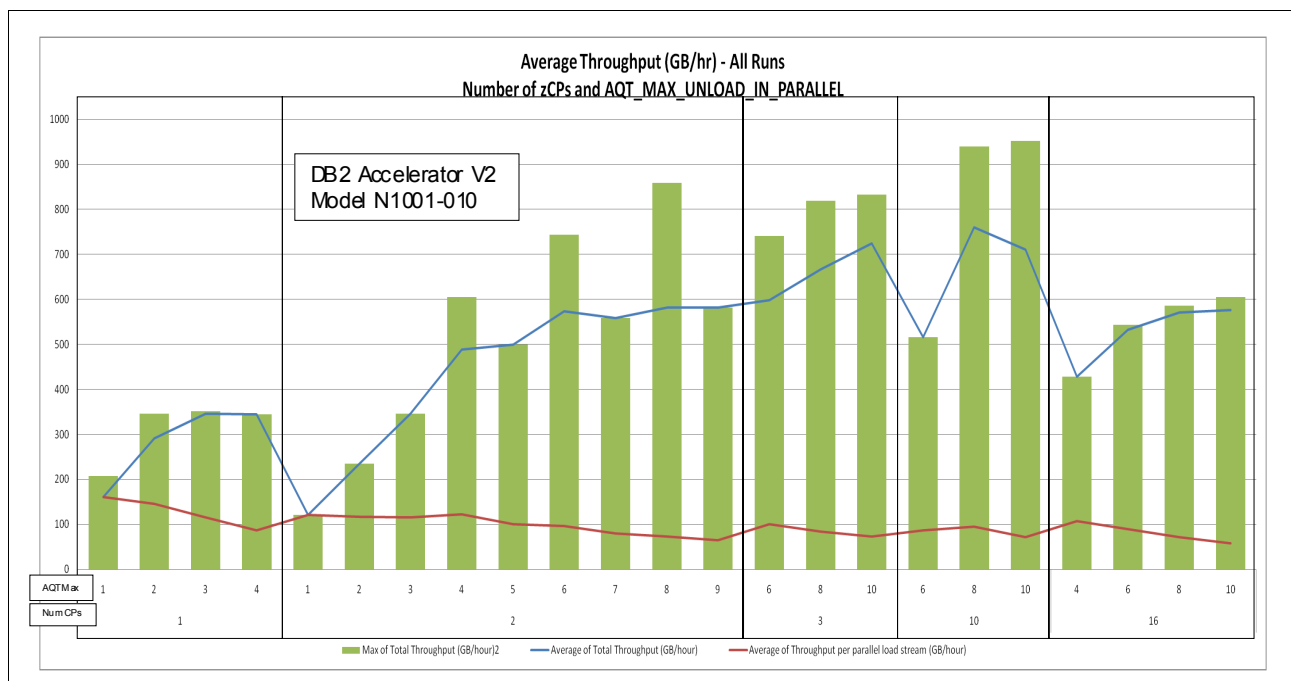


Figure 3-11 Load throughput varying number of System z processors and parallel streams

If there were plenty of System z resources, why did the throughput level off? The number of concurrent load streams that can be supported is primarily dependent on the resources that are available on System z and the Accelerator. When there is no constraint on System z, we can increasingly push more data through the pipeline. However, there is a point when the Accelerator cannot receive and process the data any faster, therefore the limit becomes the Accelerator. Figure 3-12 on page 74 shows the processor utilization on the System z and the Accelerator host processors for various numbers of available System z processors, with various degrees of parallel load streams. On the left side of the chart, you can see that the System z processors are constrained, thereby limiting the data throughput to such a point that the Accelerator has no problem processing the flow of data.

As you move to the right of the chart, the constraint on System z processors is reduced so data flow is increased consequently causing the Accelerator to become busier. At some point, the Accelerator host processors become saturated. For this workload, the Accelerator maxes out at around eight concurrent load streams.

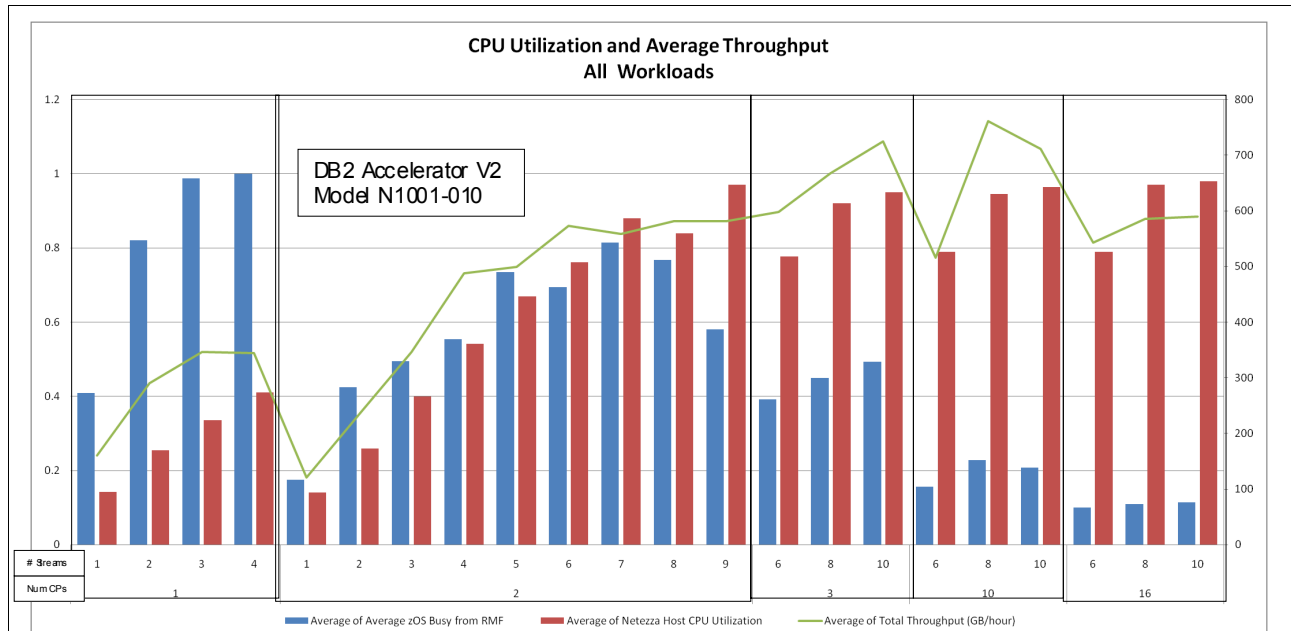


Figure 3-12 Number of parallel load streams versus available resources

The point where you reach the point of diminishing returns is also dependent on the characteristics of the data and of the table structure. The conversion for score source data types, *float* for example, cause a higher processor load on the Accelerator. The number and size of tables and partitions are a factor. If there are many small tables or partitions, the overhead of invoking stored procedures has a greater affect and reduces the overall load throughput.

In addition, the specific model of Accelerator has an affect on load performance. Typically, the multi-rack Accelerators have faster host processors and, in some cases, more host processors. Newer models also include faster host processors. Figure 3-13 on page 75 compares the N1001-010 to the N2001-010 using three different workloads, which are described in Figure 3-14 on page 75.

Moving from the N1001-010 to the N2001-010, we saw a dramatic increase in the maximum number of concurrent load streams that could be supported by an unconstrained System z EC12 along with a corresponding increase in load throughput. For the N1001, the load concurrency was in the 8 - 10 range but increased to the 16 - 18 range for the N2001; throughput doubled or nearly doubled.

Also interesting in this chart is the comparison of the two Cognos Great Outdoor sample workloads. These are essentially the same tables but with a change in one column from a *FLOAT* to a *DECIMAL*. The Accelerator processor cycles that are required to convert *FLOAT* data is costly compared to *DECIMAL* data. Changing the data type to *DECIMAL* had a significant affect on the load throughput.

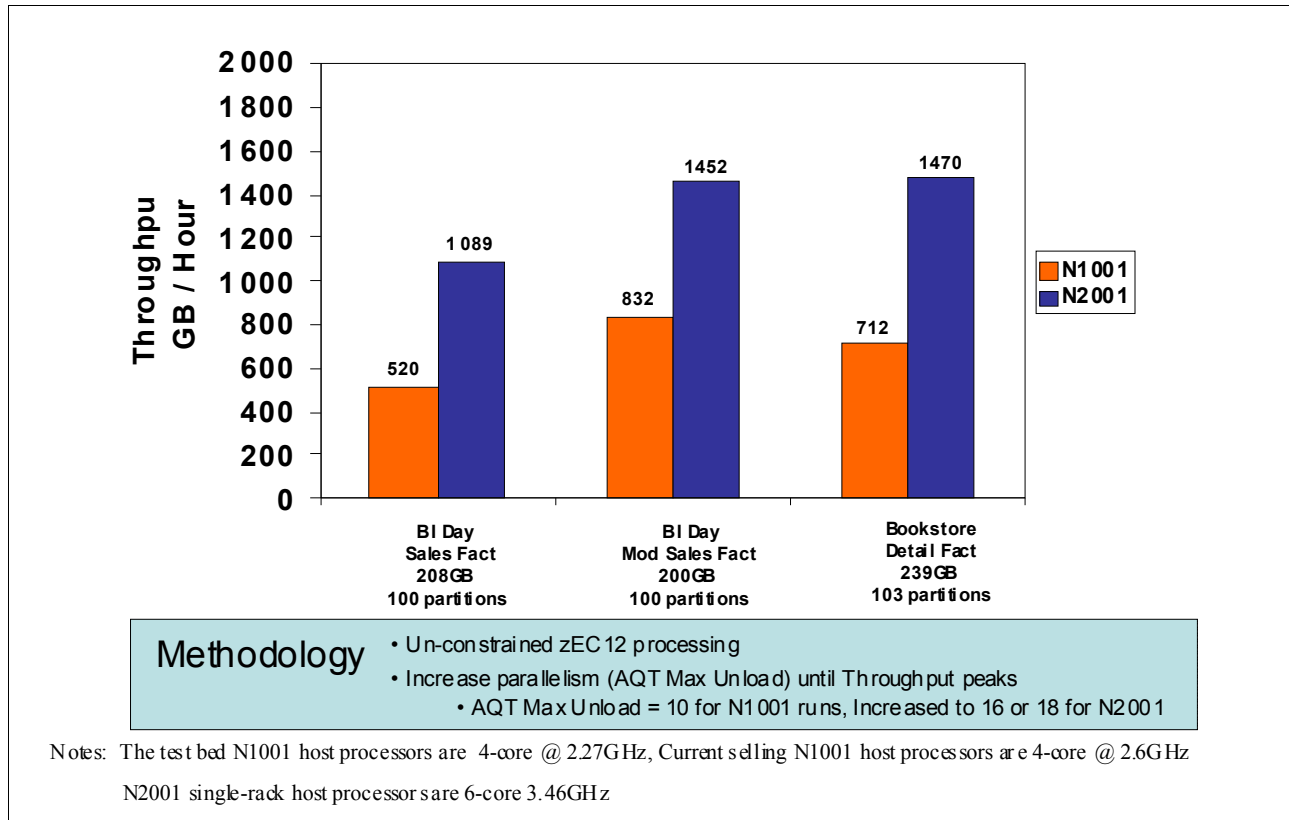


Figure 3-13 Comparing load throughput for N1001-101 and N2001-010

Table	Amount of Data Transferred	RECLENGTH (DB2 Catalog)	# rows	Columns
BI Day Sales Fact	200 GB	106	2 Billion	10 INTEGER, 5DECIMAL, 1 FLOAT
BI Day Mod Sales Fact	200 GB	102	2 Billion	10 INTEGER, 6 DECIMAL
Bookstore Detail Fact	239 GB	64	3.8 Billion	5 INTEGER, 1 BIGINT, 1 DECIMAL, 1 CHAR, 1 TIMESTAMP, 1 DATE

BIDAYSales Fact				BIDAYModSales Fact				BookorderDetail Fact			
Column Name	Col	Type	Length	Column Name	Col	Type	Length	Column Name	Col	Type	Length
ORDER_DAY_KEY	1	INTEGER	4	ORDER_DAY_KEY	1	INTEGER	4	BOD_ANALYSIS_ID	1	BIGINT	8
PRODUCT_KEY	2	INTEGER	4	PRODUCT_KEY	2	INTEGER	4	BOD_ORDER_ID	2	INTEGER	4
STAFF_KEY	3	INTEGER	4	STAFF_KEY	3	INTEGER	4	BOD_ISBN	3	CHAR	10
RETAILER_SITE_KEY	4	INTEGER	4	RETAILER_SITE_KEY	4	INTEGER	4	BOD_CUSTOMER_ID	4	INTEGER	4
ORDER_METHOD_KEY	5	INTEGER	4	ORDER_METHOD_KEY	5	INTEGER	4	BOD_STORE_ID	5	INTEGER	4
SALES_ORDER_KEY	6	INTEGER	4	SALES_ORDER_KEY	6	INTEGER	4	BOD_DATE	6	DATE	4
SHIP_DAY_KEY	7	INTEGER	4	SHIP_DAY_KEY	7	INTEGER	4	BOD_TSTAMP	7	TIMESTAMP	10
CLOSE_DAY_KEY	8	INTEGER	4	CLOSE_DAY_KEY	8	INTEGER	4	BOD_QUANTITY	8	INTEGER	4
RETAILER_KEY	9	INTEGER	4	RETAILER_KEY	9	INTEGER	4	BOD_PRICE_PER_ITEM	9	DECIMAL	7
QUANTITY	10	INTEGER	4	QUANTITY	10	INTEGER	4	BOD_SEQ_NUM	10	INTEGER	4
UNIT_COST	11	DECIMAL	19	UNIT_COST	11	DECIMAL	19				
UNIT_PRICE	12	DECIMAL	19	UNIT_PRICE	12	DECIMAL	19				
UNIT_SALE_PRICE	13	DECIMAL	19	UNIT_SALE_PRICE	13	DECIMAL	19				
GROSS_MARGIN	14	DECIMAL	7	GROSS_MARGIN	14	DECIMAL	7				
SALE_TOTAL	15	DECIMAL	19	SALE_TOTAL	15	DECIMAL	19				
GROSS_PROFIT	16	DECIMAL	19	GROSS_PROFIT	16	DECIMAL	19				

Figure 3-14 Load study workloads

Estimating the unload data size

There are two key factors in optimizing the overall load time for a set of tables. The first is the maximum number of concurrent load streams that can be supported, which is discussed in “Determining the number of parallel load streams” on page 72. The second factor is the amount of data to be loaded in terms of uncompressed bytes. This information can be used to help optimize the mix of load jobs and to determine the appropriate value of the **AQT_MAX_UNLOAD_IN_PARALLEL** environment variable.

The uncompressed size of the table or partition determines the amount of time that it takes for a single load stream to complete. If one partition of a table is twice the size of another partition, it will take roughly twice the amount of time to load. The absolute minimum load time for a set of tables will be the time to load the largest table or partition. If a non-partitioned table is very large, it might benefit from partitioning. On the other hand, there is overhead associated with the process of initializing a load stream, which must be accounted for when there are many small, or empty, tables or partitions.

Understanding the uncompressed size of individual tables and partitions helps to determine the grouping of tables in load jobs. The general rule is that all load jobs should have in sum the same amount of uncompressed data to load.

Size information in the DB2 catalog tables that is displayed in Data Studio shows sizes in terms of compressed data, which differs from the unload data size. Because it is the uncompressed data size that influences load, it is necessary to estimate the size of the uncompressed data. Estimating the unload data size depends on three factors: Number of columns, column type, and number of rows.

Example 3-2 shows an SQL statement used to determine the unload size of all partitions of all tables that have been added to the named Accelerator, <ACCELNAME>, which is replaced with the name of the Accelerator of interest. To narrow the list, the SQL can also be further qualified on the AT.CREATOR or AT.NAME columns.

Example 3-2 SQL to determine the unload size of tables

```
SELECT
  AT.CREATOR as "Table_Schema"
  , AT.NAME as "Table_Name"
  , ST.RECLENGTH as "Table_RecordLength"
  , ST.AVGROWLEN as "Table_AvgRowLength"
  , ( SELECT COUNT(SC.COLNO)
      FROM SYSIBM.SYSCOLUMNS SC
      WHERE SC.TBCREATOR = AT.CREATOR
            AND SC.TBNAME = AT.NAME
            AND SC.COLTYPE IN
              ('VARCHAR',
              'LONGVAR',
              'VARG',
              'LONGVARG',
              'CLOB',
              'BLOB',
              'DBCLOB',
              'VARBIN',
              'XML' )
      ) as "Table_NumberVarLengthColumns"
  , TS.DBNAME as "TableSpace_Database"
  , TS.NAME as "TableSpace_Name"
  , TP.PARTITION as "Partition_Number"
```

```

, TP.AVGROWLEN as "Partition_AvgRowLength"
, TP.PAGESAVE as "Partition_PageSave"
, TP.CARD as "Partition_NumberRows"
, TSS.TOTALROWS as "TableSpaceStats_TotalRows"
, (CASE WHEN TP.CARD < 0
  THEN 'Nan'
  ELSE
    (CASE WHEN
      (
        SELECT COUNT(SC.COLNO)
        FROM SYSIBM.SYSCOLUMNS SC
        WHERE
          SC.TBCREATOR = AT.CREATOR
          AND SC.TBNAME = AT.NAME
          AND SC.COLTYPE IN
            ('VARCHAR',
             'LONGVAR',
             'VARG',
             'LONGVARG',
             'CLOB',
             'BLOB',
             'DBCLOB',
             'VARBIN',
             'XML' )
      ) > 0
      THEN
        ( CASE WHEN TP.PAGESAVE > 0
          THEN
            (QUANTIZE( ROUND(
              (
                CAST(TP.AVGROWLEN AS DECFLOAT(34)) *
                (CAST(100E0 AS DECFLOAT(34)) /
                  (
                    CAST(100E0 AS DECFLOAT(34)) -
                    CAST(TP.PAGESAVE as DECFLOAT(34))
                  )
                )
              ) * CAST(TP.CARD AS DECFLOAT(34))
              ,0), DECFLOAT('1E+0'))
            )
          ELSE
            (
              CASE WHEN (TP.AVGROWLEN <> 0
                AND TP.CARD <> 0)
              THEN
                ( CAST(TP.AVGROWLEN as DECFLOAT(34)) *
                  CAST(TP.CARD as DECFLOAT(34))
                )
              ELSE
                ( CAST(ST.RECLENGTH as DECFLOAT(34)) *
                  CAST(TSS.TOTALROWS as DECFLOAT(34))
                )
              )
            )
          END
        )
      )
    )
  END
END

```

```

    )
  ELSE
    (
      CASE WHEN (TP.CARD = 0)
      THEN
        ( CAST(ST.RECLENGTH as DECFLOAT(34)) *
          CAST(TSS.TOTALROWS as DECFLOAT(34))
        )
      ELSE
        ( CAST(ST.RECLENGTH as DECFLOAT(34)) *
          CAST(TP.CARD as DECFLOAT(34))
        )
      )
    )
  END
)
END
)
END
) as "CalculatedUncompressedSize"
FROM SYSACCEL.SYSACCELERATEDTABLES AT
LEFT OUTER JOIN SYSIBM.SYSTABLES ST
  ON AT.CREATOR = ST.CREATOR
  AND AT.NAME = ST.NAME
LEFT OUTER JOIN SYSIBM.SYSTABLESPACE TS
  ON ST.DBNAME = TS.DBNAME
  AND ST.TSNAME = TS.NAME
LEFT OUTER JOIN SYSIBM.SYSTABLEPART TP
  ON TS.NAME = TP.TSNAME
  AND TS.DBNAME = TP.DBNAME
LEFT OUTER JOIN SYSIBM.SYSTABLESPACESTATS TSS
  ON TP.TSNAME = TSS.NAME
  AND TP.DBNAME = TSS.DBNAME
  AND TP.PARTITION = TSS.PARTITION
WHERE
  AT.ACCELERATORNAME = '<ACCELNAME>'
WITH UR;

```

The result column, CalculatedUncompressedSize, shows the expected unload sizes of the table or partition. The unload sizes can be used to group the table into batch jobs using the knowledge in “Running parallel load streams” on page 67.

Estimating System z processor cost for load

The processor cost of loading from DB2 to the Accelerator might be a concern in a number of shops. Since the original release of the IBM DB2 Analytics Accelerator V2.1, there have been two changes that have greatly reduced the processor cost of loading data.

The first of these changes occurred with the release of V2.1.3 of the Accelerator. A change was made to have the DB2 **UNLOAD** utility unload into the DB2 internal format, thereby avoiding the cost of data conversion into an external format. This change alone reduced the System z processor cost by approximately 50%.

The second change was an optimization in intermodel communications for XSAM (BSAM/QSAM/BPAM) used for accessing z/OS UNIX files that are used in the USS pipes to which the unloaded DB2 data is fed. This resulted in another 35 - 40% reduction in process cost that is associated with loading data to the Accelerator.

Tip: The change to XSAM is available in APAR OA41706.

With IBM DB2 Analytics Accelerator V3.1 with APAR OA41706 applied, the System z processor cost for loading is 0.5 seconds per 100 MB of data. Remember that the unit of load is a partition. By knowing which partitions are to be loaded along with the load size estimate technique that is shown in “Estimating the unload data size” on page 76, it is simple to obtain a rough estimate of processor cost of the load using the following formula:

$$\text{CP seconds} = 0.5 \# (\text{SUM}(\text{load size of partitions to be loaded}) / 1024 / 1024) / 100$$

3.3.3 Target considerations

On the Accelerator side, how the load process affects the utilization of the Accelerator host processors is the primary consideration. The tasks on the host processors receive the data sent over the network, converts the data from the internal DB2 data format to a format understood by the native Accelerator database, and then inserts the data into the underlying Accelerator database.

As the number of parallel load streams increase, so does the processor utilization of the Accelerator host system. At some point, the Accelerator host becomes saturated and increasing the number of parallel streams does not yield much increased throughput. In some cases, this can have a negative impact if the Accelerator host becomes over burdened.

Another consideration in addition to the number of parallel load streams is the type of data coming from DB2. Some DB2 data types, particularly floating point numbers, are more costly to convert, thereby increasing the processor load on the Accelerator host. This could reduce the number of parallel load streams that can be supported while loading tables containing that data type.

The **DIS ACCEL** DB2 command can be used to monitor the Accelerator host processor utilization. Keep in mind that this value is updated every 20 seconds.

3.4 Automating the load process

The Accelerator Studio is a nice graphical user interface that makes it easy to administer the Accelerator. However, it is not appropriate for production environments where the loading of data to the Accelerator needs to be integrated into normal production operations. An example is the case where there is a nightly batch process that updates the data warehouse. Having the Accelerator adds an extra step to refresh the data in the Accelerator after the data warehouse has been updated. This refresh of the Accelerator needs to be integrated into the nightly ETL process and should not need to require the effort to start the Accelerator Studio and manually refresh the data.

Fortunately, all the functions of the Accelerator Studio result in calls to stored procedures that are specific to the Accelerator. These stored procedures can be called directly and are documented in *IBM DB2 Analytics Accelerator for z/OS Version 3.1.0 Stored Procedures Reference*, SH12-6984. In this section, we describe a subset of these stored procedures that is appropriate to automating the load process.

Calling Accelerator stored procedures

The Accelerator stored procedures are just normal DB2 stored procedures. However, they do utilize XML documents as input and output parameters. The calling environment must be able to create and pass these XML strings as well as interpret the XML return strings. Of course, most programming and scripting languages can handle this. This chapter highlights a REXX example. Some tooling, however, may not easily handle XML. Check with your tool provider to see if this is supported.

In the System z world, the most common method used by automation tools is to call batch JCL jobs in some type of precedence flow. It is therefore imperative that the loading of the Accelerator can easily be accomplished within a JCL job. There is a provided C program that can be used to call a subset of Accelerator stored procedures directly from JCL while providing the XML strings using in-line DD statements. Another way is to develop a program (REXX, COBOL, Java, and so on) that does the stored procedure calls. Using a programming language is much more flexible than a call to the provided sample program. In this section, an example of using the supplied sample program and using a REXX program will be explored.

Accelerator stored procedures related to loading

A complete listing of Accelerator stored procedures is listed in Chapter 2, “Using the Studio client to manage the environment” on page 11. All of the Accelerator stored procedures have a schema of SYSPROC. For brevity, the following discussion will leave off the schema part of the stored procedure name. For example, the full name of ACCEL_LOAD_TABLES will be SYSPROC.ACCEL_LOAD_TABLES.

The most obvious Accelerator stored procedure that is associated with loading is the one that actually does the load, ACCEL_LOAD_TABLES. Other stored procedures might also be used in the context of loading data to the Accelerator. ACCEL_REMOVE_TABLES and ACCEL_ADD_TABLES can be used to remove and re-add a table when a full table refresh is required. ACCEL_GET_TABLES_DETAILS can be used to determine which tables or partitions have changed. If users should not be running queries during the load, the ACCEL_SET_TABLES_ACCELERATION could be used to disable acceleration just before the load begins, and enable acceleration after the load has finished.

Using the AQTSCALL sample program

The AQTSCALL program is a sample C program provided in the AQTSCALL member SAQTSAMP data set. It must be compiled and can be used to call a subset of Accelerator stored procedures from JCL. The XML string parameters are provided in-line using DD statements.

The following Accelerator functions are supported:

- ▶ ADDTABLES: Calls ACCEL_ADD_TABLES to add a specified list of tables to the Accelerator
- ▶ LOADTABLES: Calls ACCEL_LOAD_TABLES to load a specified list of tables to the Accelerator
- ▶ REMOVETABLES: Calls ACCEL_REMOVE_TABLES to remove a specified list of tables from the Accelerator
- ▶ SETTABLES: Calls ACCEL_SET_TABLES_ACCELERATION to enable or disable acceleration for a specified list of tables
- ▶ ARCHIVETABLES: Calls ACCEL_ARCHIVE_TABLES to archive specified partitions of a set of tables

See the member AQTSJI03 of SAQTSAMP for JCL to compile and bind AQTSCALL, define and load some sample DB2 tables, and call various Accelerator stored procedures.

Example 3-3 shows the JCL to do a full reload of a DB2 table to the Accelerator. In the Accelerator Studio, this is accomplished with just a few clicks. In JCL, AQTSCALL will be invoked for each function of remove a table, add a table, load a table, and enable a table for acceleration.

The stored procedure that is to be called is specified in the **PARTS** parameter of the RUN PROGRAM(AQTSCALL) statement. The function then looks for certain DD statements that contain two or three parameters that are specific for the stored procedure call. Another DD, AQTMSGIN, is optional but controls tracing and sets a compatibility level.

The first step, RMTABLE, in Example 3-3, removes the existing table from the Accelerator, which is specified by the **REMOVETABLES** parameter to AQTSCALL. The REMOVETABLES function requires that parameter 1, **AQTSP1**, is the name of the Accelerator and is DZA1IDAA. Parameter 2, **AQTSP2**, is the XML string that defines the list of DB2 tables to remove from the Accelerator. In this example, there is just one table in the list, APERKIN.GOSLDW_SALES_FACT. The optional AQTMSGIN is not provided.

Note: The format of the XML strings is published in the *IBM DB2 Analytics Accelerator for z/OS Version 3.1.0 Stored Procedures Reference*, SH12-6984, with examples provided in the SAQTSAMP data set.

The second step, ADDTABLES, adds the table, APERKINS.GOSLDW_SALES_FACT, back to the Accelerator using the AQTSCALL parameter **ADDTABLES**. We might want to do this if we need to reload the entire table. It is a large table and we do not want to occur the overhead of the delete processing on the Accelerator. AQTSP1 specifies the name of the Accelerator, and AQTSP2 defines the list of tables to add, in this case just the one. The DD, AQTMSGIN, contains an XML string that defines the level of tracing for this call and where in the USS file system to store the trace.

The third step, LOADFULL, loads the data from DB2 into the table that was just added to the Accelerator. It uses the AQTSCALL parameter **LOADTABLES** and has three parameters. AQTSP1, as in the previous two steps, defines the name of the Accelerator to use. However, AQTSP2 now specifies the DB2 locking level of NONE. AQTSP3 is the list of tables to load. ATQMSGIN is again used to specify the trace level and location of the trace output.

The final step, SETACC, enables the loaded table for acceleration. In the Accelerator Studio, this can be done automatically after the load completes by checking the appropriate option in the load table's dialog. When using stored procedures, it must be done explicitly. The AQTSCALL function is SETTABLES. AQTSP1 is the name of the Accelerator. AQTSP2 specifies whether to enable or disable acceleration, which in this case is on to enable acceleration. Then, AQTSP3 specifies the list of tables. This step will not contain a trace because the DD AQTMSGIN is not provided.

Example 3-3 JCL to remove, add, load, and enable a table for acceleration

```
//*****
//*   Remove APERKIN.GOSLDW_SALES_FACT from DZA1IDAA
//*****
//RMTABLE EXEC PGM=IKJEFT01,DYNAMNBR=20
//* parameter #1 specify accelerator nme
//AQTP1 DD *
DZA1IDAA
/*
//* parameter #2 specify xml listing tables to be removed
//AQTP2 DD *
<?xml version="1.0" encoding="UTF-8" ?>
```

```

<aqt:tableSet
xmlns:aqt="http://www.ibm.com/xmlns/prod/dwa/2011" version="1.0">
<table schema="APERKIN" name="GOSLDW_SALES_FACT" />
</aqt:tableSet>
/*
/* optional parameter for message input is omitted (NULL)
//SYSTSPRT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
    DSN SYSTEM(DZA1)
        RUN PROGRAM(AQTSCALL) PLAN(AQTSCALL) -
            LIB('WORKLOAD.ISAS.IDAA31.SAMPLoad') PARMS('REMOVETABLES')
    END
//*****
/* Add table APERKIN.GOSLDW_SALES_FACE to DZA1IDAA
//*****
//ADDTABLE EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
/* parameter #1 specify accelerator name
//AQTP1 DD *
DZA1IDAA
/*
/* parameter #2 specify xml listing tables to be added
//AQTP2 DD *
<?xml version="1.0" encoding="UTF-8" ?>
<aqt:tableSpecifications
xmlns:aqt="http://www.ibm.com/xmlns/prod/dwa/2011" version="1.0">
<table schema="APERKIN" name="GOSLDW_SALES_FACT" />
</aqt:tableSpecifications>
/*
/* last parameter for optional message input to control trace
/* and backward compatibility.
/* this example would create a trace (unicode) in /tmp/spadd.txt
/* and enforce output that matches Accel. V3.1 (SP interface version 4)
//AQTMMSGIN DD *
<?xml version="1.0" encoding="UTF-8" ?>
<spctrl:messageControl
xmlns:spctrl="http://www.ibm.com/xmlns/prod/dwa/2011" version="1.1">
<compatibilityLevel>4</compatibilityLevel>
<traceConfig location="/tmp/GOSLDW.txt" keepTrace="true">
    <component name="PROCEDURE" level="INFO"/>
</traceConfig>
</spctrl:messageControl>
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
    DSN SYSTEM(DZA1)
        RUN PROGRAM(AQTSCALL) PLAN(AQTSCALL) -
            LIB('WORKLOAD.ISAS.IDAA31.SAMPLoad') PARMS('ADDTABLES')
    END
//*****
/* Load table APERKIN.GOSLDW_SALES_FACT to DZA1IDAA
/* If the table is already loaded, this will enforce a full reload.

```

```

//*****
//LOADFULL EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
/* parameter #1 specify accelerator name
//AQTP1 DD *
DZA1IDAA
/*
/* parameter #2 specify LOCK_MODE
//AQTP2 DD *
NONE
/*
/* parameter #3 xml listing the tables to be loaded
//AQTP3 DD *
<?xml version="1.0" encoding="UTF-8" ?>
<aqt:tableSetForLoad
xmlns:aqt="http://www.ibm.com/xmlns/prod/dwa/2011" version="1.1">
<table schema="APERKIN" name="GOSLDW_SALES_FACT"
forceFullReload="true"/>
</aqt:tableSetForLoad>
/*
//AQTMMSGIN DD *
<?xml version="1.0" encoding="UTF-8" ?>
<spctrl:messageControl
xmlns:spctrl="http://www.ibm.com/xmlns/prod/dwa/2011" version="1.1">
<compatibilityLevel>4</compatibilityLevel>
<traceConfig traceFileSizeInMB="20" keepTrace="true"
location="/tmp/cdcload.txt">
<component name="PROFILING" level="INFO"/>
</traceConfig>
</spctrl:messageControl>
/* optional parameter for message input is omitted (NULL)
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(DZA1)
RUN PROGRAM(AQTSCALL) PLAN(AQTSCALL) -
LIB('WORKLOAD.ISAS.IDAA31.SAMPLoad') PARMS('LOADTABLES')
END
//*****
/* Set ACCLERATION enabled for APERKIN.GOSLDW_SALES_FACT
//*****
//SETACC EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
/* parameter #1 specify accelerator name
//AQTP1 DD *
DZA1IDAA
/*
/* parameter #2 for enable ON or enable OFF
//AQTP2 DD *
ON
/*
/* parameter #3 xml listing tables to be enabled/disabled
//AQTP3 DD *
<?xml version="1.0" encoding="UTF-8" ?>
<aqt:tableSet
xmlns:aqt="http://www.ibm.com/xmlns/prod/dwa/2011" version="1.0">

```

```

<table schema="APERKIN" name="GOSLDW_SALES_FACT" />
</aqt:tableSet>
/*
/* optional parameter for message input is omitted (NULL)
//SYSTSPRT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(DZA1)
  RUN PROGRAM(AQTSCALL) PLAN(AQTSCALL) -
    LIB('WORKLOAD.ISAS.IDAA31.SAMPLOAD') PARMS('SETTABLES')
END

```

Using REXX to call Accelerator stored procedures

The sample program, AQTSCALL, described in section “Using the AQTSCALL sample program” on page 80, can call only a subset of Accelerator stored procedures, albeit the most common ones. If you need to call other stored procedures, need more flexibility, need to use some type of procedural logic, or maybe call some DB2 functions as well, some type of language needs to be used. One widely used language on the System z platform is REXX.

Calling Accelerator stored procedures in REXX is fairly easy. The authors had a need to repeatedly reset a test scenario for testing incremental update. Functions included populating a DB2 table with a specified number of rows, re-adding the table to the Accelerator, and loading the data to the Accelerator. In addition, other Accelerator functions needed to be called to stop a table from being replicated, and to stop and restart replication (that is, incremental update).

It is fairly simple to call an Accelerator stored procedure from REXX. Example 3-4 shows a template for calling an Accelerator stored procedure, ACCEL_REMOVE_TABLES. Recall from section “Using the AQTSCALL sample program” on page 80, that this stored procedure requires two parameters: the Accelerator name and a listing of tables to remove. These two parameters are supplied in the REXX variables, :Acce1Name and :TableSetSpec. :Msg is an input and output parameter. Input is optional, which will control tracing. An output XML string will be returned containing any output messages. The :MsgInd variable is a numeric value that can be used to indicate if there are any messages returned. Other Accelerator stored procedures might have a slightly different call pattern due to having different parameter requirements.

Example 3-4 Calling an Accelerator stored procedure

```

ADDRESS DSNREXX "EXEC SQL CALL SYSPROC.ACCEL_REMOVE_TABLES (",
    ":Acce1Name,",
    ":TableSetSpec,",
    ":Msg      INDICATOR :MsgInd )"

```

Example 3-5 on page 85 shows a snippet of REXX code calling the ACCEL_REMOVE_TABLES stored procedure, including setting the variables that are used in the call. TableSetSpec and MsgSpec are strings that contain the XML required for those parameters. After the call, there is some rudimentary error handling. The SQLCODE should be checked to ensure that the call was successful. If so, the return message needs to be checked for unexpected conditions.

Example 3-5 REXX code snippet calling ACCEL_REMOVE_TABLES

```
AccelName = 'DZA1IDAA'

TableSetSpec = '<?xml version="1.0" encoding="UTF-8" ?> ',
               '<aqttables:tableSet ',
               'xmlns:aqttables="http://www.ibm.com/xmlns/prod/dwa/2011" ',
               'version="1.0"> ',
               '<table name="GOSLDW_SALES_FACT" schema="APERKIN" />',
               '</aqttables:tableSet>'

MsgSpec = '<?xml version="1.0" encoding="UTF-8" ?> ',
          '<spctrl:messageControl ',
          'xmlns:spctrl="http://www.ibm.com/xmlns/prod/dwa/2011" ',
          'version="1.0" versionOnly="false" > ',
          '</spctrl:messageControl>',
          copies(' ',32000)

Msg = msgSpeed.

ADDRESS DSNREXX "EXECSQL CALL SYSPROC.ACCEL_REMOVE_TABLES (",
               ":AccelName,",
               ":TableSetSpec,",
               ":Msg      INDICATOR :MsgInd ) "
say 'The SQLCODE is: ' SQLCODE

if SQLCODE < 0 then
  do
    SQLERRORPOSITION = 'Call Stored Procedure'

    if MsgInd >= 0 then
      say "Message:" Msg

    call SQLERRORROUTINE
  end
else /* Stored Procedure completed successfully */
  do
    say "-----"
    say "Table has been removed from accelerator"
    say "-----"

    if MsgInd >= 0 then do
      say "Message:" Msg
      if (pos('AQT10000I',Msg) = 0) then
        exit 4
      end
    else
      say "No message available"
    end
  end
```

See Appendix E, “Additional material” on page 343 for how to download the complete REXX program listing and JCL to run the REXX program.

3.5 Summary

This chapter introduced the management of loading data from DB2 to the Accelerator. There are two basic methods to accomplish this task:

- There is the method of loads that do bulk data movement of data from DB2 tables to the Accelerator.
- In addition, there is a capability of picking up data changes in DB2 as they happen and updating the Accelerator with that changed data in near real-time. This is described in detail in Chapter 7, “Incremental update” on page 161.

In this chapter, we described in some detail the bulk, or batch, load process, how it works, and some performance considerations. Getting the best throughput for moving bulk data from DB2 to the Accelerator is a balancing act between resource utilization on the System z side and resource utilization on the Accelerator. Having a resource constraint, mostly in terms of the processor, on either side can limit the total throughput that is achievable.

Load throughput can be increased by smartly designing parallel load streams. Each partitioned table automatically uses parallel load streams up to the number specified in the **AQT_MAX_UNLOAD_IN_PARALLEL** environment variable. Additional parallel load streams can be started by additional invocations of load. There is a practical limitation to how many parallel load streams can be supported by a given set of tables and available resources. Sometimes, care must be given in the design of the load jobs to make the most effective use of available resources. We observed that the N1001 has a sweet spot of around 8 - 10 parallel load streams. Some clients have reported less than that and some have reported getting more parallel loads going up to 16. We have observed that the N2001 has a sweet spot in the 14 - 16 range. As always, it is dependent on the specific scenario.



Accelerator resource management

In this chapter, we provide DB2 Analytics Accelerator resource management considerations.

The following new Accelerator V3.1 resource management concepts are described:

- ▶ Query prioritization
- ▶ Accelerator maintenance task prioritization
- ▶ Workload isolation

For each control, we describe what it is, how to use it, and conduct experiments. The experiments are designed to understand how the new controls change the accelerator performance behavior and to exhibit some anticipated client use-cases.

We include an Accelerator priority summary table, which summarizes the priority controls and default for different types of work and tasks that run on the Accelerator.

We also provide a summary of all related accelerator resource management modifications made to our test environment since its inception (that is, including V2.1 and V3.1), and why. This is provided to serve as an Accelerator resource management checklist for clients.

The contents encompass changes made to System z Workload Manager (WLM) service definition and the DB2 Analytics Accelerator configuration console.

We assume some basic knowledge of WLM in this chapter. For a primer on WLM and WLM for DB2, we recommend Appendix F, “WLM refresher” in *Co-locating Transactional and Data Warehouse Workloads on System z*, SG24-7726.

The following topics are covered:

- ▶ Accelerator resource management
- ▶ Accelerator prioritization within a DB2 subsystem
- ▶ Experiments: Prioritization within a DB2 subsystem
- ▶ Allocating resources with workload isolation for a shared Accelerator
- ▶ Workload isolation experiments
- ▶ Accelerator resource management checklist

4.1 Accelerator resource management

Before we start our evaluation, the first question to ask yourself is “Is Accelerator resource management important?”. Given the multitude of workloads concurrently running on a typical z/OS system, a System z performance administrator fully appreciates the importance of System z WLM and its ability to effectively and efficiently manage resources. Though, the question above is relative to the Accelerator.

The answer of course is a resounding “YES!”. Consider what the Accelerator is designed for, massively parallel processing (MPP). Therefore, it is expected that a significant portion of time, the total resources (processor, disk, channels) of the Accelerator will all be working on one or a just a few queries. From a business perspective, ensure that it is putting all those resources *first* towards the work that is most important to the business. Having quality resource management policies in place is also key to capacity planning. Quality policies can avoid or delay the need to expand, which saves money.

What is new in DB2 Analytics Accelerator V3.1 resource management?

V3.1 provides new controls for prioritizing work on an Accelerator. These controls can be divided into two categories: Accelerator prioritization for a given DB2 subsystem, and Accelerator prioritization across multiple DB2 subsystems.

For a given DB2 subsystem, you now have the ability to prioritize accelerated distributed data facility (DDF) queries and Accelerator maintenance tasks. When a given Accelerator is shared by multiple DB2 subsystems, you have the ability to set a relative percentage of Accelerator processing resources for each of the DB2 subsystems that are sharing it. We cover this in depth in this chapter, though you might want to also reference the “Beyond the basics” section within *IBM DB2 Analytics Accelerator for z/OS Version 3.1.0 Installation Guide*, SH12-6983.

It is an appliance!

During this project, we often had to remind ourselves of this point. A System z performance administrator is accustomed to having several WLM controls within their means and a deep understanding of the underlying technology. One of the benefits of an appliance is that there is less administrative management required. We struggled with getting accustomed to this concept. Perhaps, in a smaller way it can be related to what a DB2 SQL tuning specialist is going through. Given that we cannot get “under the hood”, we thought it would be of some value to provide some general Accelerator resource management behaviors that we ascertained via experiments and some discussion with development. The following are some generalities that we learned, some of which are pronounced in our experiment results:

- ▶ *Query priority*

Queries with higher priority will obtain more processing time slices than queries with lower priority. The larger the gap in priority, between competing queries, the larger the gap in the percentage of processing resources that get distributed.

- ▶ *Fair-share mentality*

Assuming that all queries have the same priority, the Accelerator will attempt to provide each concurrent query its fair share of the Accelerator resources. When the current mix of actively running queries reaches some point of Accelerator resource saturation, the additional work starts to queue.

- ▶ *Prioritization when queuing occurs*

When the Accelerator reaches a point of resource saturation, additional queries are queued for processing, regardless of their priority. One exception is short-running queries,

which are discussed next. It is important to note that the Accelerator orders the queued queries by their priority.

► *Favoritism for short-running queries*

This type of query should not be confused with WLM's period-aging. However, the Accelerator provides some favorable treatment via reserving some resources for short-running queries for times that the Accelerator is saturated with longer running queries.

► *Dynamic changes to Accelerator priorities*

Changes to any of the Accelerator priorities affects only newly arriving work (after the change). Work (such as queries) that is already actively running is not affected.

Experiment objectives

We conducted several experiments using the newly available controls. Following are our objectives:

- Understand the value of the new controls and specific priority settings that are now available. For example, if we prioritized a query or task higher than others, how did the performance of the favored task change?
- Gain best practices on how to utilize the new controls.
- Provide use-case scenarios for you to use in your environment.

4.2 Accelerator prioritization within a DB2 subsystem

In this section, we describe the following new Accelerator resource management controls for prioritizing work within a given DB2 subsystem:

- Query prioritization
- Accelerator data maintenance task prioritization

We then follow this with experiments exploiting those controls.

4.2.1 Query prioritization

System z extends some of its resource management capabilities to the Accelerator by using the already known WLM "Importance" that is associated with each query. In particular, the WLM Importance is passed to the Accelerator along with the query itself. This Importance is mapped to internal Accelerator priorities, resulting in accelerated queries that are being managed in a manner that is consistent with System z. Table 4-1 provides the current four levels of priority that are utilized to differentiate work on the accelerator. Notice how work that is running at Importance 3, 4, or 5 all run at the same priority on the Accelerator.

Table 4-1 Mapping of WLM Importance to Accelerator priorities

WLM Importance	Accelerator priorities
System	Critical
Importance 1	Critical
Importance 2	High
Importance 3, 4, 5	Normal
Discretionary	Low

Note: When a query is accelerated, DB2 passes to the Accelerator the associated WLM service class *first period* Importance, regardless if there are multiple periods in the WLM service class. Hence, to effectively prioritize your accelerated queries, you need to ensure to differentiate first period Importance of the various WLM service classes that your queries are classified to. Reference the service class definitions that we used to serve distributed data facility (DDF) queries in our project (see Figure 4-23 on page 119).

Although this *query prioritization* extension of WLM's current capabilities is quite powerful and useful (as you will see from our experiments), it is important to clarify the following:

- ▶ WLM *period aging* is not used. It is the service class first period Importance that is utilized.
- ▶ WLM resource groups have no significance for the work that is running on the Accelerator.
- ▶ When a given query is submitted to the Accelerator, dynamically changing the associated WLM service class or active WLM policy has no impact. Only additional queries that get submitted after the WLM modification are affected.
- ▶ When the Accelerator is saturated and cannot actively execute additional queries, except for queries that are determined to be short-running, any new queries are queued in the order of their priority.
- ▶ The Accelerator work is not managed to a traditional WLM goal.

4.2.2 Accelerator data maintenance task prioritization

The V3.1 Accelerator provides the ability to change the default priorities of some of the Accelerator data maintenance tasks, such as Accelerator table loads, Accelerator statistics collection (*genstats* - similar to RUNSTATS in DB2), and Accelerator data reorganization (*groom* - similar to REORG in DB2).

One of the values of this new control is providing performance administrators the ability to either prioritize table loads over concurrent query activity or just the opposite, prioritize some query activity over table loads. These are two scenarios that we examined in our experiments in 4.3, "Experiments: Prioritization within a DB2 subsystem" on page 94.

The means to change data maintenance task priorities is through the DB2 Analytics Accelerator configuration console. The following list describes the necessary steps.

1. Obtain the IP address of the Accelerator from your network administrator.
2. Start a 3270 emulator and log on to TSO/ISPF.
3. Enter the following command to use Telnet to connect to the DB2 Analytics Accelerator console:

```
tso telnet <hostname> 1600
```

The values are defined as follows:

<hostname>	This is the IP address of the Accelerator that is connected to the DB2 for z/OS data server.
1600	This is the number of the port that is configured for accessing the IBM DB2 Analytics Accelerator console using a Telnet connection between the DB2 for z/OS data server and the Accelerator.

Tip: The TSO/ISPF Telnet session does not scroll automatically. When the window is filled, the message *HOLDING* will display on the bottom right. To display the next window, press **CLEAR** (Pause key on the emulator).

4. Press Enter until you receive a prompt to enter the console password. Enter your console password. The initial password is *dwa-1234*. You must change this password at the first logon:

Enter password (use PF3 to hide input):

Then, press Enter.

5. You are now presented with the IBM DB2 Analytics Accelerator configuration console (Figure 4-1). Type **3** and press Enter to go to the “Run Accelerator Functions” menu.

```
Enter password (in TSO, use PF3 to hide input):
Licensed Materials - Property of IBM
5697-DAA
(C) Copyright IBM Corp. 2009, 2013.
US Government Users Restricted Rights -
Use, duplication or disclosure restricted by GSA ADP Schedule
Contract with IBM Corporation

*****
* Welcome to the IBM DB2 Analytics Accelerator Configuration Console
*****

You have the following options:
(1) - Change the Configuration Console Password
(2) - (Menu) Run Netezza Commands
(3) - (Menu) Run Accelerator Functions
(4) - (Menu) Manage Incremental Updates
-----
(x) - Exit the Configuration Console
```

Figure 4-1 DB2 Analytics Accelerator Configuration Console main menu

6. You are presented with seven options (Figure 4-2). Type **5** and press Enter to continue.

```
-----
You have the following options:

(0) - Go back one level
(1) - Obtain pairing code, IP address, and port
(2) - List paired DB2 subsystems
(3) - Set resource limits for DB2 subsystems
(4) - Clear query history
(5) - Specify the priority of maintenance tasks
(6) - Set the DB2 subsystem for time synchronization

(Default 0) >
```

Figure 4-2 Run Accelerator Functions menu

7. You are now prompted to select a database from the list of DB2 subsystems that share the Accelerator. In our test environment we had two subsystems, as shown in Figure 4-3. Type the number next to the subsystem of interest and press Enter.

```
Select a database system:
  1 : DBZZDDF
  2 : DZA1DDF

Select database system by name or id (0 to go back):
```

Figure 4-3 Select the DB2 subsystem

8. Referencing Figure 4-4, you are now prompted to set the priority by typing the appropriate number. For example, if you want to change the priority to *HIGHEST*, simply type **3** and press Enter.

```
Queries are run with the priority that is set in the corresponding
System z Workload Manager (WLM) environment. You can set the priority
of maintenance operations, such as loading data, analogously for
the 'DZA1DDF' subsystem:
* 1: DEFAULT
  2: SYSTEM
  3: HIGHEST
  4: HIGH
  5: NORMAL
  6: LOW
  7: LOWEST
  8: DISCRETIONARY

Enter the appropriate number to set the priority level.
To return to the main menu of the Configuration Console, just press <enter>.
(Default 0) >
```

Figure 4-4 Menu for setting the priority of the Accelerator maintenance tasks

The * signifies the current selection; the default selection is DEFAULT.

We provide Table 4-2 on page 93 so that administrators can appreciate the relationship of these priority settings to priorities that are associated with Accelerator queries. This is important to understand when having query activity concurrent with data maintenance task activity, which is one of our test scenarios described later (4.3.4, “Test scenario C: Mixed workload query and Accelerator table loads” on page 105).

Table 4-2 Relating the maintenance operation options to WLM Importance

Maintenance task priority	Comparable WLM Importance
SYSTEM	Importance 1
HIGHEST	Importance 1
HIGH	Importance 2
NORMAL	Importance 3
LOW	Importance 3
LOWEST	Importance 3
DISCRETIONARY	Discretionary

Just to reiterate, notice that SYSTEM and HIGHEST result in the same priority. Similarly, NORMAL, LOW, and LOWEST result in the same priority.

Selecting any option other than DEFAULT will result in all data maintenance tasks running at that same priority. The DEFAULT option is a little different because it favors table loads over statistics collection (similar to RUNSTATS in DB2) and data reorganization (similar to REORG in DB2) tasks. Reference Table 4-3.

Table 4-3 DEFAULT priorities of maintenance tasks

Maintenance task	Comparable WLM Importance
Table loading	3
Statistics and data reorganization	Discretionary

4.2.3 Accelerator prioritization summary table

We built the table in Figure 4-5 on page 94 as a useful summary of all the resource management controls that are available for prioritization of Accelerator work and tasks within a single DB2 subsystem.

Accelerator Workload / Task	Description	Resource Management Control	Default Priority in WLM Importance
Query (DDF)	Accelerator queries that originate via DDF	WLM Service Definition	Based on active WLM policy
Query (Local)	Accelerator queries that originate locally (ex. Batch, TSO)	WLM Service Definition (requires PTF PM88071)	If PM88071, then based on active WLM policy. Otherwise = 3
Table Loads	Accelerator table Loads & Removes	Accelerator Configuration Console	3
Incremental Update	Accelerator activity associated with Incremental Update (ex. Inserts)	not available today	3
HPSS Archive	Accelerator activity associated with HPSS archive (ex. Inserts)	not available today	3
Background Maintenance	Accelerator Maintenance (ex. Statistics collection, reorganization)	Accelerator Configuration Console	Discretionary

Figure 4-5 Accelerator resource management controls within a single DB2 subsystem

Note: A dynamic change to the WLM service class managing Accelerator queries or the DB2 Accelerator maintenance task priority is only applicable to newly arriving work, not to work already actively running.

4.3 Experiments: Prioritization within a DB2 subsystem

We conducted several experiments to evaluate the resource management controls within a single DB2 subsystem. We first provide some background on the test environment and measurement methodology, then discuss the actual experiments, which are categorized into test scenarios. We conclude with a table summarizing the resource management controls:

- ▶ Test environment and measurement preparation
- ▶ Test scenario A: Prioritizing one user (query) higher than the others
- ▶ Test scenario B: Query prioritization as concurrency increases
- ▶ Test scenario C: Mixed workload query and Accelerator table loads

4.3.1 Test environment and measurement preparation

Our tests were conducted using a zEnterprise Analytics System 9700 core infrastructure.

Table 4-4 on page 95 represents the key hardware and software that were used for our experiments.

Table 4-4 9700 test configuration for resource management experiments

Key hardware
zEC12 Model 2827-740 ▶ LPAR1: 2CPs + 2 zIIPs, 128 GB Memory ▶ LPAR2: 2CPs + 2 zIIPs, 128 GB Memory IBM storage subsystem DS8870 Model 2107-961
Key software
z/OS V1.13 DB2 10 DB2 Analytics Accelerator V3.1 PTF2 ▶ Including <i>PureData for Analytics</i> N2001-010 w/NPS® 702 IBM Tivoli® OMEGAMON® XE for DB2 Performance Expert V5.1.1
DB2 subsystems
DZA1 running on LPAR1 DBZZ running on LPAR2

Performance data collection

To obtain consistent and reliable performance measurements, we adopted a testing protocol to be followed on each measurement. Its main steps are listed and described in Table 4-5.

Table 4-5 Measurement data collection methodology

Step	Description
Stop/Start DB2	To get a consistent clean situation at every start of measurements.
Stop/Start Query Accelerator	To get a clean Query Accelerator status before each run. It also resets some of the Query Accelerator counters that keep averages.
Verify key DB2 traces are started	Verify that the following DB2 traces are started: ▶ DB2 Accounting: Classes (1, 2, 3, 7, 8) ▶ DB2 Statistics: Classes (1, 3, 4, 5, 6)
Prime buffer pools	If necessary, preload data in buffer pools.
Open data sets of the DB2 database objects of interest	Preload data in buffer pools. Only part of the data can be contained in real storage.
Set the SMF and IBM RMF™ interval to 1 minute	To obtain better system-level reporting granularity.
Resource management preparation: ▶ WLM ▶ Accelerator configuration console	Make any necessary changes to the WLM service definitions or Accelerator resource definitions.
Switch SMF	Switch to a new SMF data set before running the test. This step makes analysis easier.
Modify the statistics trace	MOD the statistics trace so that we start a clean new statistics interval that starts with our measurement.

Step	Description
Verify DB2 Analytics Accelerator status	Start and stop the Accelerator, depending on the scenario.
RUN test	Run the necessary workload for the given test scenario.
Set the SMF and RMF interval back to normal settings	In our case, set back to 30-minute intervals.
Switch SMF	Switch to a new SMF data set after running the test.
Extract DB2 and RMF records from SMF	Dump SMF 70:79, 100:102, among others.
Reporting	<ul style="list-style-type: none"> ► Create DB2, RMF, and other reports. ► Load DB2 accounting and statistics data into the OMPE Performance Database.
Analysis	Analyze the results.

To automate these operations, many of the commands can be executed from job control language (JCL). Example 4-1 shows how to start DB2 traces, prime the target database, and stop the DB2 Analytics Accelerator. This JCL would be used before a workload run without the DB2 Analytics Accelerator available.

Example 4-1 Setting up the DB2 and DB2 Analytics Accelerator environment with commands

```

/* -----
/* DESCRIPTION: SETUP DB2 ENV FOR Analytics Accelerator TEST
/* -----
//SETUPENV EXEC PGM=IKJEFT01,DYNAMNBR=20,TIME=60
//STEPLIB DD DISP=SHR,DSN=SYS1.DSN.V100.SDSNLOAD
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSTSIN DD *
    DSN S(DZA1)
    -STA TRACE(S) CLASS(1,3,4,5,6)
    -STA TRACE(A) CLASS(1,2,3,7,8)
    -DIS TRACE
    -ACCESS DATABASE(BAGOQ) SPACENAM(*) MODE(OPEN)
    -DIS ACCEL(*) DETAIL
//SYSIN DD DUMMY

```

Important: No special traces are required to collect DB2 Analytics Accelerator activity. DB2 instrumentation support (traces) for DB2 Analytics Accelerator does not require additional IFCIDs to be started. Data collection is implemented by adding new fields to the existing trace classes after application of the required software maintenance.

Workload management preparation

In order to prepare our system for query prioritization experiments, we took the following steps:

1. Modified our existing service classes serving DDF queries.
2. Modified our existing DDF classification rules to properly qualify the work into those service classes.

Service Class preparation

Based on existing System z DB2 business analytics best practices, our WLM service definition was already defined with three WLM service classes to manage and differentiate business users according to their individual WLM goals and importance to the business.

To best understand the value and usage of this new Accelerator query prioritization control, we added a fourth WLM service class, allowing us to evaluate the four different levels of Importance that Accelerator queries can be managed to. Figure 4-6 captures our service class definitions used for the test. In order to not confuse you, we highlighted in red the only part of the service class definition that is used for Accelerator query prioritization.

BUSRCRIT - Critical Importance business analytics users / applications				
Period	Importance	Duration	Goal Type	Goal
1	1	25,000	Resp time	90% < 1s
2	2	100,000	Resp time	80% < 10s
3	3		Velocity	40
BUSRHIGH - High Importance business analytics users / applications				
Period	Importance	Duration	Goal Type	Goal
1	2	25,000	Resp time	90% < 1s
2	3	1,000,000	Velocity	40
3	4		Velocity	20
BUSRMED - Medium Importance business analytics users / applications				
Period	Importance	Duration	Goal Type	Goal
1	3	1,000,000	Velocity	40
2	4		Velocity	10
BUSRLOW - Lower Importance business analytics users / applications				
Period	Importance	Duration	Goal Type	Goal
1	Discretionary		Discretionary	n/a

Figure 4-6 WLM service classes used for query priority experiments

The BUSRCRIT service class is a new service class that we added specifically to test query prioritization control. In practice, for our business analytics workloads, we previously did not classify any queries or transactions into a service class with an Importance 1. From our experience, most IBM System z clients also do not typically use Importance 1 for transactions or queries. On the other hand, we wanted to evaluate all the options provided with Accelerator query prioritization; therefore, we added it.

WLM DDF classification preparation

There are several options available to classify DDF queries. More recently, especially for query workloads, System z performance administrators are making more use of the set of “client information” qualifiers, which are useful for more granular-level classification. The client information qualifiers are set by the client information API:

AI	Client accounting string
PC	Client transaction name
SPM	Concatenation of client user ID and workstation name

For more information about the use of DB2 client information for classification and reporting, see Chapter 13 of *Co-locating Transactional and Data Warehouse workloads on System z*, SG24-7726.

For our experiments, we used the client's user ID (sometimes referred to as an *end user ID*) to qualify our DDF queries into the appropriate service classes. Figure 4-7 exhibits the specific DDF classification rules that we utilized for these experiments.

Modify Rules for the Subsystem Type				Row 32 to 40 of 40		
Command ==> _____				Scroll ==> CSR		
Subsystem Type . : DDF		Fold qualifier names? Y (Y or N)				
Description . . . Rules for DDF						
Action codes: A=After C=Copy M=Move I=Insert rule						
B=Before D=Delete row R=Repeat IS=Insert Sub-rule						
More ==>						
-----Qualifier-----						
Action		Type	Name	Start	-----Class-----	
					Service	Report
					DDFTRANS	
_____ 1	SI	DZA1	_____	DDFTRANS	DZA10THR	
_____ 2	SPM	BUSRC*	_____	BUSRCRIT	RBUSRCRI	
_____ 2	SPM	BUSRH*	_____	BUSRHIGH	RBUSRHI	
_____ 2	SPM	BUSRM*	_____	BUSRMED	RBUSRMED	
_____ 2	SPM	BUSRL*	_____	BUSRLOW	RBUSRLOW	

Figure 4-7 Classification rules for query prioritization experiments

To set and pass the client user ID on each query invocation, we utilized the DB2 supplied stored procedure, WLM_SET_CLIENT_INFO, within our command-line interface (CLI) query tool. Example 4-2 provides the syntax for calling it.

Example 4-2 Syntax for invoking the WLM_SET_CLIENT_INFO stored procedure

```
call sysproc.wlm_set_client_info (client_userid, client_wrkstation,
client_application, client_application_accounting)
```

Example 4-3 shows how we set this in our test scripts for client user ID, BUSRC001.

Example 4-3 Sample WLM_SET_CLIENT_INFO call in our test script

```
call wlm_set_client_info ('BUSRC001',NULL,'Q3A','COMPLEX')
```

Table 4-6 provides how the mapping is done between the values passed to WLM and DB2 accounting.

Table 4-6 WLM_SET_CLIENT_INFO stored procedure mapping

Stored procedure parameter	WLM classification qualifier	DB2 accounting data fields
CLIENT_USERID	SPM	QWHEUID - end user ID
CLIENT_WRKSTATION	SPM	QWHCEUW - end user workstation name
CLIENT_APPLICATION	PC	QWHCEUTX - end user transaction name

Stored procedure parameter	WLM classification qualifier	DB2 accounting data fields
CLIENT_APPLICATION_ACCOUNTING	AI	QMDAAINFO - accounting string

To verify this from the DB2 perspective, we displayed the thread via the **DIS THD DETAIL** command. See Figure 4-8 and notice that the CLIENT information passed, which we highlighted in **bold black**, and the resulting WLM service class, underlined and in **bold blue**.

```
-DZA1 DIS THREAD(*),DETAIL
CRE106I (THREAD ) DONE
DSNV401I -DZA1 DISPLAY THREAD REPORT FOLLOWS -
DSNV402I -DZA1 ACTIVE THREADS - 932
NAME      ST A   REQ ID          AUTHID   PLAN      ASID TOKEN
SERVER    AC *    6 Agent.exe    BMFT00LS DISTSERV 004F 56220
V437-WORKSTATION=DWH1, USERID=BUSRM001,
      APPLICATION NAME=Q3A
V441-ACCOUNTING=COMPLEX
V436-PGM=NULLID.SYSSN100, SEC=4, STMT=0, THREAD-INFO=BMFT00L:DWH1:
      BUSRM001:Q3A:*:*:*<9.12.44.44.2336.130615133727>
V442-CRTKN=9.12.44.44.2337.130615133728
V482-WLM-INFO=BUSRMED:1:3:110
V445-G90C2C2C.G921.130615133728=56220 ACCESSING DATA FOR
      ( 1)::9.12.44.44
V444-G90C2C2C.G921.130615133728=56220 ACCESSING DATA AT
      ( 2)DZA1STPR-::135.25.80.250..1400
V447--INDEX SESSID          A ST TIME
V448--( 1) 446:2337          W S2 1316609372802
      V448--( 2) 3519:1400          N R2 1316609372805

SERVER    AC *    6 Agent.exe    BMFT00LS DISTSERV 004F 56226
V437-WORKSTATION=DWH1, USERID=BUSRH001,
      APPLICATION NAME=Q3A
V441-ACCOUNTING=COMPLEX
V436-PGM=NULLID.SYSSN100, SEC=4, STMT=0, THREAD-INFO=BMFT00L:DWH1:
      BUSRH001:Q3A:*:*:*<9.12.44.44.2336.130615133727>
V442-CRTKN=9.12.44.44.2340.130615133731
V482-WLM-INFO=BUSRHIGH:1:2:110
V445-G90C2C2C.G924.130615133731=56226 ACCESSING DATA FOR
      ( 1)::9.12.44.44
V444-G90C2C2C.G924.130615133731=56226 ACCESSING DATA AT
      ( 2)DZA1STPR-::135.25.80.250..1400
V447--INDEX SESSID          A ST TIME
V448--( 1) 446:2340          W S2 1316609372802
      V448--( 2) 3521:1400          N R2 1316609372805
```

Figure 4-8 Display thread results utilizing WLM_SET_CLIENT_INFO

To verify proper z/OS WLM classification, we displayed the enclave within the WLM service class, BUSRHIGH. See Figure 4-9 and again notice the client information, this time passed to WLM, highlighted in bold.

```

RMF Enclave Classification Data

Details for enclave ENC00003 with token 00000354 007E6A79
Press Enter to return to the Report panel.

- CPU Time -      -zAAP Time--      -zIIP Time--
Total   0.016      Total   0.000      Total   0.000
Delta   0.010      Delta   0.000      Delta   0.000

State  ---- Using ---- ----- Delay ----- IDL  UNK
Samples CPU AAP IIP I/O  CPU AAP IIP I/O STO CAP QUE
      50   0.0 0.0 0.0 0.0   0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 100

Classification Attributes:

Subsystem Type: DDF      Owner: DZA1DIST      System: P59
Accounting Information . . :
    SQL09075NT           Agent.exe           BMFTOOL COMPLEX

Collection Name . . . . . :
Connection Type . . . . . : SERVER
Correlation Information . . : Agent.exe
LU Name . . . . . :
Netid . . . . . :
Package Name . . . . . :
Plan Name . . . . . : DISTSERV
Procedure Name . . . . . :
Process Name . . . . . : Q3A
Transaction Class/Job Class:
Transaction Name/Job Name :
Userid . . . . . : BMFTOOL
Scheduling Environment . . :
Priority . . . . . :
Subsystem Collection Name :
Subsystem Instance . . . . : DZA1
Subsystem Parameter . . . . :
BUSRH001 DWH1

```

Figure 4-9 RMF Monitor III display of end user BUSRH001's enclave

4.3.2 Test scenario A: Prioritizing one user (query) higher than the others

In this section, we evaluate the value of the different Importance levels when running a concurrent mix of queries. In order to fairly evaluate the priority control, we decided that all users competing for Accelerator resources should have the same resource demands. Therefore, unless specified, in all our tests, all users ran the same query load. Additionally, all user queries are started at essentially the same exact time.

Table 4-7 shows our test case matrix.

Table 4-7 Test scenario: Query prioritization test cases

Test case	Description	Value
QP_Usr1A	1 BUSRHigh user running Query Q3A stand-alone	This is the base test case, exhibiting the best possible query elapsed time as it is running stand-alone with no other work running.
QP_Usr5A	5 DDF users running Query Q3A 1 BUSRCrit user has WLM Imp 1 4 BUSRHigh users have WLM Imp 2	Exhibit the benefit of running at Imp 1 (the best available) versus other work running at Imp 2 (the next best).
QP_Usr5B	5 DDF users running Query Q3A 1 BUSRCrit user has WLM Imp 1 4 BUSRMed users have WLM Imp 3	Exhibit the benefit of running at Imp 1 (the best available) versus other work running at the Imp 3.
QP_Usr5C	5 DDF users running Query Q3A 1 BUSRHigh user has WLM Imp 1 4 BUSRLow users have WLM Discretionary	Exhibit the benefit of running at Imp 1(the best available) versus other work running at Discretionary (the lowest available).
QP_Usr5D	5 DDF users running Query Q3A 1 BUSRHigh user has WLM Imp 2 4 BUSRLow users have WLM Discretionary	Exhibit the benefit of running at Imp 2 (the second best available) versus other work running at Discretionary (the lowest available).

Workload

For this set of tests, Query Q3A was used. Query Q3A is a non-selective query that requires multiple joins with dimensions and aggregations across the entire fact table (billions of records aggregated, less than 10,000 rows returned). When running Q3A stand-alone, we notice that the average CPU utilization of the Accelerator worker nodes is 100%, hence it is a CPU-bound query. For the specific SQL, see Appendix A.1, “Query CPU bound” on page 300.

Query submission tool

For all of our experiments requiring queries arriving via DDF, we utilized a CLI-based query submission tool running on a Windows system in the same private network.

Test scenario A results

For each test case, we reduced the DB2 accounting data and loaded the data into OMPE Performance Database tables. Then, we queried those tables to obtain the DB2 Class 1 elapsed times for each end user, in each test case. See Figure 4-10 for the elapsed times for test case QPusr5A.

CLIENT_ENDUSER	WLM_SERVICE_CLASS	CLASS1_ELAPSED	ACCEL_NAME
BUSRC001	BUSRCRIT	351.338243	DZA1STPR
BUSRH001	BUSRHIGH	617.384687	DZA1STPR
BUSRH002	BUSRHIGH	617.833145	DZA1STPR
BUSRH003	BUSRHIGH	619.204962	DZA1STPR
BUSRH004	BUSRHIGH	619.755106	DZA1STPR

Figure 4-10 Test case QPusr5A DB2 Class 1 elapsed times by CLIENT_ENDUSER

We then took the DB2 Class 1 elapsed time results for all test cases just described (Table 4-7 on page 101) and plotted them in Figure 4-11.

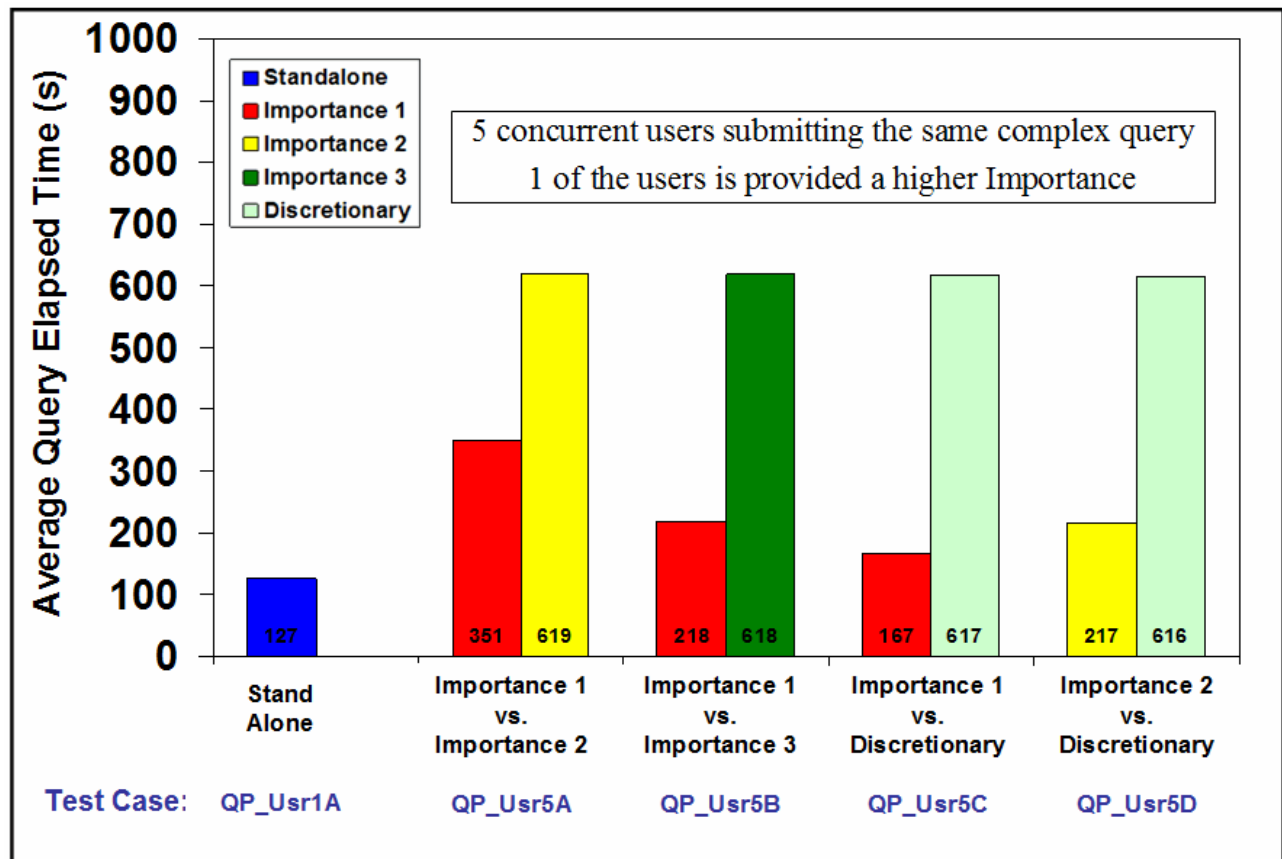


Figure 4-11 Query prioritization: One user is favored over the rest

Observations

We make the following key observations:

- ▶ In all tests, we can see that query prioritization provided value. The favored query always obtained a better response time.
- ▶ As the gap in Importance (between competing queries) widens, the larger percentage of resources is provided to the higher prioritized query. For example, notice the elapsed time of the Importance 1 query in test case QP_Usr5C is 167s, which is the best of all queries, closest to the standalone elapsed time.
- ▶ Comparing test case QP_Usr5B with QP_Usr5D, we notice that having two levels of difference in Importance has a similar effect on the competing work.
- ▶ Unfortunately, we did not have an explanation for why in all four test scenarios, the average elapsed time for the unfavored queries remained constant. We had anticipated that as the priority gap increased, not only would the favored query improve, but, consequently the unfavored queries would further degrade.

4.3.3 Test scenario B: Query prioritization as concurrency increases

Here, we evaluate the value of query prioritization as concurrency increases. Again, in order to fairly evaluate the priority control, we decided that all users competing for Accelerator resources should have the same resource demands. All user queries are started at

essentially the same exact time, except for the last test case, where we purposely delay the higher Importance query to understand the behavior of prioritization when Accelerator queuing occurs. Table 4-8 shows the test case matrix.

Table 4-8 Test scenario B: Query prioritization as concurrency increases

Test case	Description	Value
QP_Usr1e	1 BUSRHigh user running Query Q3A stand-alone	This is the base test case, exhibiting the best possible query elapsed time as it is running stand-alone with no other work running.
QP_Usr5E	5 DDF users running Query Q3A 1 BUSRHigh user has WLM Imp 2 4 BUSRLow users have WLM Discretionary	Exhibit the benefit of running at Imp 2 versus other work running at the Discretionary.
QP_Usr10E	10 DDF users running Query Q3A 1 BUSRHigh user has WLM Imp 2 9 BUSRLow users have WLM Discretionary	As concurrency increases, how does it affect prioritization?
QP_Usr15E	15 DDF users running Query Q3A 1 BUSRHigh user has WLM Imp 2 14 BUSRLow users have WLM Discretionary	As concurrency increases, how does it affect prioritization?
QP_Usr15E2	15 DDF users running Query Q3A 1 BUSRHigh user has WLM Imp 2 14 BUSRLow users have WLM Discretionary Note: From previous experiments, we have noticed that Accelerator internal queuing starts to occur when a concurrency of 11 complex Q3A queries is started. Hence, we added this test scenario, where we purposely delayed the 1 favored BUSRHigh users query until after all 14 Discretionary queries were submitted.	As concurrency increases to the point where queries are queuing within the Accelerator, how beneficial is the priority of a newly submitted query?

Again, we plot the average DB2 Class 1 elapsed time for the different users. See Figure 4-12 on page 104 for results.

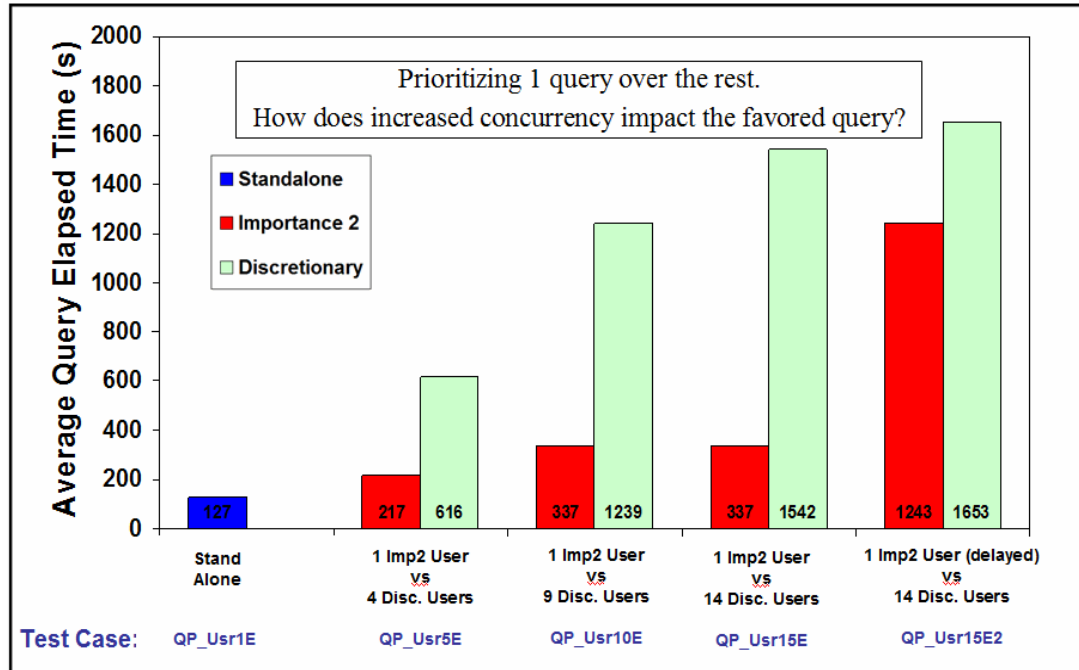


Figure 4-12 Test Scenario B: The impact of concurrency on query prioritization

Observations

As we increase concurrency from five users to 10 users, we notice that the value of prioritization is less because the elapsed time of our favored query went from 217s to 337s.

When we went to 15 concurrent users (test case QP_Usr15E), we unexpectedly noticed no change in the Importance 2 user's elapsed time. After analyzing the individual elapsed times for all 14 Discretionary users, we discovered a different pattern than what we previously witnessed with the 9 Discretionary users. In test case QP_Usr10E, all 9 Discretionary users had essentially the same elapsed time of 1240s. But in test case QP_Usr15, although 9 users had similar elapsed times of 1240s, the remaining five Discretionary users had similar elapsed times of 1540s. From this, we determined that there were some queuing type delays when we went beyond 10 Q3A queries on the Accelerator.

Based on this scenario, we had the idea to add test case QP_Usr15E2. We wanted to understand the value of query prioritization if a query was submitted into an Accelerator that was already saturated and queries were being queued. So in test case QP_Usr15E2, we purposely delayed the Importance 2 user's query submission until after the other 14 Discretionary users submitted theirs. As you can observe from the results that are charted in Figure 4-12, despite having a higher priority than all of the actively running Discretionary queries, the Importance 2 query was delayed (or queued) until one of the initial set of queries finished.

We have the following two conclusions from this set of tests:

- As the amount of concurrency increases, the aggregate processing that is associated with the less-favored queries (queries with lower priorities) will have greater impact on the favored queries.
- When the Accelerator is saturated, all additional queries will be queued regardless if they have higher priority than all of the actively running queries. We should note, though we did not test it, one exception to this are queries that the Accelerator determined would be short-running, regardless of priority.

4.3.4 Test scenario C: Mixed workload query and Accelerator table loads

In these tests our objective was to examine the potential usefulness of modifying the data maintenance prioritization. Our test scenario reflects perhaps a client use case of Accelerator loads taking precedence over concurrent query workload. For example, all Accelerator loads need to complete within a batch window. We ran a concurrent Accelerator load and query workload and then varied the priorities of each.

Workloads

- **Query:** Similar to the QP experiments above, we utilized query Q3A, though in these tests we always used five concurrent DDF users. Query Q3A is a non-selective query requiring multiple joins with dimensions and aggregations across the entire fact table (billions of records aggregated, less than 10,000 rows returned).
- **Accelerator load:** This was an Accelerator load of a 200 GB table, submitted via a batch job, similar to what was described in Example 3-3 on page 81.

Table 4-9 explains the various test cases that we used to evaluate mixed workloads.

Table 4-9 Test scenario C: Mixed workload: Query versus Accelerator table loads

Test case	Description	Value
QP_LoadQA	Load a 200 GB table into the Accelerator via a batch job, similar to Example 4-3 on page 98 JCL to remove, add, load a table.	This is the base test case, exhibiting the best possible Accelerator load elapsed time when run stand-alone.
QP_LoadQB	Five DDF users running Query Q3A, all prioritized with an Importance of 2.	This is the base test case, exhibiting the best possible average response time for the query workload when run stand-alone.
QP_LoadQC	Run the following 2 workloads at the same time: <ul style="list-style-type: none">► Accelerator Load at comparable WLM Importance of 3. (No changes to the data maintenance priorities, hence DEFAULT).► Five DDF Queries at Importance 2.	A baseline of performance for the mixed workload.
QP_LoadQD	Run the following two workloads at the same time: <ul style="list-style-type: none">► Accelerator Load at comparable WLM Importance of 1. (Modified the data maintenance priorities setting to HIGHEST).► Five DDF queries at Importance Discretionary.	How does increasing the Importance of the Accelerator load workload and decreasing the Importance of the query workload affect the performance of both?

See Figure 4-13 on page 106 for the results of these experiments, and then review the ensuing observations.

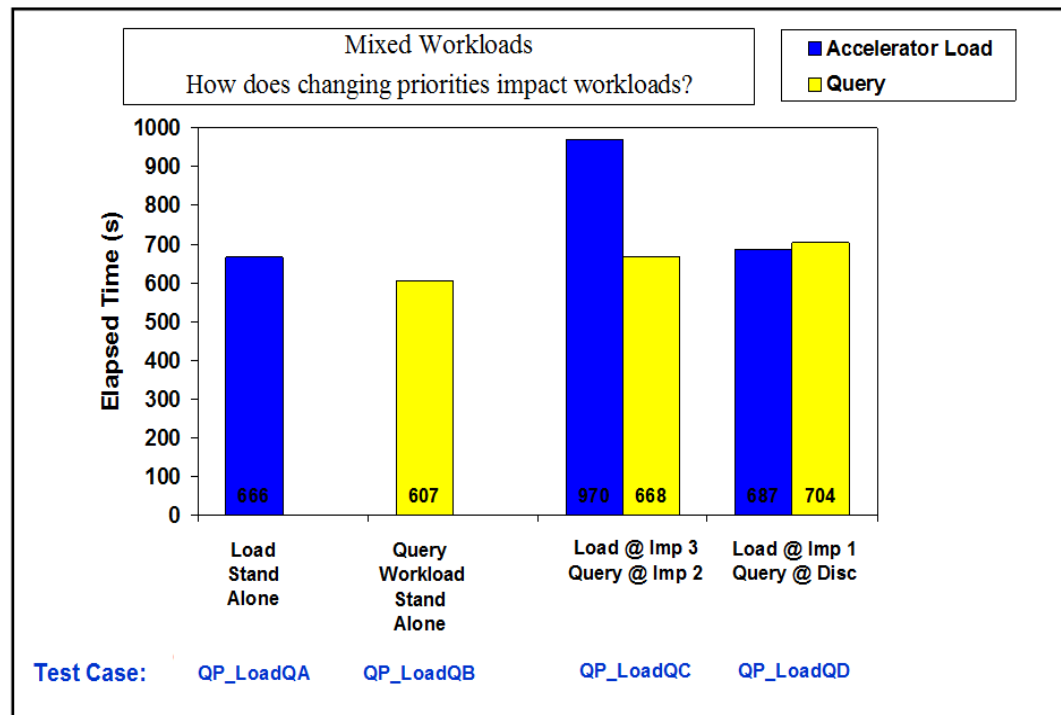


Figure 4-13 Test Scenario C: Mixed workload results

Observations and summary

The best way to analyze these results is to compare each of the workload's performance back to the stand-alone performance of the same workload. For instance, in test case QP_LoadQC we notice that the same 200 GB Accelerator load that is run stand-alone took 666 sec., and now with concurrent queries, takes 970 sec.

- ▶ As anticipated, we observed changes in performance that is associated with the changes made with priorities when load was favored over queries.
- ▶ The query workload that we utilized drove the worker node to an average utilization of 100%, but only drove the coordinator utilization to an average of 5% busy. Meanwhile, we know from monitoring the Accelerator loads, they often drive the coordinator nodes to 100% utilization. Hence, for query workloads that have more demands on the coordinator nodes, we can expect to see even more contention between these workloads, which would result in more gaps in performance that is associated with priority differentiation between queries and concurrent loads.
- ▶ Consider that the workloads used for this test are relatively small as compared to say, a client's production Accelerator refresh, which would likely include 100s to 1000s tables, and production query workload.

4.4 Allocating resources with workload isolation for a shared Accelerator

In this section, we describe workload isolation and procedures for setting it. In section 4.5, "Workload isolation experiments" on page 109, we show some experiments exploiting it with two DB2 subsystems concurrently accessing the Accelerator.

4.4.1 Workload isolation

If an Accelerator is shared by multiple DB2 subsystems, you can distribute the available processing resources of the Accelerator among the subsystems. This allows you to adapt the processing resources to the size and priority of the query workloads generated by each subsystem. The authentication process (pairing) must first have been completed for the DB2 subsystems sharing the Accelerator.

Default assignment

By default, the processing resources of an Accelerator are distributed equally. Each connected subsystem gets the same share. This prevents a resource drain, that is, a situation in which a single DB2 subsystem occupies all available resources and blocks resource access for other subsystems. Resources are allocated as a percentage of all available resources, which always add up to 100%. If you allocate more than 100% in sum, the quotas are reduced proportionally. For example, if two DB2 subsystems are attached to one Accelerator, and you assign 80% to subsystem A and 30% to subsystem B, the result will be that 73% (80/110) of the resources are reserved for subsystem A, and the remaining 27% for subsystem B. You assign Accelerator processing resources on the IBM DB2 Analytics Accelerator Configuration Console.

If you are familiar with the WLM Resource Group MIN control¹, the Accelerator workload isolation control works in a similar manner. It provides the minimum relative percentage of resource that is specified, assuming there is demand.

Workload isolation and priority

The allocated Accelerator resources set the boundaries for the usage of the single DB2 subsystem level priorities discussed previously (Figure 4-5 on page 94). If, for example, two subsystems share an Accelerator, and subsystem A is allowed to use 90% of the resources, while subsystem B gets only 10%, priority settings for subsystem B will be considered only within the 10% share. This signifies that a query with a low priority from subsystem A might still be processed faster than a query with a higher priority from subsystem B. The actual outcome, however, depends on the workload that is assigned to the shares.

4.4.2 Procedure for setting the workload isolation resource limits

The setting of resource limits is conducted via the DB2 Analytics Accelerator Configuration Console, as follows:

1. You must first log on to the Accelerator Configuration Console via steps 1 - 5, described earlier (4.2.2, “Accelerator data maintenance task prioritization” on page 90). You are then presented with the IBM DB2 Analytics Accelerator Configuration Console (Figure 4-14 on page 108). Type **3** and press Enter to go to the “Run Accelerator Functions” menu.

¹ For details about WLM Resource Groups, see this site:
<http://publib.boulder.ibm.com/infocenter/zos/v1r12/index.jsp?topic=%2Fcom.ibm.zos.v12.ieaw100%2Frgr.htm>

```

Enter password (in TSO, use PF3 to hide input):
Licensed Materials - Property of IBM
5697-DAA
(C) Copyright IBM Corp. 2009, 2013.
US Government Users Restricted Rights -
Use, duplication or disclosure restricted by GSA ADP Schedule
Contract with IBM Corporation

*****
* Welcome to the IBM DB2 Analytics Accelerator Configuration Console
*****

You have the following options:
(1) - Change the Configuration Console Password
(2) - (Menu) Run Netezza Commands
(3) - (Menu) Run Accelerator Functions
(4) - (Menu) Manage Incremental Updates
-----
(x) - Exit the Configuration Console

```

Figure 4-14 DB2 Analytics Accelerator Configuration Console main menu

2. You are now presented with seven options (Figure 4-15). Type **3** and press Enter to continue.

```

main -> 3
-----
You have the following options:

(0) - Go back one level
(1) - Obtain pairing code, IP address, and port
(2) - List paired DB2 subsystems
(3) - Set resource limits for DB2 subsystems
(4) - Clear query history
(5) - Specify the priority of maintenance tasks
(6) - Set the DB2 subsystem for time synchronization

(Default 0) >

```

Figure 4-15 DB2 Analytics Accelerator: Run Accelerator Functions menu

3. You are now provided the current resource limit settings for the DB2 subsystems known to this Accelerator. Just to reiterate, the “relative percentage” is simply the minimum resource specification for the given DB2 subsystem, divided by the sum of the minimum resource specifications and is computed for you. You are now prompted to select a database from the list of DB2 subsystems. See Figure 4-16 on page 109. Type the number next to the subsystem of interest, and press Enter.

```
(Default 0) > 3
```

Index	Location Name	Minimum Resources	Relative Percentage
1	DBZZDDF	100	50.00
2	DZA1DDF	100	50.00

Select database system by location name or index (use empty string or 0 to go back):

Figure 4-16 Minimum resources menu

4. Figure 4-17 exhibits an example of changing the minimum resource specification for DZA1DDF to 50. We then press Enter and the resulting changes are displayed.

```
Minimum resource allocation for system with location name 'DZA1DDF' (1-100,
0 or empty to abort): 50
Successfully updated minimum resource allocation for system with location
name 'DZA1DDF' to 50.
```

Press <return>

Index	Location Name	Minimum Resources	Relative Percentage
1	DBZZDDF	100	66.66
2	DZA1DDF	50	33.33

Select database system by location name or index (use empty string or 0 to go back):

Figure 4-17 Example of changing DZA1DDF's minimum resource allocation

4.5 Workload isolation experiments

This section details the experiments that are related to the workload isolation function described at 4.4, “Allocating resources with workload isolation for a shared Accelerator” on page 106.

In these tests, our objectives were to understand the following:

- ▶ How well does workload isolation’s management of the shared Accelerator resource match the current desired distribution, when there is contention for resources?
- ▶ How responsive is workload isolation’s management when a change is made to the desired distribution?
- ▶ When there is no contention for resources, will workload isolation allow a given DB2 subsystem to go beyond its minimum specification?

4.5.1 Test environment and data collection procedures

The test environment and data collection procedures are the same as what was described earlier (4.3.1, “Test environment and measurement preparation” on page 94). However, in this test, the data collection procedures were run on both logical partitions (LPARs).

Workload

For this set of tests, CPU-bound queries were utilized on both DB2 subsystems, specifically so that they could individually consume the Accelerator resources themselves. On DZA1, we utilized two iterations of query Q1B, and on DBZZ we utilized six iterations of query Q8B. Reference Appendix A.3, “Other CPU bound queries” on page 304 for the SQL used.

Query submission tool

For all of our experiments requiring queries arriving via DDF, we utilized a CLI-based query submission tool running on a Windows system in the same private network.

Test scenario

Table 4-10 describes the steps that we took to test out the workload isolation control.

Table 4-10 Workload isolation test scenario sequence

Time	Action or observation	Min. resource specification	
		DZA1	DBZZ
08:26	On DZA1, we submitted a large batch workload of complex queries, each query capable of driving the Accelerator worker nodes to 100%. Note: Using JES2 initiators, we only allowed two queries to run concurrently because we purposely wanted to avoid any potential Accelerator queuing for this experiment.	100	100
08:44	On DBZZ, we submitted a large batch workload of complex queries, each query capable of driving the Accelerator worker nodes to 100%. Note: Using JES2 initiators, we only allowed six queries to run concurrently because we purposely wanted to avoid any potential Accelerator queuing for this experiment.	100	100
09:03	On DZA1, we observed the last of the large batch of complex queries finished. Essentially this allowed for DBZZ to consume all the Accelerator resources.	100	100
09:15	Again, on DZA1 we submitted a large batch workload of complex queries, each query capable of driving the Accelerator worker nodes to 100%. As before, we only allowed two queries to run concurrently.	100	100
09:45	We logged into the Accelerator configuration console and modified the workload isolation minimum resource specifications in a manner to provide DZA1 minimally 80% of the total Accelerator resources, should it demand, and DBZZ 20%.	80	20
10:13	We logged into the Accelerator configuration console and modified the workload isolation minimum resource specifications in a manner opposite of what we previously did. So in this case, we provided DBZZ minimally 80% of the total Accelerator resources, should it demand, and DZA1 20%.	20	80
11:03	On DBZZ, we observed the last of the large batch of complex queries finished. Essentially, this allowed for DZA1 to consume all the Accelerator resources.	20	80
11:23	On DZA1, we observed the last of the large batch of complex queries finished.	20	80

As in earlier experiments, we loaded our DB2 accounting and statistics trace data in the OMPE Performance Database tables. See 9.3.4, “The Accelerator support in the OMPE Performance Database” on page 249.

We then plotted the ACCEL_CPU_TIME metric from the DB2PM_STAT_ACCEL PDB table, by DB2 subsystem, by time (minutes). See Figure 4-18 for the annotated plot for our workload isolation test scenario.

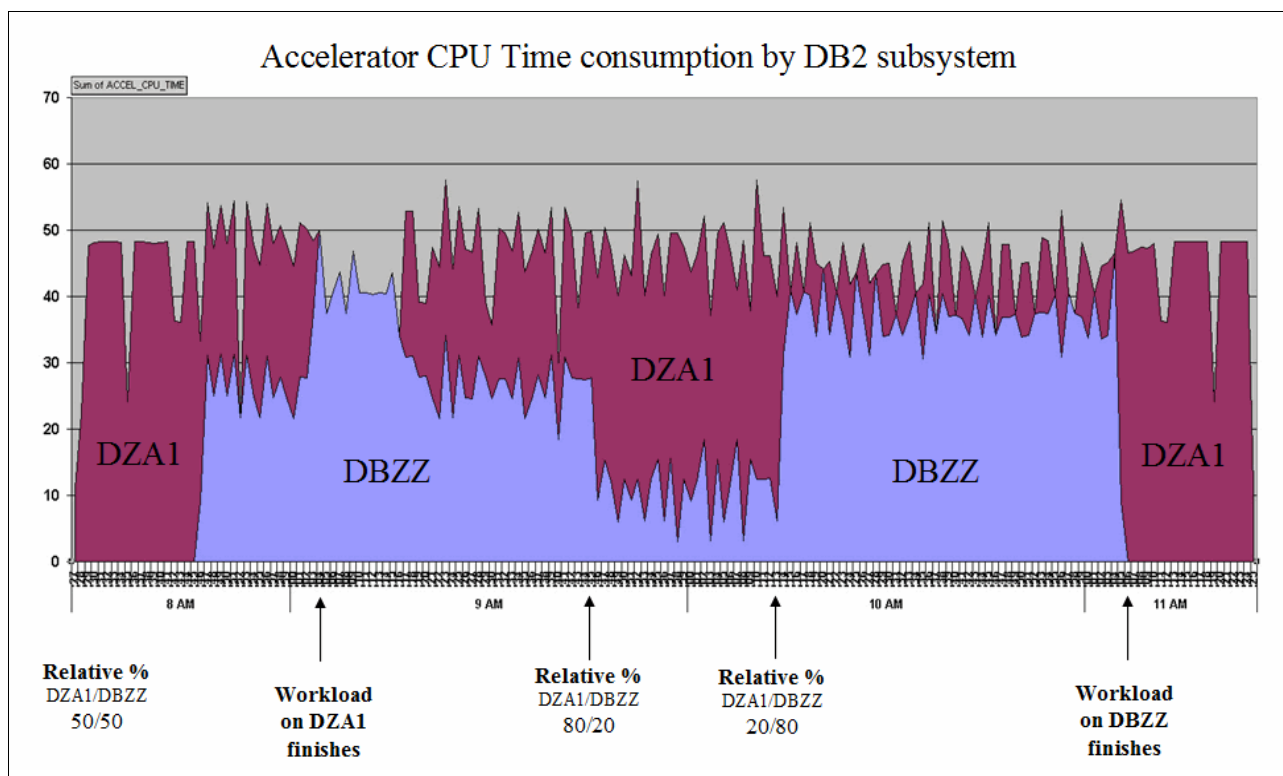


Figure 4-18 Workload isolation test scenario plot of Accelerator CPU time consumption

4.5.2 Key observations

First, during the complete test scenario time, we were intermittently monitoring the Accelerator worker node utilization via the **DISPLAY ACCELERATOR** command, and it was always 100%.

After introducing the query workload on DB2 subsystem DBZZ at 08:44, within only a couple of minutes, workload isolation reduced the Accelerator CPU time consumption rate from subsystem DZA1, now providing roughly 50% of the total Accelerator CPU time to DB2 subsystem DBZZ.

We witnessed that when DZA1’s demand for resources seized at 09:03, DBZZ was able to now consume the complete Accelerator resources.

After modifying the minimum resource specifications (hence relative percentages) at 09:45 to 80/20, we noticed within only a few minutes, the DBZZ consumption rate was reduced down to roughly the 20% anticipated. Then, reversing the relative percentages at 10:13 yielded the anticipated opposite effect.

When the DBZZ query workload finished at 11:03, you can observe that the DZA1 consumption rate again increases.

4.6 Accelerator resource management checklist

In this section, we provide a summary checklist of all Accelerator resource management considerations after adding an Accelerator to a System z infrastructure.

4.6.1 Resource management checklist

When a decision is made to add an Accelerator to a System z configuration, it is suggested that the performance administrators again collaborate with the BI administrator, lead DBA, and business users to reassess the end-to-end Analytics resource management/monitoring policies that are currently in place and determine any modifications. The Accelerator checklist in Table 4-11 can be utilized as part of or after that review. The table is our attempt at a comprehensive checklist for implementing Accelerator resource management. In some instances, we provide our own sample implementation for reference.

The steps listed are not in a strict order (other than step 1), though they are in what we believe is a natural sequence. The *Version* column reflects the initial Accelerator version the step should be considered. The *Control* column represents the interface to be utilized for implementation.

- “WLM” represents the interface for defining WLM controls. This can either be the traditional ISPF WLM menus or the more recent IBM z/OS Management Facility described at the following website:

<http://www.ibm.com/systems/z/os/zos/zosmf>

- “DACC” represents the DB2 Accelerator console configuration described previously.

Table 4-11 Accelerator resource management checklist

	Item	Version	Control	Considerations
1	Defining the WLM Application Environment for Accelerator DB2 stored procedures	V21+	WLM	This is a mandatory Accelerator installation step and is described well in the <i>IBM DB2 Analytics Accelerator for z/OS Version 3.1.0 Installation Guide</i> , SH12-6983, and <i>Optimizing DB2 Queries with IBM DB2 Analytics Accelerator for z/OS</i> , SG24-8005. One additional consideration is the use of the MNSPAS parameter within the stored procedure JCL to set the minimum number of stored procedure address spaces to be started all the time, reducing the start-up time associated with every Accelerator load. Reference “Defining the application environment” on page 114.
2	WLM Resource Management for Accelerator stored procedures	V21+	WLM	<ul style="list-style-type: none">► Need to create new and review existing WLM classification rules, service classes, and report classes for:<ul style="list-style-type: none">a. The Accelerator DB2 stored procedures.b. The associated WLM-managed address spaces in which they run.► Calls to these stored procedures can either be initiated via Accelerator Studio, or locally on z/OS. Reference our residency implementation and potential pitfalls (“WLM for DB2 Accelerator stored procedures” on page 114).► For optimal “Load” performance, see the APARs listed in 3.3, “Load performance considerations” on page 66.

	Item	Version	Control	Considerations
3	WLM resource management for incremental update	V31+	WLM	If exploiting incremental update, one of the necessary components requiring resource management is the IBM InfoSphere® CDC agent running on z/OS. We suggest adjusting WLM STC classification rules, and adding a report class. Incremental update is typically exploited to minimize latency between DB2 and the Accelerator. Assuming this is your case, the CDC agent should be classified to a WLM service class with aggressive goals, similar to that of the DB2 DBM1 address space. Reference our residency implementation ("WLM for incremental update: Service and report classes for CDC" on page 118)
4	Allocating resources for shared Accelerators	V31+	DACC	<ul style="list-style-type: none"> ▶ It is vital to ensure Accelerator resources are allocated properly among the potential various DB2 subsystems sharing it. The performance administrators should get with the business users to determine the appropriate allocations. Recall that the allocations set for each DB2 subsystem are minimums. If capacity is available, any of the attached DB2 subsystems can consume all of it. ▶ The Accelerator priority settings within a given DB2 subsystem (such as query and maintenance activities covered in steps 5 and 6) will be applied to the allocated portion of the Accelerator determined in this Step 4. <p>Reference 4.4, "Allocating resources with workload isolation for a shared Accelerator" on page 106 for details about using this control.</p>
5	Review and modify WLM definitions for query workloads	V31+	WLM	<ul style="list-style-type: none"> ▶ Since each query's WLM service class "Importance" is now being utilized for prioritization on the Accelerator, it is vital that the current WLM definitions are reviewed and refined to ensure consistency with various priorities of the users and applications originating the queries. Reference our suggestions and residency implementation ("Review and modify WLM definitions for query workloads" on page 118). ▶ At the time of this writing, DB2 just released PTF PM88071, which provides Accelerator query prioritization for locally submitted (such as batch, TSO) queries. Without this PTF, locally submitted accelerated queries will be prioritized with an Importance of 3, assuming PTF3 is installed. ▶ Consider multiple WLM policies. If at different points in time (day/week) the various priorities change, consider implementing an alternate WLM policy, which could be deployed at those periods of time. For example, if Accelerator loads are done during a refresh window, consider reducing the Importance of queries and increasing the prioritization of loads via DACC (see Step 6).
6	Setting prioritization of Accelerator maintenance tasks	V31+	DACC	<ul style="list-style-type: none"> ▶ Determine prioritization between maintenance tasks and concurrent accelerated queries and make any necessary adjustments. ▶ As mentioned in Step 5, consider if an alternate WLM policy is required for certain periods of time. If so, consider whether procedures also need to be put in place to adjust the maintenance task prioritization during this same time frame. <p>Reference 4.2.2, "Accelerator data maintenance task prioritization" on page 90) for how to utilize this control.</p>

	Item	Version	Control	Considerations
7	Continue to monitor resource consumption and refine resource management	All	WLM DACC	<ul style="list-style-type: none"> ▶ Adding an Accelerator to a System z configuration is likely to result in significant changes to native System z resource consumption, as well as a new resource (the Accelerator) to monitor/manage. As the Accelerator continues to mature and BA workloads grow, we anticipate additional rapid changes. Performance administrators need to monitor the changes in resource consumption via WLM's service and report classes along with their other resource monitoring and capacity planning tools, and consider resource management refinements if necessary. Reference Chapter 9, "Monitoring DB2 Analytics Accelerator environments" on page 207. ▶ One area of investigation, not previously mentioned, is re-evaluation of any current WLM resource groups that might be in place for the traditional native data warehouse query workloads.

4.6.2 Details referenced in resource management checklist

This section contains the residency implementation details and additional information referenced from Table 4-11 on page 112.

Defining the application environment

The general IBM recommendations suggest:

- ▶ To avoid conflicts with environment variables that are set for stored procedures of other applications, use a dedicated WLM Application Environment for the IBM DB2 Analytics Accelerator for z/OS stored procedures.
- ▶ The DB2 supplied stored procedures SYSPROC.ADMIN_INFO_SYSPARM, SYSPROC.DSNUTILU, and SYSPROC.ADMIN_COMMAND_DB2 must use a WLM environment that is separate from the one used by the IBM DB2 Analytics Accelerator for z/OS stored procedures. Verify that this and a few other requirements are met by following these steps:
 - Verify that SYSPROC.ADMIN_INFO_SYSPARM, SYSPROC.DSNUTILU, and SYSPROC.ADMIN_COMMAND_DB2 each use a separate WLM environment.
 - Make sure that NUMTCB is set to 1 (NUMTCB=1) for the SYSPROC.ADMIN_INFO_SYSPARM and SYSPROC.DSNUTILU WLM environments.
 - Consider using the **MNSPAS** parameter to keep a minimum number of WLM-managed address spaces running all the time to minimize start-up delays every time that you are running Accelerator loads. However, the consequences are that these address spaces will always consume some storage resources. See Chapter 3, "Data latency management" on page 55.

WLM for DB2 Accelerator stored procedures

It is important to define WLM performance goals so that the WLM service class for the Accelerator stored procedures will get appropriate service and can provide a sufficient number of additional WLM address spaces in a timely manner. Under favorable conditions, starting an address space takes two seconds. Under good conditions, this action takes about 10 seconds. However, if the workload is quite high, the time needed to start an address space can be considerably longer.

WLM service class for Accelerator stored procedures

For the residency, we defined a *new* service class, entitled IDAA_SPS, for WLM to manage our Accelerator stored procedures. We chose a very aggressive IBM Velocity™ goal of 90 to ensure that WLM provided ample resources. See Figure 4-19.

Service Class Name	: IDAA_SPS
Description	SC for Accel stored procedures
Workload Name	DB2WKLD (name or ?)
Base Resource Group	_____ (name or ?)
Cpu Critical	NO (YES or NO)
Specify BASE GOAL information. Action Codes: I=Insert new period, E=Edit period, D=Delete period.	
-- Period -- ----- Goal -----	
Action #	Duration Imp. Description
—	1 ———— 2 ————— Execution velocity of 90

Figure 4-19 Accelerator stored procedures WLM service class

As opposed to using an existing service class, we specifically chose to create a new service class, providing us an easy means to dynamically change the goals of this work via a WLM policy change. An example might be overriding the goal to make it less Important, due to other concurrent native z/OS work that is a higher priority. We thought this might be useful in some client environments, for the same reason. For example, at different times of the day or week potentially goals of the concurrent work might change.

The Accelerator stored procedures are called from either a remote graphical user interface, such as the Accelerator Studio or locally (such as batch or TSO). We created WLM classification definitions to handle both, which are described next.

WLM classification for Accelerator loads run locally (such as batch TSO)

All local calls of DB2 stored procedures inherit the performance attributes of the calling address space and are considered continuations of existing address space transactions (also known as a *dependent enclave*). Hence, when Accelerator loads are initiated locally, ensure that they are classified into the appropriate service class and report class. In our residency experiments, all our local Accelerator loads had job names that started with “LOD”. Within the WLM JES subsystem, we utilized the Transaction Name (TN) qualifier. Figure 4-20 on page 116 provides the details.

Modify Rules for the Subsystem Type					Row 1 to 11 of 30	
Command ==> _____					Scroll ==> CSR	
Subsystem Type . : JES		Fold qualifier names?			Y (Y or N)	
Description . . . Rules for Batch Jobs						
Action codes:		A=After	C=Copy	M=Move	I=Insert rule	
		B=Before	D=Delete row	R=Repeat	IS=Insert Sub-rule	
		More ==>				
		-----Qualifier-----			-----Class-----	
Action	Type	Name	Start	Service	Report	
				DEFAULTS: BATCHLOW		
_____	1	TN	LOD*	IDAA_SPS	ACCLDNAT	

Figure 4-20 Classifying batch Accelerator load jobs

We utilized a report class of ACCLDNAT so that we can monitor the resource consumption of our locally submitted Accelerator loads.

WLM classification for Accelerator loads initiated remotely (DDF)

DB2 stored procedures that are called remotely (DDF) are managed under a WLM independent enclave, and hence will be classified via the WLM DDF subsystem. Figure 4-21 exhibits how we classified all Accelerator stored procedures.

Subsystem Type . : DDF					
Fold qualifier names? Y (Y or N)					
Description . . . Rules for DDF					
Action codes:		A=After	C=Copy	M=Move	I=Insert rule
		B=Before	D=Delete row	R=Repeat	IS=Insert Sub-rule
		More ==>			
		-----Qualifier-----		-----Class-----	
Action	Type	Name	Start	Service	Report
				DEFAULTS: DDFTRANS	
_____	1	PR	ACCEL_L*	IDAA_SPS	RACCEL_L
_____	1	PR	ACCEL*	IDAA_SPS	RACCEL

Figure 4-21 Classifying remote (DDF) calls of the Accelerator stored procedures

The name of the Accelerator load stored procedure is ACCEL_LOAD_TABLES. Using the WLM qualifier PR (stored procedure), we classified all Accelerator loads (using “ACCEL_L”) into service class IDAA_SPS and report class RACCEL_L. The report class is valuable to specifically capture just the resource consumption of remote called Accelerator load activity. We also classified all the other Accelerator stored procedures into the same service class.

The additional advantage of having a unique service class or report class for Accelerator load activity is the ability to understand resource delays associated specifically with that work.

WLM for DB2 Accelerator stored procedures: WLM-related timeouts

When inadequate resources were provided to Accelerator-load stored procedures, we saw several WLM-related failures during a load data into DB2 Analytics Accelerator process.

Example 4-4 shows the messages as reported in the stored procedure WLM address space procedure.

Example 4-4 Error in WLM address space: -471

```
[02:21:13] Call to SYSPROC.DSNUTILU returned SQLCODE=-471 with reasoncode 00E79002
[02:21:13] SYSPROC.DSNUTILU might need to be started or its WLM Environment to be resumed.Call to
SYSPROC.DSNUTILU is re-tried in 60
seconds
[02:25:33] Call to SYSPROC.DSNUTILU returned SQLCODE=-471 with reasoncode 00E79002
[02:25:33] SYSPROC.DSNUTILU might need to be started or its WLM Environment to be resumed.Call to
SYSPROC.DSNUTILU is re-tried in 60
seconds
```

Example 4-5 shows the information provided by the DB2 Analytics Accelerator Data Studio graphical user interface (GUI).

Example 4-5 Failure as reported in the DB2 Analytics Accelerator GUI

Stored procedure call "ACCEL_LOAD_TABLES"

Parameters:

Accelerator name: IDAATF3

Lock Mode: None

AQT10200I - The CALL operation failed. Error information: "DSNT408I SQLCODE = -471, ERROR: INVOCATION OF FUNCTION OR PROCEDURE SYSPROC.DSNUTILU FAILED DUE TO REASON 00E79002 DSNT418I SQLSTATE = 55023 SQLSTATE RETURN CODE DSNT415I SQLERRP = DSNX9WCA 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 ". The unsuccessful operation was initiated by the "SYSPROC.DSNUTILU('AQT002C00010004', 'NO', 'TEMPLATE UD PATH /tmp/AQT.DA12.AQT002C00010004 FILEDATA BINARY RECFM VB LRECL 32756 UNLOAD TABLESPACE "BAGOQ"."TSLARGE" PART 4 FROM TABLE "GOSLDW"."SALES_FACT" HEADER CONST X'0101' ("ORDER_DAY_KEY" INT,"PRODUCT_KEY" INT,"STAFF_KEY" INT,"RETAILER_SITE_KEY" INT,"ORDER_METHOD_KEY" INT,"SALES_ORDER_KEY" INT,"SHIP_DAY_KEY" INT,"CLOSE_DAY_KEY" INT,"RETAILER_KEY" INT,"QUANTITY" INT,"UNIT_COST" DEC PACKED(19,2),"UNIT_PRICE" DEC PACKED(19,2),"UNIT_SALE_PRICE" DEC PACKED(19,2),"GROSS_MARGIN" DOUBLE,"SALE_TOTAL" DEC PACKED(19,2),"GROSS_PROFIT" DEC PACKED(19,2)) UNLDDN UD NOPAD FLOAT IEEE SHRLEVEL CHANGE ISOLATION CS SKIP LOCKED DATA', :hRetCode)" statement.

Explanation:

This error occurs when an unexpected error from an SQL statement or API call, such as DSNRLI in DB2 for z/OS, is encountered. Details about the error are part of the message, for example, the SQL code and the SQL message.

User actions:

Look up the reported SQL code in the documentation of your database management system and try correct the error.

The issue was that DB2 received an SQL CALL statement for a stored procedure or an SQL statement containing an invocation of a user-defined function. The statement was not accepted because the procedure could not be scheduled before the installation-defined time limit expired. This can happen for any of the following reasons:

- ▶ The DB2 **STOP PROCEDURE(name)** or **STOP FUNCTION SPECIFIC** command was in effect. When this command is in effect, a user-written routine cannot be scheduled until a DB2 **START PROCEDURE** or **START FUNCTION SPECIFIC** command is issued.
- ▶ The dispatching priority that is assigned by WLM to the caller of the user-written routine was low, which resulted in WLM not assigning the request to a task control block (TCB) in a WLM-established stored procedure address space before the installation-defined time limit expired.
- ▶ The WLM application environment is quiesced, so WLM will not assign the request to a WLM-established stored procedure address space.

WLM for incremental update: Service and report classes for CDC

Incremental update requires running the InfoSphere Change Data Capture agent on z/OS. This agent has a z/OS started task, with a default name of CHCPROC. We classified CHCPROC into the same aggressive service class serving our DB2 started tasks, and we added a report class so that we could monitor its resource consumption. See Figure 4-22.

Subsystem Type . : STC		Fold qualifier names? Y (Y or N)	
Description . . . Rules for Started Tasks			
Action codes:		A=After	C=Copy
		B=Before	D=Delete row
		M=Move	I=Insert rule
		R=Repeat	IS=Insert Sub-rule
		More ==>	
		-----Qualifier-----	
		-----Class-----	
Action	Type	Name	Start
DEFAULTS: STCLOW			
_____	1 TN	%%%DBM1	_____
_____	1 TN	%%%DIST	_____
_____	1 TN	%%%MSTR	_____
_____	1 TN	%%%IRLM	_____
_____	1 TN	DZA1WLM*	_____
_____	1 TN	DB2WLM*	_____
_____	1 TN	CHCPROC	_____
		Service	Report
		DB2HIGH	RDB2DBM1
		DB2HIGH	RDB2DIST
		DB2HIGH	RDB2MSTR
		SYSSTC	RDB2IRLM
		DB2HIGH	RDB2WLM
		DB2HIGH	RDB2WLM
		DB2HIGH	RDCAGNT

Figure 4-22 Classification of the InfoSphere CDC agent

Review and modify WLM definitions for query workloads

We provide the following suggestions:

- In order to effectively utilize Accelerator query prioritization, be sure to use multiple service classes, differentiating the WLM Importance of the first period.
- We continue to suggest using multiple periods per service class, for period-aging of the native DB2 queries/transactions.
- We continue to suggest using separate service classes to manage locally submitted query work versus the query work arriving via DDF.
- Recall that accelerated queries returning large answer sets can still potentially consume significant native System z processing resources, and the associated native tasks will move through any WLM service class periods as normal.

Figure 4-23 contains the service class definitions serving DDF queries, which we utilized for our residency and intended as a sample that clients could base their definitions on.

BACrit - Critical Importance business analytics users / applications				
Per	Imp	Duration	Goal Type	Goal
1	1	25,000	Resp time	90% < 1s
2	2	100,000	Resp time	80% < 10s
3	3		Velocity	40

BAHigh - High Importance business analytics users / applications				
Per	Imp	Duration	Goal Type	Goal
1	2	25,000	Resp time	90% < 1s
2	3	1,000,000	Velocity	40
3	4		Velocity	20

BAMed - Medium Importance business analytics users / applications				
Per	Imp	Duration	Goal Type	Goal
1	3	1,000,000	Velocity	40
2	4		Velocity	10

BALow - Lower Importance business analytics users / applications				
Per	Imp	Duration	Goal Type	Goal
1	Discretionary		Discretionary	n/a

BUSRCrit - Critical user (Added just for our redbook evaluation)				
Per	Imp	Duration	Goal Type	Goal
1	1		Velocity	80

Figure 4-23 Residency WLM service classes serving DDF queries

Figure 4-24 contains the service class definitions serving JES2 (batch) queries that we put in place for our residency. Though the support for Accelerator query prioritization for locally submitted queries (such as batch, TSO) was not yet available, the samples that we provide have the modifications assuming that it was.

BABATCR - Critical Importance business analytics Batch Reporting				
Per	Imp	Duration	Goal Type	Goal
1	1	25,000	Resp time	90% < 1s
2	2		Velocity	40

BABATMD - Medium Importance business analytics Batch Reporting				
Per	Imp	Duration	Goal Type	Goal
1	3		Velocity	20

BABATLO - Lower Importance business analytics Batch Reporting				
Per	Imp	Duration	Goal Type	Goal
1	Discretionary		Discretionary	

Figure 4-24 Residency WLM service classes serving local (batch reporting) queries

Note: In our experience, most System z clients only use WLM Importance “1” for address spaces that *serve* transactions (such as DB2, WebSphere), not transactions (or queries) themselves. This is also a general zWLM IBM recommendation. The service class in Figure 4-24 contains a WLM Importance “1” service class solely for the purpose of our evaluation of the highest priority available with the Accelerator. It is *not* a recommendation. If a customer considers using an Importance “1” for a first period service class serving queries, we highly recommend that there is a very short duration associated with it, such as 1000. Recall, WLM *duration* is the amount of System z resource that is consumed before a unit of work (such as query) will move to the next WLM period.

WLM classification of DDF queries

Review and make any necessary modifications to your WLM classification rules to differentiate the work appropriately into your refined service class definitions. We suggest considering the use of the DB2 client information attributes along with potentially the WLM_SET_CLIENT_INFO stored procedure, for more granular classification of the various users/applications contending for resources. See “WLM DDF classification preparation” on page 97 for an example of how we exploited for our query prioritization experiments. Additional information is provided in *Co-locating Transactional and Data Warehouse Workloads on System z*, SG24-7726.



Query acceleration management

IBM DB2 Analytics Accelerator, and IBM DB2 for z/OS provide a single unique system for online transaction processing (OLTP) and analytical dynamic query workloads.

DB2 automatically makes a choice on the most efficient and cost-optimized platform for execution. The chapter “Query acceleration management” of the book, *Optimizing DB2 Queries with IBM DB2 Analytics Accelerator for z/OS*, SG24-8005, covers extensively query acceleration foundation.

In this chapter, we mainly describe the new query acceleration features that were introduced in the IBM DB2 Analytics Accelerator V3.1. Also, some existing V2 functionalities are described as the background of the V3 new features. To avoid confusion, throughout the chapter, new features are high lighted in **bold**.

The following topics are described:

- ▶ Query acceleration processing flow
- ▶ Query acceleration settings and enablement
- ▶ Query acceleration criteria
- ▶ How IBM DB2 Analytics Accelerator handles correlated subqueries
- ▶ INSERT from subselect query acceleration processing
- ▶ Profile tables
- ▶ Accelerator EXPLAIN support
- ▶ DB2 Analytics Accelerator hardware considerations

5.1 Query acceleration processing flow

IBM DB2 query acceleration is viewed as a new query access path for DB2. Figure 5-1 illustrates a high-level query execution flow. There is an application on the left, DB2 in the middle, and DB2 Analytics Accelerator on the right. When the application submits a dynamic SQL query, DB2 analyzes it.

If query acceleration is not enabled or the query does not qualify for acceleration, it is executed locally within DB2.

If query acceleration is enabled, and the query qualifies for acceleration, DB2 converts the query into the Accelerator-supported syntax and routes it to the DB2 Analytics Accelerator through a Distributed Relational Database Architecture (DRDA) requestor connection. DB2 talks to the Accelerator DRDA server on the SMP host in the PureData System for Analytics appliance powered by Netezza, which takes over the query execution.

The Accelerator sends the result back to DB2 through DRDA. And the result is passed to the application. Also shown on the chart are the heartbeat messages from the Accelerator to DB2, with the Accelerator availability and performance indicators.

There is no change to the high-level query execution processing flow in V3.

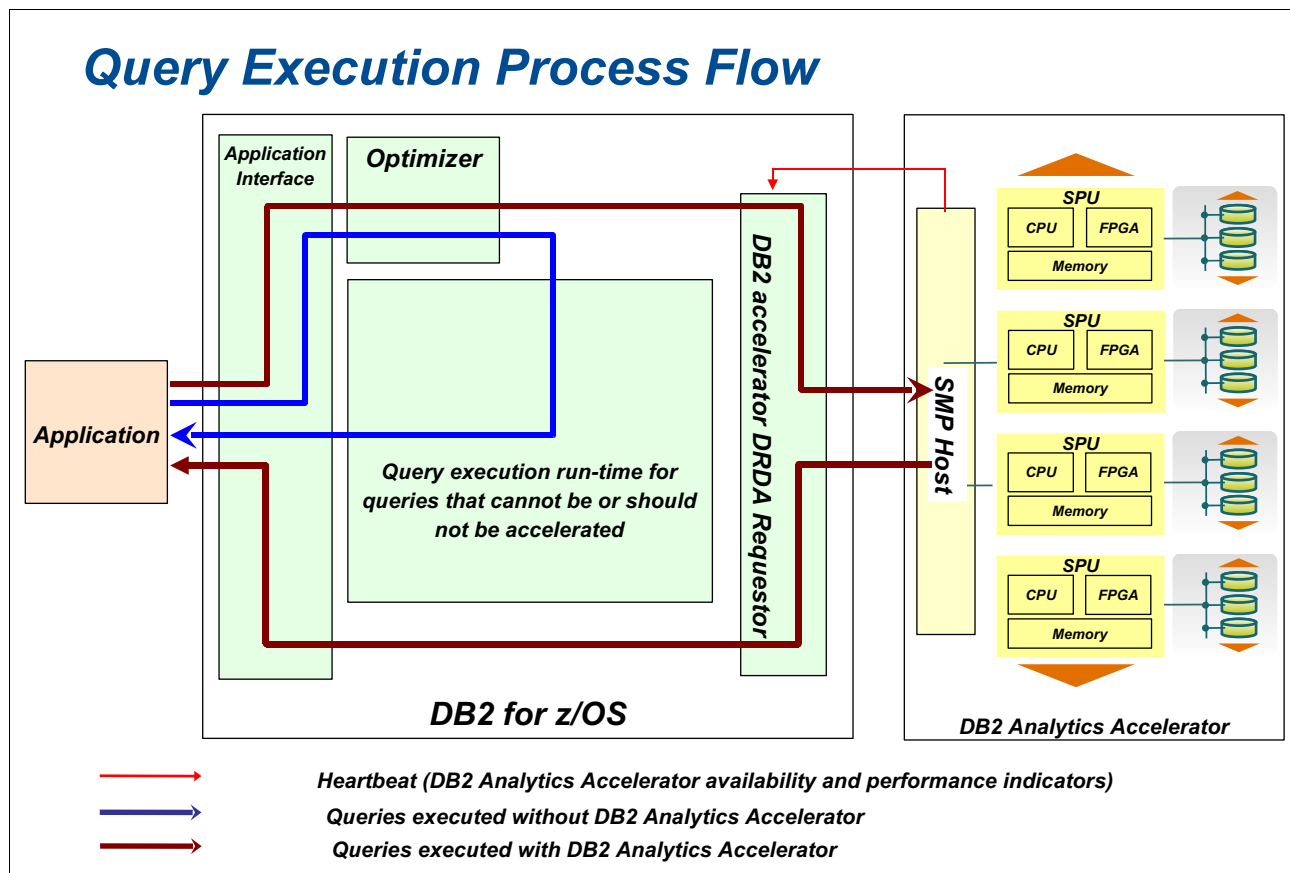


Figure 5-1 Query execution flow

5.2 Query acceleration settings and enablement

Query acceleration is controlled by both DSNZPARM parameters and the new special register, CURRENT QUERY ACCELERATION.

Following is the list of DSNZPARM parameters and the special register setting for the query acceleration. The DSNZPARM parameters are for the system level. In most cases, setting the QUERY_ACCELERATION DSNZPARM to ENABLE is all that is required. However, based on business requirements, you might have to configure different settings for different applications. In this case, you can modify the application source code by setting the special register CURRENT QUERY ACCELERATION after establishing the database session or connection, and before issuing any SQL statements.

5.2.1 DSNZPARM settings for accelerated queries

In this section, we briefly describe the DSNZPARMs related to Accelerator settings. For a more comprehensive and detailed list of DSNZPARMs to be evaluated for warehousing applications, see Appendix C, “DSNZPARM parameters setting for data warehouse” on page 315.

ACCEL DSNZPARM

ACCEL DSNZPARM specifies whether Accelerator servers can be used with the DB2 subsystem, and how the Accelerator servers are to be enabled and started. An Accelerator server cannot be started unless it is enabled.

Following are possible values of ACCEL:

NO	This specifies that the Accelerator servers cannot be used with the DB2 subsystem.
AUTO	This specifies that the Accelerator servers are automatically enabled and started when the DB2 subsystem is enabled.
COMMAND	This specifies that the Accelerator servers are automatically enabled when the DB2 subsystem is started. The Accelerator servers can be started with the DB2 START ACCEL command.

To enable query acceleration, ACCEL should be set as AUTO or COMMAND. These settings remain the same as V2.

QUERY_ACCELERATION DSNZPARM

The QUERY_ACCELERATION keyword on the DSN6SPRM macro defines the default value for the CURRENT QUERY ACCELERATION special register. The QUERY_ACCELERATION default is used when the SET CURRENT QUERY ACCELERATION SQL statement has not been issued.

Following are the possible options of QUERY_ACCELERATION:

NONE	No query acceleration takes place. (default)
ENABLE	<p>A query is accelerated only if DB2 determines there is an advantage to running the query on the Accelerator. Costing and heuristics contribute to the decision of DB2 to send to the Accelerator.</p> <p>If there is an Accelerator failure while running the query, or the Accelerator returns an error, DB2 returns a negative SQL code to the application.</p>

ENABLE_WITH_FAILBACK

This option is similar to ENABLE with a slight change to the behavior of DB2 for the cases when there is an Accelerator failure or when the Accelerator returns an error.

If the Accelerator returns an error during a FETCH or a subsequent OPEN, DB2 returns the error to the user, and does not execute the query.

ELIGIBLE

Indicates that any given query will be accelerated if it is eligible for acceleration without heuristics and costing. Queries that are not eligible will be executed in DB2. Costing and heuristics are not considered.

If there is an Accelerator failure while running the query, or the Accelerator returns an error, DB2 returns a negative SQL code to the application.

ALL

Indicates that any given query will be accelerated if it is eligible for acceleration without heuristics and costing, and queries that are not eligible will not be executed in DB2, but will raise an SQL error instead. There is no failback with this option.

Note: ELIGIBLE and ALL are newly introduced in V3 by APAR PM72265. With these two new options added to the DSN6SPRM.QUERY_ACCELERATION parameter and CURRENT QUERY ACCELERATION register, users have further control on the query acceleration. For example, if they want the query to run in the Accelerator regardless, without comparing the performance of running locally in DB2, the option ELIGIBLE can be specified.

ELIGIBLE and ALL have also been added to DB2 9 for z/OS and Accelerator V2.1 by APAR PM72264.

QUERY_ACCEL_OPTIONS DSNZPARM

QUERY_ACCEL_OPTIONS DSNZPARM specifies additional types of SQL queries that are not by default included in query acceleration. This DSNZPARM is introduced so that the user can selectively enable the query acceleration of certain functionality. Typically, this functionality may potentially behave differently in the DB2 than in the Accelerator and the user wants to be aware of it.

The QUERY_ACCEL_OPTIONS parameter can support up to 32 options. Five possible options are supported now. They are: NONE, 1, 2, 3, and 4. The installation panel DSNTIP8A allows you to set up the options:

NONE

This is the default setting. It means that query acceleration is restricted to SQL statements that do not have the attributes described in the remaining options (given below).

1

This specifies that the queries that include data encoded by multi-byte character set EBCDIC encoding scheme are not blocked from executing on the Accelerator. Although, the Accelerator encodes the same data in the UTF-8 Unicode encoding scheme.

EBCDIC and Unicode implement different collating sequences. The collating sequence for Unicode is numeric, uppercase characters, and then lower case characters (1, 2, 3, A, B, C, a, b, c). In EBCDIC, the collating sequence is lower case, upper case, and then numeric (a, b, c, A, B, C, 1, 2, 3). There are also differences in collating for the national characters. This affects both data ordering and the results

from range predicates. Therefore, if the tables include character columns where more than one of these groups can be found in the column values and the SQL statements include range predicates or ordering on these columns, a query executed in DB2 may return a different result set than the same query that is executed in the Accelerator.

2 This specifies that in an INSERT FROM SELECT statement, the SELECT part is enabled for executing on the Accelerator, although the SELECT operates on the data in the Accelerator that might not be current.

3 This specifies that the queries that include DB2 byte-based string functions, for example, SUBSTR, on data encoded by multi-byte character sets encoding schemes (like Unicode) are not blocked from executing on the Accelerator. Although, the Accelerator supports only character-based string functions. If the data on which the string function is specified contains only single-byte characters, executing the function on the Accelerator will return the same result as executing the function on DB2 irrespective of what encoding scheme is used for the data. However, if the data contains multi-byte characters, the results might not be the same.

Option 3 is also used to enable the query acceleration of the scalar function when CODEUNITS16 or OCTETS is specified, and option 3 is specified in DSNZPARM QUERY_ACCEL_OPTIONS.

4 This specifies that the queries that reference an expression with a DATE data type that uses a LOCAL format are not blocked from executing on IBM DB2 Analytics Accelerator. IBM DB2 Analytics Accelerator will use dd/mm/yyyy format to interpret the input and output date value.

Option 4 should be specified only when the DATE FORMAT field of the install panel DSNTIP4 specifies LOCAL or when application programs that process SQL on DB2 have been precompiled with the DATE (LOCAL) option. In either case, the LOCAL date exit routine must also define the specific “dd/mm/yyyy date” format. If the LOCAL format is not defined as “dd/mm/yyyy”, the query might return unpredictable results.

These options can be specified individually or together. Each of the following is permitted:

```
QUERY_ACCEL_OPTIONS=NONE
QUERY_ACCEL_OPTIONS=1
QUERY_ACCEL_OPTIONS=3
QUERY_ACCEL_OPTIONS=(1, 2, 3)
QUERY_ACCEL_OPTIONS=(3, 1)
QUERY_ACCEL_OPTIONS=(,1)
QUERY_ACCEL_OPTIONS=(1, 2, 3, 4)
```

QUERY_ ACCELERATION DSNZPARM was introduced after V2. This DSNZPARM can be set to selectively enable certain query acceleration functionality.

5.2.2 CURRENT QUERY ACCELERATION special register

CURRENT QUERY ACCELERATION is a special register variable that identifies when DB2 sends queries to an Accelerator server, and what DB2 does if the Accelerator server fails. The data type is VARCHAR(255).

Following are possible values of CURRENT QUERY ACCELERATION:

NONE	This specifies that no query acceleration is done.
ENABLE	This specifies that the queries are accelerated only if DB2 determines that it is advantageous to do so. If there is an Accelerator failure while a query is running, or the Accelerator returns an error, DB2 returns a negative SQLCODE to the application.
ENABLE WITH FAILBACK	This specifies that the queries are accelerated only if DB2 determines that it is advantageous to do so. If the Accelerator returns an error during the PREPARE of first OPEN for the query, DB2 executes the query without the Accelerator. If the Accelerator returns an error during a FETCH or a subsequent OPEN, DB2 returns the error to the user, and does not execute the query.
ELIGIBLE	This specifies that the queries are accelerated if they are eligible for acceleration. DB2 does not use cost information to determine whether to accelerate the queries. Queries that are not eligible for acceleration are executed by DB2. If there is an Accelerator failure while a query is running, or the Accelerator returns an error, DB2 returns a negative SQLCODE to the application.
ALL	<p>This specifies that the queries are accelerated if they are eligible for acceleration. DB2 does not use cost information to determine whether to accelerate the queries. Queries that are not eligible for acceleration are not executed by DB2, and an SQL error is returned. If there is an Accelerator failure while a query is running, or the Accelerator returns an error, DB2 returns a negative SQLCODE4742 to the application. Except that when the query satisfies either of the conditions below, the query will run in DB2 instead:</p> <p>If all the tables referenced in the query have the qualifier SYSIBM, SYSACCEL, DB2GSE, SYSXSR, or declared global temp table.</p> <p>If the top query block is pruned, that is, the query contains a pruning predicate like 1 = 2. The query will stay in DB2.</p>

QUERY_ACCELERATION DSNZPARM determines the default value that is to be used for the CURRENT QUERY ACCELERATION special register. The default for the initial value of QUERY_ACCELERATION DSNZPARM is NONE unless your installation has changed the value.

You can change the value of the register by executing the SET CURRENT QUERY ACCELERATION statement. For more information about this statement, see the SET CURRENT QUERY ACCELERATION topic in *DB2 10 for z/OS SQL Reference*, SC19-2983.

To enable query acceleration, CURRENT QUERY ACCELERATION should be set as a value other than NONE.

5.2.3 ODBC and JDBC application query acceleration

With Open Database Connectivity (ODBC) and Java Database Connectivity (JDBC) drivers, you can control query acceleration using the special register, but you can also do so *without having to modify your applications*. Starting with IBM DB2 Connect™ Version 10.1 with Fix Pack 2 (PTFs IP23387 - IP23396, depending on the platform), ODBC data sources can be configured to enable any valid DB2 for z/OS special register. Refer to the document *Enabling query acceleration with IBM DB2 Analytics Accelerator for ODBC and JDBC applications without modifying the applications*, which is available at the following site:

<http://www.ibm.com/support/docview.wss?uid=swg27038078>

See also Appendix D.5, “InfoSphere Optim Configuration Manager” on page 342 for changing registered client properties.

5.3 Query acceleration criteria

In general, eligible queries for DB2 Analytics Accelerator are mostly long-running queries. Short-running OLTP queries will seldom qualify.

This section describes other basic acceleration criteria that are currently being used by the DB2 optimizer to make a decision about whether or not to accelerate a given query. The criteria described in this section is valid at the time of writing and might change in the future.

Figure 5-2 on page 128 shows the acceleration criteria without any consideration to the sequence of evaluation.

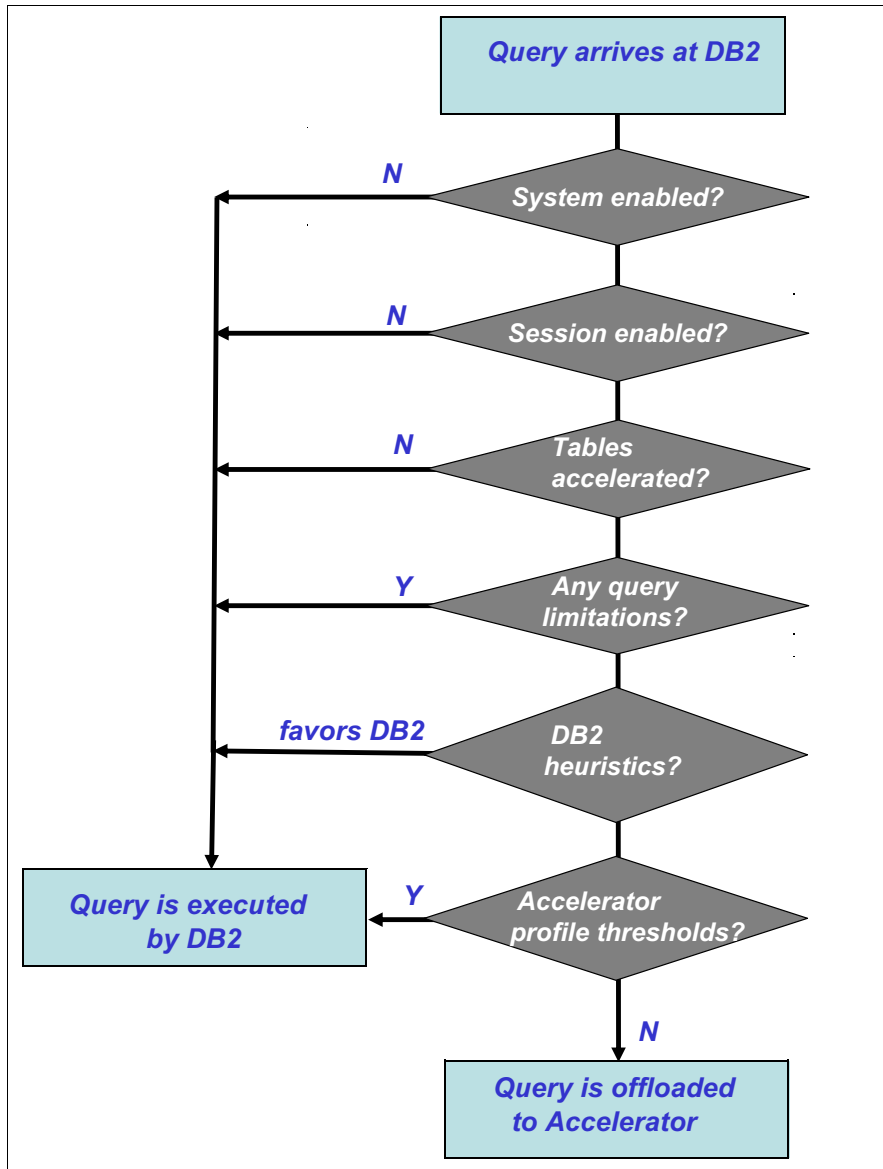


Figure 5-2 DB2 Analytics Accelerator accelerate criteria

As shown in Figure 5-2, accelerating a dynamic query is based on the following criteria:

1. Query acceleration DSNZPARMs or special registers described in the previous section are set correctly to enable the query acceleration.
2. An Accelerator is started and available.
3. All the tables associated with the query are accelerated and the data of all the referenced columns in the query is loaded and resides in the same Accelerator.
4. The SQL query is among the query types that DB2 for z/OS can be accelerated.
5. The SQL functionality that is required to execute the query is supported by the Accelerator. DB2 decides not to accelerate a query if the Accelerator does not support the SQL feature, or if there is no easy way that the query can be mapped for DB2 Analytics Accelerator to return the same results as DB2. In some cases, users can rewrite the query to make it eligible for the acceleration. See “The query types that are supported by the Accelerator” on page 129.

6. The DB2 optimizer rule-based heuristics determine that the query would be a long-running query in DB2. The estimate is based on query semantics, available indexes, and statistics that are collected in catalog tables. **If the CURRENT QUERY ACCELERATION is set as ELIGIBLE or ALL, this step is skipped.**
7. The thresholds defined in the Accelerator profile table are evaluated to favor the Accelerator. Read 5.4, “How IBM DB2 Analytics Accelerator handles correlated subqueries” on page 133 for additional information about how to manipulate the threshold settings. **If the CURRENT QUERY ACCELERATION is set as ELIGIBLE or ALL, this step is skipped.**

DB2 accelerates a query only when all the preceding criteria are satisfied; otherwise, the query runs in DB2.

5.3.1 The query types that are supported by the Accelerator

A query needs to satisfy the following mandatory criteria to be considered for query acceleration¹:

- ▶ The cursor is not defined as a scrollable or row set cursor.
- ▶ The query is defined as read-only.
- ▶ The query is dynamic.
- ▶ The query is from a package (not a plan with DBRMs).
- ▶ The query is a SELECT statement, **or INSERT FROM SELECT statement.**

If the query is a SELECT statement, DB2 will only accelerate the whole query, not a query block (part of a query), which means that the whole query will either run in DB2 or run in the Accelerator, not from both.

If a query is an INSERT from a select statement, and DSNZPARM QUERY_ACCEL_OPTIONS includes number 2, DB2 will accelerate the whole select statement. The INSERT operation will be processed by DB2. Query acceleration criteria of the INSERT from the subselect statement is the same as the select statement.

5.3.2 The query functionality limitations

DB2 will accelerate a query only when the SQL functionality that is required to execute the query is supported by the Accelerator. The following list provides the query functionality that is not supported by the Accelerator:

- ▶ The encoding scheme of the statement is multiple encodings. This can be either because tables are from different encoding schemes, or because the query contains a coded character set identifier (CCSID)-specific expression, for example, a cast specification expression with a CCSID option.
- ▶ The query FROM clause specifies data-change-table-reference; that is, the query is select from FINAL TABLE or select from OLD TABLE.
- ▶ The query contains a *correlated table expression*. A correlated table expression is a table expression that contains one or more correlated references to other tables that are in the same FROM clause. Regular correlated queries might qualify for accelerate.
- ▶ The query contains a recursive common table expression reference.
- ▶ The query contains one of the following predicates:
 - <op> ALL, where <op> can be =, <>, >, >=, <=
 - NOT IN <subquery>

¹ This is subject to change as more support is added to the Accelerator environment.

- ▶ The query contains a string expression (including a column) with an unsupported subtype.

The supported subtypes are:

- EBCDIC SBCS
- ASCII SBCS
- UNICODE SBCS
- UNICODE MIXED
- UNICODE DBCS (graphic)
- **EBCDIC MIXED**
- **EBCDIC DBCS (graphic)**

Note: EBCDIC MIXED and EBCDIC DBCS (GRAPHIC) are supported only when the QUERY_ACCEL_OPTIONS includes number 1 and SYSACCEL.SYSACCELERATEDTABLES.SUPPORTLEVEL value is 2.

- ▶ The query contains an expression with an unsupported result data type.

Supported result data types² are:

- CHAR
- VARCHAR
- **GRAPHIC (non-ASCII)**
- **VARGRAPHIC (non-ASCII)**
- SMALLINT
- INT
- BIGINT
- DECIMAL
- FLOAT
- REAL
- DOUBLE
- DATE
- TIME
- TIMESTAMP with precision 6

- ▶ The query refers to a column that uses a field procedure (FIELDPROC).

- ▶ The query (SQL statement) uses a special register other than:

- CURRENT DATE
- CURRENT TIME
- CURRENT TIMESTAMP

- ▶ The query contains a date or time expression in which LOCAL is used as the output format, or a CHAR function in which LOCAL is specified as the second argument.

Exception: If QUERY_ACCEL_OPTIONS 4 is specified, this restriction does not apply. Refer to the QUERY_ACCEL_OPTIONS DSNZPARM option 4 for more details.

- ▶ The query contains a sequence expression (NEXTVAL or PREVVAL).

- ▶ The query contains a user-defined function (UDF).

- ▶ The query contains a ROW CHANGE expression.

- ▶ A date, time, or time stamp duration is specified in the query. Only labeled durations are supported.

- ▶ The query contains a string constant that is longer than 16000 characters.

- ▶ A new column name is referenced in a sort-key expression, for example:

```
SELECT C1+1 AS X, C1+2 AS Y FROM T WHERE ... ORDER BY X+Y;
```

² The listed types refer to the built-in types. User-defined types (UDTs) are not allowed.

- ▶ The query contains a correlated scalar-fullselect. Here “correlated” means that the scalar-fullselect references a column of a table or view that is named in an outer subselect.
- ▶ **The query contains one of the scalar functions listed below, and also satisfies these three conditions:**
 - **CODEUNIT16, OCTETS, or NO CODEUNITS is specified**
 - **QUERY_ACCEL_OPTION does not specify 3**
 - **Contain non-SBCS string argument**

Function list:

- **SUBSTRING**
- **CHARACTER_LENGTH**
- **CAST specification**
- **CHAR**
- **VARCHAR**
- **LOCATE**
- **LOCATE_IN_STRING**
- **LEFT**
- **RIGHT**
- **POSITION**
- ▶ The query contains a cast specification with a result data type of GRAPHIC or VARGRAPHIC.
- ▶ The query contains one of the following scalar functions or cast specifications with a string argument that is encoded in UTF-8 or UTF-16, **and QUERY_ACCEL_OPTIONS does not specify option 3:**
 - **CAST(arg AS VARCHAR(*n*))** where *n* is less than the length of the argument
 - **VARCHAR(arg, *n*)** where *n* is less than the length of the argument
 - **LOWER(arg, *n*)** where *n* is not equal to the length of the argument
 - **UPPER(arg, *n*)** where *n* is not equal to the length of the argument
 - **CAST(arg as CHAR(*n*))**
 - **CHAR**
 - **LEFT**
 - **LPAD**
 - **LOCATE**
 - **LOCATE_IN_STRING**
 - **POSSTR**
 - **REPLACE**
 - **RIGHT**
 - **RPAD**
 - **SUBSTR**
 - **TRANSLATE** if more than one argument is specified
- ▶ The query uses a LENGTH function, but the argument of this function is not a string, or is encoded in UTF-8 or UTF-16.
- ▶ The query uses a DAY function where the argument of the function specifies a duration.
- ▶ The query uses a MIN or MAX function with string values or more than four arguments.
- ▶ The query uses an EXTRACT function, which specifies that the SECOND portion of a TIME or TIMESTAMP value must be returned.
- ▶ The query uses one of the following aggregate functions with the DISTINCT option:
 - **STDDEV**
 - **STDDEV_SAMP**
 - **VARIANCE**

- VAR_SAMP
- ▶ The query uses any table functions (ADMIN_TASK_LIST, ADMIN_TASK_STATUS, and so on).
- ▶ The query references a predicate or expression in which one side is string date type and the other side is numeric data type. For example:

```
SELECT .. FROM TABLE WHERE VARCHAR_COL = INT_COL
```
- ▶ The query references a scalar function that is not included in the table below.

Table 5-1 List of DB2 scalar functions supported in DB2 Analytics Accelerator

Scalar functions			
ABS	FLOAT	MIDNIGHT_SECONDS	SIGN
ADD_MONTHS	FLOOR	MIN	SMALLINT
BIGINT	HOURL	MINUTE	SPACE
CEILING	IFFULL	MOD	SORT
CHAR	INTEGER	MONTH	STRIP
COALESCE	JULIAN_DAY	MONTHS_BETWEEN	SUBSTR
CONCAT	LAST_DAY	NEXT_DAY	TIME
DATE	LCASE	NULLIF	TIMESTAMP
DAY	LEFT	POSSTR	TIMESTAMP_FORMAT
DAYOFMONTH	LENGTH	POWER@	TRANSLATE
DAYOFWEEK	LN	QUARTER	TRUNCATE
DAYOFWEEK_ISO	LOCATE	RADIANS	UCASE
DAYOFYEAR	LOCATE_IN_STRING	REAL	UPPER
DAYS	LOG10	REPEAT	VALUE
DECIMAL	LOG	REPLACE	VARCHAR
DEGREE	LOWER	RIGHT	VARCHAR_FORMAT
DIGIT	LPAD	ROUND	WEEK_ISO
DOUBLE	LTRIM	RPAD	YEAR
EXP	MAX	RTRIM	
EXTRACT	MICROSECOND	SECOND	
SUBSTRING	CHARACTER_LENGTH	POSITION	BITAND
BITANDNOT	BITOR	BITNOT	TIMESTAMPDIFF

Note: APAR PM88071 introduces support of offload of implicit cast for DECFLOAT, bit operations, and TIMESTAMPDIFF.

5.4 How IBM DB2 Analytics Accelerator handles correlated subqueries

This section explains what correlated subqueries are, lists the current limitations of IBM DB2 Analytics Accelerator for z/OS regarding this type of subquery, and describes possible circumventions. For details, see the technote “*How IBM DB2 Analytics Accelerator for z/OS handles correlated subqueries*”, which is available at the following site:

<http://www.ibm.com/support/docview.wss?uid=swg27037928>

5.4.1 Background of the correlated subqueries

A *subquery* is a form of a fullselect that is enclosed within parentheses. For example, a subquery can be used in a search condition. A fullselect that is used to retrieve a single value as an expression within a statement is called a *scalar fullselect*. A fullselect that is used in the FROM clause of a query is called a *nested table expression*.

A subquery can include search conditions of its own, and these search conditions can, in turn, include further subqueries. Thus, an SQL statement can contain a hierarchy of subqueries. Hierarchy elements that contain subqueries are at a higher level than the subqueries they contain.

Subqueries might return multiple rows and multiple columns, and can exist in a FROM clause or as an argument of an EXISTS, IN, ANY, or ALL test. For example: “Find all customers that had an URGENT order” can be written as:

```
SELECT Name
FROM Customer
WHERE CustomerID IN (
    SELECT CustomerID
    FROM Orders
    WHERE Priority = 'URGENT')
```

There is a special case of subquery called *scalar subquery*, which returns exactly one value in the format of one row (or zero rows), one column table. For example, “List all stores where sales are higher than one percent of all company sales,” can be written in the SQL query-within-a-query form using a scalar subquery:

```
SELECT StoreID
FROM Stores
WHERE TotalSale > 0.01 * (
    SELECT SUM( TotalSales )
    FROM Stores );
```

The system calculates the sum of sales in the inner subquery first and then uses the calculated value when running the outer query.

The system evaluates all subqueries once, and then calculates and stores their SELECT expressions. When the outer query runs, the system uses the computed values as a substitute for the subquery.

A *correlated subquery* is a subquery that references (or correlates with) the outer query. With correlated subqueries, the system evaluates the subqueries repeatedly, once for every row that is selected from the outer table.

Correlated subqueries are not self-contained, but instead refer to columns of the outer query. For example, the query “Find all stores where the sales of dairy products are more than 10% of the total sales” can be written in SQL as:

```
SELECT StoreID
FROM Stores S
WHERE S.TotalSales * 0.1 < (
    SELECT SUM( I.Price )
    FROM Item I
    WHERE S.StoreID = I.StoreID
    AND I.ItemType = "Dairy" );
```

This is a correlated subquery because the inner query uses the StoreID column from the Stores table in the outer query to calculate the total dairy sales for specific stores. In many cases, you can use JOIN expressions instead of correlated subqueries. For example, you could rewrite the previous query by joining the Stores table with a nested table expression that is based on the Items table:

```
SELECT S.StoreID
FROM Stores S, (
    SELECT I.StoreID, SUM( I.Price ) DairySales
    FROM Items I
    WHERE I.ItemType = "Dairy"
    GROUP BY I.StoreID ) AS D
WHERE S.StoreID = D.StoreID
AND S.TotalSales * 0.1 < D.DairySales;
```

5.4.2 Correlated subqueries processing in the Accelerator

IBM DB2 Analytics Accelerator for z/OS supports non-correlated and certain correlated subqueries. Whenever it encounters a non-correlated subquery, it precalculates the subquery once. When it encounters correlated subqueries in WHERE clauses, it transforms these subqueries to equivalent JOIN expressions. Depending on the form and the placement, IBM DB2 Analytics Accelerator for z/OS might be unable to execute correlated subqueries and issue an error. DB2 cannot decide up-front for all the cases whether a query execution on the Accelerator is possible. If the CURRENT QUERY ACCELERATION special register was set to ENABLE WITH FAILBACK, the query will be executed by DB2. The error will not be reported to the calling application. For all other values of this special register except NONE, that is, ENABLE, ELIGIBLE, and ALL, DB2 will return a negative SQL code to the application as a result of the Accelerator error.

The following restrictions apply to the form and the placement of correlated subqueries:

- ▶ You can use correlated subqueries in WHERE clauses.
- ▶ You can use correlated subqueries in inner join conditions and with the operator for equal join conditions.
- ▶ You can use correlated subqueries in mixed correlated expressions, but only in the following form:


```
expr(correlated_columnA, correlated_columnB, ...) = expr(local_columnX, local_columnY, ...)
```
- ▶ You can use only one level of nesting, that is, a correlated subquery can refer only to columns in the next outer SELECT statement.
- ▶ You cannot use correlated subqueries in SET operations (UNION, INTERSECT, EXCEPT, and MINUS).

- ▶ You cannot use correlated subqueries in aggregates with GROUP BY and HAVING clauses.
- ▶ You cannot use correlated subqueries in clauses that are connected by the OR operator.
- ▶ You cannot use correlated subqueries in CASE or WHEN expressions.
- ▶ You cannot use correlated subqueries in IN lists.
- ▶ You cannot use correlated subqueries in SELECT lists.

5.4.3 Circumventions

If a query with a correlated subquery cannot be run on the Accelerator, set the CURRENT QUERY ACCELERATION to ENABLE WITH FAILBACK or rewrite the query so that it contains only supported constructs. Rewriting a query can be a complex task and special care must be taken to ensure that the query returns the same results. See the following examples, which show how a subquery in the SELECT clause can be rewritten:

Example 1

```
SELECT C.LastName, (
    SELECT MAX(O.TotalPrice)
    FROM Orders AS O
    WHERE C.CustomerID = O.CustomerID)
    AS BiggestOrder
FROM Customers AS C;
```

This query returns the most expensive order for each customer. For customers who did not order anything, the BiggestOrder value will be NULL.

The subquery in this type of query can usually be changed into a JOIN expression:

```
SELECT C.LastName, O.BiggestOrder
FROM Customers AS C
LEFT JOIN (
    SELECT CustomerID, MAX(TotalPrice) AS BiggestOrder
    FROM Orders
    GROUP BY CustomerID) AS O
ON C.CustomerID = O.CustomerID
```

This query still contains a subquery. However, the subquery is no longer correlated with the outer query. A LEFT JOIN is used to preserve the NULL values that arise from customers with no orders.

Example 2

A more complex example with a BETWEEN JOIN condition:

```
SELECT C.START_DATE, C.END_DATE, (
    SELECT COUNT(*)
    FROM HOLIDAYS AS H
    WHERE H.HOLIDAY_DATE BETWEEN C.START_DATE AND C.END_DATE
    AND H.COUNTRY = 'UK' ) AS HOLIDAY_COUNT
FROM CONTRACTS AS C
WHERE C.START_DATE >= '2000-01-01'
```

The query in “Example 2” on page 135 returns the number of holidays in the United Kingdom between the start and end dates of a contract, but only for those contracts that started after 1 January 2000.

The query can be rewritten as follows:

```
SELECT C.START_DATE, C.END_DATE,
       COALESCE(F.HOLIDAY_COUNT, 0) AS HOLIDAY_CNT
FROM CONTRACTS AS C
-- For each pair of dates, find number of holidays between them
LEFT JOIN (
  SELECT E.START_DATE, E.END_DATE, COUNT(*) AS HOLIDAY_COUNT
    -- Get all the unique combinations of the two dates
  FROM (
    SELECT DISTINCT D.START_DATE, D.END_DATE
    FROM CONTRACTS AS D
    WHERE D.START_DATE >= '2000-01-01') AS E
  JOIN HOLIDAYS AS H
  ON H.HOLIDAY_DATE BETWEEN
    E.START_DATE AND E.END_DATE
  WHERE H.COUNTRY = 'UK'
  GROUP BY E.START_DATE, E.END_DATE) AS F
ON C.START_DATE = F.START_DATE
   AND C.END_DATE = F.END_DATE
WHERE C.START_DATE >= '2000-01-01'
```

The COALESCE function is used to convert the NULL values from the left outer join into valid zero values.

5.5 INSERT from subselect query acceleration processing

If a query is an INSERT FROM SELECT statement and satisfies the query acceleration criteria that are listed in the previous section, and also the DSNZPARM QUERY_ACCEL_OPTIONS includes number 2, DB2 will accelerate the select portion of the INSERT statement. The Accelerator will return the select result back to DB2 and the INSERT operation will be processed by DB2 locally.

Example 5-1 shows the access path of the INSERT statement.

Example 5-1 Simple INSERT statement that is eligible for DB2 Analytics Accelerator offload

```
INSERT INTO BID10TB.GOSLDWF.GOSLDW_SALES_FACT SELECT *
FROM BID1TB.GOSLDWF.GOSLDW_SALES_FACT
WHERE ORDER_DAY_KEY BETWEEN 20120204 AND 20120304
AND SALE_TOTAL <> 999999
```

Figure 5-3 on page 137 shows the sample access plan diagram of the query that is not accelerated because the value of the CURRENT QUERY ACCELERATION register is set to NONE.

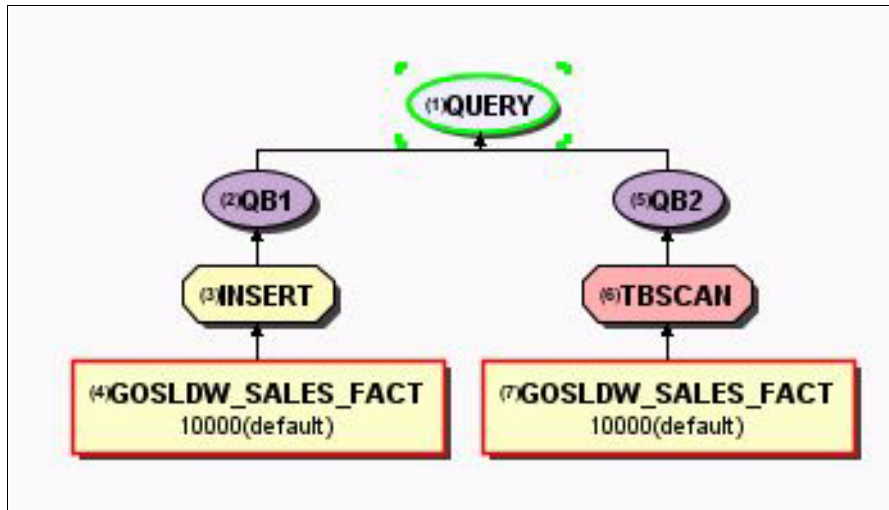


Figure 5-3 Access plan diagram of the query that is not accelerated

Figure 5-4 shows the access plan diagram for the same query when it satisfies all the eligibility criteria for DB2 Analytics Accelerator query acceleration. When comparing this access plan diagram with Figure 5-3, notice the new symbol named “accelerated” appearing in the accelerated query. This corresponds to the column value of ACCESTYPE='A' on the PLAN_TABLE. From the access plan diagram, you can see that one query block (for SELECT) is accelerated and the other query block (for INSERT) is still in DB2.

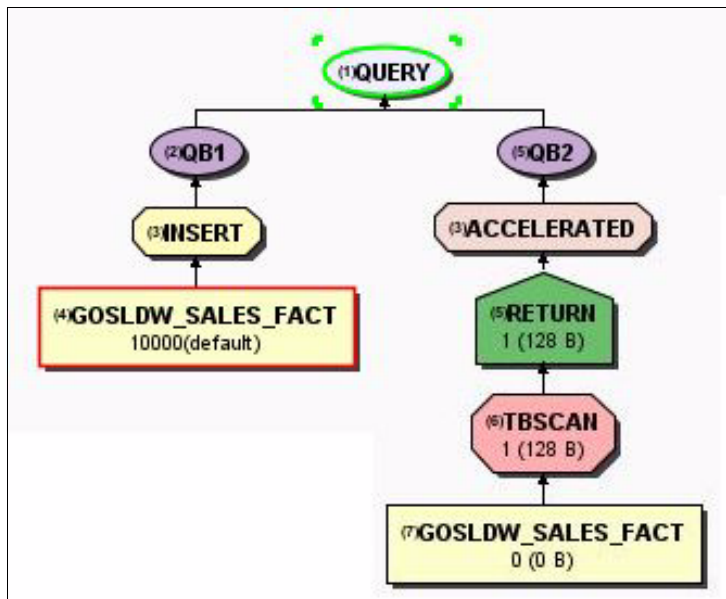


Figure 5-4 Access plan diagram of the query that is accelerated

INSERT from subselect query acceleration was newly introduced after V2.

EXPLAIN

EXPLAIN output shows the routed subquery with ACCESTYPE='A' and QBLOCK_TYPE='NCOSUB'. Table 5-2 shows the PLAN_TABLE information.

Table 5-2 PLAN_TABLE

QUERYNO	QBLOCKNO	TNAME	ACCESS TYPE	QBLK_TY PE	PARENT_ QBLKNO	TABLE_ TYPE
1	1	GOSLDW_SALES_FACT		INSERT	0	T
1	2	GOSLDW_SALES_FACT	A	NCOSUB	1	T

The first row TNAME is target table BID10TB.GOSLDWF.GOSLDW_SALES_FACT and the second row TNAME is source table BID1TB.GOSLDWF.GOSLDW_SALES_FACT.

In one of its rows, the DSN_QUERYINFO_TABLE that is shown in Table 5-3, contains the reason why a query is not qualified for routing. If option 2 is not specified, the reason_code will be 17.

Table 5-3 DSN_QUERYINFO_TABLE

Column name	Data type	Description
...		
REASON_ CODE	SMALLINT NOT NULL WITH DEFAULT	The reason why the query is not offloaded: 1 No active Accelerator was found when EXPLAIN was executed. 17. The query is an INSERT statement, and DB2 subsystem parameter DSN6SPRM.QUERY_ACCEL_OPTIONS does not specify option 2 to enable its acceleration.
...		

5.6 Profile tables

This section describes all the profile tables and how to manipulate the data in the tables to influence query acceleration.

Heuristics adds three parameters that can be adjusted through profile tables.

Table 5-4 on page 139 summarizes the KEYWORDS and default setting for each parameter. In general, if no profile is created or the profile is disabled or stopped, the default setting that is indicated in the table is used by heuristics.

Attention: When profile monitoring is not running, the default behavior is to ignore the default value for ACCEL_RESULTS_SIZE_THRESHOLD.

Table 5-4 DSN_PROFILE_ATTRIBUTES table

KEYWORD	Data type	Default value	Description	DSN_QUERYINFO_TABLE.QI_DATA (REASON_CODE =3 for all of the subreason codes below)
ACCEL_TABLE_THRESHOLD	Integer	1000000	Specifies the maximum total table cardinality for a query to be treated as a short-running query. -1 means that this check is disabled.	300 The query is a DB2 short running query or routing is not advantageous.
ACCEL_TOTALCOST_THRESHOLD	Integer	5000	Represents the minimum estimated cost that is used to determine whether the query should be offloaded.	301 The query is a DB2 short running query or routing is not advantageous.
ACCEL_RESULTSIZE_THRESHOLD	Integer	-1	Represents the maximum number of thousand rows for the result set to allow query to be offloaded. The unit is thousand rows, for example, 2000 means 2000 K rows, that is, 2 million rows. The default value is -1, meaning the result size checking is disabled.	302 The query is a DB2 short running query or routing is not advantageous.

When profile monitoring is started, the heuristics parameters can be adjusted through profile tables.

When DB2 decides to execute the query locally instead of sending it to the Accelerator with reason code 3, meaning that the query is a short-running query or routing is not advantageous, DSN_QUERYINFO_TABLE.QI_DATA contains a subreason code to tell which criteria causes reason code 3, for example, if the query references only small tables.

5.7 Accelerator EXPLAIN support

The DB2 EXPLAIN function is enhanced to provide basic information about Accelerator usage. It tells whether a query qualifies for acceleration and, if not, why is not qualified. The access path details associated with the query execution by the Accelerator are provided independently of DB2 EXPLAIN by the IBM DB2 Analytics Accelerator Studio.

When a query is accelerated, for each query, regardless of the number of query blocks that a query contains, the whole query has one row in both PLAN_TABLE and DSN_QUERYINFO_TABLE. PLAN_TABLE column ACCESSTYPE is 'A'. And DSN_QUERYINFO_TABLE column QI_DATA contains the converted Accelerator query text.

If the query is not accelerated, REASON_CODE and QI_DATA columns provide the details about why the query is not accelerated. The following query can be used to retrieve the reason why a query is not accelerated, so that the corresponding action can be taken:

```
SELECT QUERYNO, REASON_CODE, QI_DATA FROM SYSADM.DSN_QUERYINFO_TABLE;
```

Here are a few examples of reason codes and actions to influence the query acceleration:

- ▶ When REASON_CODE is 2, set CURRENT QUERY ACCELERATION to a value that is not NONE.
- ▶ When REASON_CODE is 3, set CURRENT QUERY ACCELERATION as ELIGIBLE, or adjust the profile attribute that is based on the subreason code that is described in the previous section.
- ▶ When REASON_CODE is 4, change the query to add “FOR FETCH ONLY” clause.
- ▶ When REASON_CODE is 5, change the protocol to use DRDA instead of private.
- ▶ When REASON_CODE is 6, change the program so that it uses single row fetch, for example, both DSNTDP4 and DSNTIAUL use rowset cursor as default. The follow options should be used to make it single row fetch:
 - For DSNTDP4, specify this control statement: #SET MULT_FETCH 1
 - For DSNTIAUL, specify: PARMS('SQL,1'), and bind a new package:


```

          BIND PACKAGE(DSNTIB92) MEM(DSNTIAUL) ACTION(REPLACE) -
            CURRENTDATA(NO) ISOLATION(CS) RELEASE(COMMIT) -
            VALIDATE(RUN) EXPLAIN(NO)
          BIND PLAN(DSNTIB92) PKLIST(*.DSNTIB92.DSNTIAUL) -
            ACTION(REPLACE) ISOLATION(CS) VALIDATE(BIND)
          
```

Below is a list of reason codes that describe why the query is not accelerated:

- | | |
|----|--|
| 1 | No active Accelerator was found when an EXPLAIN statement was executed. |
| 2 | The special register CURRENT QUERY ACCELERATION is set to NONE. |
| 3 | The query is a DB2 short-running query or routing is not advantageous. |
| 4 | The query is not read-only. |
| 5 | The query uses DB2 private protocol for distributed processing (V9 only). |
| 6 | The cursor is defined as scrollable or rowset cursor. |
| 7 | The query is a multiple encoding scheme statement. |
| 8 | The query FROM clause specifies a data-change-table-reference. |
| 9 | The query contains a correlated table expression. |
| 10 | The query contains a recursive common table expression reference. |
| 11 | The query contains an unsupported expression. |
| 12 | The query references a table that is either not defined in Accelerator, or defined but on a different Accelerator from another table in the query, or the table is defined but not enabled for query acceleration. |
| 13 | The Accelerator containing the tables referenced in the query is not started. |
| 14 | A column referenced in the query was altered in DB2 after the data is loaded in the Accelerator. |
| 15 | The query uses functionality that is available only in DB2 for z/OS Version 10 new-function mode or later (DB2 10 and above). |
| 16 | The query is not from a package (DB2 9 only). |

- 17 **The query is an INSERT FROM SELECT statement. Subsystem parameter, QUERY_ACCEL_OPTIONS does not specify option 2 to enable the acceleration of INSERT FROM SELECT statements (DB2 10 and above).**
- 18 **The query contains DB2 11 new functionality.**
- 19 **The Accelerator server is not at the correct level and does not support the following function in the SQL statement.**

The EXPLAIN tables can be populated with information described above, even if there is no Accelerator connected to DB2. Specifying EXPLAINONLY on the **START ACCEL** command does not establish any communications with an actual Accelerator, but enables DB2 to consider its presence in the access path selection process. This is the virtual Accelerator feature. It is useful when resource limit facility (RLF) limits the execution of some long-running queries and you want to find out whether these long-running queries qualify for acceleration.

5.8 DB2 Analytics Accelerator hardware considerations

Figure 5-5 shows the linear relationship between the response time and different DB2 Analytics Accelerator models for single query execution. The scan times are in seconds and are estimated times for scanning a 1 TB size table (SALES_FACT).

Table	Creator	Rows (CARDF)	Row length	Size (DASD)	Uncomp.Size	1000-12 scan	1000-6 scan	1000-3 scan
SALES_FACT	GOSLDW	10295.39 M	65	693.32 GB	n/a	77.27	148.01	295.92

Figure 5-5 Linear relationship between response time and N1000 appliance models

From Figure 5-6, you can determine the response time improvement for different models for the N1001. The query response time also varies linearly regarding processing capacity.

Systems and Sizes

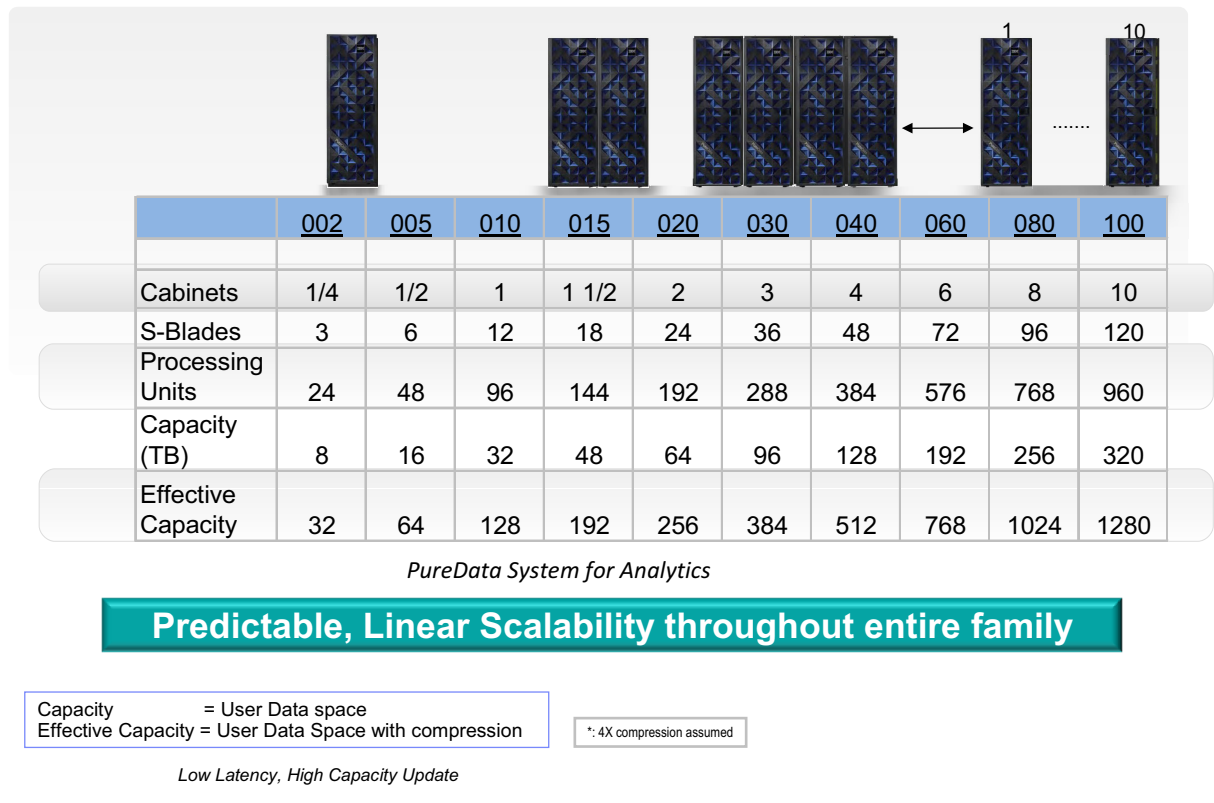


Figure 5-6 Systems and sizes for N1001

More information about configurations and performance for the Accelerator is in Chapter 8, “Impact of new Netezza hardware and software components” on page 187.



High-Performance Storage Saver

In this chapter, we introduce the High-Performance Storage Saver (HPSS) feature and its functional capabilities that allow users to archive partitions of a table or an entire table on the IBM DB2 Analytics Accelerator.

The HPSS feature is designed to further integrate online transaction processing (OLTP) and analytics in a single database system and as an online archive. The IBM DB2 Analytics Accelerator for DB2 for z/OS has been augmented to host archived time-partitioned historic data tables and at the same time, reduce DB2 storage requirements, and improve throughput performance for both non-archived and archived data query workloads.

The following topics are discussed in this chapter:

- ▶ High-Performance Storage Saver design principles
- ▶ Archive process and operations
- ▶ Query processing
- ▶ Monitoring and instrumentation
- ▶ Restore archived partition from the Accelerator

6.1 High-Performance Storage Saver design principles

IBM DB2 Analytics Accelerator (Accelerator) V3.1 can function as a High-Performance Storage Saver (HPSS) online archive. This feature stores static, historical, time-partitioned partitions, or full table data solely on the Accelerator as shown in Figure 6-1. This effectively reduces the host storage requirement and data volume in DB2 and it improves query performance. Subsequently, it removes the requirement for the data to be replicated on both the DB2 for z/OS and the Accelerator storage.

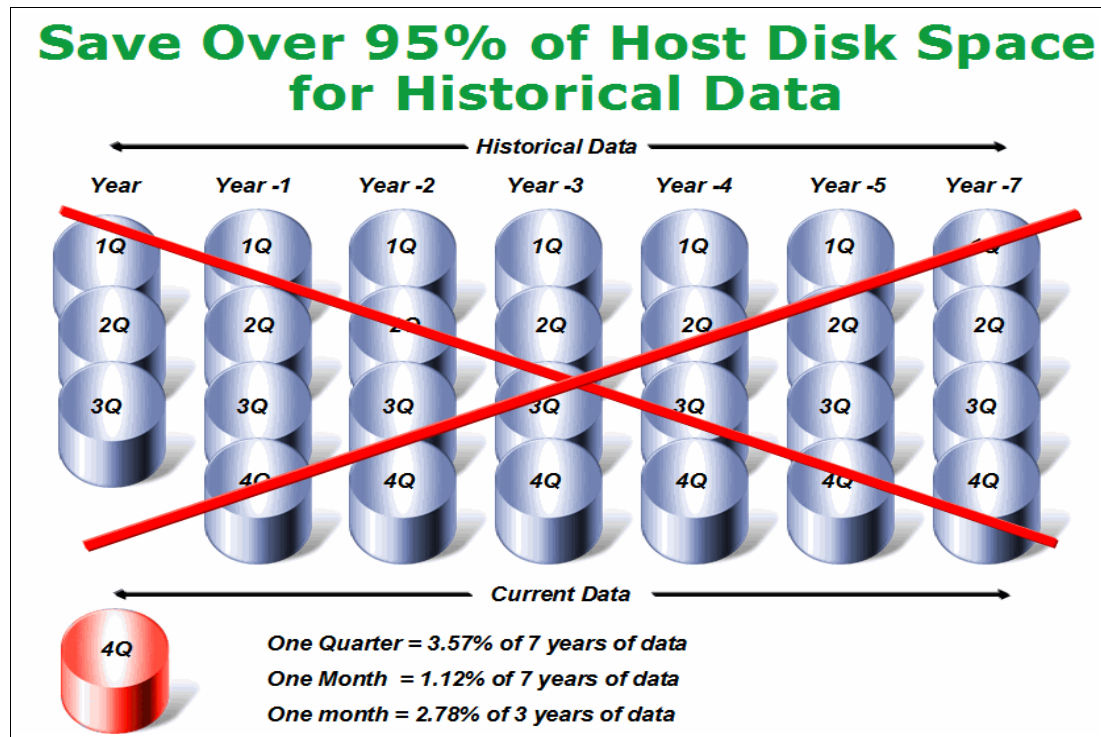


Figure 6-1 95% of host disk space are used for historical data

Tables or partitions can now be divided between traditional database resident partitions on DB2 for z/OS and the Accelerator. This allows users to continue high speed access query data while substantially reducing the overhead in storing and maintaining the data being moved to the Accelerator. Furthermore, there is no need to create a separate data warehouse or extract, transform, and load (ETL), or additional data warehouse system.

The more recent data partitions are typically used with frequent data changes in a transactional context and with short running queries. The entire table is typically used for analytics that are data intensive or to aggregate complex queries.

Today, the typical business analytic query systems are based on data where more than 95% of host disk space is occupied by static historical data. Figure 6-2 on page 145 shows an example of an organization keeping seven years worth of history data. The history data generally is not subject to change or updates regularly, if at all.

Following are the base requirements for HPSS exploitation:

- ▶ IBM DB2 Analytics Accelerator V3.1
- ▶ IBM DB2 10 for z/OS
- ▶ IBM DB2 Analytics Accelerator Studio V3 or later

With HPSS, DB2 for z/OS still has ownership of all data, and all the queries that target the data archived in the Accelerator are now only directed to the Accelerator. This not only provides dramatic storage savings on System z, it also allows the organization to substantially increase the amount of history data to be continuously archived and stored in the Accelerator over time. Figure 6-2 demonstrates that once the data partitions have successfully been archived and committed to the Accelerator, the archived partitions no longer reside in DB2. The Accelerator now hosts both the recent “Active” operational partition as well as the newly “Archive” partitions.

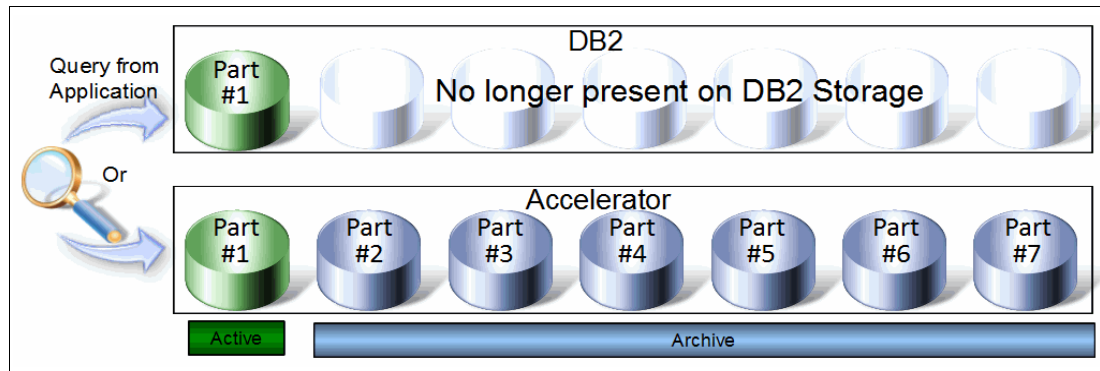


Figure 6-2 Archived partitions reside in the Accelerator and no longer in DB2

6.1.1 HPSS overview

Most, if not all, of the data in an operation data store (ODS) or enterprise data warehouse (EDW) is static. The large tables are partitioned by time and the older partitions rarely, if ever, change.

Currently, there are many database management system (DBMS) vendors providing multi-temperature data solutions. By comparison, they are not the same as the IBM HPSS archive process and its functional capabilities, even though there are similarities. IBM HPSS provides an integrated seamless solution with exceptional performance throughput for analytical query processing against archived data that resides in the Accelerator.

There is no requirement for network-attached external storage with HPSS. The Accelerator provides large disk capacity at a much lower cost by:

- Providing the basis for access to data irrespective of where it resides, either on DB2 or the Accelerator.
- Supporting both transactional and analytical query access patterns.

HPSS is designed to seamlessly integrate into DB2 at SQL level, which provides transparency for any user application accessing the online archive in the Accelerator.

Accelerator: DZA1STPR @ DZA1DDF1

Acceleration:

Stopped [Start](#)

Credentials valid since:

2/21/13 8:29 PM [Update](#)

Status:

Unknown

Trace:

DEFAULT / OFF [Configure](#) [Save](#) [Clear](#)

Used space:

N/A

Active queries:

0

Replication:

Error [Start](#)

Replication latency:

n/a [Show events](#)

▶ Query Statistics

▶ About

▶ Tables (405 of 406 loaded / 405 of 406 enabled for acceleration)

+

 Add...

✎

 Alter Keys...

✖

 Remove

↑

 Load...

⚙

 Acceleration ▼

📦

 Storage Saver ▼

↺

 Replication ▼

Cancel Tasks

☐ Name like:

📄

+

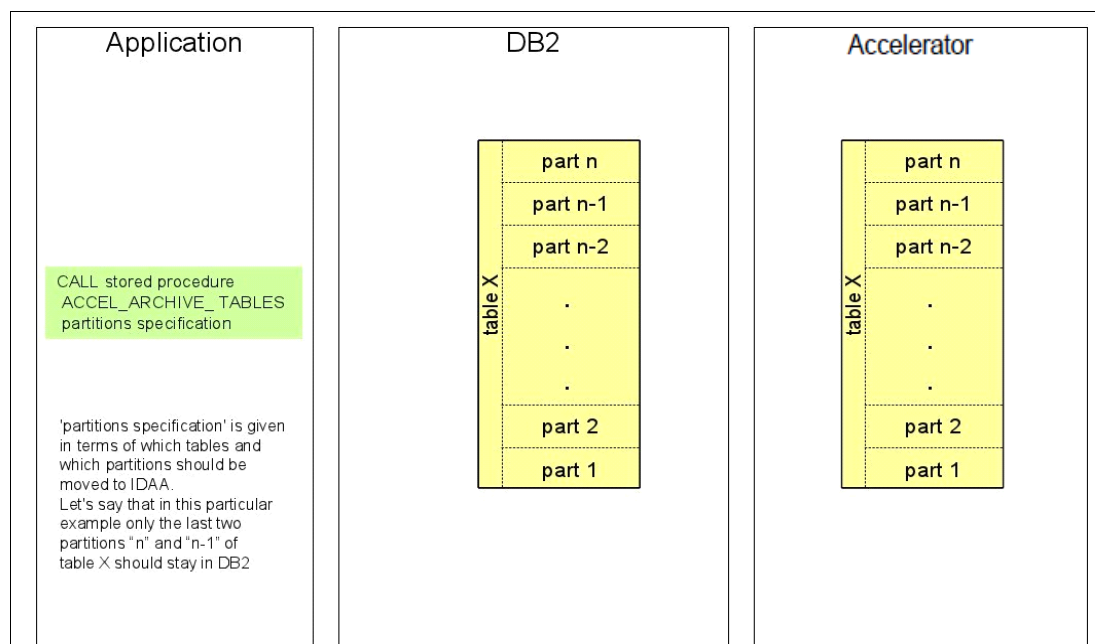
⊖

📦

 Move Partitions to Accelerator...

Name	Size	Rows	Acceleration	Last Load	Storage Saver Partitions	Replication
<div>+</div> <div>📦</div> BID100KB	1.14 GB	-	80 of 80	80 of 80 tables	0 of 80 tables	0 of 80

The new ACCEL_ARCHIVE_TABLES stored procedure is displayed, and you are able to specify the actions that are shown in Figure 6-4.



The supplied stored procedure shown in Figure 6-5 on page 147 illustrates “partitions specification” in terms of which tables and partitions should be moved to the Accelerator. In this example, only the last two partitions, “n” and “n-1” of table x, should stay in DB2.

146 Hybrid Analytics Solution using IBM DB2 Analytics Accelerator for z/OS V3.1

It is possible to archive data of a table that is already loaded in the Accelerator, and to archive data of a table that was just added, but not yet loaded.

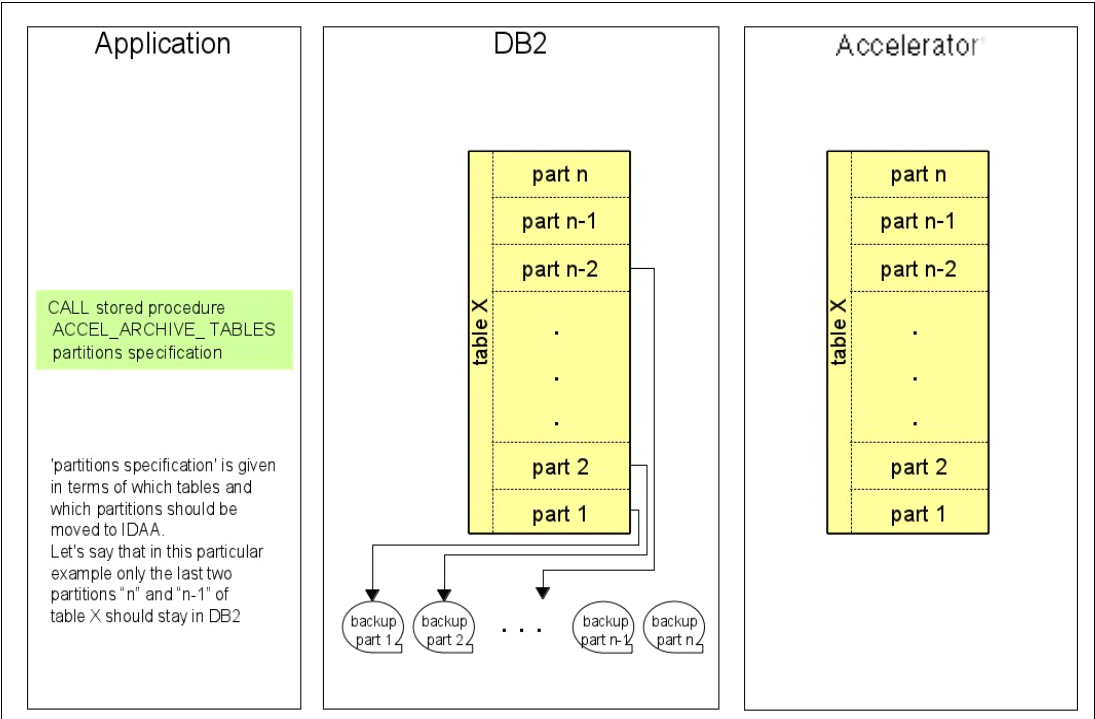


Figure 6-5 HPSS creates new image copies

After the image copy has been created for each of the archive partitions, the data is copied to the Accelerator. Multiple partitions of the same table are archived in parallel. Each partition is committed individually in the Accelerator. As shown in Figure 6-6 on page 148, after the copy process has been completed, the DB2 partitions are pruned in DB2.

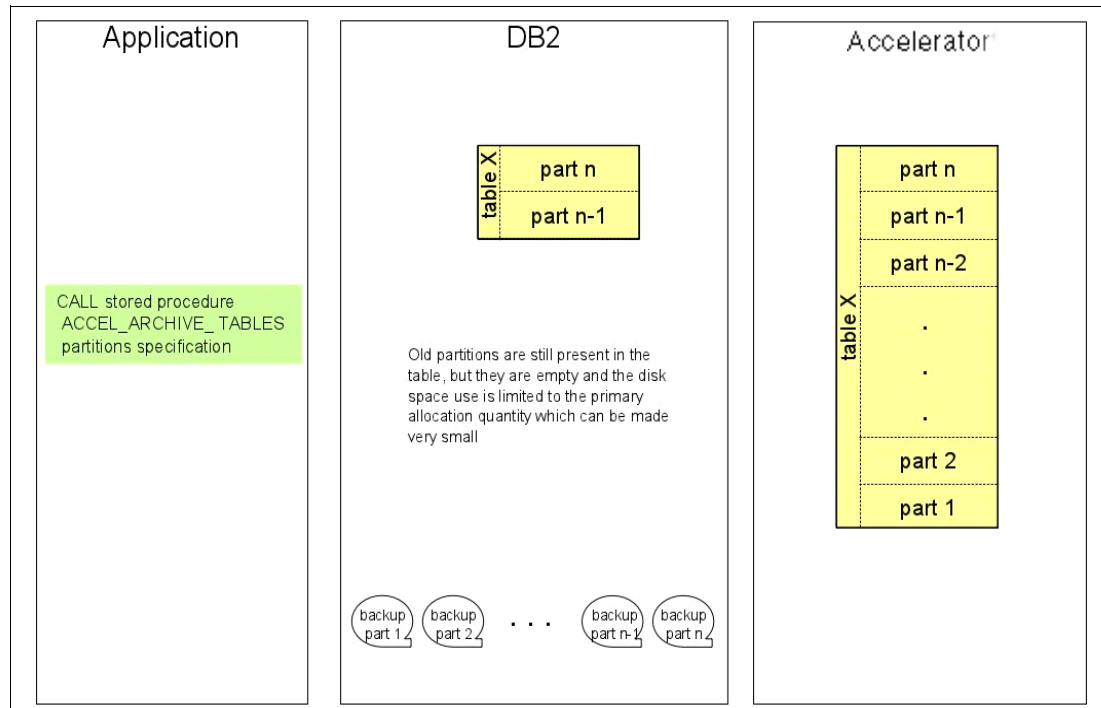


Figure 6-6 Pruning archived partitions in DB2

The **DB2 LOAD REPLACE** utility is used to place an empty data set for each affected partition. The associated data sets remain in DB2 (the schema information that is, partition boundaries for old partitions are empty but still present in the DB2 catalog), and are reduced to primary allocation quantity. Therefore, it is better to define a small primary allocation (PQTY) of the table space before archiving data via an ALTER of PQTY.

For DB2, these archived partitions still exist, they are just without any data. Therefore, SQL statements such as UPDATE or DELETE have no effect on empty partitions. INSERT should not add new rows to empty partitions. Set the archived partitions to the *read-only* state to prevent unintentional data modification.

After the archive operation, the chosen partitions are moved to the Accelerator. However, DB2 for z/OS is still responsible for the recovery and maintains all backup image copies. The active, non-archived data is synchronized by partition refresh or incremental update. Having the most recent partition in both DB2 and the Accelerator provides Accelerator-driven performance acceleration for analytical queries accessing only the most recent partitions. By default, queries access only the data from the most recent partitions. The queries can be executed in DB2 or the Accelerator based on standard query routing criteria. The default can be overridden by DSNZPARM.

The user application needs to specify the query's scope, that is, whether the query accesses only the partitions that reside both in DB2 and the Accelerator, or all the partitions that reside in Accelerator including the archived partitions. The specification is given explicitly by using the special register CURRENT GET_ACCEL_ARCHIVE, or implicitly by inheriting the value specified by the corresponding DSNZPARM GET_ACCEL_ARCHIVE (the default value is NO).

If all the data needs to be accessed for query processing, one of the following mechanisms is setting a DSNZPARM, which activates the “all data” scope for the DB2 data sharing group. This way, no application needs to be changed. Setting the special register CURRENT

GET_ACCEL_ARCHIVE allows switching between the “all data” scope and the “most recent data” scope at any time. The user application can use both scopes within the same execution at the chosen scope at SQL statement level.

6.1.2 Current restrictions

The initial HPSS function implementation imposes several restrictions for partitions/tables that are moved to the Accelerator. Some restrictions currently apply to the Accelerator, which would also apply to HPSS. Following are the details:

- ▶ Range-partition tables only
 - Table-controlled or index-controlled are both acceptable
 - Non-partitioned tables or tables partitioned by growth are not supported
- ▶ Only supported for tables that are not parent in a foreign key relationship
- ▶ Tables that have the column data types that are not supported by the Accelerator cannot be archived via HPSS
- ▶ Support dynamic SQL only. The static SQL queries will operate on the non-archived partitions only
- ▶ HPSS supports partition-level archive only
- ▶ No support for individual partition restores back to DB2
- ▶ No disaster recovery capabilities
 - Not able to move partition data of the same table to multiple/different Accelerators
 - Queries accessing moved data partitions must find all archived data on the same Accelerator because all partitions selected for archive operation must always be moved to the same Accelerator

6.2 Archive process and operations

The first step should be to ensure that your environment has being updated to the supported hardware and software levels, including the IBM DB2 Analytics Accelerator Studio.

ACCEL_ARCHIVE_TABLES stored procedure is designed to handle all the steps necessary in moving selected historical data from DB2 to the Accelerator. This includes the removal of the targeted archive data (partitions and table) from the DB2 table with the maintenance of related indexes. See Figure 6-7 on page 150.

Familiarize yourself with all aspects of HPSS functional capabilities to ensure proper use of this new function. An instance of mistake or improper use of HPSS can result in possible loss of data and integrity exposure.

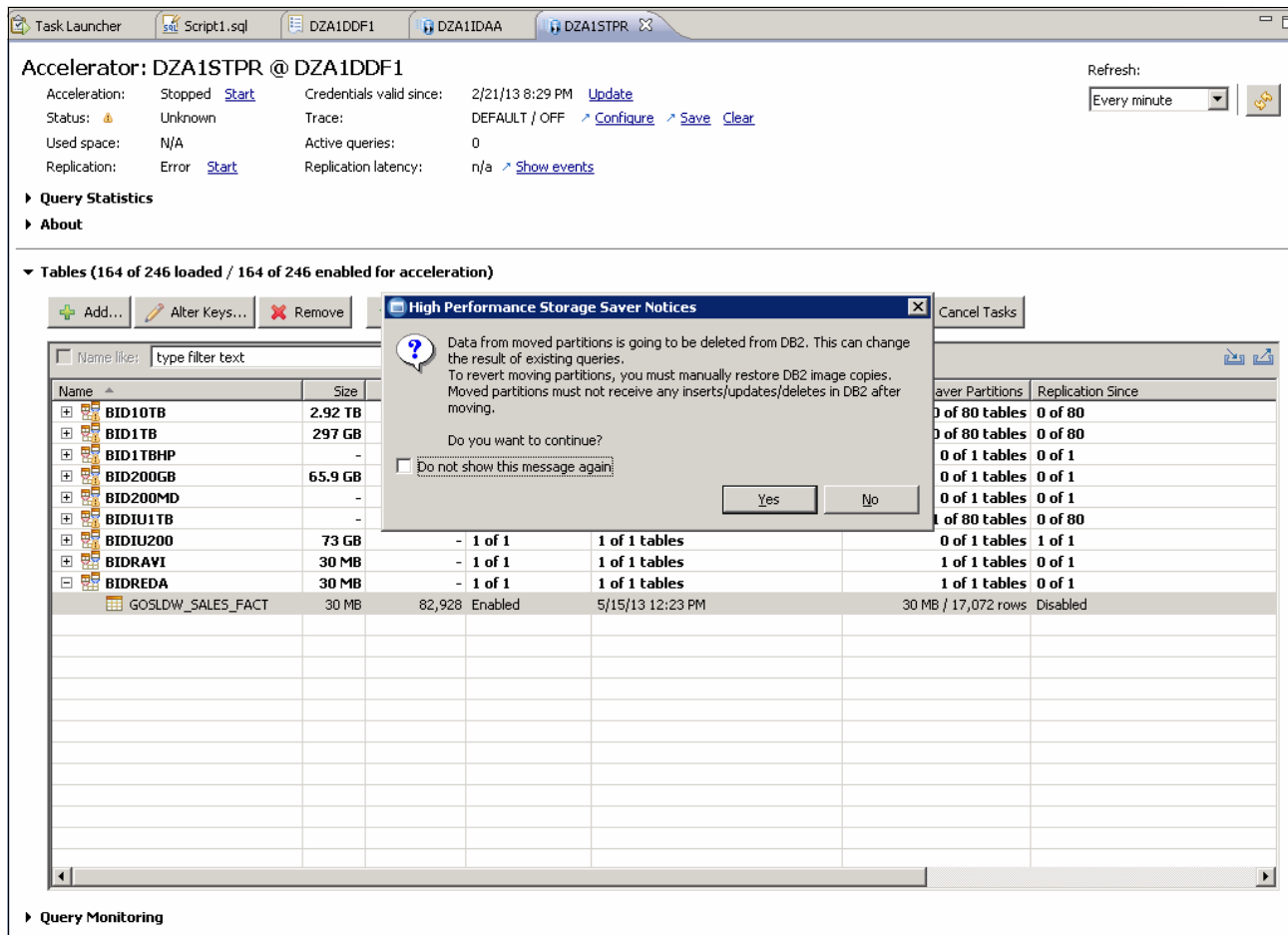


Figure 6-7 HPSS notices: Moved partitions must NOT receive any inserts, updates, or deletes in DB2 after moving

6.2.1 Prepare for archiving partitions and tables

In this section, we look at preparing for HPSS archive function utilization.

The first step is to ensure that the target table, in this case the GOSLDW_SALES_FACT table, has been defined in the Accelerator Data Studio graphical user interface (GUI).

Second, reference this checklist as part of the preparation work:

1. Checking the latest updates on the candidate table

The number of changes that are kept in table SYSIBM.SYSTABLESPACESTATS can therefore be utilized to check if potentially unexpected updates on a table have been performed. Even if no utility has been run against the table, you might still want to run “reinstates” first and check for the counters for a certain period of time to ensure read-only access before archiving the candidate partitions/tables.

The other option is to check the real-time statistic to capture changes to a DB2 table space after the last run of the **Reinstates**, **Reorg**, **Load**, or **Copy** utility.

2. Checking statements in dynamic state cache

It is important to ensure that none of the related SQL statements have any syntax elements that cannot run on the Accelerator. You can check the snapshot of your daily SQL statement from the dynamic statement cache of DB2. Leverage the DB2 Explain

report to detect both updates to the table and syntactical limitations that are preventing query DB2 routing to the Accelerator.

3. Checking static SQL statements

Currently, the Accelerator supports dynamic SQL statements only. When considering specific partitions/tables candidates, it is *important* to ensure that the data is *not* accessed by static SQL statements.

4. The table/partitions must be in “Initial Load Pending” or “Loaded” state

5. Set the partition's table space to read-only

```
START DATABASE(<data-base-name>)
SPACENAM(<tablespace-na,e>) PART(n) ACCES(RO)
```

Consult with the IBM technical support team for assistance, if needed.

Click to select target GOSLDW_SALES_FACT table as shown in Figure 6-8. Then, click the **Storage Saver** tag and pull down to select **Move Partitions to Accelerator...** to initiate archive partition operations, as shown in Figure 6-8.

Accelerator: DZA1STPR @ DZA1DDF1

Acceleration: Stopped [Start](#)

Credentials valid since: 2/21/13 8:29 PM [Update](#)

Status: Unknown

Trace: DEFAULT / OFF [Configure](#) [Save](#) [Clear](#)

Used space: N/A

Active queries: 0

Replication: Error [Start](#)

Replication latency: n/a [Show events](#)

▼ Query Statistics

Statistics collected since accelerator server process was started.

Successful queries: 3,096

Maximum queue wait time: N/A

Maximum number of queries in queue: 111

Failed queries: N/A

Average queue wait time: N/A

► About

▼ Tables (405 of 406 loaded / 405 of 406 enabled for acceleration)

+ Add...

✎ Alter Keys...

✖ Remove

⬆ Load...

⚙ Acceleration ▼

📦 Storage Saver ▼

🔄 Replication ▼

Cancel Tasks

🔍 Name like: type filter text

📄 + -

📦 Move Partitions to Accelerator...

Name	Size	Rows	Acceleration	Last Load	Storage Saver Partitions	Replication Since
BID100KB	1.14 GB	-	80 of 80	80 of 80 tables	0 of 80 tables	0 of 80
BID10TB	2.92 TB	-	80 of 80	80 of 80 tables	0 of 80 tables	0 of 80
BID1TB	297 GB	-	80 of 80	80 of 80 tables	0 of 80 tables	0 of 80
BID1TBHP	-	-	0 of 1	0 of 1 tables	0 of 1 tables	0 of 1
BID200GB	65.7 GB	-	1 of 1	1 of 1 tables	0 of 1 tables	0 of 1
BID200MD	57.3 GB	-	1 of 1	1 of 1 tables	0 of 1 tables	0 of 1
BIDIU1TB	329 GB	-	80 of 80	80 of 80 tables	0 of 80 tables	1 of 80
BIDIU200	77.4 GB	-	80 of 80	80 of 80 tables	0 of 80 tables	0 of 80
BIDRAVI	30 MB	-	1 of 1	1 of 1 tables	1 of 1 tables	0 of 1
BIDRAVIK	30 MB	-	1 of 1	1 of 1 tables	0 of 1 tables	0 of 1
BIDREDA	30 MB	-	1 of 1	1 of 1 tables	1 of 1 tables	0 of 1
GOSLDW_SALES_FACT	30 MB	82,928	Enabled	5/15/13 12:23 PM	30 MB / 17,072 rows	Disabled

Figure 6-8 Storage Saver: Move Partitions to Accelerator function

As shown in Figure 6-9 and Figure 6-10 on page 153, the Accelerator Data Studio “Move Selected Partitions” window displays all partitions belonging to the GOSLDW_SALES_FACT table. For demonstration purposes, we checked and selected partitions 8 - 16 to be archived to the Accelerator.

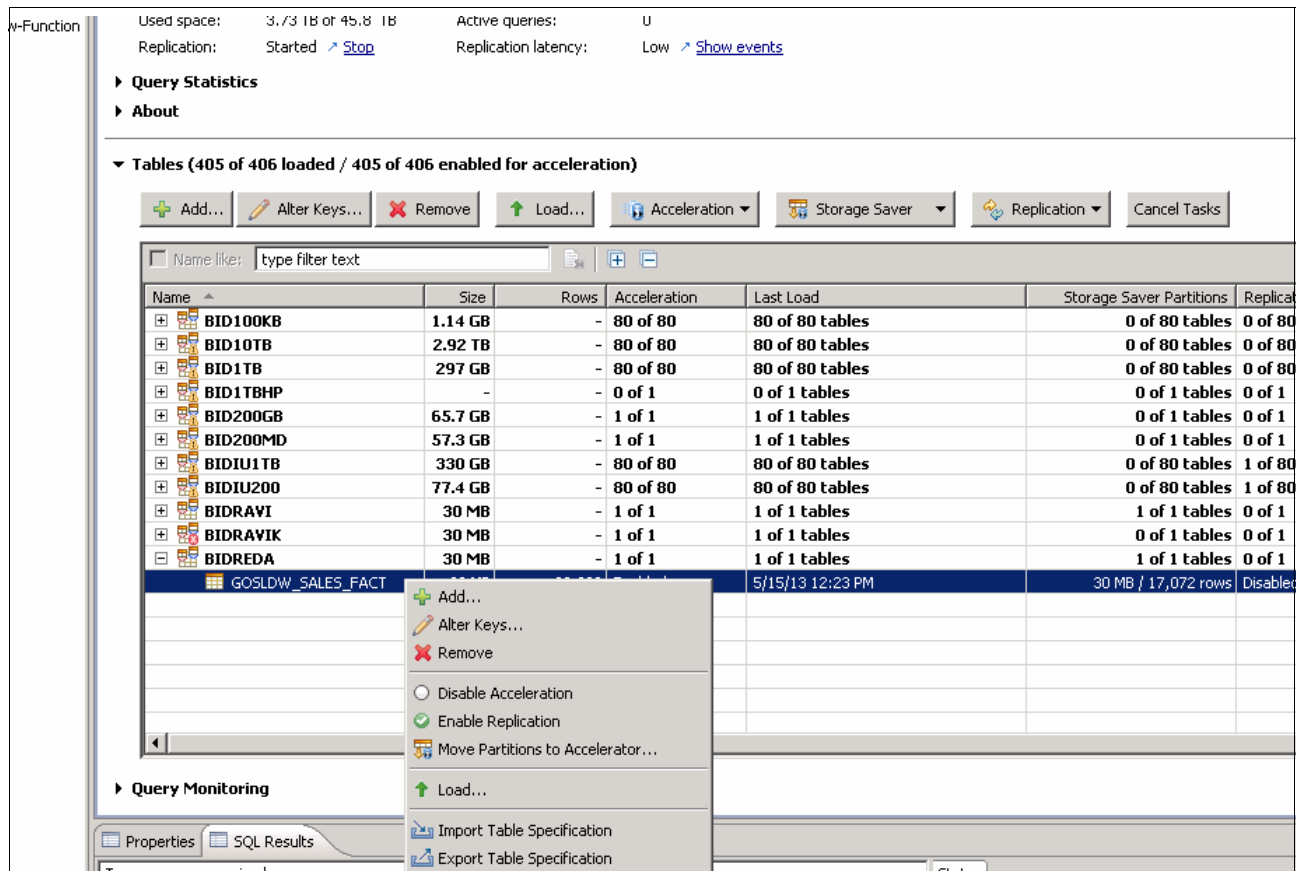


Figure 6-9 New Storage Saver tag enables Move Partitions to Accelerator operation

The other option is to right-click the highlighted GOSLDW_SALES_FACT table and select **Move Partitions to Accelerator...**

In this example, we choose to manually select partitions. The other option is to move all partitions up to and including the limit key ending at 20041001.

Manually selecting partitions 8 - 16 and clicking **OK** starts the “Move Partitions Job”. Reference Figure 6-10 for moving partitions in progress.

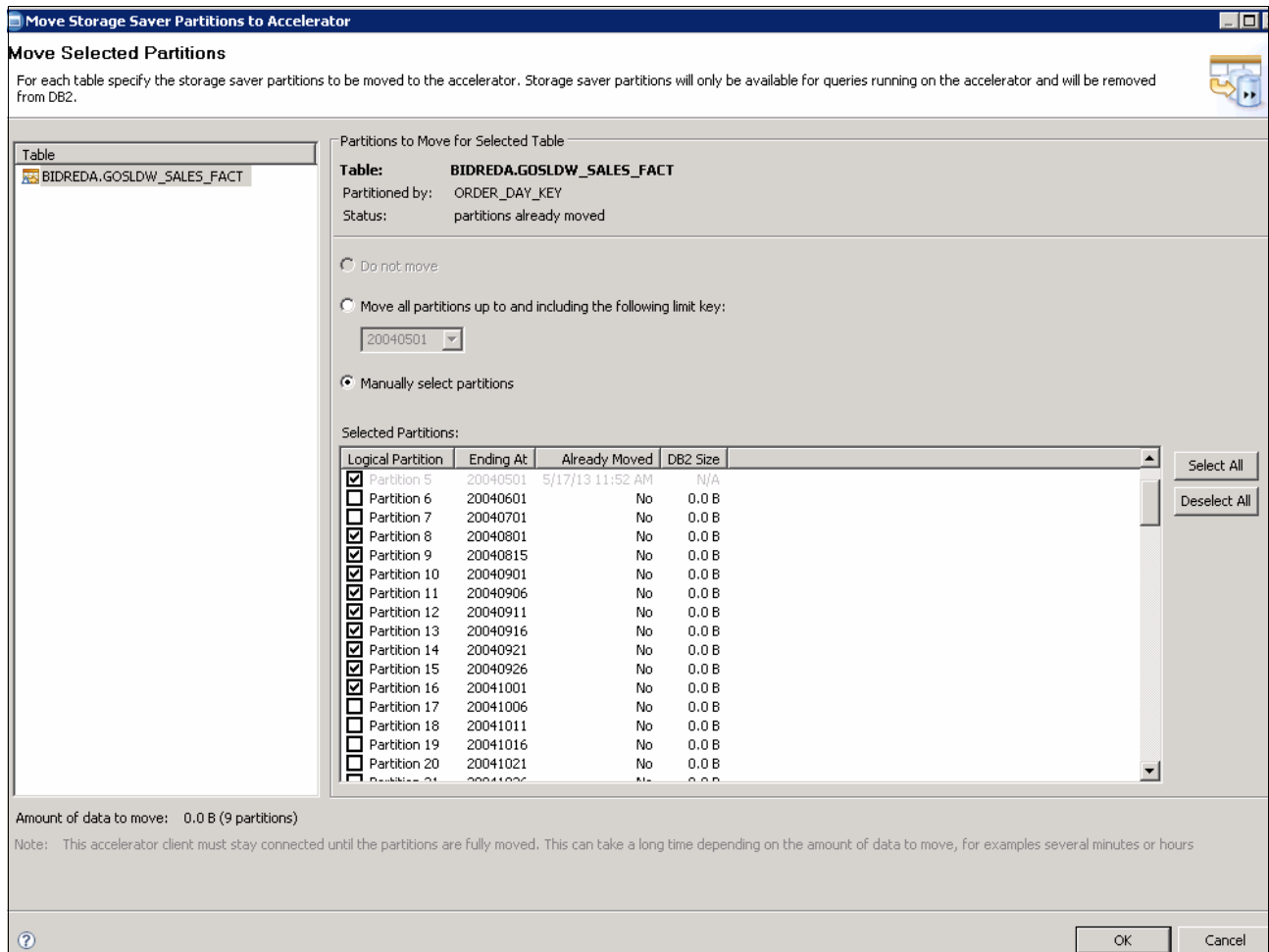


Figure 6-10 Partitions 8 - 16 have been selected for archive into the Accelerator

In Figure 6-11, “Move Selected Partitions” indicates for each table the storage saver partitions to be moved to the Accelerator. The archived storage saver partitions are only available for queries running on the Accelerator. In addition, the Accelerator Data Studio must stay connected until the partitions are fully moved.

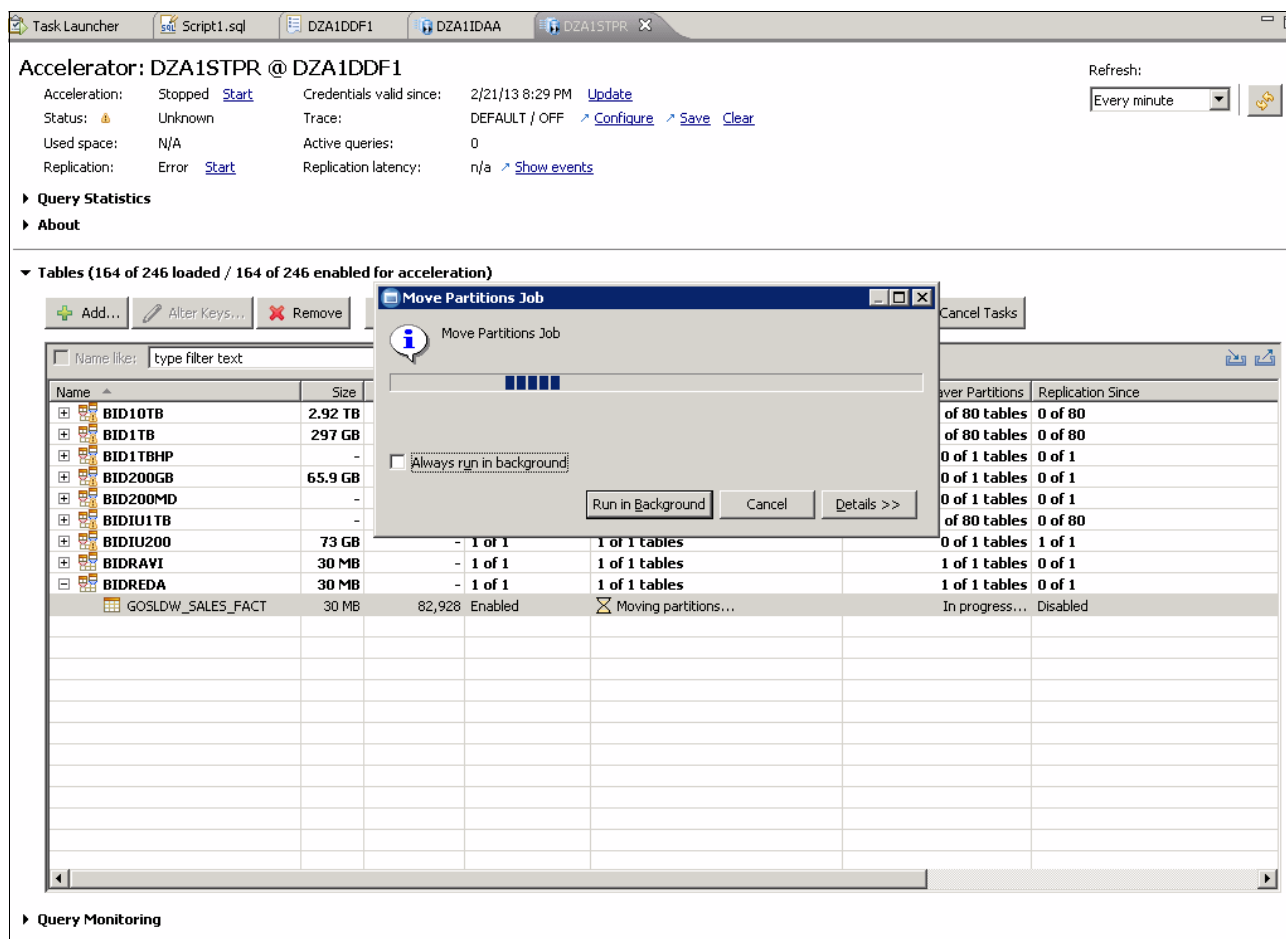


Figure 6-11 Archive: Move partition in progress

Once you have selected and launched the data partitions for archive to Accelerator, there are multiple processing steps being carried out automatically behind the scene during the archive operation. Following are the processing steps:

- ▶ Lock partition
 - This guarantees consistent backup image copies and data transfer to the Accelerator
 - The table partition in shared mode to prevent concurrent updates on the data
- ▶ Automatically creates a new image copy of the partition via the DB2 **COPY** utility:
 - Data set name: <HLQ>.AQT.<dbname>.<tsname>.P<4-digit part number>
 - The high-level qualifier (HLQ) identifies the location of the backup data sets to be placed
 - HLQ can be influenced via the **AQTENV** environment variable file
- ▶ Copy the data from the partition to the Accelerator using the **UNLOAD** utility
- ▶ Commit the archive data in the Accelerator and unlock partition

- Remove the partition in DB2 using the LOAD REPLACE utility with an empty input data set
There will be no DB2 logs written with the execution of the **LOAD REPLACE** utility
- Enable archive flag in SYSACCEL.SYSACCELERATEDTABLES, if needed
- Multiple partitions of the same table are archived in parallel

As shown in Figure 6-12, the “Load Tables/Partitions”, partitions 8 - 16 (between partition 7 and partition 17) are no longer part of the GOSLDW_SALES_FACT table nor available for archive.

Load Tables / Partitions
Select the whole table or partitions to be loaded from DB2 for z/OS to the accelerator.

Name	Partitioned By	Load Recommended	DB2 Size	Schema
<input checked="" type="checkbox"/> GOSLDW_SALES_FACT	ORDER_DAY_KEY	No - data unchanged	29 MB	BIDREDA
<input type="checkbox"/> Partition 6	Ending at '20040601'	No - data unchanged	0.0 B	
<input type="checkbox"/> Partition 7	Ending at '20040701'	No - data unchanged	0.0 B	
<input checked="" type="checkbox"/> Partition 17	Ending at '20041006'	No - data unchanged	0.0 B	
<input checked="" type="checkbox"/> Partition 18	Ending at '20041011'	No - data unchanged	0.0 B	
<input checked="" type="checkbox"/> Partition 19	Ending at '20041016'	No - data unchanged	0.0 B	
<input checked="" type="checkbox"/> Partition 20	Ending at '20041021'	No - data unchanged	0.0 B	
<input checked="" type="checkbox"/> Partition 21	Ending at '20041026'	No - data unchanged	0.0 B	
<input checked="" type="checkbox"/> Partition 22	Ending at '20041101'	No - data unchanged	0.0 B	
<input checked="" type="checkbox"/> Partition 23	Ending at '20041106'	No - data unchanged	0.0 B	
<input checked="" type="checkbox"/> Partition 24	Ending at '20041111'	No - data unchanged	0.0 B	
<input checked="" type="checkbox"/> Partition 25	Ending at '20041116'	No - data unchanged	0.0 B	
<input checked="" type="checkbox"/> Partition 26	Ending at '20041121'	No - data unchanged	0.0 B	
<input checked="" type="checkbox"/> Partition 27	Ending at '20041126'	No - data unchanged	0.0 B	
<input checked="" type="checkbox"/> Partition 28	Ending at '20041201'	No - data unchanged	0.0 B	
<input checked="" type="checkbox"/> Partition 29	Ending at '20041206'	No - data unchanged	0.0 B	
<input checked="" type="checkbox"/> Partition 30	Ending at '20041211'	No - data unchanged	0.0 B	
<input checked="" type="checkbox"/> Partition 31	Ending at '20041216'	No - data unchanged	0.0 B	
<input checked="" type="checkbox"/> Partition 32	Ending at '20041221'	No - data unchanged	0.0 B	
<input checked="" type="checkbox"/> Partition 33	Ending at '20041226'	No - data unchanged	0.0 B	
<input checked="" type="checkbox"/> Partition 34	Ending at '20050101'	No - data unchanged	0.0 B	
<input checked="" type="checkbox"/> Partition 35	Ending at '20050201'	No - data unchanged	0.0 B	
<input checked="" type="checkbox"/> Partition 36	Ending at '20050301'	No - data unchanged	0.0 B	
<input checked="" type="checkbox"/> Partition 37	Ending at '20050401'	No - data unchanged	0.0 B	

☐ Lock DB2 tables while loading: All Tables
☒ After the load, enable acceleration for disabled tables.
 Amount of data to load: 0.0 B
 Note: This accelerator client must stay connected until the tables are fully loaded. This can take a long time depending on the amount of data to load, for example several minutes or hours.

Buttons: Select All, Deselect All, Select Recommended, Expand All, Collapse All, Next: Unknown, Show Recommended: All, OK, Cancel

Figure 6-12 Partitions 8 - 16 have been loaded into the Accelerator

Note: Incremental update can be used on any table with archived partitions. However, incremental update has to be stopped and resumed after the partitions' archive operations have completed.

Ensure proper interaction between incremental update Change Data Capture (CDC) and HPSS and ONUTILITYACTION=IGNORE setting.

ONUTILITYACTION=... specifies the behavior of InfoSphere CDC for z/OS after detecting that a DB2 utility (such as LOAD or RECOVER) has run on a table space containing source tables that are being mirrored. InfoSphere CDC for z/OS will determine if the actions might have resulted in the source tables being out of sync with their target tables. This keyword provides configured options for resynchronization.

6.2.2 Batch stored procedure invocation

For users that prefer batch submit over Accelerator Data Studio GUI, a direct invocation of the ACCEL_ARCHIVE_TABLES stored procedure can be used to achieve the same result using SQL directly. The XML document to be passed to the stored procedure only has to be adjusted to use the proper information.

In Figure 6-13, the stored procedure would archive partitions 8 - 16, just the same as though it were being done under the Accelerator Data Studio GUI.

```
<dwa:tableSetForArchiving xmlns:dwa="http://www.ibm.com/xmlns/prod/dwa/2011"
  version="1.0"> <table schema="BIDREDA" name="GOSLDW_SALES_FACT">
    <partitions>8:16</partitions>
  </table> </dwa:tableSetForArchiving>
```

Figure 6-13 Archive store procedure: Partitions 8 - 16

When the initial archive processing has been completed for the target BIDREDA.GOSLDW_SALES_FACT table partitions 8 - 16, the user can add and archive an additional partition or partitions to the Accelerator continuously on a regular basis. The example in Figure 6-14 shows the process of archiving the additional partition 17 to the Accelerator. If a partition or partitions has already been archived before, the partition is automatically skipped because it is already in the Accelerator. In addition, the HPSS online archive is at the partition level and not at the row and column level.

```
<dwa:tableSetForArchiving xmlns:dwa="http://www.ibm.com/xmlns/prod/dwa/2011"
  version="1.0"> <table schema="BIDREDA" name="GOSLDW_SALES_FACT">
    <partitions>8:16</partitions>
  </table> </dwa:tableSetForArchiving>
```

Figure 6-14 Archiving the additional partition 17 to the Accelerator

In addition, you can automate the partition archive operations by leveraging the ACCEL_ARCHIVE_TABLES stored procedure using the SQL interfaces. The XML document is used to archive all partitions of the target table excluding the last 10 partitions, as shown in Figure 6-15.

```
<dwa:tableSetForArchiving xmlns:dwa="http://www.ibm.com/xmlns/prod/dwa/2011"
  version="1.0"> <table schema="BIDREDA" name="GOSLDW_SALES_FACT">
  <partitions>8:16</partitions>
</table> </dwa:tableSetForArchiving>
```

Figure 6-15 Archiving selected groups of partitions

6.3 Query processing

With the introduction of HPSS online archive, for each query entering DB2, the *DB2 query optimizer* looks at query processing against recent, non-archived partitions, or archived partitions. For every query queued for processing, DB2 query optimizer checks if the following conditions are met:

- ▶ Query acceleration is enabled
- ▶ All tables referenced by the query have been loaded on the Accelerator
- ▶ The query is qualified and meets the routing criteria
Supported query constructs (same routing criteria apply to the Accelerator)
- ▶ Same heuristics decisions, optimal query access plan, and cost-based optimization for query routing apply

6.3.1 Query execution on the Accelerator with HPSS online archive

It is your choice whether to include archive data for query processing. This is accomplished by setting the special register CURRENT GET_ACCEL_ARCHIVE in the application, or it can be implicitly inherited from the value specified by the corresponding DSNZPARM GET_ACCEL_ARCHIVE with a default value of NO:

If CURRENT GET_ACCEL_ARCHIVE = YES

- ▶ With archive, the query can only run in the Accelerator
- ▶ DB2 optimizer heuristics check and query routing rule do not apply in this case

With HPSS, the query can be routed and needs to access the online archive data. DB2 rewrites the query to combine the active partitions (on a table on the DB2 side) with the archived partitions (which reside in their table on the Accelerator) via the UNION ALL SQL operator.

Figure 6-16 illustrates query acceleration with and without the HPSS online archive.

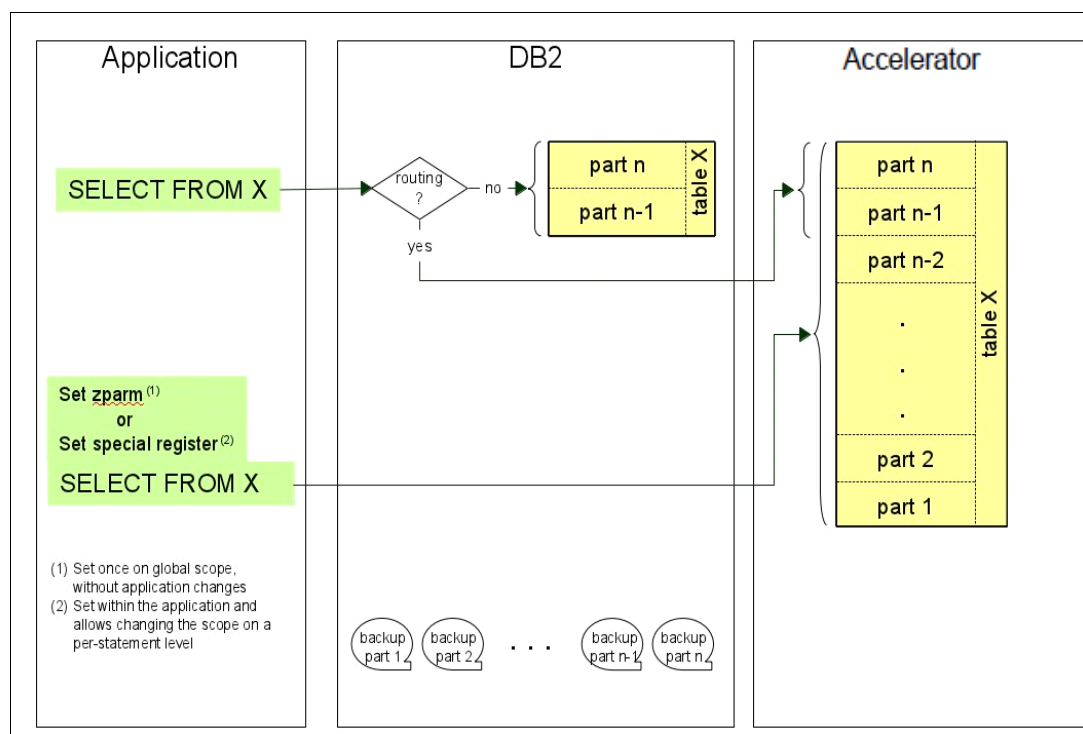


Figure 6-16 Query execution with and without archive data

6.3.2 Query execution on Accelerator without HPSS online archive

By default, queries run on the Accelerator against the non-archived, active operational data, with:

- ▶ `CURRENT GET_ACCEL_ARCHIVE = NO`
 - Non-archived only, even if archived data (partitions) for the same tables are present on the Accelerator
 - Follow the same DB2 optimizer heuristics checks and query routing rule

6.3.3 Query execution on DB2 without HPSS online archive

When access to the HPSS online archive data is not required, queries can be executed in DB2 just the same as before the introduction of the HPSS feature.

Depending on the query characteristics, if it contains unsupported expressions, non-eligible queries, or accessing non-accelerated tables, the DB2 optimizer may choose to do so because of heuristics, query routing rule, cost optimization, and index-backed queries.

In addition, the Accelerator table `SYSACCEL.SYSACCELERATEDTABLES` contains a new `CHAR(1)` column with the name, `ARCHIVE` (added by DB2 10 APAR PM72274):

- ▶ A - Table is archived in the Accelerator and the corresponding Accelerator contains both active and archived data
- ▶ C - Table is archived in another Accelerator and the corresponding Accelerator contains the active (current) data only
- ▶ Blank: Table is not archived in the IBM DB2 Analytics Accelerator

With HPSS, query processing needs to consider whether the query should include the archived data (partitions) of the archived tables. For non-archived, active operational data, the DB2 query optimizer will decide whether the query should be executed natively in DB2 or be routed to the Accelerator by following the existing query routing criteria and decision.

6.3.4 Query acceleration controls

System parameter GET_ACCEL_ARCHIVE special register:

- ▶ Value NO (default)
 - Specifies when a table is archived in the Accelerator
 - The table reference does not include the archived data
- ▶ Value YES
 - Specifies when a table is archived in the Accelerator
 - The table reference includes the archived data
 - Query can run *only* in the Accelerator

Special register CURRENT GET_ACCEL_ARCHIVE:

Can be set implicitly by inheriting the value of the system parameter, or explicitly with the SET CURRENT GET_ACCEL_ARCHIVE for each individual SQL statement.

Note: DB2 for z/OS is the only point of access and administration of data, this includes backup and recovery strategies regardless of where the data is physically located. The DB2 optimizer component determines the optimal query access plan and cost-based optimization for query routing.

6.4 Monitoring and instrumentation

For monitoring purposes, users can look at the ACCEL GET_TABLES_INFO and ACCEL GET_TABLES_DETAILS stored procedures that have been extended to return and echo necessary corresponding archived partitions/table information. The source information and statistics are gathered by the Accelerator at the following levels:

- ▶ Table level:
 - Archived data volume in the Accelerator by the number of rows and bytes
- ▶ Partition level:
 - Timestamp on when a partition was archived
 - How much data was archived from DB2 to the Accelerator
 - The name of the backup data set

See Chapter 9, “Monitoring DB2 Analytics Accelerator environments” on page 207.

6.5 Restore archived partition from the Accelerator

The restore operations are a set of administrative manual steps that are necessary in restoring data, rebuilding index, and support structures in DB2. It will restore one or more archived partitions from the Accelerator back into DB2. We understand that historical and archived data typically will not be updated and most likely will not be targeted for query processing. Therefore, they are being backed up and stored separately from the more recent, active operational data.

However, there are circumstances where it might become necessary to restore the archived partitions/tables back into DB2. On the Accelerator, the archived data is directly moved back into the shadow table for the non-archived data and the corresponding catalog information is updated.

Note: HPSS supports restore of *all* archive data and unregisters the table from the Accelerator.



Incremental update

To accelerate queries, the IBM DB2 Analytics Accelerator (Accelerator) must have access to the data of interest. This, naturally, requires that data must be copied from the DB2 table to corresponding tables defined to the Accelerator.

There must be an initial population of the data, which is accomplished by using the batch load process. After a table has been populated initially, the data needs to be updated periodically to keep in synch with the data in the source DB2 table. There are two methods: batch refreshing, and incremental update (or trickle feed).

Chapter 3, “Data latency management” on page 55 introduces and compares both the batch and incremental update methods. It describes in detail the batch method.

In this chapter, we describe the incremental update method in relation to the Accelerator. We go into some detail about the inner workings of incremental update and look at the behavior of a few scenarios.

The following topics are covered:

- ▶ Overview of incremental update
- ▶ Incremental update architecture
- ▶ Installation and configuration
- ▶ Defining incremental update between a DB2 subsystem and the Accelerator
- ▶ Defining tables to incremental update
- ▶ Access server
- ▶ z/OS capture agent
- ▶ Accelerator target agent
- ▶ Incremental update scenarios
- ▶ Summary

7.1 Overview of incremental update

There are cases in which the batch method of refreshing data from DB2 to the Accelerator is not quite adequate. Some business processes must have access to the latest data. In some cases, that need might be driven by technical reasons such as having an extremely small batch window in which there is not enough time for a batch refresh.

Many companies are building real-time business intelligence systems. These systems need low end-to-end latency, taking data from operational systems to analytic data store for supporting reporting, dashboards, and predictive analytics. The most effective way to accomplish this is to use a change propagation system. Change propagation systems capture data changes occurring in the operational system, typically from the database log. These changes are then continuously fed to the analytic systems and applied to those data stores, such as DB2, asynchronously in near real-time. This implies then that the Accelerator must also be kept in synch with the updates that are occurring in those DB2 tables.

There is a case where businesses need to do reporting and some level of analytics on operational data. The complexity and processing time of ETL systems to move and transform the data for analytic usage might not be acceptable. Using the Accelerator to offload analytics makes analytics possible directly on operational data with minimal to no impact to the operational applications. However, a major characteristic of operational systems is that it is a dynamic environment with many data changes occurring. These changes need to be reflected in the Accelerator in near real-time.

There are cases in which change propagation may be used for a purely technical reason. For example, if the number of changes to a large DB2 table occur to a small subset of the total number of rows in a table, or if changes occur across many table partitions, it could be more effective to use incremental update to copy those rows to the Accelerator compared to the time for a full refresh. This might be particularly helpful where there is already a tight batch processing window.

Another technical reason for using incremental update to copy data from DB2 to the Accelerator is for high-availability reasons. It is possible to apply the changes from one capture of changes from DB2 and apply those changes to multiple Accelerators. If an Accelerator fails, DB2 can detect this and use the surviving Accelerators for rerouting queries. For more information, see Chapter 11, “High availability considerations” on page 285.

Incremental update is the change propagation function of the IBM DB2 Analytics Accelerator and became generally available with V3. It is based on, and is a special use of, the existing Change Data Capture (CDC) component of IBM InfoSphere Data Replication V10.1.3. It is deeply integrated into the IBM DB2 Analytics Accelerator product and managed using the Accelerator Studio. It is an included feature with IBM DB2 Analytics Accelerator V3.

All required CDC components are included with the Accelerator:

- ▶ The capture agent is IBM InfoSphere Change Data Capture for DB2 for z/OS.
- ▶ The target agent is IBM InfoSphere Change Data Capture for IBM Netezza Linux x86.
- ▶ The management interface component is the IBM InfoSphere Change Data Capture Access Server for Linux x86.

7.2 Incremental update architecture

Incremental update consists of individual components on both System z and the Accelerator. These components work together to define and manage the entire process of capturing data changes from DB2 tables and applying those changes to tables in the Accelerator. The goal is to accomplish this data synchronization in near real-time.

Figure 7-1 shows the high-level architecture of the incremental update feature.

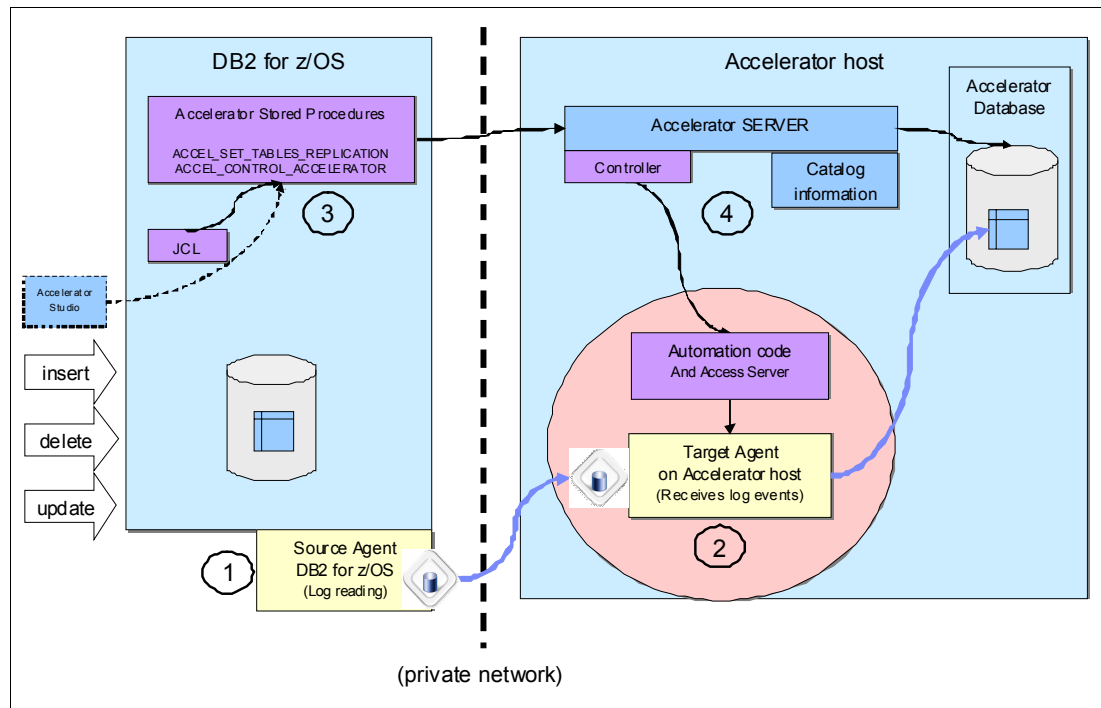


Figure 7-1 Incremental update architecture

There is a source agent running as a z/OS started task, Circle 1 in Figure 7-1. The source agent is responsible for monitoring the DB2 log for changes to specified tables. As these changes are found, they are staged in memory until a commit record is seen.

The changes for the complete unit of work are sent over the network to the target agent, Circle 2, running in the Accelerator host. The target agent then stages the changes in its staging area until such time that the apply process updates the target tables with those changes.

Of course, the source and target agents need to know which tables are participating in incremental update. Functions in the Accelerator Studio are used to select the tables that will be updated using incremental update. See Circle 3 in Figure 7-1.

Explicit calls to Accelerator stored procedures can also be used. By way of a controller function running on the Accelerator, these requests are turned into API calls to the CDC management component, Access Server, Circle 4. Access Server is responsible for setting up the definitions to make the source and target agents aware of what to replicate.

After the definitions are made, the table has to be loaded. This creates an initial capture point. After the capture point has been created, changes are replicated from the source DB2 tables to the target Accelerator tables. This causes a near real-time synchronization of data. Notice

that the actual change propagation process is under the control of the included CDC components and is independent of the normal Accelerator processes.

The Accelerator incremental update feature is a specific use of the IBM CDC product. A simple 1:1 mapping with no transformations is used and no user exits are involved. As all management functions are wrapped into the Accelerator Studio and Accelerator stored procedures, the CDC Management Console is not used or provided. Because the Accelerator is an appliance, connectivity information to directly connect to the Access Server is not exposed.

The architecture is optimized for throughput. During normal operations, there is no disk I/O involved. Captured changes flow directly from the DB2 log buffers to the source agent capture staging area in memory. They are held there until a unit of work completes. If it is a rollback, the changes are simply discarded. After the source agent receives a commit, the changes for that unit of work are sent over the network to the target agent. The target agent stores the changes in the apply staging area in memory. Changes are consolidated on the target, resulting in a single change to a row. To optimize for the Accelerator platform, UPDATES are changed into DELETE, INSERT pairs. Changes are applied in mini-batches to make effective use of the Accelerator bulk processing capability. The target applies all units of work that arrive in a window that uses an algorithm that is based on time, typically 60 seconds, and a minimum size.

Incremental update leverages the normal Accelerator load functionality to do the initial load that establishes the capture point. This load process is optimized for the bulk moving data from DB2 to the Accelerator. For information about the Accelerator load process, see Chapter 3, “Data latency management” on page 55.

As changes are applied to the Accelerator table, the data tends to get disorganized over time. The Accelerator monitors the disorganization. When a threshold of disorganization is reached, data maintenance operations are automatically scheduled. The operations run in the background and have minimal impact on running queries.

If you are familiar with CDC terminology, there is one *subscription* that is defined for a DB2 subsystem and Accelerator pair. This subscription is automatically defined when the Accelerator is set up for incremental update using the Accelerator administration console. When tables are enabled for incremental update using the Accelerator Studio, mappings are defined for that subscription.

In addition to the incremental update management functions, other incremental update functions are also deeply integrated into the Accelerator. All the incremental update components are integrated into the Accelerator trace collection and trace management facilities. The Accelerator side incremental update components are integrated into the Accelerator fail-over concept. If the active Accelerator host fails, the secondary host will take over and restart all incremental update components. Incremental update errors and warning events are reported to the z/OS operator console using the DSNX881I message ID.

Monitoring information that is related to the incremental update functionality is available via the Accelerator accounting records. Maintenance of the incremental update components on the Accelerator is fully integrated into the Accelerator software update procedure. Updates are provided in packages that are loaded into the unformatted system service (USS) file system on System z. By using the Accelerator Studio, those updates are pushed across the private network to the Accelerator and applied to the target Accelerator table.

7.3 Installation and configuration

Installation of the incremental update System z components is accomplished using normal SMP/E installation and maintenance techniques while the Accelerator components are integrated into the Accelerator installation and upgrade processes.

For a description about how to install InfoSphere CDC for z/OS on a z/OS system in a source or target environment, see “Installing or upgrading InfoSphere CDC for z/OS” at the following site:

http://pic.dhe.ibm.com/infocenter/iidr/v10r2m0/index.jsp?topic=%2Fcom.ibm.cdc.doc.homepage.doc%2Fic-homepage_iidr.html

7.3.1 Installing the System z components for incremental update

The capture agent component on z/OS is IBM InfoSphere Change Data Capture for z/OS V6.5 (FMID HCHC650). This component is not part of the integrated installation because it resides on z/OS. It is installed and maintained using normal SMP/E installation processes. An existing installation of V6.5 of the CDC capture agent can be used. However, be aware that some of the configuration values, especially the defaults, might not be suitable for using the Accelerator incremental update feature. Example 7-1 shows the suggested values as defined in the *IBM DB2 Analytics Accelerator for z/OS Version 3.1.0 Installation Guide*, SH12-6983. See this guide for further information about these parameters.

Example 7-1 Suggested configuration values

```
CHCCFGxx
CONFIG PALCLEANUPTIME=23:59,
      TIMEZONE=<YOUR TIMEZONE AS LISTED IN SCHCDATA.CHCTMZON>,
      ADMININACTTIMEOUT=15,
      AUTORESTARTINTERVAL=2
CHCDBMxx
DB2 SSID=<YOUR DB2 SSID>,
  PLANSUFFIX=<YOUR DEFINED SUFFIX>,
  ONUTILITYACTION=IDLE,
  ONSCHEMACHANGE=IDLE,
  ADDCOLUMNISSCHEMACHANGE=YES,
  LOGPOLLINTERVAL=(3,ALWAYS),
  LOGCACHEDELAY=5,
  CACHEBLOCKSIZE=2,
  CACHELEVEL1RESERVED=160,
  CACHELEVEL1SIZE=200
```

Important: Use the setting ONUTILITYACTION=IGNORE if you use the High Performance Storage Saver (HPSS) to move partitions.

Attention: With this configuration, the CDC STC can allocate up to 2.2 GB (2 GB staging space + 200 MB L1 cache) of z/OS storage resources (above the bar storage) that can be paged to auxiliary storage. Ensure that there are sufficient system resources for this configuration; paging can significantly reduce replication throughput.

Enable DATA CAPTURE CHANGES on the SYSIBM.SYSTABLES catalog table. This helps CDC to detect most Data Definition Language (DDL) changes on tables that are involved in incremental update, also called *in-scope tables*. If it is not enabled, CDC will be aware of DDL changes only when a log record is encountered that does not match the definition in the CDC metadata. The action that DB2 takes depends on the type of DDL operation and the version of DB2. For details, see the information center, “Understanding the behavior of InfoSphere CDC for z/OS when DDL changes occur in the InfoSphere CDC for z/OS” at the following site:

<http://pic.dhe.ibm.com/infocenter/cdc/v6r5m1/index.jsp?topic=%2Fcom.ibm.cdc.doc.cdcforzos.doc%2Fconcepts%2FunderstandingthebehaviorwhenDDLchangesoccur.html>

For setting of the CDC parameters, also see the IBM Technote: Preventing outages of the incremental update process, at the following site:

<http://www.ibm.com/support/docview.wss?uid=swg27039147>

7.3.2 Installing the Accelerator components for incremental update

The initial installation of the components for incremental update is installed by the IBM installation personnel during the initial installation of the Accelerator. However, maintenance updates to the incremental update components are integrated into the maintenance process for the base Accelerator software. This consists of:

- ▶ The components for incremental update are delivered by the normal program temporary fix (PTF) delivery mechanism. The SMP/E installation procedure places Accelerator update packages, including the incremental update packages for the Access Server and Replication Engine, in a USS HFS file system. The directory is specified in the **AQT_INSTALL_PREFIX** environment variable that is found in <HLQSP>.SAQTSAMP(AQTENV). The default directory is usr/lpp/aqt/packages.
- ▶ The Accelerator Studio is used to transfer the packages to the Accelerator. Studio copies the packages from the HFS directory across the network to the Accelerator.
- ▶ The Accelerator Studio is used to activate the components. This accomplishes the installation of the selected package on the Accelerator.
- ▶ The transfer and activation of packages can also be done by an explicit call to the Accelerator stored procedure SYSPROC.ACCEL_UPDATE_SOFTWARE.

7.4 Defining incremental update between a DB2 subsystem and the Accelerator

After installing the capture agent on z/OS, you must still enable incremental updates (we also use the term, *replication*) for the DB2 subsystems from which you want to capture changes. This allows the incremental update components on the Accelerator to communicate with the source agent on z/OS. Of course, the DB2 subsystem and the Accelerator must have already gone through the pairing process.

In CDC terminology, there is one CDC subscription definition for a DB2 subsystem and Accelerator pair. All table mappings for a DB2 subsystem are a part of this subscription.

Note: In V3 of the IBM DB2 Analytics Accelerator, there is a limit of two DB2 subsystems that can be enabled for incremental updates for any one Accelerator system.

The following information needs to be obtained:

- ▶ The name of the DB2 subsystem.
- ▶ The IP address of the source agent on z/OS.
- ▶ The port that is assigned to the source agent on z/OS. This is defined by using the **SERVICENAME** environment variable that is found in the configuration member <CHCHLQ>.SCHCDATA(CHCCMMxx).
- ▶ The DB2 credentials used for incremental update. The user ID must have SELECT authorization for all tables that are to be enabled for incremental update. It must also have ALTER authority so DATA CAPTURE CHANGE can be defined on the table, which is done automatically when a table is enabled for incremental update. It is suggested to use a non-expiring password to avoid having incremental fail when the password expires. The DB2 credentials can manually be changed via the Configuration Console.

The Accelerator Configuration Console is used to enable and disable incremental updates for a DB2 subsystem, and to update the DB2 subsystem connection credentials. An administrator connects to the Configuration Console, selects the Enable incremental update function, and enters the preceding information at the prompts to connect the Accelerator and DB2 for incremental updates. Example 7-2 shows a sample dialog using the Configuration Console to enable an incremental update for a DB2 subsystem.

Example 7-2 Configuration Console dialog for enabling incremental update

```
Enter password (in TSO, use PF3 to hide input):
Licensed Materials - Property of IBM
5697-DAA
(C) Copyright IBM Corp. 2009, 2013.
US Government Users Restricted Rights -
Use, duplication or disclosure restricted by GSA ADP Schedule
Contract with IBM Corporation

*****
* Welcome to the IBM DB2 Analytics Accelerator Configuration Console
*****

You have the following options:
(1) - Change the Configuration Console Password
(2) - (Menu) Run Netezza Commands
(3) - (Menu) Run Accelerator Functions
(4) - (Menu) Manage Incremental Updates
-----
(x) - Exit the Configuration Console
4

main -> 4
-----
You have the following options:

(0) - Go back one level
(1) - Enable incremental updates
(2) - Disable incremental updates
(3) - Update DB2 subsystem credentials

(Default 0) > 1
```

```
Select the DB2 subsystem to be enabled for Replication
Database system DZA1DDF(state=STARTED) is already configured for replication
Using the only applicable database system DBZZDDF
Enter the Capture Agent IP address on z/OS ('' or '0' to exit): 135.25.80.5
Enter the Capture Agent TCP port: (Default 5999) > 5999
Enter the DB2 UserID for replication ('' or '0' to exit): CHCUSER
Enter the password (in TSO, use PF3 to hide input):
Enter the password again to confirm (in TSO, use PF3 to hide input):
Press 'y' to configure DB2 Subsystem 'DBZZDDF' for replication: Y
```

7.5 Defining tables to incremental update

Incremental update management functions are done by using the Accelerator Studio graphical user interface or using explicit calls to Accelerator stored procedures. Incremental update can be started and stopped for the DB2 subsystem. Tables can be enabled for incremental update or removed from incremental update. In the Accelerator Studio, the status of incremental update is displayed as seen in Figure 7-2 on page 169. The number of source captures and target updates are displayed in a pop-up message and an indicator of latency is displayed as Low/Medium/High. For more information about the Accelerator Studio, see Chapter 2, “Using the Studio client to manage the environment” on page 11.

To enable a table for incremental update, the table must first be added to the Accelerator. Then, it is just a matter of highlighting one or more tables in the table list and selecting **Enable Replication**. The underlying table is ALTERed to set DATA CAPTURE CHANGE on the table and the table is set to INITIAL_LOAD_PENDING.

After a table has been enabled for replication, a capture point must be established. The capture point is established by doing a load of the table. The capture point of the table is reflected in the Tables section of the Accelerator Studio as seen in Figure 7-2 on page 169.

Establishing a capture point is enforced by the Accelerator. This enforcement is done by setting the Accelerator table status to the INITIAL_LOAD_PENDING state at the time a table is enabled for replication. The normal Accelerator load function is used to accomplish the initial load that sets the capture point. To prevent in-flight changes on tables that are involved in incremental updates, the source DB2 tables must be locked. Lock mode TABLE (single table) or TABLESET (multiple tables) is used on the source DB2 tables during the unload process to prevent changes that occur while establishing the capture point.

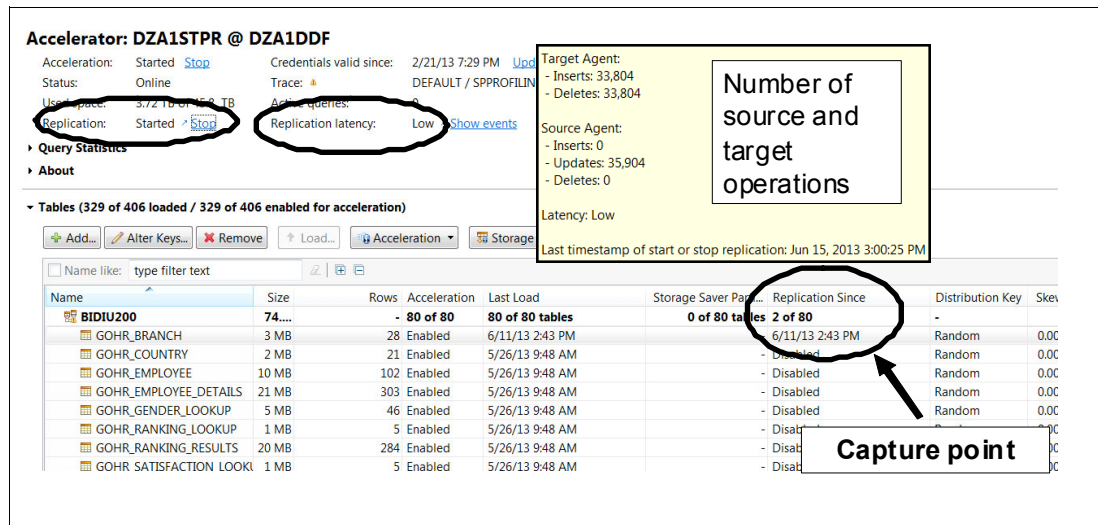


Figure 7-2 Capture point reflected in the Replication Since column of the Accelerator Studio

Hint: If a newly added table is to be enabled for incremental update, do not do a load until after the Enable for Replication action. If you add a table and then load it before enabling for replication, the Accelerator table status will still be set to INITIAL_LOAD_PENDING and the table will need to be loaded again.

Enabling replication on an existing Accelerator table will reset that table to INITIAL_LOAD_PENDING and a load table will need to be performed to establish the capture point. It is also possible to reset the capture point for a table that is already participating in incremental update. This might be necessary because of non-logged operations on the source DB2 table. Either a full table reload or partition reload, assuming partition reload caught all the changes, is supported. Replication will have to be stopped to do the reload. Replication will then continue from the newly established capture point.

7.6 Access server

The access server is the CDC component that is responsible for the management of incremental update and runs on the Accelerator host. It manages the subscriptions and table mappings for the Accelerator. Definitions are stored in a metadata store on the Accelerator. It communicates with the source and target agents via TCP/IP to create these definitions. It also starts and stops incremental update with DB2 subsystems.

All incremental update management functions are integrated into the Accelerator, as discussed in section 7.5, “Defining tables to incremental update” on page 168. The Accelerator Studio, or Accelerator stored procedure calls, communicates with Accelerator processes that are running on the Accelerator host. These requests are translated to calls to the access server via its application programming interface, as seen in Figure 7-1 on page 163. The access server then performs the requested action, communicating with the source agent and target agent as necessary.

When a DB2 subsystem is enabled for incremental update, a call to the Access Server is made to create a subscription for that DB2 subsystem and Accelerator pair. Only one subscription is used for the pair. See section 7.4, “Defining incremental update between a DB2 subsystem and the Accelerator” on page 166 for more information.

7.7 z/OS capture agent

The capture agent on System z runs as a started task or a batch job. There is one capture agent for each DB2 subsystem or data sharing group. The capture agent reads data from the DB2 z/OS log, sends the changes for a captured unit of work to the appropriate target agent, responds to requests from the Access Server, and responds to commands entered from the z/OS console. In the case of data sharing groups, DB2 handles retrieving log records from the other members.

7.7.1 Reading the DB2 log

The source agent uses the DB2 for z/OS instrumentation facility interface (IFI) to access the DB2 log. Log records for all tables that have DATA CAPTURE CHANGES configured are read. The source agent log reading process filters the records to capture only the records for the tables that are defined for it to capture.

Upon startup, the source agent will start a DB2 trace to output log records to a DB2 online performance (OP) buffer. The size of this buffer is specified using the **BUFSIZE** keyword of the CHCDBMxx CDC configuration member. This defines the size of the OP buffer in a range 8 - 1024 KB in 4 KB increments. When the source agent requests log records, IFI attempts to find the log record in the OP buffer before it resorts to reading the log file. If the system is very active in terms of data changes being captured, then consider increasing this to increase the chance of a buffer hit.

CDC uses an asynchronous method to read DB2 log data via IFI:

1. The CDC source agent issues an asynchronous read (READA) to IFI.
2. When a certain buffer utilization threshold is reached, the CDC source agent will be notified.
3. Establishes a new DB2 log reading position where it last detected that no more DB2 log data could be read.
4. Acquires DB2 log data one record at a time passing it to other CDC tasks for processing.
5. Continues reading until there is no more new DB2 log data to read.
6. The CDC source agent issues another asynchronous read to IFI.

The amount of time that the source agent waits varies depending on two other CDC configuration parameters, **BUFTHRESHOLD** and **LOGPOLLINTERVAL**. Both of these keywords are specified in the CHCDBMxx configuration member.

The **BUFTHRESHOLD** keyword specifies the percentage of the OP buffer that must contain data before IFI notifies the CDC source agent. Recall that the **BUFSIZE** keyword specifies the size of the OP buffer. By default, **BUFTHRESHOLD** is 1%. Every time that the CDC source agent reaches the end of the log, a new asynchronous read is issued, and the CDC source agent waits until the **BUFTHRESHOLD** is reached and IFI notifies the CDC source agent. During very active periods, this may seldom happen and therefore there will be continuous reading of the log. On the other extreme, at times of very low change activity the **BUFTHRESHOLD** might not be reached for a long time or conceivably never be reached. Those data changes might end up having an extremely high latency or might not be captured until a higher level of activity is reached.

LOGPOLLINTERVAL determines the maximum amount of time, in seconds, that the CDC source agent waits for a notification from IFI. If this interval is reached, an unprompted read of the log is performed. This can help when activity is very low, which causes the

BUFTHRESHOLD not to be reached in a desirable time frame. The required configuration for the Accelerator is LOGPOLLINTERVAL=(3,ALWAYS) for non-sharing and data sharing mode to allow the waitForReplication accelerator feature to work correctly.

If you are replicating changes to more than one Accelerator from one DB2 subsystem, consider configuring a CDC log cache. Each Accelerator that is paired for incremental update with the DB2 subsystem has a separate subscription. Each subscription uses the IFI directly to read log records that require additional work for DB2 to satisfy all the requests for data. When a CDC log cache is configured, data is read once from IFI and retained in the cache. Each subscription attempts to read from the log cache. If the data is not in the cache, only then will the subscription call DB2 IFI to read the log.

The log cache consists of a level 1 cache in memory that is backed by a level 2 cache in a VSAM data set. CDC job CHCCRCCH creates the VSAM file. Keywords **CACHEBLOCKSIZE**, **CACHELEVEL1SIZE**, and **CACHELEVEL1RESERVED** must all have non-zero values to enable the log cache. Implement a log cache that covers one day of DB2 log entries as described at:

<http://www.ibm.com/support/docview.wss?uid=swg27039147>

7.7.2 Staging changes

As log records are read, they are filtered to select only the log records for tables that are associated with the subscription, or DB2 subsystem and Accelerator pair. These records are stored in an in-memory staging area and organized into unit of recovery (UR) commit groups. When a COMMIT record is found for a commit group, the changes for that commit group are sent over the network to the CDC target agent and the commit group removed from the staging area. If a ROLLBACK is found for a commit group, the changes are simply removed from the staging area.

The staging area memory resides above the bar. Each active subscription has its own staging area. Each subscription acquires 1 MB of storage when it starts and then acquires more storage as needed to hold incomplete commit groups. When storage is no longer required, it is released. As a staging area nears capacity a warning message, CHC2532W, is issued. Seeing these warnings is an indication that the maximum size should be increased. If the staging reaches the maximum size, an error message, CHC1524E, is generated and incremental update ends for that subscription.

The maximum size of a subscription's staging area is specified using the keyword, **MAXSUBSCRSTAGESIZE** in the CHCDBMxx configuration member. The default is 2 GB. Care must be taken to ensure that other memory specifications are set appropriately. The **STG64LIMIT** keyword of the CHCCFGxx configuration member defines the maximum amount of above the bar storage that can be used by all components of CDC. In the context of incremental update, this needs to be set to MAXSUBSCRSTAGESIZE times the number of subscriptions that can be active at any one time. Care must be taken to specify a MEMLIMIT value in the CDC JCL large enough to accommodate this and ensure that z/OS has sufficient auxiliary storage available.

7.7.3 Capture agent considerations

The amount of System z processor that the capture agent consumes depends on the amount of changed data activity present in the DB2 subsystem in terms of number of megabytes that the capture agent has to process. The capture agent must read every change record in the DB2 log to determine which changes are relevant to the Incremental Update subscriptions. Changes that are for tables that are defined to incremental update for replication to the Accelerator are called *in-scope changes*. All others are called *out-of-scope changes* and are

most likely the result of having another replication product causing DB2 to capture changes for other tables. The processor cost of reading out-of-scope changes could have an impact on the overall cost of incremental update if there is a significant number of out-of-scope changes.

If LOGPOLLINTERVAL is set in low change activity scenarios, this might cause additional processor usage. Frequently waking the capture agent to process just a few or perhaps no changes will incur some amount of overhead each time, which could add up over time. When change activity is high, there is little to no impact to using LOGPOLLINTERVAL as the capture agent stays busy. If this is a concern, the cost of this overhead must be balanced with the business need to reduce latency during low change activity periods.

7.8 Accelerator target agent

The CDC target agent runs on an Accelerator host. Its function is to take the changes sent from the source agent, stage them in memory, and apply the changes to the target tables. There is one target agent per Accelerator. The starting and stopping of the target agent is automatically managed by the Accelerator. The CDC components, both access server and the target agent, are integrated into the Accelerator's fail-over concept. If the active Accelerator host fails, the secondary host takes over and restarts all incremental update-related components.

Changes are applied to the target tables in mini-batches to take advantage of the Accelerator's bulk handling capabilities, and occur approximately every 60 seconds.

7.8.1 Staging changes

As commit groups are transmitted from the source agent to the target agent, they are staged in memory until such time that the apply process runs. As changes are staged, a couple of things happen to help make the apply more efficient. Any UPDATES from the source agent are decomposed into DELETE, INSERT pairs. Bulk inserts and deletes are more efficient than doing updates against the target table. In addition, multiple changes to a row are consolidated into one change with just the latest values.

7.8.2 Applying changes

The target agent applies the changes from the target staging area using mini-batches at approximately 60-second intervals that are modified somewhat by reaching a minimum size threshold. All commit groups that arrived during the interval are applied in bulk to the target tables.

Since updates are converted into deletes and inserts, the apply process does only inserts and deletes. First, a bulk delete is done qualifying on the rows in the cache. This scans the table and flags those rows as deleted. They are not physically deleted until cleanup maintenance task runs. Following the bulk deletes, a bulk insert is done to the table to insert all the new rows from the staging area. Then, the changes are committed.

Figure 7-3 on page 173 shows a typical pattern of capture activity at the source agent and the periodic apply activity taking place at the target. The x-axis is elapsed time and y-axis is the rate of activity in operations per second. Operations for the source include inserts, updates, and deletes being captured from the DB2 source tables. Operations for the target include inserts and deletes being applied to the Accelerator tables.

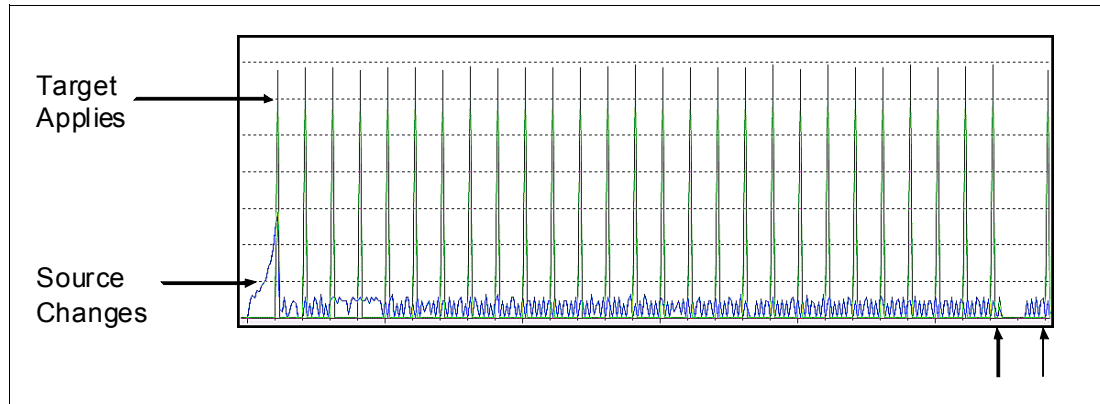


Figure 7-3 Graphic representing pattern of source change captures and target apply changes

The lower line is the changes being captured by the source agent and represents the number of changes per second. Although there is some variability in the change capture rate, over time it is somewhat continuous. The upper line is the rate at which the target agent applies those changes. Notice that the interval, which is 60 seconds, between the apply peaks is consistent except in the lower right corner. It might be difficult to see but at the first arrow, there is a small peak where the agent applied some changes just a few seconds after the 60-second interval and just before there was a quiet period of no source changes. After some time, source changes began again and about 60 seconds after that, the target agent applied those changes.

7.8.3 About delete processing

The Accelerator was architected as a high-performance query engine without the need to create performance artifacts such as indexes and summary tables. It does this using a divide and conquer method of highly parallelized and special hardware, each working on a small slice of data. This results in being able to scan data at extremely high speeds.

Unfortunately, this does have an effect on operations that operate on individual rows, for example, updates and deletes such as those found in incremental update scenarios. There are no indexes to help locate the rows, which result in scanning the table to find the row. That leads to the design point of applying changes in mini-batches by the incremental update function of the Accelerator. Because a scan would most likely be required, why not do many operations during that scan. Decomposing updates into deletes and inserts eliminates the need to do a scan to update rows.

Inserts are very efficient and are generally not a concern for the performance of incremental update processing. Because deletes require a scan of the table, the performance of a delete operation is similar to an I/O bound query. Performance is related to how fast the data can be scanned to locate the rows that need to be marked as deleted. Data that results in smaller data slices that are spread across many worker nodes will have much better scan performance than data that has large data slices that are spread across fewer nodes.

The performance of delete processing is a major contributor to the overall performance of incremental update processing. If the scan of an Accelerator table takes a long time, this could cause the apply agent to fail to keep up with the rate of changes coming from the source agent causing latency to increase.

It is worth noting that delete processing does not cause rows to be physically removed. Delete processing in the Accelerator just flags the rows as deleted. The Accelerator will run a maintenance task to physically remove the row after a certain level of disorganization is

reached. This task runs in the background at a low priority and has minimal impact on query processing.

A simple rough method to understand delete performance is to perform a `SELECT COUNT_BIG(*)` on an accelerated table. The query might need to be forced to the Accelerator by using the `SET CURRENT QUERY ACCELERATION ALL` statement. This forces a complete scan of the Accelerator table and the time it takes to complete is roughly the time to perform the delete process of incremental update. This time in consideration with the scan times of all tables participating in incremental update should give some idea if there might be a negative impact to incremental update processing. Just as in query processing, delete processing time can be decreased by increasing the number of worker nodes such that the data is spread across the nodes in smaller slices.

The way that data is distributed across work nodes also has an impact on scan speed. The minimum time that it takes to scan a table is related to the size of the largest data slice. If a distribution key results in uneven distribution of data, this could have a negative impact on scan speed and therefore a negative impact on delete performance.

The use of an organizing key can have a positive impact on scan and therefore delete performance. Specifying an organization key caused the data on a disk to be ordered by that key. For each data block on disk, there is a zone map that keeps track of the minimum and maximum value for every mappable field in that block. The zone map is looked at first to determine if that data block needs to be accessed. Incremental update delete processing is qualified by the primary key of the source DB2 table if one exists. If the Accelerator table has an organization key that is the same as the DB2 primary key, a significant number of data blocks could be skipped, thereby increasing the delete performance.

It is suggested to have a primary key, or a unique constraint, on a unique index for another reason as well. If there is a unique constraint on the source DB2 table, the delete processing is qualified on the columns in the primary key. If there is no uniqueness, delete processing is qualified on all the columns of the table. This is fine in the case where one or more columns make up a logical unique key. If there is no unique identifying information, more rows could potentially be deleted than intended because all rows with that set of values across all columns will be deleted.

7.8.4 About latency

Latency is measured at the amount of time that passes between the time the data changes on a DB2 table and the time it changes on the Accelerator table. Latency depends on a number of factors:

- ▶ The time from when a change is made and the commit. Only after a commit is received can the change be sent to the target system.
- ▶ The time before a target agent wakes up and starts reading the log. In a busy system, this is probably minimal. In a system with a lower rate of change, this could be significant depending on the environment variables as discussed in 7.7.1, “Reading the DB2 log” on page 170.
- ▶ The time of transmission across the private network between the System z and the Accelerator. Unless the network is extremely busy due to concurrent load operations, this should be minimal.
- ▶ The time that the changes stay in the target staging area before the target agent applies the changes to the target database. At the Accelerator, this happens approximate every 60 seconds and is probably the largest factor in latency time for replicating to the Accelerator.

The latency of any individual change is mostly due to the timing of when it arrives at the Accelerator and when the Accelerator target agent runs. If the system is running well, this could vary from a few milliseconds to upwards of 60 seconds.

If the source and the target are on different systems, as in the case of the Accelerator incremental update, the accuracy of the latency calculation depends on how well the clocks between the two systems are synchronized.

The clock on the Accelerator is synchronized with a selected DB2 subsystem and is selected using the Accelerator Configuration Console. Time synchronization occurs during query operations. If the Accelerator goes for an extended period of low or no query activity, the clocks could drift apart, therefore affecting the latency calculation. If an unexpectedly high latency value is observed even when there is no change activity, it is usually not an issue. The clocks will again be synchronized when query activity resumes.

An indicator of current average latency is displayed in a pop-up of Accelerator Studio, as shown in Figure 7-2 on page 169, and discussed in Chapter 2, "Using the Studio client to manage the environment" on page 11. The latency is indicated by the values *low*, *medium*, and *high*. The latency can also be obtained via a call to the Accelerator stored procedure `SYSPROC.ACCEL_CONTROL_ACCELERATOR` by using the function `getAcceleratorInfo`. Information regarding the status of replication will be in the `replicationInfo` section of the result message.

Example 7-3 shows the `replicationInfo` section resulting from the `getAcceleratorInfo` action. The average latency at this point is 38 seconds. There have been 610,380 updates to the source tables. At this point, 491,384 of those updates have been applied; 491,384 inserts and 491,384 deletes to the target tables.

Example 7-3 <replicationInfo> section of result message from the getAcceleratorInfo function

```
<replicationInfo state="STARTED" lastChangeTimestamp="2013-06-15T20:00:25.949697Z"
latencyInSeconds="38" activeAccessServerVersion="6.5.1663.0"
activeReplicationEngineVersion="6.5.2 InterimFix10">
  <sourceAgent insertCount="0" updateCount="610380" deleteCount="0" />
  <targetAgent insertCount="491384" updateCount="0" deleteCount="491384" />
</replicationInfo>
```

The chart on the left side of Figure 7-4 on page 176 shows average latency during a 5-minute test of updates against a DB2 table that is being replicated to the Accelerator. There were 1,711,120 updates to a 1 TB DB2 table over a period of 5 minutes and 5 seconds for an average of 5,610 updates per second. There is an average latency of zero until changes are first applied to the target table. The average latency for the first set of changes is 36 seconds. If we could look at the latency of each individual row, we might find that the first row that was updated in DB2 had a latency of maybe a minute or even more, while the last row in that set might have had a latency of milliseconds.

Notice that the average latency stays the same for approximately 1 minute. That is because with the mini-batch nature of incremental update, there are no changes being applied during that time interval, therefore no changes in latency. The average latency of the next mini-batch drops to 20 seconds, and so on, until the last mini-batch is applied and the latency drops to zero when there are no changes.

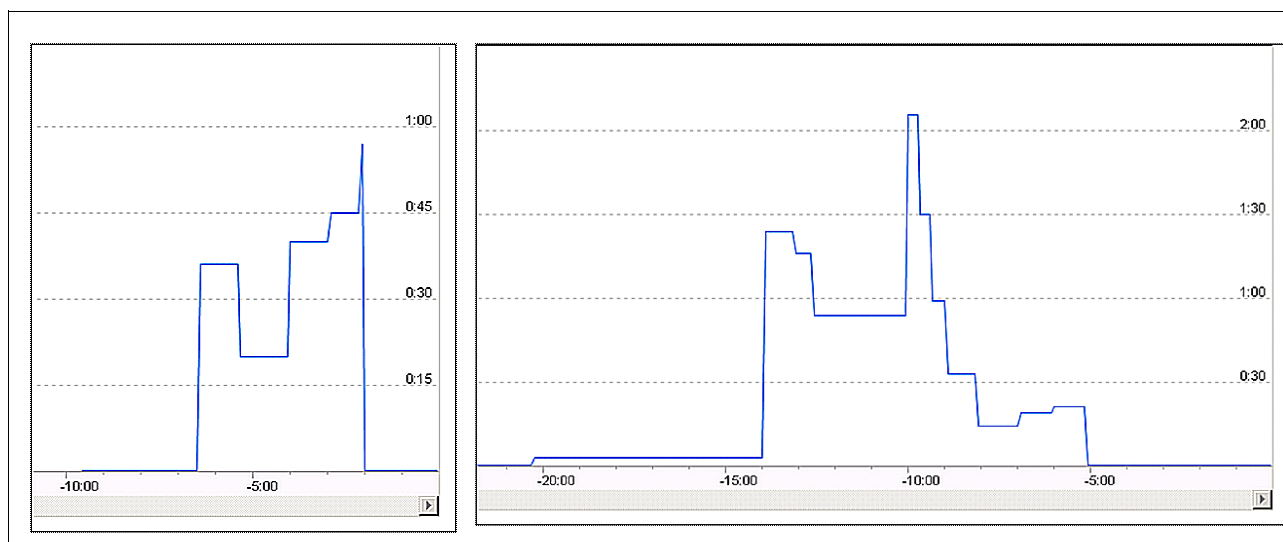


Figure 7-4 Chart of average latency

The chart on the right side shows a little more variability in the average latency. It was very low during the first 6 minutes then jumped up and stayed higher for around 5 minutes, up to over 2 minutes for a short time. It then took a steep drop for a bit before dropping off to zero. What happened during this time frame? There was 5 minutes of intense query activity against a different table during that period, which absorbed most of the Accelerator resources and had some impact on the target agent. There is more discussion about concurrently running queries and incremental update in 7.9.7, “Concurrent query with replication” on page 183.

Many factors affect latency. The settings of the z/OS source agent, such as BUFSIZE and BUFTHRESHOLD, determine how quickly the source agent can get changes from the DB2 login to the staging area. The length of time that a commit group stays open determines how quickly the source agent can get the changes out of the staging area and over the network to the target agent. On the Accelerator target side, how often the apply process runs and how long it takes to apply the changes, especially deletes, also influences the latency. If the Accelerator host is busy with other tasks like high-priority queries or even batch load, the apply process might get delayed due to processor availability. Deletes to a table that have a long scan time can also slow down the apply rate and negatively affect latency. Most of the conditions are usually temporary and the apply catches up when the bottleneck is cleared.

Although latency is an important measurement in an incremental update scenario, the value of one point-in-time latency measurement is not so important. Latency should be watched over time. If it is consistently in the medium or high range, further investigation to find the bottleneck needs to be done. The most likely cause is too much other activity, particularly high-priority queries, on the Accelerator.

A simple way to monitor the latency as well as change activity is to periodically use the **getAcceleratorInfo** command of SYSPROC.ACCEL_CONTROL_ACCELERATOR. See the replication information section of the results in Example 7-3 on page 175. This can easily be called from a REXX program and the replication information that is parsed and stored in a DB2 table. Using an automation tool to run this once a minute gives a nice log of latency trends as well as source and target activity. A sample REXX program, LATMONTRX, to do this is provided as described in Appendix E, “Additional material” on page 343, along with DDL to create the table and JCL to run the REXX program.

TIMESTAMP	LATENCYINSECONDS	SOURCE_INSERTS	SOURCE_UPDATES	SOURCE_DELETES	TARGET_INSERTS	TARGET_DELETES
2013-06-15 19:57:43.812371	0	0	4637044	0	4637044	4637044
2013-06-15 20:20:22.624326	0	0	4637044	0	4637044	4637044
2013-06-15 20:21:11.189512	0	0	4800840	0	4637044	4637044
2013-06-15 20:22:10.523145	34	0	5305848	0	5030748	5030748
2013-06-15 20:23:10.30481	19	0	5805180	0	5565948	5565948
2013-06-15 20:24:09.36564	19	0	5970672	0	5929632	5929632
2013-06-15 20:25:08.751776	35	0	6529900	0	6484824	6484824
2013-06-15 20:26:08.603794	20	0	6617552	0	6484824	6484824
2013-06-15 20:27:14.57402	0	0	6617552	0	6617552	6617552

Figure 7-5 Sample data from a table that is populated by the LATMONRX REXX program

7.9 Incremental update scenarios

In this section, we explore some incremental update scenarios to understand the characteristics of how incremental update works for various situations. Some scenarios that are covered include:

- ▶ Insert only replication
- ▶ Mass insert replication
- ▶ Delete only replication
- ▶ Mass delete replication
- ▶ Update only replication
- ▶ Multi-table replication
- ▶ Concurrent query with replication

For these scenarios, we used a System z EC12 with 10 processors, DB2 10, DB2 Analytics Accelerator Model N2001 Single Rack system with seven worker nodes, and DB2 Analytics Accelerator software V3.1.2.

The primary DB2 table that is used for incremental update in these scenarios is a 1 TB fact table with 9 billion rows. A 200 GB, 2 billion row version of the same table is also used in some scenarios. In addition, a few small dimension tables are used in the queries.

The source agent is configured with 1024 KB for the log buffer (BUFSIZE), notification at 1% of log buffer (BUFTHRESHOLD), and a polling interval of 5 seconds (LOGPOLLINTERVAL).

The tool that is used to capture the metrics has a measurement interval of 5 seconds, and reports numbers in operations per second.

Note: Incremental update performance is extremely workload-dependent. Table size, row size, configuration of the capture agent, other DB2 work, other Accelerator work, and even the specific model of Accelerator hardware can dramatically affect incremental update performance.

7.9.1 Insert only replication

In this scenario, we look at the behavior for replicating inserts only.

- ▶ Table: 1 TB fact table
- ▶ Uncompressed row length: 119 bytes
- ▶ Operations: Inserts only
- ▶ Number of inserts per commit group: 10

- Think time between insert statements: zero
- Time of execution: 15 minutes

Key metrics are:

- Total number of inserts: 4,980,260
- Average replication rate: 5533 operations per second
- Total bytes of source data replicated: 565 MB
- All target apply operations that are completed within the 5-second period of granularity
- Apply rates varied from a low 66,015 inserts per second to a maximum of 68,228 for each 5-second period of granularity

Figure 7-6 shows a graph of insert rates (inserts per second) for the source agent and the target agent. The source agent had a fairly steady rate of around 5500 inserts per second. The target agent ran approximately every minute, inserting approximately 330,000 - 340,000 inserts each time that it ran.

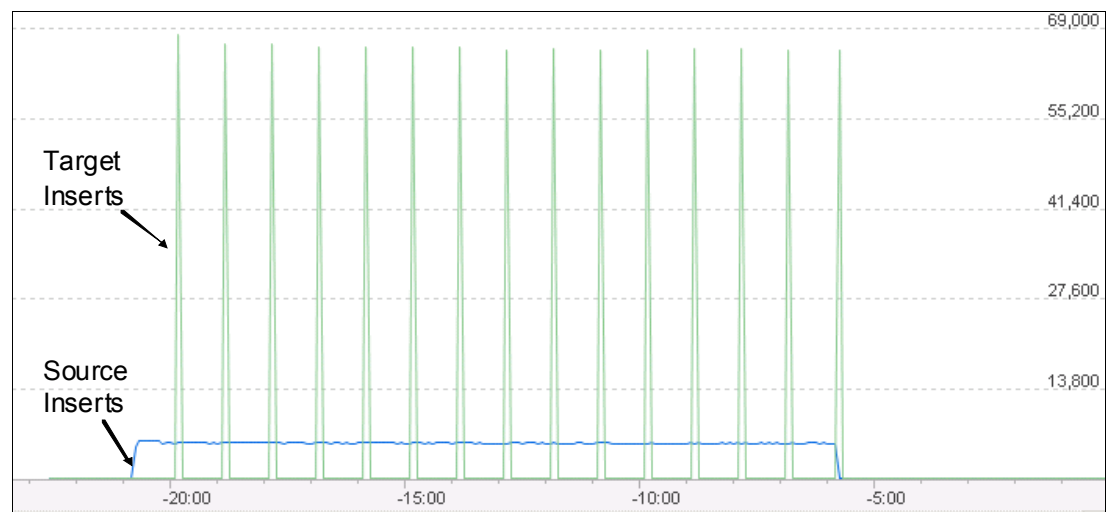


Figure 7-6 Operations per second for source and target agents: inserts only

7.9.2 Mass insert replication

In some situations, a batch load of one or more partitions is not practical. For example, it could be that inserts are relatively few, scattered across many partitions, and the refreshing of those partitions cannot be contained in the batch window. Perhaps the use of incremental update could alleviate the problem if the replication can be accomplished in a more timely fashion.

This scenario explores the replication behavior of doing a mass insert to a table. We turned off replication, and ran an insert job that inserted approximately 5.1 rows into the DB2 table. We then turned replication on to capture all the changes:

- Table: 1 TB fact table
- Uncompressed row length: 119 bytes
- Operations: Mass inserts

Key metrics are:

- Total number of inserts: 5,052,730
- Total bytes of source data replicated: 574 MB

Look at Figure 7-7 and compare it to Figure 7-6 on page 178. It is immediately obvious that the time frame is compressed. In this scenario, all 5.1 million inserts were propagated in approximate 2 minutes. The capture and apply rates (operations per second) were also a lot higher. The capture rate was well over 50,000 per second and were all sent in approximate 90 seconds. Notice that the apply did not wait 60 seconds between runs. As the staging area reaches some threshold, the apply runs even if the 60-second interval has not expired. With the final apply, the staging area did not reach that threshold and the apply ran approximately 60 seconds after the previous apply. The apply agent can adapt to high velocity situations assuming that the Accelerator has enough spare resources available for replication activity.

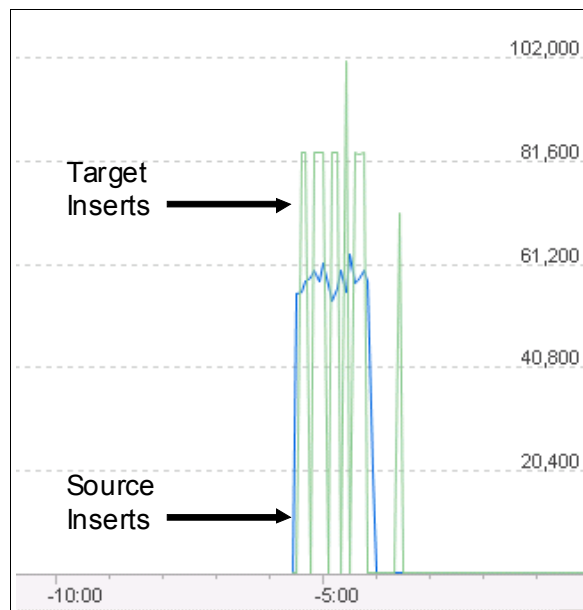


Figure 7-7 Operations per second for source and target agents - mass inserts

7.9.3 Delete only replication

In a typical data warehouse dimensional model, the preponderance of activity is inserts to a fact table. Updates and deletes usually occur to relatively smaller dimensional tables. However, in an operational data store or an operational system, deletes are normal.

This scenario looks at the behavior of replicating many deletes to the Accelerator.

- ▶ Table: 1 TB fact table
- ▶ Uncompressed row length: 119 bytes
- ▶ Operations: Deletes only
- ▶ Number of deletes per commit group: five
- ▶ Think time between delete statements: zero
- ▶ Time of execution: 15 minutes

Key metrics are:

- ▶ Total number of deletes: 961,140
- ▶ Average replication rate: 974 operations per second
- ▶ Total bytes of source data replicated: 109 MB

Figure 7-8 on page 180 shows the operations per seconds. The first you might notice is that the line for the source delete rate looks like teeth of a comb. This is the effect of setting the LOGPOLINTERVAL to 5 seconds as discussed in 7.7.1, “Reading the DB2 log” on page 170. In this run, the logging of changes cannot keep up with the capture agent

processing. After each run, the capture agent must wait for the notification from DB2 IFI after the LOGPOLINTERVAL expires.

The target agent runs every 60 seconds and deletes the rows from the Accelerator table, as discussed in 7.8.3, “About delete processing” on page 173. In this scenario, the apply agent kept up very well, applying deletes against a 1-terabyte table. Notice that there is one small apply at the end to apply the last few deletes.

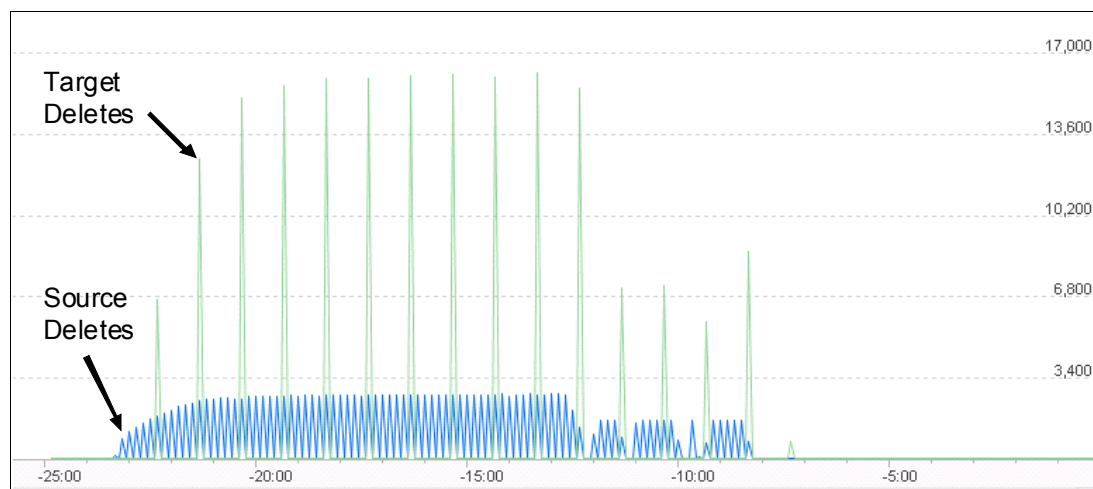


Figure 7-8 Operations per second for source and target agents: Deletes only

7.9.4 Mass delete replication

Occasionally in a data warehouse, there is a need to purge old data. In cases where data is organized such that this is accomplished by the removal of one or more partitions, the Accelerator handles this easily. In some instances, however, deletes are relatively few, scattered across many partitions, and the refreshing of those partitions cannot be contained in the batch window. As with mass inserts, perhaps, incremental update might be a good option.

In this scenario, we explore replication behavior for mass deletes. We used single delete statements to first delete 2 million rows. After those changes were replicated, we did a larger delete of 3.2 million rows:

- ▶ Table: 1 TB fact table
- ▶ Uncompressed row length: 119 bytes
- ▶ Operations: Deletes only
- ▶ Number of deletes per commit group: 2,000,001 and 3,283,859

Key metrics in this scenario: Total number of deletes is 5,283,860.

Figure 7-9 on page 181 shows the results of these two deletes. In the first delete, it took less than 1 minute to transmit the changes from the source system to the target system staging area. Then approximately after the start of the replication, the apply ran and made one pass through the data to flag all 2 million rows as deleted. The apply completed within the 5-second resolution of the graphing tool.

In the second run, it still took less than a minute for all 3.2 million changes to be captured and staged in the target system. Notice that two applies ran. The first apply ran before the 60 second mini-batch interval expired and was due to reaching a threshold for the staging area. The first apply completed in the 5-second interval averaging approximately 476,000 deletes per second for 5 seconds, for a total of about 2.4 million rows deleted. The second

apply ran at the normal 60-second interval and applied the rest of the rows of 0.8 million. It also completed within 5 seconds.

Each of the three applies deleted a widely varying number of rows: 0.8 million, 2.4 million, and 2 million. However, each took less than 5 seconds to complete. Now look back at Figure 7-8 on page 180. Each of those apply runs completed within the 5-second measurement interval and deleted fewer rows at a time, 80,000 rows or less. This shows that the delete process is primarily dependent on how long it takes to scan the target table and flag those rows as deleted.

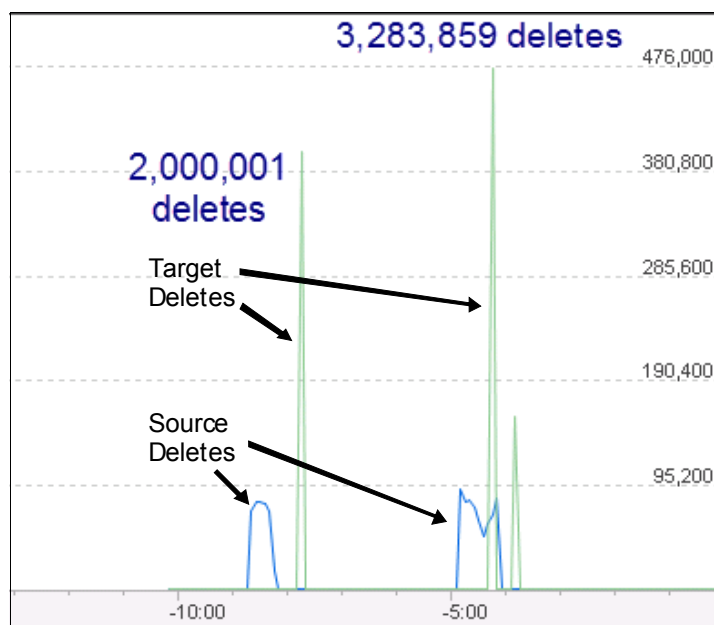


Figure 7-9 Operations per second for source and target agents: mass deletes

7.9.5 Update only replication

In this scenario, we look at the behavior for replicating updates only. Updates in DB2 are transformed into insert and delete pairs at the Accelerator. At each mini-batch, the target agent will first perform a mass delete for all deletes that are in the staging. These could be deletes as a result of deleting rows in the DB2 table or generated from updates to rows in the DB2 table. This is then followed by a mass insert. Again, this could be the result of inserts to the DB2 table or generated from updates to rows in the DB2 table.

The mass inserts are very efficient, so they are not our main concern when looking at performance of replication. The performance of delete processing is driven by the time it takes to scan the target table and flag rows as deleted. The time that it takes to scan the data is somewhat dependent on the time that it takes to scan the largest data slice.

This scenario was designed to explore replicating updates:

- ▶ Table: 1 TB fact table
- ▶ Uncompressed row length: 119 bytes
- ▶ Operations: Updates only
- ▶ Number of updates per commit group: four
- ▶ Time of execution 15 minutes

Key results are:

- ▶ Total number of updates: 5,846,520
- ▶ Average replication rate: 6500 operations per second
- ▶ Total bytes of source data replicated: 664 MB

Figure 7-10 shows a graph of updates per second for the source agent and the total number of deletes and inserts per second for the target agent. The pattern of source updates indicates waiting for the LOGPOLLINTERVAL. There were some longer pauses on the capture side in this run, which we believe was caused by resource contention somewhere along the line. But notice that the replication had no problem catching up.

When an update comes across to the target system, it is converted into a delete and insert pair. The target agent then does only deletes and inserts. When the apply runs, all deletes in the staging that are for a table are processed in one scan of the target table. This is followed by a mass insert of all the rows in the staging area to the target table.

Each of the apply runs in Figure 7-10 is doing both deletes and inserts. Most of the time both the delete processing and the insert processing completes in one 5-second measurement interval. However, sometimes we see delete processing in one 5-second interval and the insert processing in the next 5-second interval. The result is that the target agent had no problems keeping up with the replication rate.

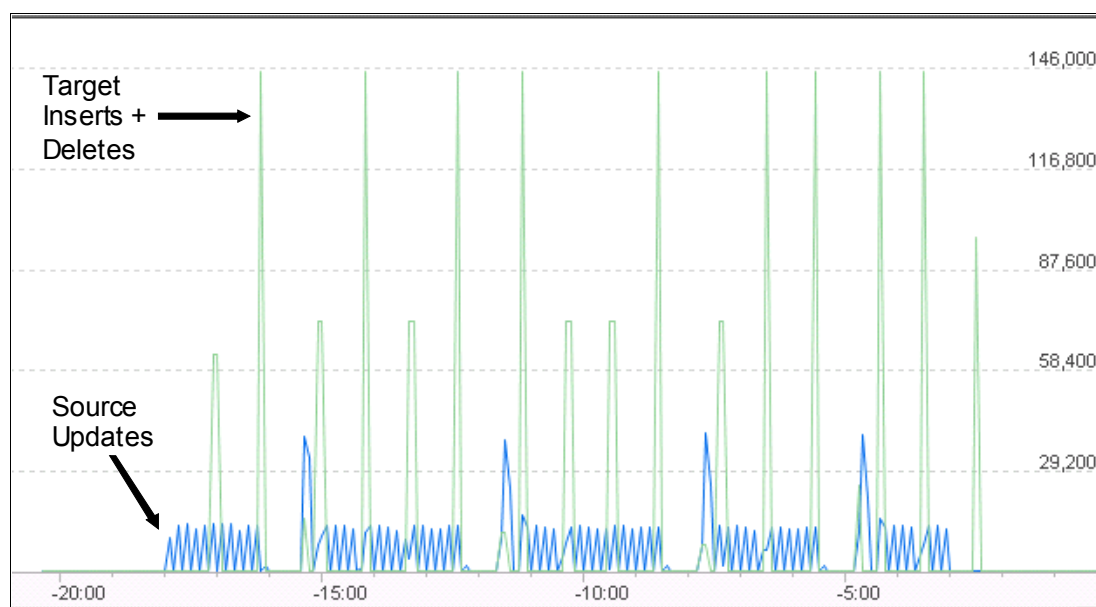


Figure 7-10 Operations per second for source and target agents: updates only

7.9.6 Multi-table replication

So far, we have only looked at replication scenarios involving just one table. This scenario will replicate changes to two tables. Both tables are the same layout except that one is 200 GB and one is 1 TB. We used DB2 updates because that results in 50% deletes and 50% inserts to the target tables. We generated twice as many updates to the 1 TB table compared to the 200 GB table:

- ▶ Table: 1 TB fact table/200 GB fact table
- ▶ Uncompressed row length: 119 bytes
- ▶ Number of rows: 9 billion/2 billion

- Operations: Updates only
- Number of updates per commit group: four updates to 1 TB table; two updates to 200 GB table
- Time of execution: 15 minutes

Key results are:

- Total number of updates: 6,704,070
- Average replication rate: 7450 operations per second
- Total bytes of source data replicated: 761 MB

Figure 7-11 shows the behavior of replicating updates to both tables. Notice that there is a pattern of an apply that has a rate nearing 120,000 per second followed by an apply that has about half that rate. It follows then that the changes for the 1 TB table are processed first, followed by another apply to process the changes for the 200 GB table. With one exception, both the deletes and inserts for each apply completed within the 5-second measurement interval.

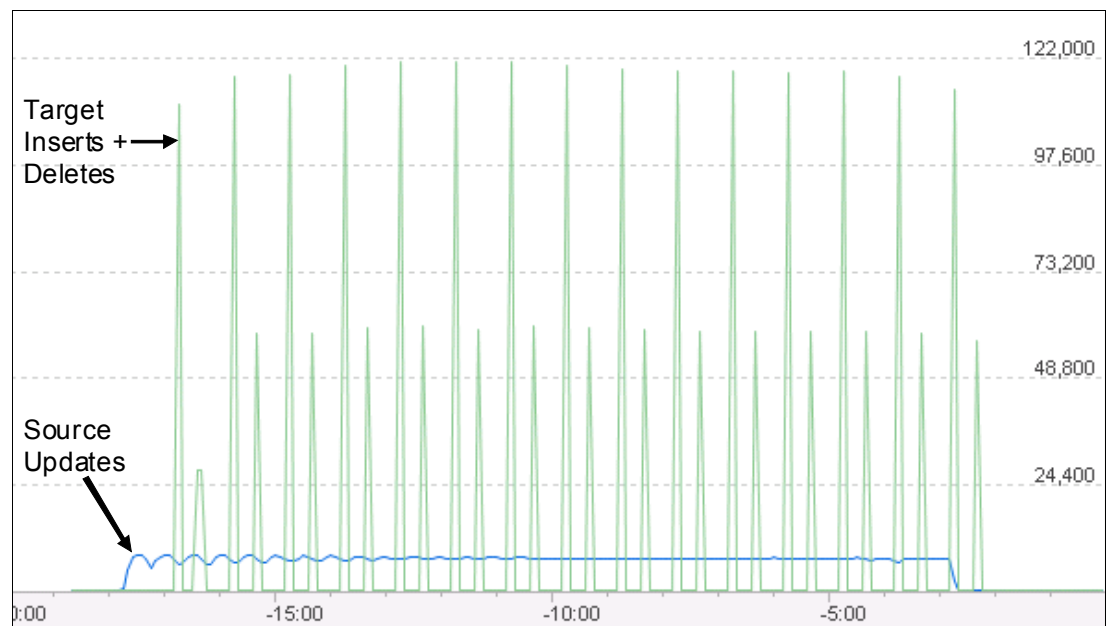


Figure 7-11 Operations per second for source and target agents: two table replication

7.9.7 Concurrent query with replication

The replication scenarios so far have only looked at the behavior of replication without any queries running. However, the most prominent use case for replication is to incrementally update the Accelerator from DB2 while running queries at the same time.

Base queries

We used a set of four queries. Three of these queries do aggregations over a join of fact and dimension tables. The fourth query performs a count of all rows in the table to force an I/O-bound query that scans the entire table.

There were four simulated users executing the four queries concurrently, with no think time in between. The test was executed for a total of 5 minutes. This was designed to simulate an extreme peak period where the Accelerator worker nodes are driven to 100% utilization over the 5-minute time period for this set of queries.

Table 7-1 shows the results of a 5-minute query run with no replication occurring. This provides some basic performance metrics to be used to see the impact on queries while concurrently replicating data.

Table 7-1 Query performance with no replication

Total elapsed time: 5 min., 25 sec.	Query 1	Query 2	Query 3	Count
Average response time (seconds)	17	24	161	32
Number of executions	18	13	2	10
Total number of rows returned	3600	1560	12,600	10

Replication

The replication component of this test is the same as documented in section 7.9.5, “Update only replication” on page 181.

The process

The following process was used for the tests in this section:

1. Start the 15-minute job to update the source DB2 table and replicate to the Accelerator
2. After approximately 5 minutes, start the 5-minute query job
3. The query job will complete while the updates continue for another 5 minutes

With this test, we are looking to see how concurrent queries affect the replication rate, how replication affects the query response time, and how long it takes for replication to catch up after the queries complete.

Queries while replicating to a different table

In this test, we update the 200 GB table replicating the changes to the Accelerator. The concurrent queries execute against the 1 TB table.

Replication:

- ▶ Table: 1 TB fact table
- ▶ Uncompressed row length: 119 bytes
- ▶ Number of rows: 9,000,000,000
- ▶ Operations: Updates only
- ▶ Number of users: 20
- ▶ Number of updates per commit group: four
- ▶ Think time between update statements: zero
- ▶ Time of execution: 15 minutes

Key results are:

- ▶ Total number of updates: 8,167,068
- ▶ Average replication rate: 9075 operations per second
- ▶ Total bytes of source data replicated: 926 MB

Looking at Figure 7-12 on page 185, we can clearly see when the queries started consuming 100% of the Accelerator worker nodes. For the first 6.5 minutes, replication was feeding and applying changes steadily. When the queries started, the replication rate became sporadic but it was still occasionally able to get some resources to process changes, albeit a bit slower. However, after the queries stopped consuming 100% of the worker nodes, replication was able to catch up in about a minute.

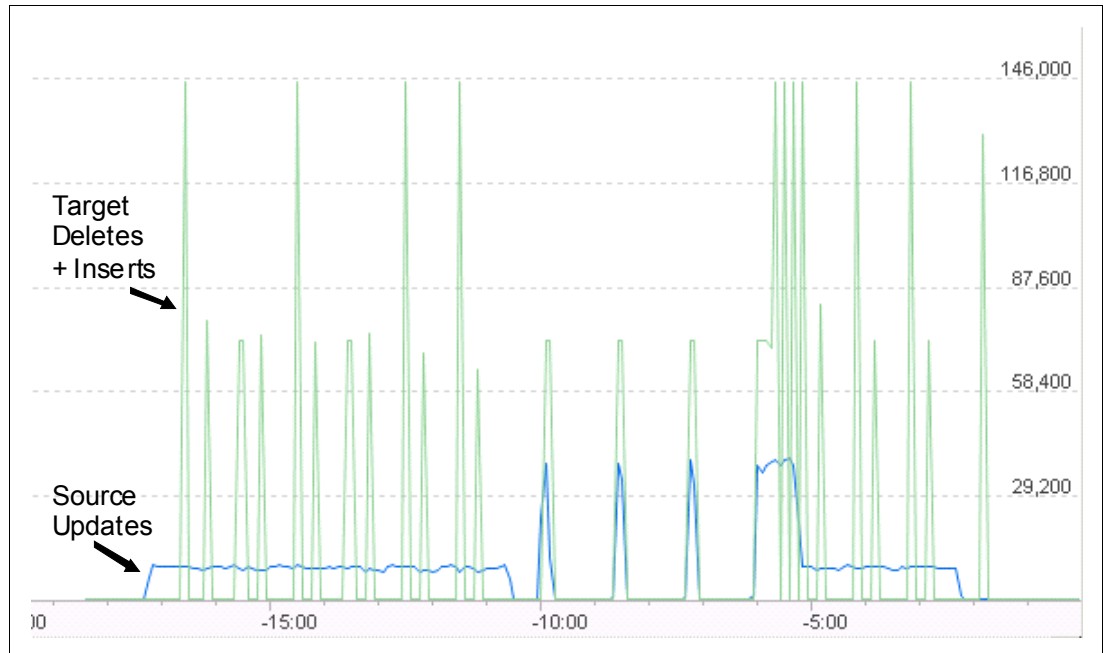


Figure 7-12 Replication result with concurrent query to different table

Figure 7-13 shows the impact on latency. While the queries were consuming the worker nodes, the slowdown in the replication rate had an impact to latency of course. You can see that the latency averaged about 5 seconds before the queries started. When the queries started, the latency peaked after 3 minutes but rapidly recovered as the replication rate increased after the queries completed, returning to an average of about 5 seconds.

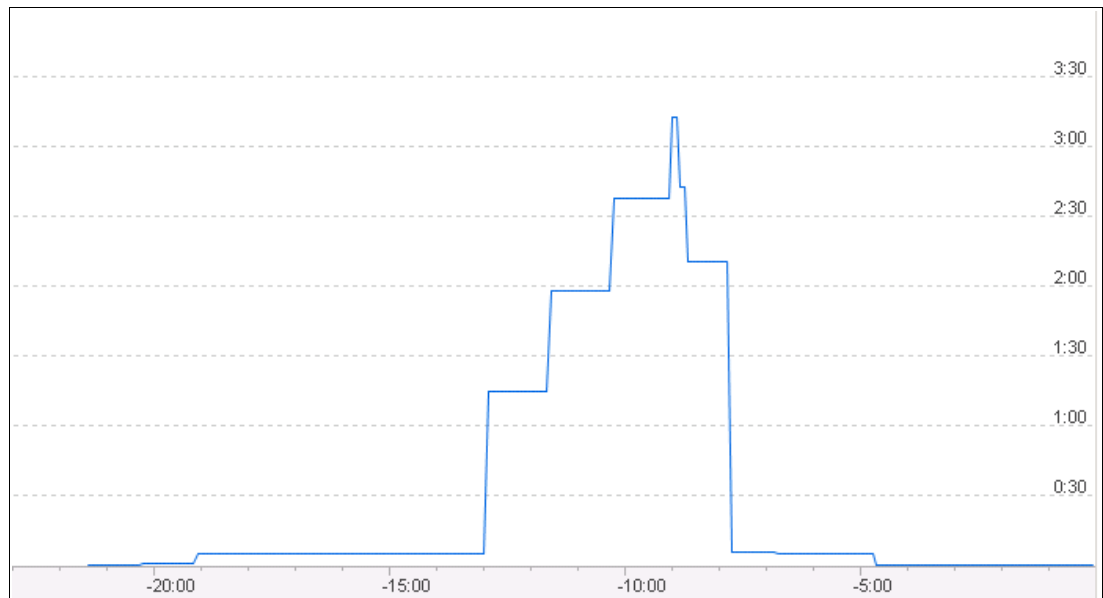


Figure 7-13 Impact on latency with concurrent query to a different table

Queries

How did the queries perform with concurrent replication? Look at Table 7-2. The average response time for each query did increase, but by less than 10%. This resulted in one less execution for some of the queries for the same time period.

Table 7-2 Comparison of query metrics without and with concurrent replication

Total elapsed time: 5 min., 38 sec.	Query 1	Query 2	Query 3	Count
Average response time (seconds)				
without replication	17	24	161	32
with replication	18	26	165	35
Number of executions				
without replication	18	13	2	10
with replication	17	12	2	9
Total number of rows returned				
without replication	3600	1560	12,600	10
with replication	3400	1400	12,600	9

Conclusion

This test simulated a very heavy peak query usage by driving the Accelerator worker nodes to 100% utilization during the query processing. We saw a minor impact on query response time, less than 10%.

7.10 Summary

This chapter introduced the new feature to the IBM DB2 Analytics Accelerator known as *incremental update*. This feature enables near real-time change propagation from a DB2 source table on System z to the corresponding table deployed on the Accelerator. This chapter explored some of the key details of incremental update and its behavior in several common scenarios. While performance of incremental update is extremely dependent on the specific workload, this information should help design an incremental update strategy for your shop.

Not every table has a requirement for near real-time updates. The IBM DB2 Analytics Accelerator enables you to design the best strategy for data latency management using a combination of bulk data loading. See Chapter 3, “Data latency management” on page 55, as well as incremental update.

The results of further lab experiments for incremental update capturing CP consumption and providing the SQL for estimating it for a given workload are provided in the white paper, *Synchronizing data in IBM DB2 Analytics Accelerator for z/OS*, which is available at the following website:

<http://www.ibm.com/support/docview.wss?uid=swg27038501>



Impact of new Netezza hardware and software components

IBM Netezza 1000 is a purpose-built, standards-based data warehouse appliance that architecturally integrates database, server, storage, and advanced analytic capabilities into a single, easy-to-manage system. The IBM Netezza 1000 appliance is designed for rapid and deep analysis of data volumes scaling into the petabytes, delivering 10 - 100x performance improvements at a fraction of the cost of other options that are available from traditional vendors. N1000 has been the base for the IBM DB2 Analytics Accelerator V2.1 solution.

With the IBM DB2 Analytics Accelerator V3.1, both the IBM Netezza 1000 and IBM Netezza 2000 appliances are available in the IBM PureData System for Analytics offering. Both appliances can use the updated operating system, Netezza Performance System (NPS), from NPS 6.0.8 to NPS 7.0.2.

This chapter provides a description of the hardware and software levels and query performance considerations that are based on lab measurements, which highlight the provided benefits.

For LOAD throughput enhancements on N2001 versus N1001, and Data Facility Storage Management Subsystem (DFSMS) enhancements, see Chapter 3, “Data latency management” on page 55.

In this chapter, we discuss:

- ▶ Hardware and software evolution
- ▶ PureData System for Analytics N2001
- ▶ Netezza Performance System 7.0.2
- ▶ Query performance measurements

8.1 Hardware and software evolution

When IBM acquired Netezza, the processing capability that is enabled by the Netezza Field Programmable Gate Arrays (FPGA) technology made Netezza's query speeds clear winners for business analytics. Today, a new generation of Netezza, branded the *IBM PureData System for Analytics*, is powered by the new Netezza N2001 systems as well as the N1001 systems.

The new N2001 system has a three times faster scan rate than the N1001, while supporting 50% greater storage capacity per rack, and improved resiliency and fault tolerance. This performance is due to more spare drives in the cabinet and faster drive regeneration.

The operating system has also been upgraded to NPS 7.0.2 for both Netezza systems, the N1001 and N2001.

8.2 PureData System for Analytics N2001

Figure 8-1 shows the three main areas of benefits with the next generation PureData System for Analytics.

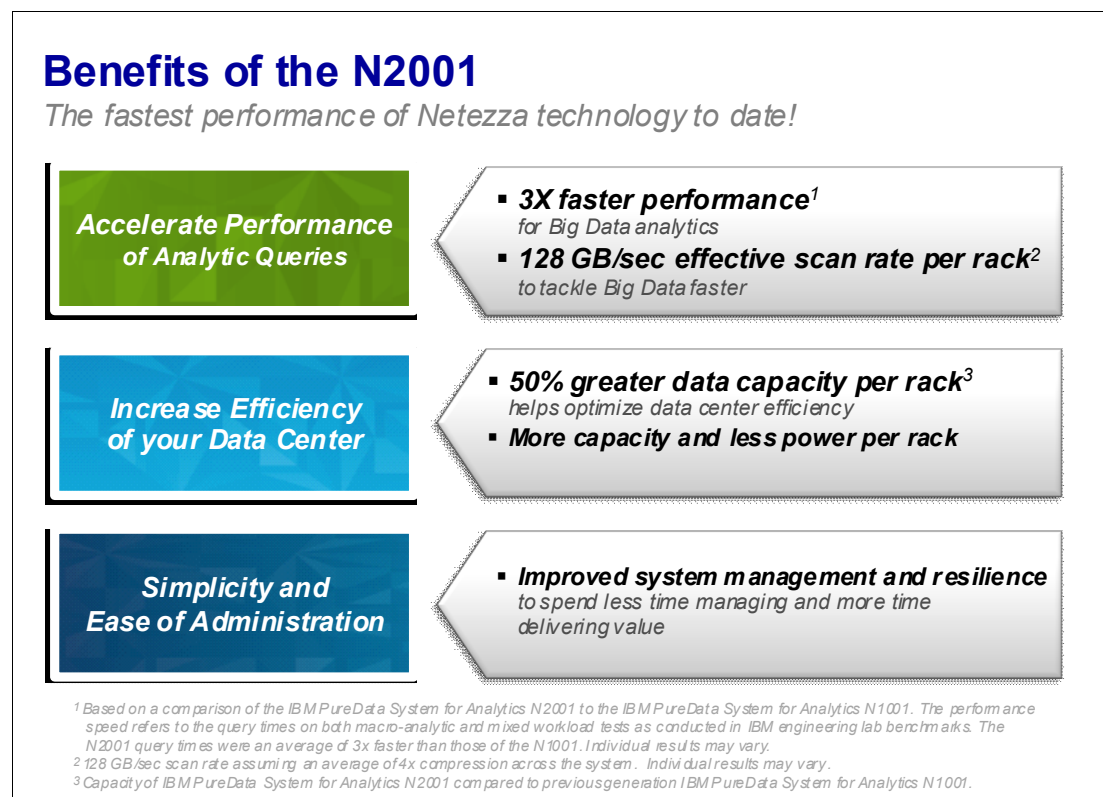


Figure 8-1 Benefits of the new PureData for Analytics N2001

- It is the fastest performing Netezza platform to date. It is three times or more faster for mixed workloads. That is, it accelerates the analytics three times or more faster than the current generation PureData System for Analytics N1001.

This means that you are able to tackle Big Data faster than ever before, with a 128-gigabyte per second effective scan rate per rack. If you have a two-rack system, you

get two times that scan speed. If you have a four-rack system, you get four times that speed. The N2001 delivers 3x scan rate improvement and 3x or more performance improvement over the existing N1000, which is the TwinFin® configuration.

- ▶ Beyond accelerating your queries, your data center efficiency improves by having 50% greater capacity per rack without increasing the power or cooling requirements. The new system delivers more capacity with less power and cooling per rack.
- ▶ And it is still the simplest, easiest to use, and easiest to administer. There is no change on ease of use.


The same values of ease of use, such as speed and simplicity are there, but with faster scan and more capacity per rack than the previous version.

8.2.1 PureData for Analytics benefits

PureData for Analytics is powered by Netezza technology to optimize performance and simplicity of data warehouse services for analytics applications. See Figure 8-2.

IBM PureData System for Analytics

Optimized exclusively for analytic data workloads



PureData
System for Analytics

- Speed**
 - 10-100x faster than traditional custom systems¹
 - Patented, hardware accelerated MPP (Massively Parallel Processing)
- Simplicity**
 - Data load ready in hours
 - No database indexes
 - No tuning
 - No storage administration
- Scalability**
 - Peta-scale data capacity²
- Smart**
 - Designed to run complex analytics in minutes, not hours

¹ Based on IBM customers' reported results. "Traditional custom systems" refers to systems that are not professionally pre-built, pre-tested and optimized. Individual results may vary.
² Peta Scale capacity offered in the N1001 model

Figure 8-2 The N2001 architecture

This model of PureData offers:

- ▶ **Speed**

The patented integration of Netezza software and hardware Accelerators deliver performance that is typically 10 - 100 times faster than custom integration of components.
- ▶ **Simplicity**

The system is fully integrated so it can be data-load ready in less than 4 hours. And the Netezza technology also eliminates the need for complex database management tasks, such as defining and optimizing indexes and manually administering storage.

- Scalability

Available in various size configurations, this system can be expanded to handle peta-scale capacity of user data.

- Smartness

It is able to run advanced analytic algorithms in minutes and includes the richest set of in-database analytics functions.

The core value of PureData System for Analytics has not changed. The next generation N2001 system still provides the four Ss: speed, simplicity, scalability, and smart.

This new system is still 10 - 100 times faster than the traditional custom system and probably more, given the speed increase. It is a patented hardware, asymmetric, MPP environment with the patented hardware acceleration of the FPGA.

There is no need for indexes, there is no tuning, and no storage administration. With the PureData System for Analytics, you do not have the hours worth of maintenance and work that you have to do daily with other systems.

This system is data-load ready in hours and then it continues to run. It can go to petabytes of data within a single rack and it is smart. It has the richest set of database analytics in the industry. And it can run those complex analytics in seconds or minutes, not hours or days.

How is the 3x scan performance increase obtained? See Figure 8-3 on page 191.

Previously, in the N1001 and in the TwinFin, on the left, there was one drive. Now you see three drives. What we always had prior to the N2001 was a 1:1:1 ratio; one disk drive to one FPGA core to one CPU core. But the fundamental bottleneck with that is that the drive can deliver only about 100 megabytes per second. That was the limiting factor.

The FPGA core can process significantly more than that. The CPU core can process significantly more than that. By the time it gets to the CPU core, the FPGA core has already thrown away most of what it scanned by projecting and filtering and applying the predicate, storing away any extraneous columns that are not needed. So the FPGA core really gets very, very small subsets of what the FPGA core processes.

The solution is that there are now three drives for every FPGA core, and every CPU core. There is no bottleneck on the FPGA core. We are still using less data, and fewer cycles than the FPGA core has, and the CPU core is still sitting in the front with the same amount of processing power. We have more drives per FPGA and CPU core.

We also increased the speed of the FPGA cores. We are using new FPGA cores and newer, faster CPU cores. Therefore, they are delivering well over 1000 megabytes per second. Even with 4x compression on the data, we are still not overwhelming the FPGA cores.

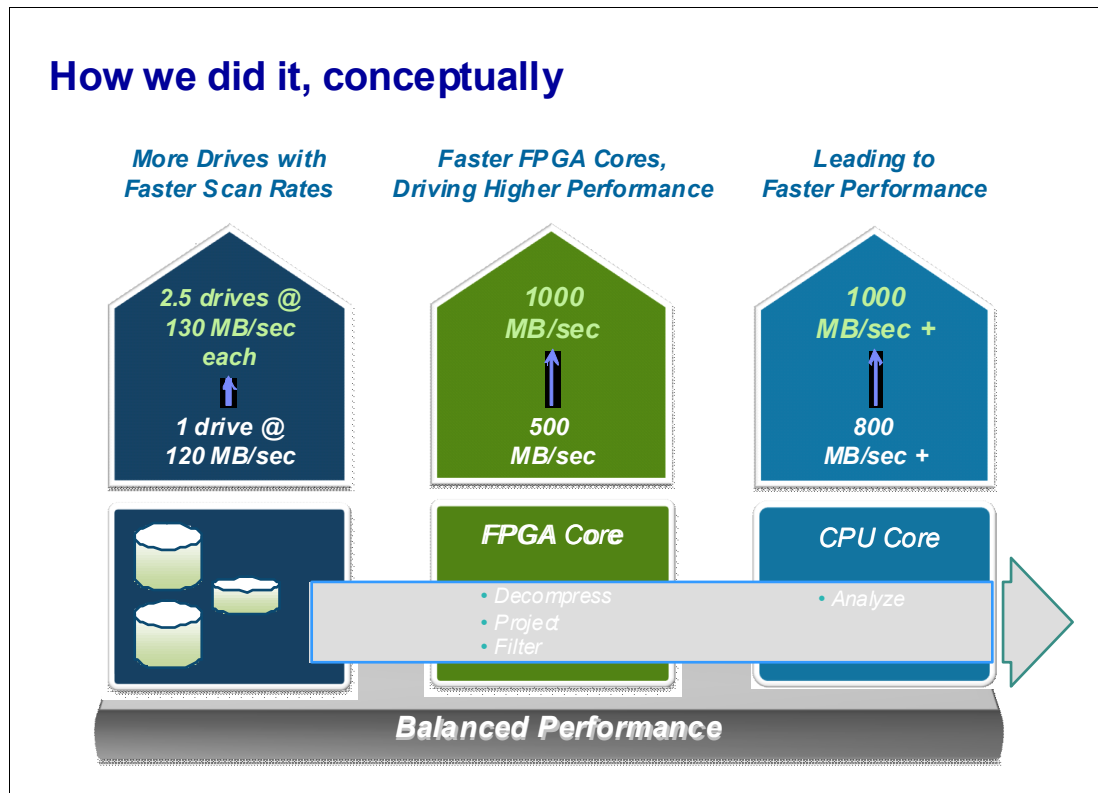


Figure 8-3 The major contributors

8.2.2 New hardware for PureData System for Analytics

Figure 8-4 on page 192 is an overview of the PureData System for Analytics. While you can buy PureData for Analytics in ¼ rack, ½ rack, full rack, and multi-rack configurations, each rack will look like what is shown in the diagram.

There are 12 SAS disk enclosures with a total of 288 600 GB drives, two front end “hosts” using dual quad core processors, and seven 300 Gb drives each.

But the real work happens on the S-Blades. In a half rack there are 4, and in a full rack there are 7. These S-Blades have two eight-core CPUs and two eight-engine FPGAs, along with 128 GB of memory plus an 8 GB slice buffer. The FPGAs act as an intelligent query filter — automatically discarding 90 - 95% of the data that is irrelevant to the query being run. The built-in expertise of the FPGA does more than that though. Certain types of processing are pushed down to run inside the FPGA — immediate decompression, projections, restrictions, and visibility lists are all handled within the FPGA. This patented hardware acceleration layer is the corner stone that provides the breathtaking performance expected from IBM PureSystems™.

All of this adds up to a system that can handle up to 192 GB of data in a full rack that can scan data at nearly 500 TB per hour and load data at up to 5 TB per hour. All of this efficiency is available without any need for indexes, storage management, or tuning.

N2001 Hardware Overview

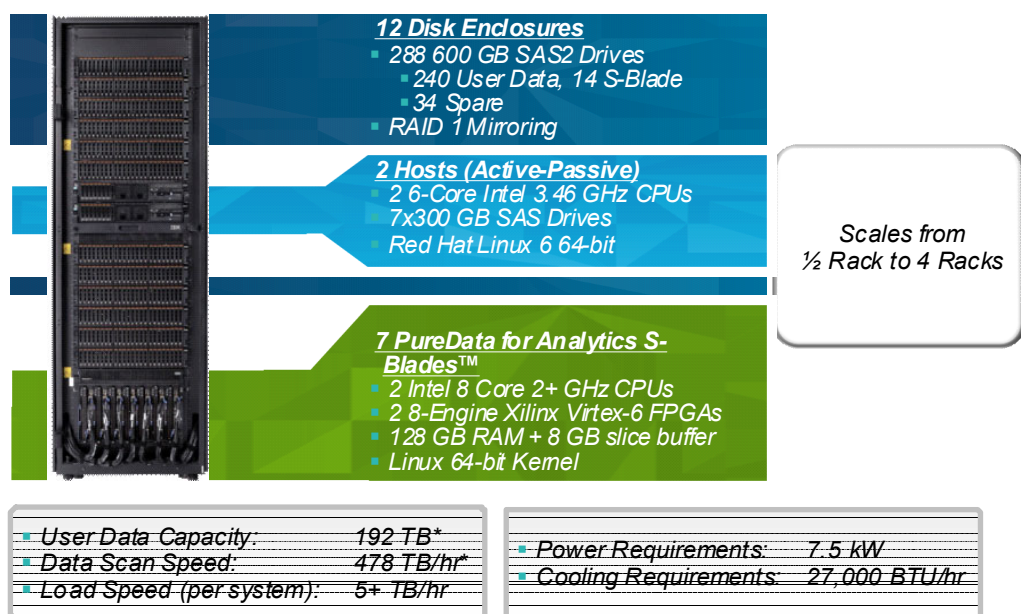


Figure 8-4 N2001 hardware

8.2.3 Scan speed

Figure 8-5 on page 193 shows how the scan speeds have increased. Basically, about each drive in the N2001 is capable of delivering about 130 MBps of throughput (compared to approximately 120 MBps in the N1001).

What we had before with the 1:1:1 ratio, one drive to one FPGA core to one CPU core, the speed of the drive was the limiting factor as far as how fast the FPGA core could process the data. This is because the FPGA core could handle way more than that and the CPU core even more than the FPGA core. So the speed of the drive was a limiting factor.

So we now have more than one drive per FPGA core and per CPU core. Using basic math, we have about 2 1/2 drives per FPGA core and per CPU core. So that drives up the speed that data can be scanned and delivered to the FPGA for processing up to about 325 megabytes per second. If you add in the 4x compression, that is going to get up to around 1300 megabytes per second.

In the N2001, we now have faster FPGA cores that can process about 1000 MBps. So we are now delivering both 2 1/2 times as much data, per second to the FPGA, and to the CPU core. That is how we fundamentally increased the scan speed and how we increase the performance of this system.

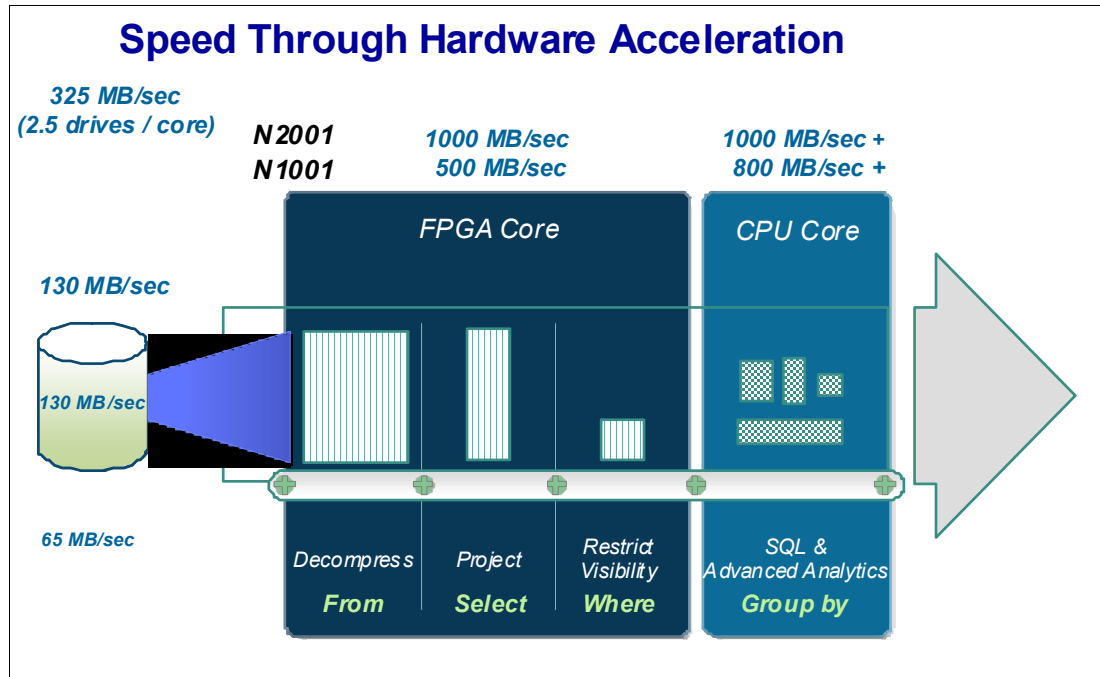


Figure 8-5 Hardware speed

8.2.4 The S-Blade components

Each new S-Blade is 16 FPGA cores and 16 CPU cores rather than eight in the N1001 system. See Figure 8-6.

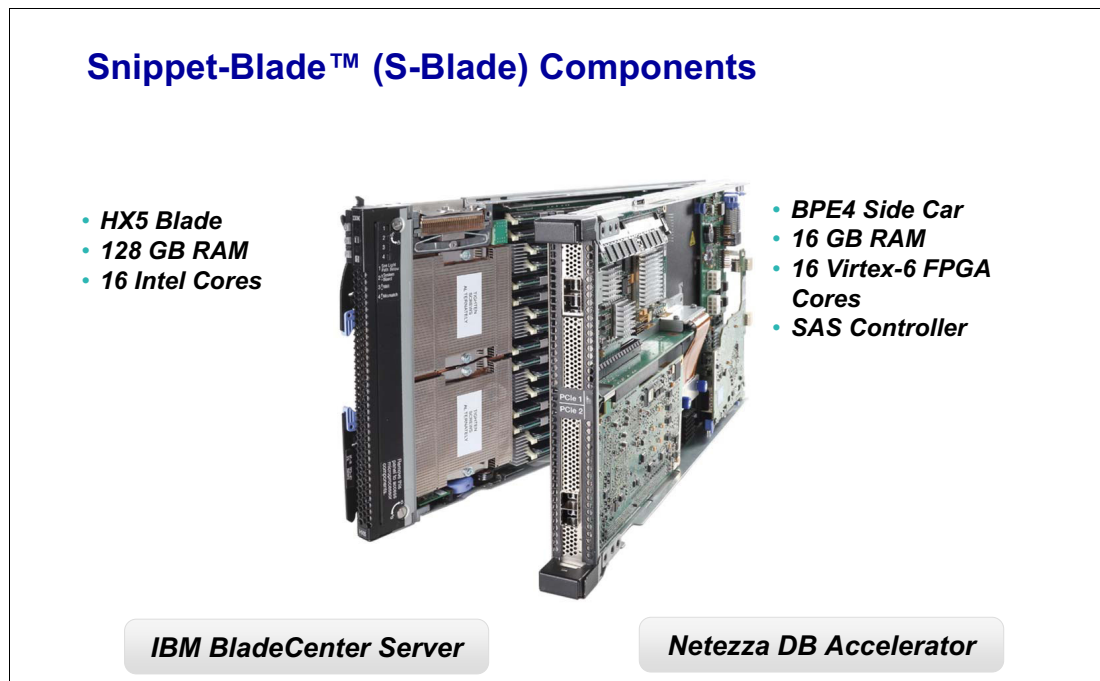


Figure 8-6 S-Blade components

8.2.5 Available PureData Systems for Analytics Models

N2001 has 288 active drives with 600 gig each. It has per rack capacity of about 173 terabytes after the mirroring attempt. It has about 192 terabytes of user data capacity after 4x compression. See Figure 8-7.

N2001 has seven blades per rack, and 112 cores, so that really has not changed from the previous version. The S blade memory now has gone up. Even though there is only twice as many cores, the S blade memory has gone up by a factor of five, which is significant.

PureData System for Analytics Models		
	Pure Data System for Analytics N1001	Pure Data System for Analytics N2001
Blade Type	HS22	HX-5
CPU Cores / Blade	2 x 4 Core Intel CPUs	2 x 8 Core Intel CPUs
# Disks	96 x 3.5" / 1 TB SAS (92 Active)	288 x 2.5" / 600GB SAS2 (240 Active)
Raw Capacity	96 TB	172.8 TB
Total Disk Bandwidth	~11 GB/s	~32 GB/s
S-Blades per Rack (cores)	14 (112)	7 (112)
S-Blade Memory	24 MB	128 MB
Rack Configurations	1/4, 1/2, 1, 1 1/2, 2 – 10	1/2, 1, 2, 4
FPGA Cores / Blade	8 (2 x 4 Engine Xilinx FPGA)	16 (2 x 8 Engine Xilinx Virtex 6 FPGA)
User Data** / Rack *	128 TB	192 TB
<p>* Assuming 4x Compression ** Max number of racks is 4 for N2001</p>		

Figure 8-7 Models of IBM PureSystems for Analytics

Figure 8-8 on page 195 shows N2001 systems and various sizes. The cabinets can be half rack, full rack, two racks, and four racks.

N2001 Systems and Sizes

PureData System for Analytics N2001

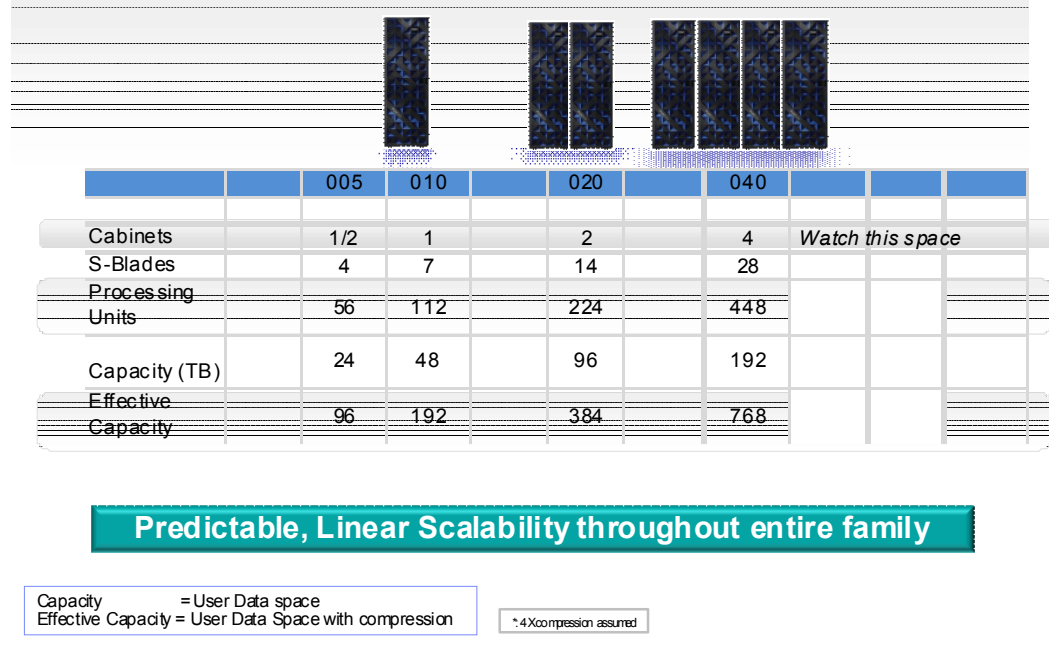


Figure 8-8 N2001 systems

8.3 Netezza Performance System 7.0.2

For recent general updates, see the “What’s New in the N2001 (Striper) Architecture?” forum, which is available at the following site:

<https://www.ibm.com/connections/forums/html/topic?id=e5106045-3be9-44cc-916f-378af57f8359>

Also, see the official N2001 Data Sheet at the following website:

<http://www.ibm.com/software/data/puredata/analytics/>

In this section, we describe some of the new features in Release 7.0.2 of Netezza Performance System (NPS), which are of interest for the Accelerator functions. In particular, we describe the concurrent processing enhancements.

Two enhancements contribute to better concurrent processing for short, tactical queries that go after small chunks of data:

- ▶ Directed data processing
- ▶ Page granular zone maps

8.3.1 Directed data processing

Directed data processing applies to a specific, but fairly common use case where there is a large fact table distributed on a specific column.

The scheduler is essentially able to manage four types of operations:

- ▶ Disk I/O
- ▶ FPGA processing
- ▶ CPU processing
- ▶ Network operations

SQL queries are broken up into snippets that consist of these types of operations. The scheduler will run snippets from different queries in parallel, provided that they do not conflict. This optimization allows for much more efficient scheduling of disk I/O operations.

Assume that the table TX_HIST is distributed on the column, CUSTOMER_ID. See Figure 8-9.

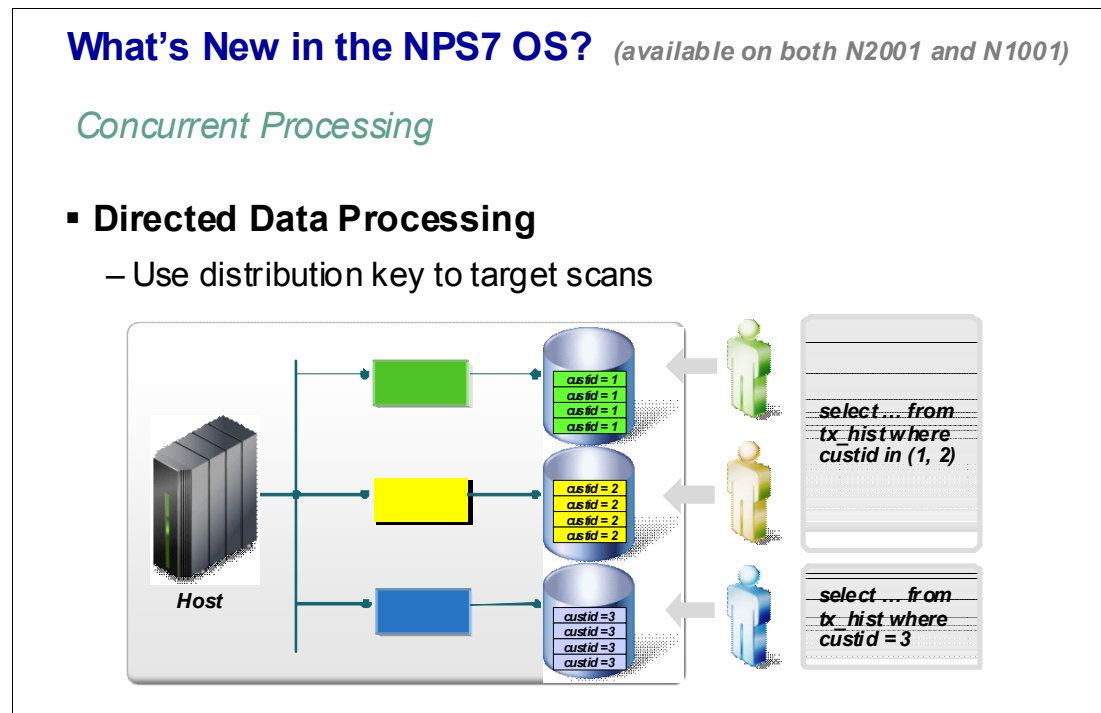


Figure 8-9 NPS7 enhancement: Directed data processing

Prior to version NTS 7, if we had the following two queries, the system would run them sequentially, assuming that we could only do one disk operation at a time:

```
SELECT ... from TX_HIST WHERE CUSTID in (1,2)
SELECT ... from ORDERS WHERE CUSTID = 3
```

However, the two queries are going after different customer IDs. Because customer ID is the distribution key for the table, NPS knows that these records reside on different data slices or disks. NPS 7 realizes this and only sends the relevant snippets down to the S-Blades that own the data slices that hold the data. So these two queries will now execute their disk reads concurrently. They may not run faster, but we will be able to run a lot more of them in parallel.

8.3.2 Page granular zone maps

Release 7.0 introduces a change to the Netezza zone map design with page-granular zone maps. Zone maps are automatically generated internal tables that the Netezza system uses to improve the throughput and response time of SQL queries against large grouped or nearly ordered date, timestamp, bigint, integer, and bigint data types. In previous releases, Netezza created zone maps for the data contained within each data slice extent (or 3 MB section). An extent contains twenty-four 128 KB pages.

With Release 7.0, the Netezza storage manager creates zone maps for each page (or 128 KB portion) within an extent. These more granular zone maps can help to improve query performance for classes of “small” queries that typically return a few records contained within a few pages of storage. Page granular zone maps help to improve performance for certain types of queries with narrow restricts by reducing I/O.

If you upgrade to Release 7.0, the upgrade enables the page granular zone maps. The system creates and maintains the zone maps as new user data is written to the system. If you downgrade from a Release 7.0.x release to a release before 7.0, note that the downgrade process removes the page granular zone maps that exist in the system.

For the potential to reduce disk I/O by as much as 24 times, consider the example in Figure 8-10, with mostly ordered data.

On the left, we scan 4 x 3 MB extents for a total of 12 MB scanned.

On the right, we scan 8 x 128 K pages for a total of only 1 MB scanned.

We obtain a significant reduction in disk I/O for tactical queries leading to increased throughput and concurrency.

What's New in the NPS 7 OS *(available on both N2001 and N1001)*

Concurrent Processing

Page Granular Zone Maps

- **24X finer granularity for less I/O**
where col = October

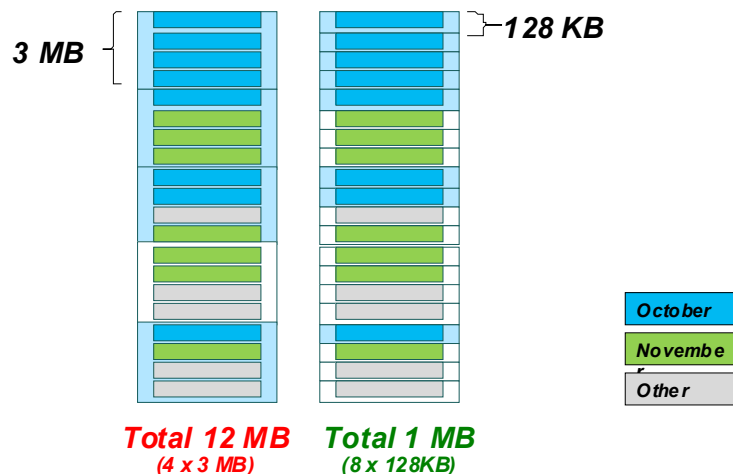
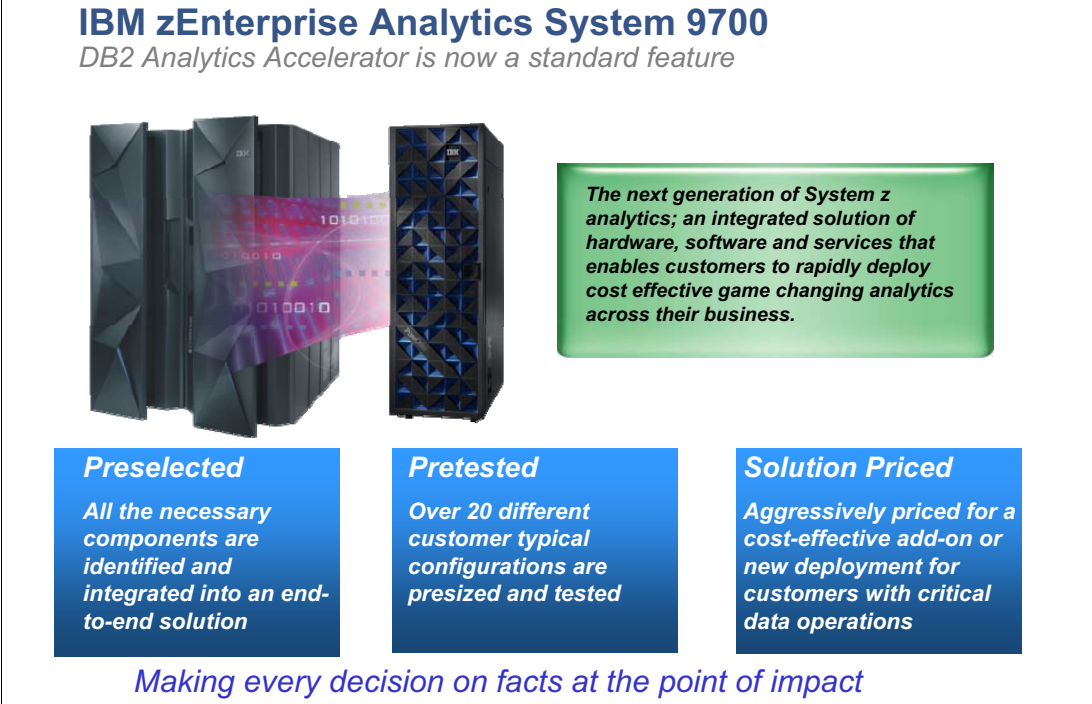


Figure 8-10 NPS enhancement: Page granular zone maps

8.4 System 9700 and 9710

The hardware and software enhancements provided by the N2001 are included in the IBM zEnterprise Analytics System 9700 offering, which is shown in Figure 8-11. See Chapter 10, “IBM zEnterprise Analytics System 9700” on page 267 for details.



IBM zEnterprise Analytics System 9700
DB2 Analytics Accelerator is now a standard feature

The next generation of System z analytics; an integrated solution of hardware, software and services that enables customers to rapidly deploy cost effective game changing analytics across their business.

Preselected
All the necessary components are identified and integrated into an end-to-end solution

Pretested
Over 20 different customer typical configurations are presized and tested

Solution Priced
Aggressively priced for a cost-effective add-on or new deployment for customers with critical data operations

Making every decision on facts at the point of impact

Figure 8-11 IBM zEnterprise Analytics System 9700

8.5 Query performance measurements

The general objective of these measurements is to provide a comparison of serial and concurrent query performance of the Accelerator with N2001 compared to N1001.

In the measurements, we compare different hardware but also different operating systems. It is N1001 going to N2001, as well as NPS 6.0.8P2 to NPS 7.0.2P0.

The N1001 system is a TwinFin HS21, not the latest HS22, with less RAM and a slightly slower processor.

The test environment is made up of the zEnterprise Analytics System 9700 offering, including:

- ▶ Cognos Great Outdoor star join database with large fact table and small dimension tables
- ▶ zEC12 with 4 or 10 CPs, 128 GB real memory
- ▶ zOS V1.13
- ▶ DB2 10 for z/OS
- ▶ DB2 Analytics Accelerator V3.1
- ▶ PureData for Analytics N1001-010
- ▶ PureData for Analytics N2001-010

Following are the test scenarios:

- ▶ Scenario #1: I/O bound query
- ▶ Scenario #2: CPU bound query
- ▶ Scenario #3: Mixed: I/O and CPU bound query
- ▶ Scenario #4: Selective query with equal predicates benefitting from zone maps
- ▶ Scenario #5: Directed data processing

8.5.1 Scenario #1: I/O bound query

In this scenario, we ran serially and concurrently I/O-bound queries executing a full table scan of the 9 billion rows and 100 partitions of the SALES_FACT table.

The query executes the following SQL statement on the fact table, with no predicate, no join, and no aggregate in the queries:

```
SELECT COUNT_BIG(*) FROM SALES_FACT
```

For elapsed time, we measured query elapsed time between the start of the first query and the end of the last query for concurrent runs.

Figure 8-12 shows the results of a test with I/O bound queries all executing the same SQL multiple times to test the concurrency on the N2001.

We can observe the query elapsed time scale linearly with the number of concurrent queries on the N2001.

Average CPU utilization on worker nodes is 10 - 34% on N2001.

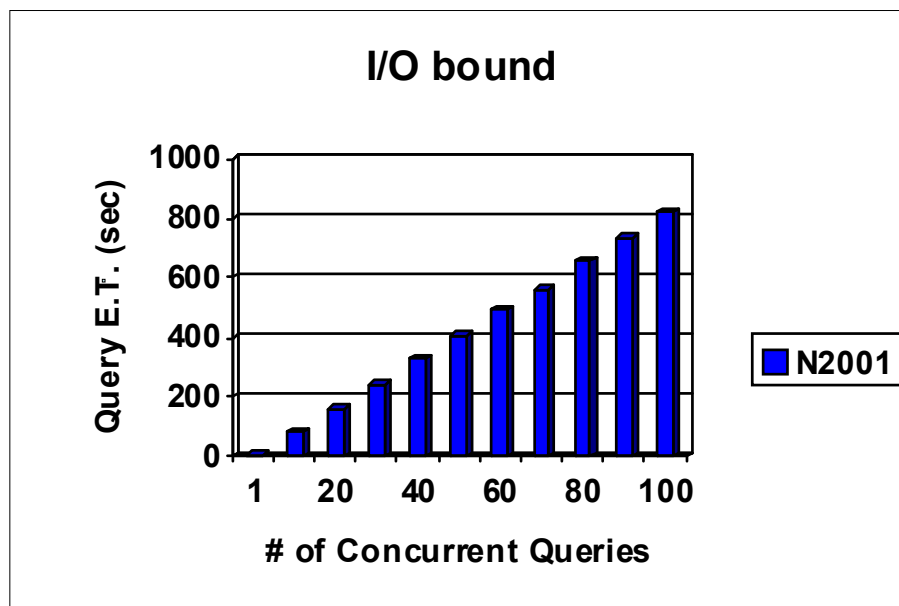


Figure 8-12 Concurrent I/O bound queries on N2001

When serial query is executed on N2001, single query elapsed time is about 8.9 seconds. When query is executed on N2001 concurrently, the average query elapsed time is about 8.6 seconds.

No query was on the Netezza wait queue for these tests.

Figure 8-13 shows the comparison of concurrency tests for I/O bound queries on N2001 versus N1001.

The query executes a full table scan of 9 billion rows of the SALES_FACT table.

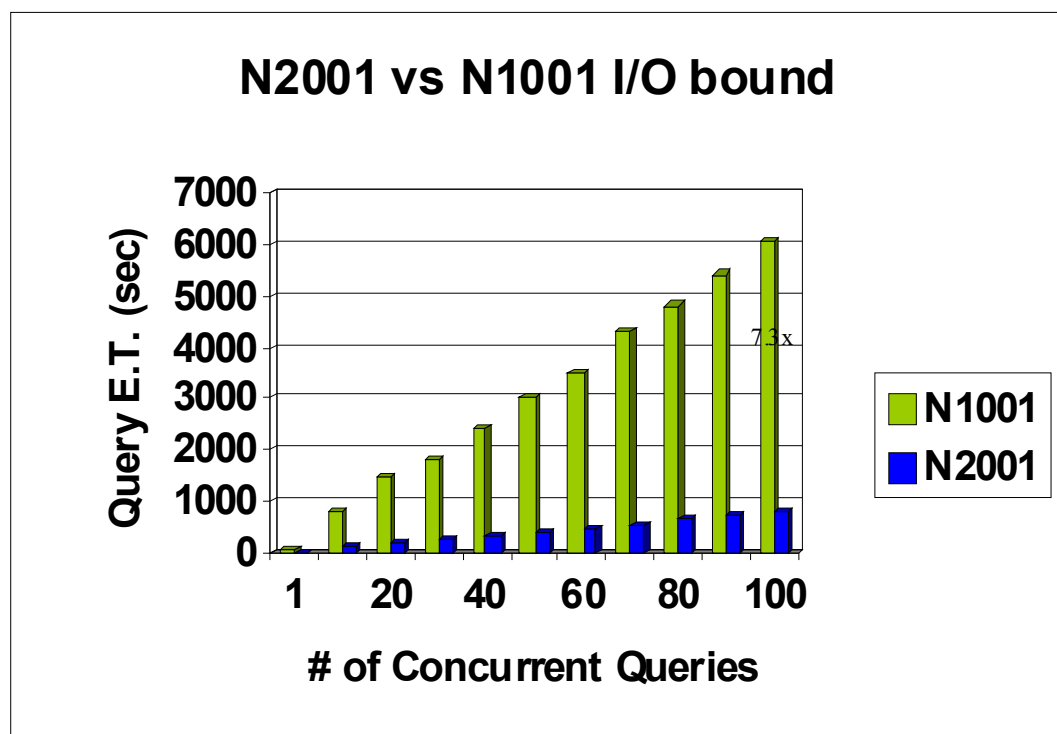


Figure 8-13 Concurrency tests for I/O bound queries on N2001 versus N1001

Query elapsed time scale linearly with the number of concurrent queries on N2001. Query elapsed time scale close to linear on N1001.

When query is executed on N1001, the single query elapsed time is 31 seconds.

When query is executed on N2001, the single query elapsed time is 8.9 seconds.

The performance improvement for single I/O bound query on N2001 versus N1001 is roughly 3.3x. This improvement is independent from the N1001 migration from NPS6 or NPS7. The I/O bound COUNT_BIG(*) query with no predicate and no aggregation does not benefit from the performance improvements of NPS7. Query CPU usage on zEC12 is negligible because the query is routed to the N2001 or N1001.

When N2001 and N1001 are not available or when query is not accelerated, the query is executed on zEC12 with 10 CPs and query parallelism degree 20. Query elapsed time on zEC12 with 10 CPs is about 37 seconds, and the CPU time on zEC12 is 328 seconds.

A few queries were put on the Netezza job queue starting at 55 concurrent query test and beyond on N1001. No query was on the job queue for all concurrency tests on N2001. This is good concurrency improvement on N2001 versus N1001.

Results of scenario #1

Serial I/O bound, full table scan query improved 3.3x on N2001 versus N1001. This is the expected improvement from the faster disk scan rate and more disks on N2001.

Concurrent I/O bound query improve from a faster scan rate, more memory, and a faster processor. Up to 7.3x improvement observed on N2001 with many concurrent COUNT_BIG(*) queries.

8.5.2 Scenario #2: CPU bound query

In this scenario, we ran concurrent CPU bound, full scan queries with poor, or no predicate selectivity with expensive joins and aggregation.

The query selects from the SALES_FACT with expensive joins and aggregation. We list the complex query in Example A-1 on page 300. The selectivity of the predicates is poor. The SALES_FACT table has 9 billion rows in 100 partitions. The database is about 1 TB.

Each test has one or multiple (the same) complex queries. The same CPU bound query ran serially and concurrently.

The measure query elapsed time is between the start of the first query and the end of the last query for concurrent runs.

Figure 8-14 shows the comparison of concurrency tests for CP bound queries on N2001 versus N1001.

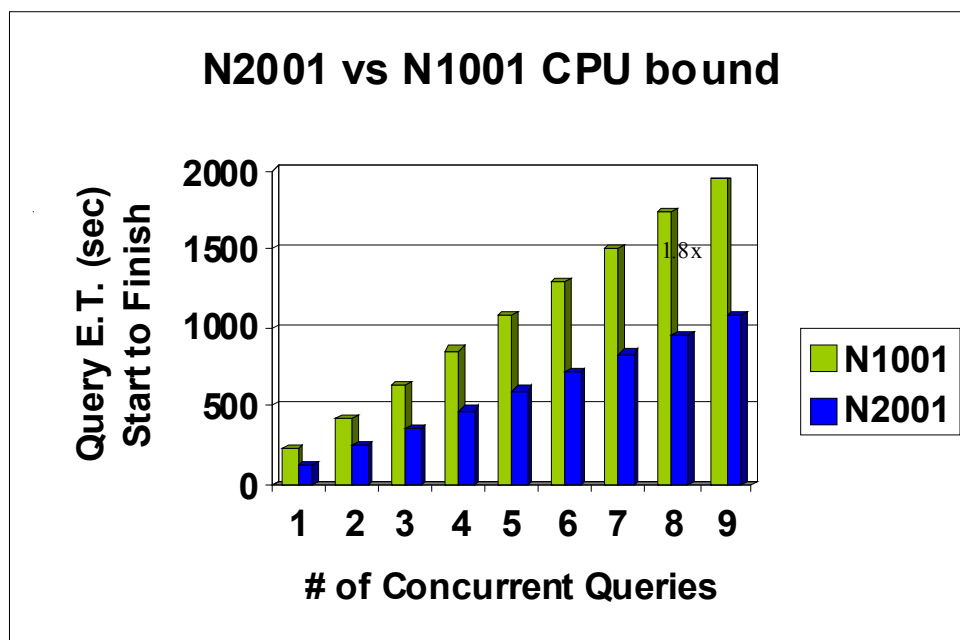


Figure 8-14 Comparing N2001 versus N1001 with concurrency test

Query elapsed time scale linearly with the number of concurrent queries on N2001. Query elapsed time scale close to linear on N1001.

The average CPU utilization on worker nodes is about 99 - 100%.

When query is executed on N1001, the single query elapsed time is 229 seconds.

When query is executed on N2001, the single query elapsed time is 125 seconds.

The performance improvement for single CP bound query on N2001 versus N1001(NPS6) is roughly 1.8x. This is relatively close to the 2x improvement on FPGA cores for N2001 versus

N1001. The performance improvement for single CP bound query on N2001 versus N1001 (NPS7) is roughly 1.5x in this scenario. The query performance improved about 18% on N1001 after the upgrade from NPS6 to NPS7. The performance improvement came from CPU reductions for multiple users and other enhancements in NPS7.

Query CPU usage on zEC12 is negligible because the query is executed on N2001 or N1001.

Figure 8-15 shows that concurrent CP bound queries scale linearly on N2001. These queries are very CP intensive as the average CPU utilization on worker nodes was about 99 - 100%.

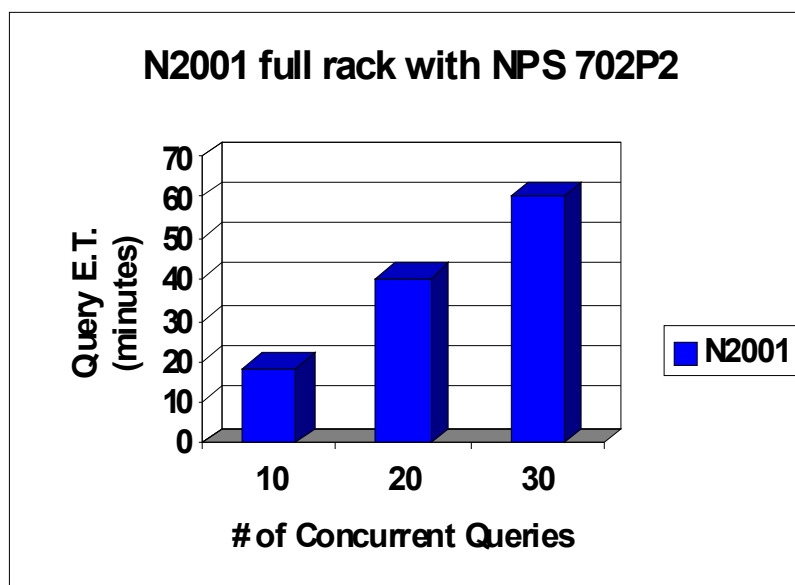


Figure 8-15 Scalability of concurrent queries on N2001

Query elapsed time scale linearly with the number of concurrent queries on N2001.

Average CPU utilization on worker nodes is about 99 - 100%

Results of scenario #2

CP bound, full scan queries with poor or no selectivity, improved 1.8x on N1001 (NPS6) and 1.5x on N1001(NPS7). Improvements are due to a faster processor and other enhancements in N2001.

Query performance scale linearly on N2001 and N1001.

It does not allow page granular zone maps or other optimizations to be used in this query due to full scan with poor or no selectivity.

8.5.3 Scenario #3: Mixed: I/O and CPU bound query

In this scenario, we ran concurrent IO bound queries and CPU bound queries as described earlier. We ran the same number of I/O bound queries and CPU bound queries in each concurrency test.

Measurement of query elapsed time is between the start of the first query and the end of the last query for concurrent runs.

The smallest SALES_FACT table used by I/O bound queries has 9 billion rows in 100 partitions. The largest SALES_FACT table used for CP bound queries has 90 billion rows in 512 partitions.

For concurrent scan queries, the same COUNT_BIG(*) query ran one or multiple times.

Figure 8-16 summarizes the results.

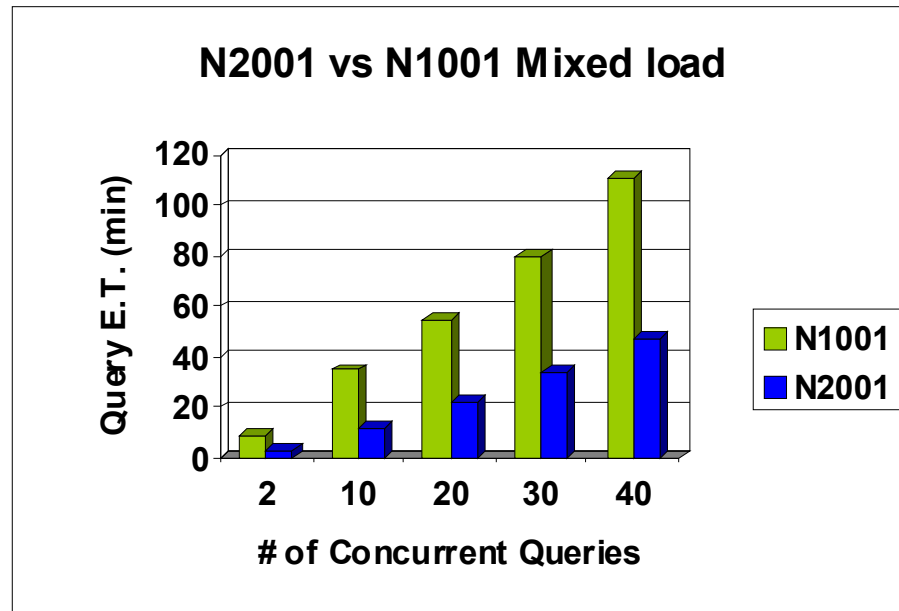


Figure 8-16 Mixed: I/O and CPU bound query: N2001 versus N1001

Query elapsed time scale linearly with the number of concurrent queries on N2001.

Average CPU utilization on worker nodes is 12 - 100%.

The elapsed time for concurrent one I/O bound query and one CP bound query is 8.8 seconds on N1001. The elapsed time for concurrent one I/O bound query and one CP bound query is 3.6 seconds on N2001. The improvement on N2001 versus N1001 is 2.4x.

Similar 2.4x performance improvement was observed for N2001 versus N1001 at higher concurrency tests.

Query elapsed time scale linearly with the number of concurrent queries on N2001 and N1001.

Results of scenario #3

Concurrent one I/O bound query with one CPU bound query improved 2.4x on N2001 versus N1001. The improvement comes from a faster processor and other hardware improvements.

Similar performance improvement is obtained at higher concurrency tests.

8.5.4 Scenario #4: Selective query with equal predicates benefitting from zone maps

In this scenario, we ran queries with selective, equal predicates. Queries have two equal predicates on dimension key columns that target search for a few rows in data slices. One row in the answer set is out of a 10 TB database. We list the query in Example A-2 on page 302.

We expect benefits from page granular zone maps in NPS 7 and the hardware improvements.

We measure elapsed time from the start of the first query to the end of the last query.

The concurrency tests use queries that have two equal predicates on two-dimension key columns. Queries also have other predicates. Each equal predicate has a different literal value for concurrent queries.

Figure 8-17 summarizes the results.

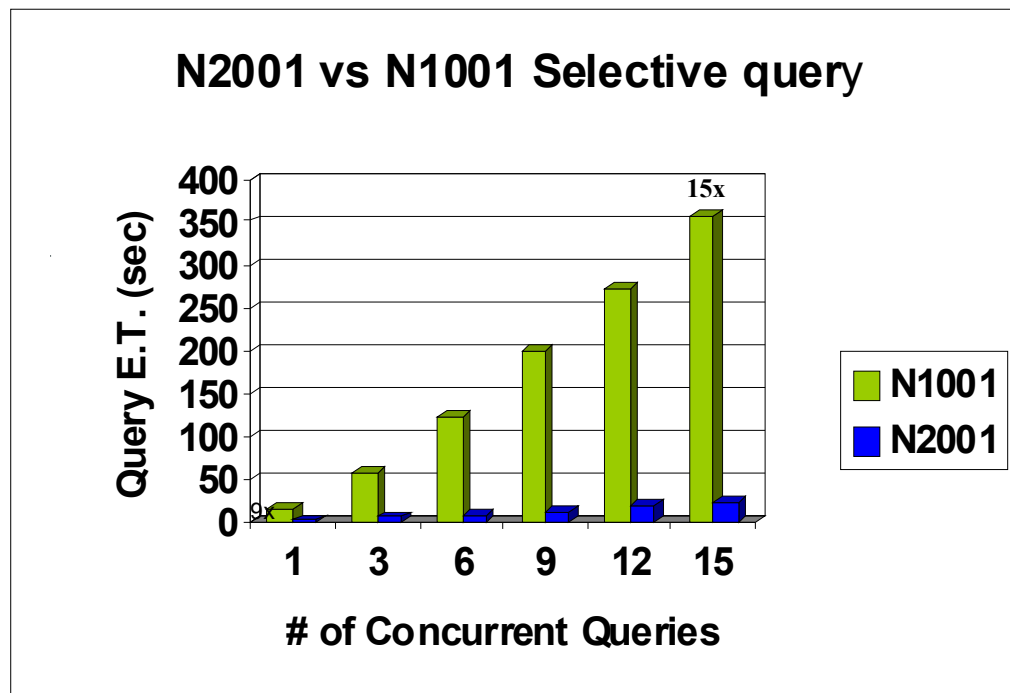


Figure 8-17 Selective query with equal predicates: N2001 versus N1001

For serial query tests, the average improvement in query elapsed time is 9x on N2001 versus N1001. For three, six, nine, twelve, and fifteen concurrent query tests, the average improvement in completion time is 15x. Improvement from N2001 hardware and NPS 702 software contributed to the up to 15x speedup. We can attribute one-third of the improvement to the page granular zone maps in NPS 702, and about two-thirds to N2001 hardware.

For the 15 query concurrency tests, the average CPU utilization on worker nodes was 21% on N2001 and 10% on N1001.

Results of scenario # 4

Serial query with very good selectivity improved 9x on N2001 versus N1001 with NPS6. This is a good improvement due to a faster disk scan rate, bigger RAM, and page-granular zone maps (scan less data via zone maps, enhancement in NPS 7).

Concurrent I/O bound queries have great improvement from a faster scan rate, bigger RAM, page-granular zone maps, and faster CPU. We observe up to 15x improvement for 15 concurrent queries.

8.5.5 Scenario #5: Directed data processing

This scenario shows the benefit of directed data processing in NPS 7. Directed data processing provides advantages similar to the limited partition scan in DB2 for z/OS.

The queries used in concurrency tests join the customer table (13.8 million rows) and fact table (10.3 billion rows) on the customer ID columns and the query search for different customer IDs in each of the concurrent query. The two tables are co-located on customer ID columns.

See Figure 8-18 for the results of our measurements. N2001 B is the case of no distribution key. N2001 A is the case with a distribution key and directed data processing.

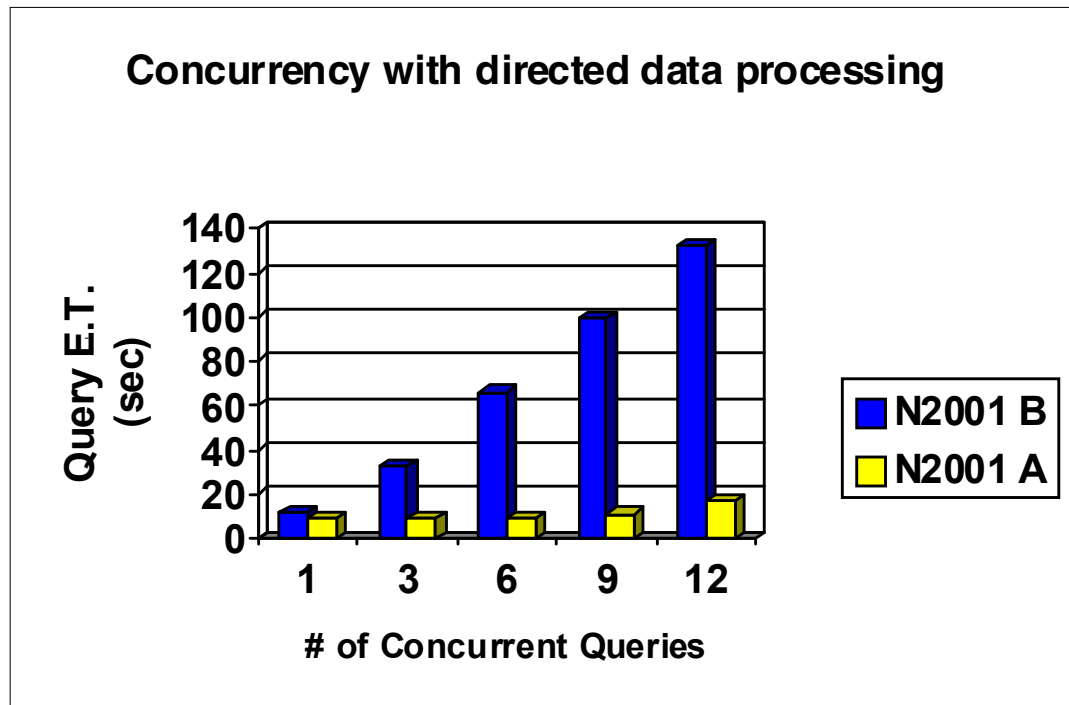


Figure 8-18 Directed data processing scenario

Single query without concurrency, completed in 9 seconds on N2001. In fact, all 12 queries completed in 9 seconds in serial tests. For the three concurrent queries test, all three completed in 9 seconds as if there was only one query. The same result for the six concurrent queries test. All six completed in 9 seconds concurrently. For the nine concurrent queries test, the completion time is 11 seconds. Only one query finished in 11 seconds, the other eight queries finished in 9 seconds. For the 12 queries concurrency test, 10 queries completed in 9 seconds and two queries completed in 17 seconds.

The directed data processing enhancement in NPS 7 is very effective in improving concurrency in the targeted queries. The improvement is nine times in nine queries concurrency tests. The directed data processing improvement on N1001 is smaller because

the number of data slices is much less on N1001 than on N2001, and other hardware differences.

8.5.6 Query performance summary

In summary, the N2001 with NPS 7 is scaling linearly and provides a range of improvements from 1.8 to 15 times when compared to N1001. We have observed:

- ▶ 3.3x improvement in serial, I/O bound, table scan query. Observed up to 7.3x improvement in high concurrent table scan queries on N2001.
- ▶ 1.8x improvement in serial and concurrent CPU bound, complex queries on N2001.
- ▶ 2.4x improvement in mixed concurrent I/O bound and CPU bound queries on N2001.
- ▶ Up to 15x query performance improvement on N2001 for concurrent queries with good selectivity.
- ▶ Query performance and concurrency improvement from a faster disk scan rate, bigger RAM, and faster and more processors on N2001 versus N1001. Selective queries benefit from improvements in NPS 7.
- ▶ For targeted, selective workloads, directed data processing has a substantial advantage in query performance, particularly on N2001. The improvement is 9x in the nine query concurrency tests.



Monitoring DB2 Analytics Accelerator environments

In this chapter, we describe the integration of the DB2 Analytics Accelerator in the DB2 for z/OS monitoring environment.

Support of DB2 of the Accelerator instrumentation is introduced by new performance counters in instrumentation facility component identifier (IFCID) 2 and 3 that are reported in Batch Reports (Accounting, Statistics, and Record Trace).

DB2 commands have been extended for the management and monitoring of query Accelerators.

DB2 Analytics Accelerator Studio allows query management and query monitoring. For information about query management, see Chapter 2, “Using the Studio client to manage the environment” on page 11. For information about query monitoring, see Chapter 5, “Query acceleration management” on page 121.

This chapter explains how to use this information to monitor sample scenarios and the environment. The following topics are described:

- ▶ Monitoring the environment
- ▶ Monitoring low latency updates
- ▶ Reporting and monitoring using OMPE
- ▶ Cataloging a DB2 for z/OS database as an ODBC data source

9.1 Monitoring the environment

From the performance perspective, a DB2 Analytics Accelerator-enabled DB2 environment can provide the following benefits:

- ▶ Reduced query elapsed time when offloading SQL
- ▶ Reduced CPU utilization on System z for applicable query workloads
- ▶ Improved system throughput by reduction of overall resource utilization in DB2 and System z

Note the following fundamental concepts about the DB2 Analytics Accelerator instrumentation:

- ▶ All the DB2 Analytics Accelerator accounting and statistics information is routed through DB2.
- ▶ The DB2 Analytics Accelerator instrumentation is added to DB2 through the extension of the current traces. No additional IFCID is introduced.

In addition to DB2 traces information, DB2 Analytics Accelerator commands help to monitor DB2 Analytics Accelerator activity in a more online fashion. In this chapter, we provide details about monitoring and reporting DB2 Analytics Accelerator and DB2 performance.

To support monitoring the DB2 Analytics Accelerator, DB2 introduced new performance counters in IFCID 2 and 3 that can be reported in IBM Tivoli OMEGAMON XE for DB2 Performance Expert (OMPE) Batch Reports including Accounting, Statistics, and Record Trace.

Important: No special traces are required to collect DB2 Analytics Accelerator activity. DB2 instrumentation support (traces) for DB2 Analytics Accelerator does not require additional classes or IFCIDs to be started. Data collection is implemented by adding new fields to the existing trace classes after application of the required software maintenance.

IBM Tivoli OMPE on z/OS provides batch reporting for DB2 Analytics Accelerator:

- ▶ Batch Statistics Trace/Report of DB2 Analytics Accelerator used by DB2 subsystem
- ▶ Batch Accounting of applications with DB2 Analytics Accelerator accelerated SQL queries and Accelerator-specific performance metrics
- ▶ Batch Record Trace reporting on single DB2 trace record with Accelerator-specific metrics

The DB2 statistics trace and the Accelerator

The DB2 statistics class 1 trace provides the following information:

- ▶ The states of the Accelerators that are in use
- ▶ The amount of processing time that is spent in Accelerators
- ▶ Counts of the amount of sent and received information
- ▶ Counts of the number of times that queries were successfully and unsuccessfully processed by Accelerators

Figure 9-1 shows the relation between DB2 and the Accelerator, and how the instrumentation data moves from the Accelerator into DB2.

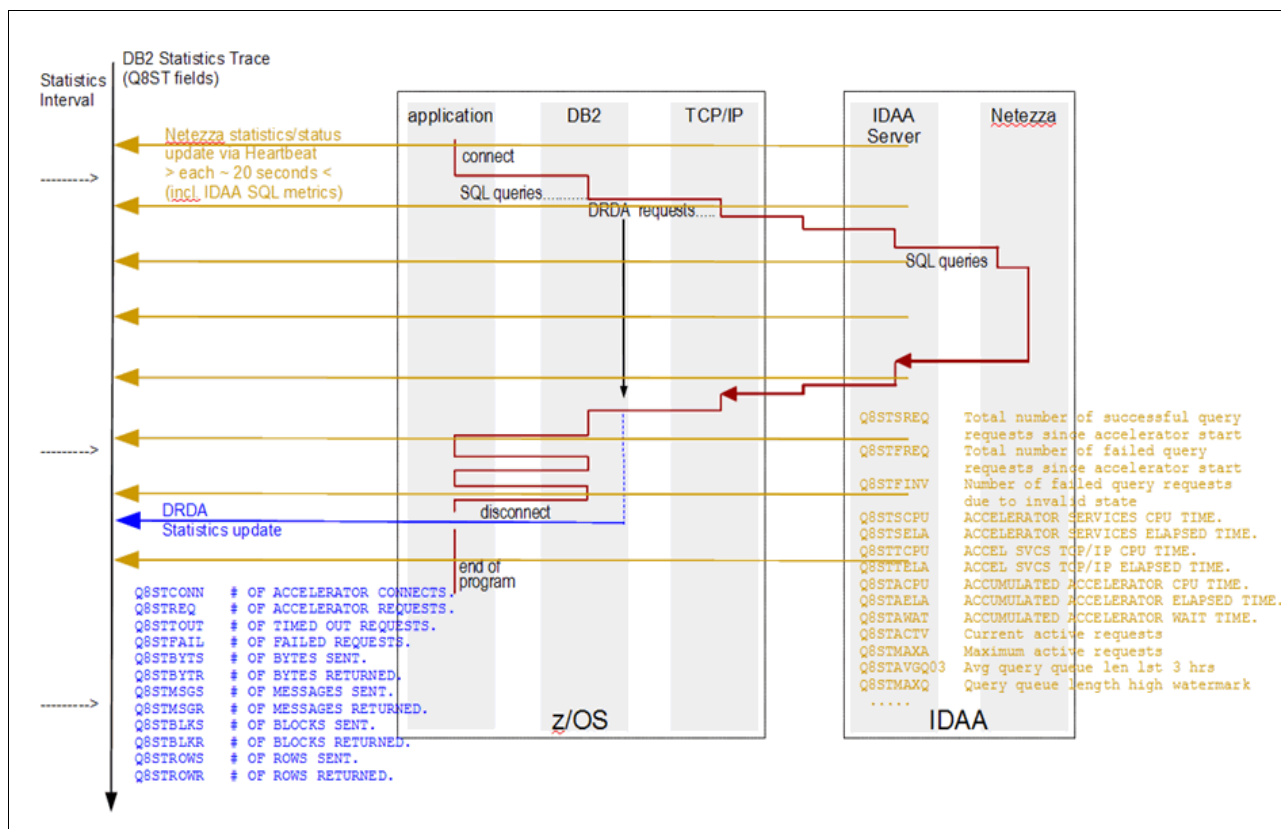


Figure 9-1 The DB2 statistics traces and the Accelerator

The DB2 Analytics Accelerator statistics and status are updated through the DB2 Analytics Accelerator heartbeat at 20-second intervals. This also includes DB2 Analytics Accelerator SQL metrics, which are transferred in an open interface (Distributed Relational Database Architecture (DRDA) extension) between DB2 and the Accelerators.

The OMPE statistics report displays DB2 Analytics Accelerator statistics, when available. No special report command syntax is required, as shown in Example 9-1.

Example 9-1 OMPE statistics report layout long command example

```
GLOBAL
  TIMEZONE (- 01:00)
  FROM(02/22/12,10:00),TO(02/22/12,10:30)
  STATISTICS
  REPORT
  LAYOUT (LONG)
EXEC
```

Tip: The OMPE STATISTICS LAYOUT (SHORT) command does not report DB2 Analytics Accelerator statistics. Use LAYOUT (LONG) instead.

Example 9-2 shows the Accelerator section of an OMPE Statistics Long report.

Example 9-2 OMPE Statistics Long report showing the Accelerator section

1	LOCATION: DWHDA12	OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V510)	PAGE: 1-31
	GROUP: N/P	STATISTICS REPORT - LONG	REQUESTED FROM: 02/22/12 10:00:00
	MEMBER: N/P		TO: 02/22/12 10:30:00
	SUBSYSTEM: DA12		INTERVAL FROM: 02/22/12 10:00:22
	DB2 VERSION: V10	SCOPE: MEMBER	TO: 02/22/12 10:26:38
----- HIGHLIGHTS -----			
INTERVAL START	: 02/22/12 10:00:22.23	SAMPLING START: 02/22/12 10:00:22.23	TOTAL THREADS : 2060.00
INTERVAL END	: 02/22/12 10:26:38.32	SAMPLING END : 02/22/12 10:26:38.32	TOTAL COMMITS : 19995.00
INTERVAL ELAPSED:	26:16.093702	OUTAGE ELAPSED: 0.000000	DATA SHARING MEMBER: N/A
IDAATF3	ACCELERATION	QUANTITY	IDAATF3 CONTINUED QUANTITY

QUERIES SUCCESSFULLY EXECUTED	24.00	AVG QUEUE LENGTH (LAST 3 HRS)	1.68
QUERIES FAILED TO EXECUTE	0.00	AVG QUEUE LENGTH (LAST 24 HRS)	1.68
ACCELERATOR IN INVALID STATE	0.00	MAXIMUM QUEUE LENGTH	16.00
CURRENTLY EXECUTING QUERIES	5.74	AVG QUEUE WAIT ELAPSED TIME	0.000001
MAXIMUM EXECUTING QUERIES	24.00	MAX QUEUE WAIT ELAPSED TIME	0.000083
CONNECTS TO ACCELERATOR	44.00	WORKER NODES	2.22
REQUESTS SENT TO ACCELERATOR	88.00	WORKER NODES AVG CPU UTILIZATION (%)	7.65
TIMED OUT	0.00	COORDINATOR AVG CPU UTILIZATION (%)	0.31
FAILED	0.00		
BYTES SENT TO ACCELERATOR	354814.00	DISK STORAGE AVAILABLE (MB)	5948861.42
BYTES RECEIVED FROM ACCELERATOR	254580.00	IN USE (%)	5.81
MESSAGES SENT TO ACCELERATOR	484.00	IN USE FOR DATABASE (MB)	354179.00
MESSAGES RECEIVED FROM ACCEL	484.00	DATA SLICES	16.31
BLOCKS SENT TO ACCELERATOR	0.00	DATA SKEW (%)	91.69
BLOCKS RECEIVED FROM ACCELERATOR	0.00		
ROWS SENT TO ACCELERATOR	0.00	PROCESSORS	17.79
ROWS RECEIVED FROM ACCELERATOR	0.00		
TCP/IP SERVICES ELAPSED TIME	4:46:07.281505	ELAPSED TIME IN ACCELERATOR	4:31:41.389834
WAIT TIME IN ACCELERATOR	1.665544	CPU TIME SPENT IN ACCELERATOR	2:14.299979

For a detailed explanation of all the report's fields, see *IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS Report Reference Version 5.1.0*, SH12-6921.

The relevant fields are listed and defined here:

- **QUERIES SUCCESSFULLY EXECUTED:** This field indicates the number of queries (sent by this DB2 system since Accelerator start) that were successfully executed in the Accelerator.
- **QUERIES FAILED TO EXECUTE:** This field indicates the number of queries (sent by this DB2 system since Accelerator start) that failed to be successfully executed for any reason, including the Accelerator being in an invalid state.
- **MAXIMUM EXECUTING QUERIES:** This field indicates the maximum number of queries executing in the Accelerator at any time since Accelerator start. This includes the queries from all DB2 systems connected to this Accelerator.
- **WAIT TIME IN ACCELERATOR:** This field indicates the accumulated wait time spent in the Accelerator when executing requests from the DB2 subsystem.
- **ELAPSED TIME IN ACCELERATOR:** This field indicates the accumulated elapsed time spent in the Accelerator when executing requests from the DB2 subsystem.
- **CPU TIME SPENT IN ACCELERATOR:** This field indicates the accumulated CPU time spent in the Accelerator when executing requests from the DB2 subsystem.

An OMPE Statistics Long report showing DB2 Analytics Accelerator-related command activity is shown in Example 9-3.

Example 9-3 OMPE Statistics report showing DB2 Analytics Accelerator command counters

1	LOCATION: DWHDA12	OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V510)				PAGE: 1-6	
	GROUP: N/P	STATISTICS REPORT - LONG				REQUESTED FROM: 02/22/12 10:00:00	
	MEMBER: N/P					TO: 02/22/12 10:30:00	
	SUBSYSTEM: DA12					INTERVAL FROM: 02/22/12 10:00:22	
	DB2 VERSION: V10	SCOPE: MEMBER				TO: 02/22/12 10:26:38	
----- HIGHLIGHTS -----							
INTERVAL START : 02/22/12 10:00:22.23		SAMPLING START: 02/22/12 10:00:22.23		TOTAL THREADS : 2060.00			
INTERVAL END : 02/22/12 10:26:38.32		SAMPLING END : 02/22/12 10:26:38.32		TOTAL COMMITS : 19995.00			
INTERVAL ELAPSED: 26:16.093702		OUTAGE ELAPSED: 0.000000		DATA SHARING MEMBER: N/A			
DB2 COMMANDS		QUANTITY	/SECOND	DB2 COMMANDS	CONTINUED	QUANTITY	/SECOND
-----				-----			
DISPLAY DATABASE		0.00	0.00	MODIFY TRACE		0.00	0.00
...							
DISPLAY PROFILE		0.00	0.00				
DISPLAY ACCEL		19.00	0.01	ACCESS DATABASE		1.00	0.00
ALTER BUFFERPOOL		0.00	0.00	UNRECOGNIZED COMMANDS		1.00	0.00
ALTER GROUPBUFFERPOOL		0.00	0.00				
ALTER UTILITY		0.00	0.00	TOTAL		2032.00	1.29
START DATABASE		0.00	0.00				
...							
START PROFILE		0.00	0.00				
START ACCEL		1.00	0.00				
STOP DATABASE		0.00	0.00				
STOP TRACE		0.00	0.00				
...							
STOP ACCEL		0.00	0.00				

OMPE Record Trace reports and the Accelerator

OMPE Record Trace reports show IFCID information as presented by DB2. This IFCID information is used to produce other OMPE reports. The difference is that for other reports, this information can be interpreted, manipulated, or not included. Use Record Trace reports if the information in other reports does not provide the required level of detail.

An OMPE Record Trace report including IFCID 002, contains an Accelerator Data section. Example 9-4 shows the OMPE report command that we used to create a Record Trace report.

Example 9-4 OMPE Record Trace report command

```
DB2PM GLOBAL (TIMEZONE(+5)
INCLUDE(SUBSYSTEM(DZA1))
INCLUDE (IFCID (2)))
RECTRACE
TRACE
LEVEL(LONG)
EXEC
```

Example 9-5 shows the Accelerator section of the generated report.

Example 9-5 OMPE Record Trace report output

ACCELERATOR DATA			
PRODUCT ID:	AQT03010		
SERVER ID:	DZA1IDAA		
STATE:	ONLINE		
QUERIES SUCCESSFULLY EXECUTED	3688	AVG QUEUE LENGTH (LAST 3 HOURS)	0
QUERIES FAILED TO EXECUTE	62	AVG QUEUE LENGTH (LAST 24 HOURS)	2
ACCELERATOR IN INVALID STATE	0	MAXIMUM QUEUE LENGTH	34
CURRENTLY EXECUTING QUERIES	0	AVG QUEUE WAIT ELAPSED TIME	0.051000
MAXIMUM EXECUTING QUERIES	41	MAX QUEUE WAIT ELAPSED	1.877000
CONNECTS TO ACCELERATOR	453	WORKER NODES	12
REQUESTS SENT TO ACCELERATOR	929	WORKER NODES AVG CPU UTILIZATION (%)	1.00
TIMED OUT	0	COORDINATOR AVG CPU UTILIZATION (%)	2.00
FAILED	0		
BYTES SENT TO ACCELERATOR	3779750	DISK STORAGE AVAILABLE (MB)	33557184
BYTES RECEIVED FROM ACCELERATOR	28862008	IN USE (%)	12.39
MESSAGES SENT TO ACCELERATOR	5006	IN USE FOR DATABASE (MB)	3532751
MESSAGES RECEIVED FROM ACCELERATOR	5019	DATA SLICES	92
BLOCKS SENT TO ACCELERATOR	0	DATA SKEW	28.29
BLOCKS RECEIVED FROM ACCELERATOR	42		
ROWS SENT TO ACCELERATOR	0	PROCESSORS	96
ROWS RECEIVED FROM ACCELERATOR	3353001		
TCP/IP SERVICES ELAPSED TIME	3 19:04:14.75084	ELAPSED TIME IN ACCELERATOR	12:25.015730
WAIT TIME IN ACCELERATOR	0.751139	CPU TIME SPENT IN ACCELERATOR	4:15.954000

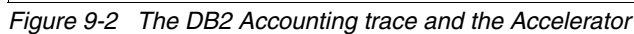
In addition to the information provided in the OMPE Statistics long report, the Record Trace report provides the field, STATE. This field shows the current Accelerator at the trace interval. State can be:

- ▶ 0 = INITIALIZED
- ▶ 1 = ONLINE
- ▶ 2 = PAUSED
- ▶ 3 = OFFLINE
- ▶ 4 = STOPPED
- ▶ 5 = MAINTENANCE
- ▶ 6 = DOWN
- ▶ 7 = UNKNOWN

The DB2 Accounting trace and the Accelerator

DB2 accounting class 1 trace records provide information about how often Accelerators were used, and how often Accelerators failed. The DB2 Analytics Accelerator accounting instrumentation is added to the DB2 traces. Figure 9-2 on page 213 illustrates how the information flows between DB2 and the DB2 Analytics Accelerator.

The DB2 Analytics Accelerator provides the values in an architected extension to DRDA, resulting in an open interface between DB2 and query Accelerators.



Example 9-6 OMPE Accounting LAYOUT(LONG) command

Tip: The **OMPE ACCOUNTING LAYOUT (SHORT)** command does not report DB2 Analytics Accelerator statistics. Use the **LAYOUT (LONG)** command instead.

Chapter 9. Monitoring DB2 Analytics Accelerator environments **213**

Example 9-7 OMPE Accounting Report Long Accelerator section

ACCELERATOR	IDENTIFIER	ACCELERATOR	AVERAGE	TOTAL	ACCELERATOR	AVERAGE	TOTAL
PRODUCT	AQT02012	OCCURRENCES	1.00	24	ELAPSED TIME		
SERVER	IDAATF3	CONNECTS	1.00	24	SVCS TCP/IP	6:31.492821	2:36:35.8277
		REQUESTS	2.00	48	ACCUM ACCEL	5:56.198064	2:22:28.7535
		TIMED OUT	0.00	0	CPU TIME		
		FAILED	0.00	0	SVCS TCP/IP	0.000341	0.008187
		SENT			ACCUM ACCEL	3.357125	1:20.570989
		BYTES	8026.33	192632	WAIT TIME		
		MESSAGES	11.00	264	ACCUM ACCEL	0.037185	0.892430
		BLOCKS	0.00	0			
		ROWS	0.00	0	DB2 THREAD		
		RECEIVED			CLASS 1		
		BYTES	7467.00	179208	ELAPSED	N/A	10:47:16.523
		MESSAGES	11.00	264	CP CPU	N/A	16:36.545365
		BLOCKS	0.00	0	SE CPU	N/A	0.000000
		ROWS	0.00	0	CLASS 2		
					ELAPSED	N/A	2:47:43.0033
					CP CPU	N/A	16:35.173834
					SE CPU	N/A	0.000000

Alternatively, you can use the OMPE Accounting **LAYOUT** subcommand option **ACCEL** to report on DB2 Analytics Accelerator accounting only. A syntax sample is shown in Example 9-8.

Example 9-8 OMPE Accounting LAYOUT(ACCEL) command

```
GLOBAL
  TIMEZONE (+ 01:00)
ACCOUNTING
  TRACE
  LAYOUT(ACCEL)
  INCLUDE(SUBSYSTEM(DA12))
EXEC
```

Example 9-9 shows this kind of layout.

Example 9-9 OMPE Accounting Layout Accel

1	LOCATION: DWHDA12	OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V510)	PAGE: 1-28
	GROUP: N/P	ACCOUNTING TRACE - ACCEL	REQUESTED FROM: 02/16/12 09:00:00.00
	MEMBER: N/P		TO: 02/16/12 12:40:00.00
	SUBSYSTEM: DA12		ACTUAL FROM: 02/16/12 09:22:00.00
	DB2 VERSION: V10		
---- IDENTIFICATION -----			
ACCT TSTAMP:	02/16/12 10:35:44.63	PLANNAME:	BIBusTKS
BEGIN TIME :	02/16/12 10:20:08.13	PROD TYP:	COMMON SERV
END TIME :	02/16/12 10:35:44.63	PROD VER:	V9 R7 M5
REQUESTER :	::FFFF:9.152.86.	CORRNAME:	BIBusTKS
MAINPACK :	BIBusTKS	CORRNMBR:	erve
PRMAUTH :	IDAA3	CONNTYPE:	DRDA
ORIGAUTH :	IDAA3	CONNECT :	SERVER
		WLM SCL:	STCCMD
		LUW NET:	G9985641
		LUW LUN:	0BB4
		LUW INS:	120216092007
		LUW SEQ:	7
		ENDUSER :	Anonymous
		TRANSACTION:	RC3 - Report 3
		WSNAME :	'BLANK'
ACCELERATOR	IDENTIFIER	ACCELERATOR	TOTAL
PRODUCT	AQT02012	OCCURRENCES	1
SERVER	IDAATF3	CONNECTS	1
		REQUESTS	12
		TIMED OUT	0
		FAILED	0
		SENT	
		BYTES	9021
		MESSAGES	21
		BLOCKS	0
		ROWS	0
		RECEIVED	
		BYTES	597291
		MESSAGES	29
		BLOCKS	10
		ROWS	0
		ELAPSED TIME	
		SVCS TCP/IP	12:56.075917
		ACCUM ACCEL	0.000000
		CPU TIME	
		SVCS TCP/IP	0.001294
		ACCUM ACCEL	0.000000
		WAIT TIME	
		ACCUM ACCEL	0.000000
		DB2 THREAD	
		CLASS 1	
		ELAPSED	15:36.498126
		CP CPU	0.029813
		SE CPU	0.000000
		CLASS 2	
		ELAPSED	N/P
		CP CPU	N/P
		SE CPU	0.000000

The Accounting Accelerator report block is shown for each Accelerator that provided services to a DB2 thread. The block consists of three adjacent columns that contain the Accelerator identification, the activity-related counters, and the corresponding times.

The Accounting trace shows values and times for each Q8AC section. The Accounting report shows not only accumulated values and times, but also average values and times calculated for one occurrence. It shows the sum of a counter, or the time of all Q8AC sections processed, divided by the number of processed Q8AC sections.

For a complete explanation of the meanings of all the report fields, refer to *IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS Report Reference Version 5.1.0*, SH12-6921.

Tip: For updated versions of all tools' manuals, and OMEGAMON XE for DB2 Performance Expert in particular, refer to the DB2 Tools Product Page, available at: <http://www.ibm.com/support/docview.wss?uid=swg27020910#omegaxepe-lib>

The most significant fields are listed here:

SERVER	The Accelerator server identifier.
OCCURRENCES	The number of sections processed for the Accelerator. The name of this Accelerator is shown in the data block ACCELERATOR IDENTIFIER.
ELAPSED TIME - SVCS TCP/IP	The Accelerator services TCP/IP elapsed time measured in DB2. It starts when sending the requests to the Accelerator and ends when receiving the results from the Accelerator.
ELAPSED TIME - ACCUM ACCEL	The elapsed time spent in the Accelerator when executing requests from the DB2 subsystem.
CPU TIME - SVCS TCP/IP	The Accelerator services TCP/IP CPU time measured in DB2 for the amount of CPU consumed by the distributed data facility (DDF) service task to perform the SEND and RECEIVE to an Accelerator service. It does not account for the TCP/IP address CPU to route the message on to the network and receive the reply into the DDF task.
CPU TIME - ACCUM ACCEL	The CPU time spent in the Accelerator when executing requests from the DB2 subsystem.

DB2 and DB2 Analytics Accelerator communicate using DRDA. This activity can be found in the Distributed activity section of the OMPE Accounting report, as shown in Example 9-10.

In this example, **SERVER: IDAATF3** identifies the DB2 Analytics Accelerator appliance.

Example 9-10 OMPE Accounting report, distributed activity section

---- DISTRIBUTED ACTIVITY -----									
SERVER	: IDAATF3	CONVERSATIONS INITIATED:	1.00	#COMMT(1)SENT:	0	MESSAGES SENT :	11.0		
PRODUCT ID	: AQT	#CONVERSATIONS QUEUED :	0	#ROLLB(1)SENT:	0	MESSAGES RECEIVED:	11.0		
METHOD	: N/P	CONVERSATION TERMINATED:	0.00	SQL SENT :	3.00	BYTES SENT :	8026.3		
REQUESTER ELAP.TIME:	6:31.492821	#RLUP THREADS :	24	ROWS RECEIVED:	0.00	BYTES RECEIVED :	7467.0		
SERVER ELAPSED TIME:	N/A					BLOCKS RECEIVED :	0.0		
SERVER CPU TIME :	N/A								
DBAT WAITING TIME :	0.000000								
#DDF ACCESSES :	24								
#COMMIT(2) SENT :	N/A	#BACKOUT(2) SENT :	N/A	#BKOUT(2) R.R:	N/A	#LASTAGN.SENT :	N/		
		SUCCESSFULLY ALLOC.CONV:	N/A	TRANSACT.SENT:	N/A	STMT BOUND AT SER:	N/		

Not accounted time and the DB2 Analytics Accelerator

When a thread is executed in DB2, a high Not Accounted time can be an indication of a lack of CPU or of the workload being executed under a low priority service class. Examine the DB2 times distribution, including Not Accounted time, in the header of the OMPE Accounting Long Trace.

Example 9-12 shows the section of an OMPE Accounting Trace Long report for a distributed thread where the SQL was routed to the DB2 Analytics Accelerator.

Example 9-12 High DB2 Not Accounted time when off-loaded to DB2 Analytics Accelerator

1	LOCATION: DWHDA12	OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V510)			PAGE: 1-79	
	GROUP: N/P	ACCOUNTING TRACE - LONG			REQUESTED FROM: 02/22/12 10:00:00.	
	MEMBER: N/P				TO: 02/22/12 10:30:00.	
	SUBSYSTEM: DA12				ACTUAL FROM: 02/22/12 10:07:53.	
	DB2 VERSION: V10					
----- IDENTIFICATION -----						
ACCT TSTAMP:	02/22/12 10:10:41.43	PLANNAME:	CLP /hom	WLM SCL:	SCREPM	CICS NET: N/A
BEGIN TIME :	02/22/12 10:07:57.99	PROD TYP:	COMMON SERV			CICS LUN: N/A
END TIME :	02/22/12 10:10:41.43	PROD VER:	V9 R7 M5	LUW NET:	G9985641	CICS INS: N/A
REQUESTER :	::FFFF:9.152.86.	CORRNAME:	db2bp	LUW LUN:	E879	
MAINPACK :	SQLC2H22	CORRNMBR:	'BLANK'	LUW INS:	120222090753	ENDUSER : RI09
PRMAUTH :	IDAA3	CONNTYPE:	DRDA	LUW SEQ:	6	TRANSACTION: CLP /home/cognos/scripts/queries
ORIGAUTH :	IDAA3	CONNECT :	SERVER			WSNAME : RI09
ELAPSED TIME DISTRIBUTION				CLASS 2 TIME DISTRIBUTION		

APPL	=====			CPU	=====	
DB2	-----> 100%			SECPU	-----> 100%	
SUSP				NOTACC		
				SUSP		

TIMES/EVENTS	APPL(CL.1)	DB2 (CL.2)	IFI (CL.5)	CLASS 3 SUSPENSIONS	ELAPSED TIME	EVENTS

ELAPSED TIME	2:43.44200	2:43.43351	N/P	LOCK/LATCH(DB2+IRLM)	0.062513	8
NONNESTED	2:43.44200	2:43.43351	N/A	IRLM LOCK+LATCH	0.052726	5
STORED PROC	0.000000	0.000000	N/A	DB2 LATCH	0.009787	3
UDF	0.000000	0.000000	N/A	SYNCHRON. I/O	0.015186	7
TRIGGER	0.000000	0.000000	N/A	DATABASE I/O	0.015186	7
				LOG WRITE I/O	0.000000	0
CP CPU TIME	0.012219	0.012131	N/P	OTHER READ I/O	0.017551	6
AGENT	0.012219	0.012131	N/A	OTHER WRTE I/O	0.000000	0
NONNESTED	0.012219	0.012131	N/P	SER.TASK SWTCH	0.065887	6
STORED PRC	0.000000	0.000000	N/A	UPDATE COMMIT	0.000140	1
UDF	0.000000	0.000000	N/A	OPEN/CLOSE	0.047466	2
TRIGGER	0.000000	0.000000	N/A	SYSLGRNG REC	0.001146	1
PAR.TASKS	0.000000	0.000000	N/A	EXT/DEL/DEF	0.010767	1
				OTHER SERVICE	0.006367	1
SECP CPU	0.000000	N/A	N/A	ARC.LOG(QUIES)	0.000000	0
				LOG READ	0.000000	0

This report shows that almost all of the accounting interval time was spent in DB2, of which all is reported as Not Accounted time. The Distributed Activity section of the same report shows that the thread communicated with the DB2 Analytics Accelerator appliance for a period of time equal to the DB2 Not Accounted time. This is shown in Example 9-13.

Example 9-13 OMPE Accounting Trace Distributed activity section

----- DISTRIBUTED ACTIVITY -----					
SERVER	: IDAATF3	CONVERSATION TERMINATED:	N/A	NBR RLUP THREADS :	1
PRODUCT ID	: AQT	COMMT(1)SENT	: 0	MESSAGES SENT :	11
PRODUCT VERSION	: V2 R1 M2	ROLLB(1)SENT	: 0	MESSAGES RECEIVED:	11
METHOD	: N/P	SQL SENT	: 3	BYTES SENT :	8699
REQUESTER ELAP.TIME	: 2:43.208437	ROWS RECEIVED	: 0	BYTES RECEIVED :	3897
SERVER ELAPSED TIME	: N/A			BLOCKS RECEIVED :	0
SERVER CPU TIME	: N/A				
DBAT WAITING TIME	: N/A				
CONVERSATIONS INITIATED:	1				
CONVERSATIONS QUEUED :	0				
COMMIT(2) SENT :	N/A	SUCCESSFULLY ALLOC.CONV:	N/A	MSG.IN BUFFER :	N/A
BACKOUT(2) SENT :	N/A	MAX OPEN CONVERSATIONS :	N/A	PREPARE SENT :	N/A

CONT->LIM.BL.FTCH SWITCH:	N/A	LAST AGN.SENT :	N/A
COMMIT(2) RESP.RECEIVED:	N/A	STMT BOUND AT SER:	N/A
BKOUT(2) R.R :	N/A	FORGET RECEIVED :	N/A
TRANSACT.SENT :	N/A		

Note the fields highlighted in bold in Example 9-14, which shows the Accelerator section:

- ▶ DB2 THREAD, CLASS 1, ELAPSED 2:43.442000 corresponds to ELAPSED TIME / APPL(CL.1) in Example 9-12 on page 217.
- ▶ DB2 THREAD, CLASS 2, ELAPSED 2:43.433514 corresponds to ELAPSED TIME / DB2 (CL.2) in Example 9-12 on page 217.
- ▶ ELAPSED TIME, SVCS TCP/IP 2:43.208437 corresponds to REQUESTER ELAP.TIME in Example 9-13 on page 217.

In addition, this section shows ACCUM ACCEL 2:43.071440, the accumulated Accelerator times.

Example 9-14 OMPE Accounting Trace Accelerator activity section

ACCELERATOR	IDENTIFIER	ACCELERATOR	TOTAL	ACCELERATOR	TOTAL
-----	-----	-----	-----	-----	-----
PRODUCT	AQT02012	OCCURRENCES	1	ELAPSED TIME	
SERVER	IDAATF3	CONNECTS	1	SVCS TCP/IP	2:43.208437
		REQUESTS	2	ACCUM ACCEL	2:43.071440
		TIMED OUT	0	CPU TIME	
		FAILED	0	SVCS TCP/IP	0.000336
		SENT	0	ACCUM ACCEL	0.016000
		BYTES	8699	WAIT TIME	
		MESSAGES	11	ACCUM ACCEL	0.004840
		BLOCKS	0		
		ROWS	0	DB2 THREAD	
		RECEIVED		CLASS 1	
		BYTES	3897	ELAPSED	2:43.442000
		MESSAGES	11	CP CPU	0.012219
		BLOCKS	0	SE CPU	0.000000
		ROWS	0	CLASS 2	
				ELAPSED	2:43.433514
				CP CPU	0.012131
				SE CPU	0.000000

We found that distributed requests being routed to the Accelerator reports the time spent in the Accelerator as Not Accounted time. This kind of thread with a small result set reports almost 100% Not Accounted time. You must refer to the Accelerator section of the report, if any, to verify that the reported Not Accounted time was not really used in the Accelerator.

OMPE and DB2 Analytics Accelerator-related information

To obtain up-to-date information about OMPE, refer to the following sources:

- ▶ IBM Tivoli OMEGAMON XE for DB2 Performance Expert information center, available at:
http://pic.dhe.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=%2Fcom.ibm.omegamon.xe.pe_db2.doc_5.1.1%2Fko2welcome_pe.htm
- ▶ PDF versions of the documentation are made available at this site:
<http://www.ibm.com/support/docview.wss?uid=swg27020910#omegaxepe-lib>

Note: Changes shipped via service stream between releases might not yet be incorporated into the PDF file documentation, but they are documented in *techdocs*, which you can reach from the previously mentioned links.

9.1.1 The DB2 message DSNX881I

An Accelerator issues DB2 messages if the state of the hardware changes or if problems are encountered. These messages start with the prefix or qualifier, DSNX881I.

When an Accelerator has been connected successfully to a DB2 subsystem, and the Accelerator has been started by the **-START ACCEL** command or the corresponding function in IBM DB2 Analytics Accelerator Studio, a heartbeat connection is established between the Accelerator and that particular DB2 subsystem. Status information about the Accelerator is sent to the DB2 subsystem every 30 seconds.

You can view most of this information by using the **-DIS ACCEL DB2** commands. Other information cannot be viewed in this way, but is written to the z/OS system log (SYSLOG). The heartbeat information that appears in the system log is identified by the prefix, DSNX881I.

Each DSNX881I message is made up of the following parts, which occur in the order indicated in Example 9-15.

Example 9-15 DSNX881I message structure

```
DSNX881I  -<SSID> <MESSAGE-ID> <SEVERITY> <ACCELERATOR_MESSAGE_COUNTER>  
(<ACCELERATOR-TIMESTAMP>) <MESSAGE-TEXT>
```

The placeholders have the following meaning:

- ▶ **SSID:** Is the DB2 subsystem ID (SSID)
- ▶ **MESSAGE-ID:** A numeric ID for the specific error message. This ID can be used for system monitoring.
- ▶ **SEVERITY:**
 - I: Information message
 - W: Warning message
 - E: Error message
- ▶ **ACCELERATOR_MESSAGE_COUNTER:** An internal counter that increases with every additional error on the Accelerator. If the text after the DSNX881I qualifier is longer than 255 characters, another DSNX881I message is issued. All messages belonging together have the same <ACCELERATOR_MESSAGE_COUNTER> value. The <MESSAGE-TEXT> block of each subsequent message contains a sequel to the information in the previous message.
- ▶ **ACCELERATOR-TIMESTAMP:** The time when the error occurred on the Accelerator. The internal clock of the Accelerator is synchronized with the first DB2 subsystem that was connected to the Accelerator.
- ▶ **MESSAGE:** A textual description of the error.

For more information, see the online publication “Structure of DSNX881I Messages”:

<http://www.ibm.com/support/docview.wss?uid=swg27037905>

The length of a DSNX881I message does not exceed 255 characters. If more characters are needed, additional DSNX881I messages are written to the SYSLOG.

If a logical partition (LPAR) contains multiple DB2 subsystems that are connected to the same physical Accelerator, error messages are issued for every subsystem. If an Accelerator is paired with a data sharing group, only the first member of the group writes a message to the system log (SYSLOG).

If an LPAR hosts multiple DB2 subsystems that are paired with the same physical Accelerator, and the Accelerator was started in each DB2 subsystem (DB2 command **-START ACCEL(...)**), the system log of the LPAR will contain the same DSNX881I messages for each subsystem. That is, you see the same messages multiple times in the log, each time with a different SSID.

If an error occurs while the acceleration state of an Accelerator is Stopped, an error message is stored on the Accelerator. As soon as the Accelerator becomes available again in DB2, the stored error messages are sent to the DB2 subsystem. It might happen that a DSNX881I message reports a past problem that has already been fixed.

The mentioned online publication “Structure of DSNX881I Messages”, which is available at <http://www.ibm.com/support/docview.wss?uid=swg27037905>, provides details about the MESSAGE-IDs, SEVERITY, and MESSAGE-TEXT that are associated with a DSNX881I message. The following list is a sample of the available information:

- ▶ **MESSAGE-ID:** 1
- ▶ **Accelerator Event Category:** HostStateChange
- ▶ **NPS Event Category:** sysStateChanged
- ▶ **Expected MESSAGE-TEXT:** "NPS system <HOST> went from <previousState> to <currentState> at <eventTimestamp> <eventSource>. <notifyMsg> Event: <eventDetail>"
- ▶ **Severity:** I
- ▶ **Impact:** Availability of the Accelerator for query processing. Everything different from Online prevents the Accelerator from answering queries.
- ▶ **Action:** If <currentState> shows a value other than Online, run the **nzstart** command from the IBM DB2 Analytics Accelerator Configuration Console.

All MESSAGE-ID values are used in Version 2.1.x and in Version 3.1.0 of IBM DB2 Analytics Accelerator for z/OS, except for MESSAGE-ID 20, which is available in Version 3.1.0 only.

The “NPS Event Category” column of the previous list uses the names of template event rules as described in the “IBM Netezza System Administrator's Guide”. This guide is included in every “IBM Netezza NPS Software and Clients” fix pack that is available on IBM Fix Central. The fix packs contain the “IBM Netezza System Administrator's Guide” in PDF format.

Example 9-16 shows the message as published in the System z console when an NPS changes status from online to offline.

Example 9-16 DSNX881I message in system log

```
DSNX881I  -DZA1 1 I 16695 (09-May-13, 16:46:49 EDT) 517
NPS system isas-h2 went from online to offliningNow at 09-May-13,
16:46:49 EDT User initiated. Event: eventType: sysStateChanged
eventTimestamp: 09-May-13, 16:46:49 EDT eventArgs:
previousState=online, currentState=off
DSNX881I  -DZA1 1 I 16695 (09-May-13, 16:46:49 EDT) 518
liningNow, eventSource=user eventSource: User initiated event
```

Example 9-17 shows the same problem occurrence as documented in the DB2 MSTR address space log.

Example 9-17 DSNX881I in DB2 syslog

```
16.46.32 STC05580 DSNX881I  -DZA1 1 I 16695 (09-May-13, 16:46:49 EDT) 517
517                NPS system isas-h2 went from online to offliningNow at 09-May-13,
```

```

517          16:46:49 EDT User initiated. Event: eventType: sysStateChanged
517          eventTimestamp: 09-May-13, 16:46:49 EDT eventArgs:
517          previousState=online, currentState=off
16.46.32 STC05580 DSNX881I -DZA1 1 I 16695 (09-May-13, 16:46:49 EDT) 518
518          liningNow, eventSource=user eventSource: User initiated event

```

Example 9-18 shows the sequence of DSNX881I messages in the DB2 MSTR syslog at NPS initialization.

Example 9-18 DSNX881I in DB2 syslog: NPS initialization and status change

```

17.06.32 STC05580 DSNX881I -DZA1 1 I 16696 (09-May-13, 17:06:43 EDT) 225
225          NPS system isas-h2 went from discovering to initializing at
225          09-May-13, 17:06:43 EDT User initiated. Event: eventType:
225          sysStateChanged eventTimestamp: 09-May-13, 17:06:43 EDT eventArgs:
225          previousState=discovering, curren
17.06.32 STC05580 DSNX881I -DZA1 1 I 16696 (09-May-13, 17:06:43 EDT) 226
226          tState=initializing, eventSource=user eventSource: User initiated
226          event
17.07.32 STC05580 DSNX881I -DZA1 1 I 16697 (09-May-13, 17:07:47 EDT) 233
233          NPS system isas-h2 went from initializing to initialized at
233          09-May-13, 17:07:47 EDT User initiated. Event: eventType:
233          sysStateChanged eventTimestamp: 09-May-13, 17:07:47 EDT eventArgs:
233          previousState=initializing, curre
17.07.32 STC05580 DSNX881I -DZA1 1 I 16697 (09-May-13, 17:07:47 EDT) 234
234          ntState=initialized, eventSource=user eventSource: User initiated
234          event
17.07.32 STC05580 DSNX881I -DZA1 1 I 16698 (09-May-13, 17:07:49 EDT) 235
235          NPS system isas-h2 went from initialized to preOnlining at 09-May-13,
235          17:07:49 EDT User initiated. Event: eventType: sysStateChanged
235          eventTimestamp: 09-May-13, 17:07:49 EDT eventArgs:
235          previousState=initialized, current
17.07.32 STC05580 DSNX881I -DZA1 1 I 16698 (09-May-13, 17:07:49 EDT) 236
236          State=preOnlining, eventSource=user eventSource: User initiated event
17.07.32 STC05580 DSNX881I -DZA1 1 I 16699 (09-May-13, 17:07:53 EDT) 237
237          NPS system isas-h2 went from preOnlining to resuming at 09-May-13,
237          17:07:53 EDT User initiated. Event: eventType: sysStateChanged
237          eventTimestamp: 09-May-13, 17:07:53 EDT eventArgs:
237          previousState=preOnlining, currentSta
17.07.32 STC05580 DSNX881I -DZA1 1 I 16699 (09-May-13, 17:07:53 EDT) 238
238          te=resuming, eventSource=user eventSource: User initiated event
17.07.32 STC05580 DSNX881I -DZA1 1 I 16700 (09-May-13, 17:07:55 EDT) 239
239          NPS system isas-h2 went from resuming to online at 09-May-13,
239          17:07:55 EDT User initiated. Event: eventType: sysStateChanged
239          eventTimestamp: 09-May-13, 17:07:55 EDT eventArgs:
239          previousState=resuming, currentState=onlin
17.07.32 STC05580 DSNX881I -DZA1 1 I 16700 (09-May-13, 17:07:55 EDT) 240
240          e, eventSource=user eventSource: User initiated event

```

Sample scenario: communication error

The following lines describe our environment setup, and the feedback that is received in DB2 when there is a communication error between DB2 and an Accelerator.

Example 9-19 shows the contents of the SYSACCEL.SYSACCELERATORS table in our environment. In this case, the DB2 subsystem is connected to two Accelerators (DZA1IDAA and DZA1STPR) and has a virtual Accelerator defined (RAVIRTUE).

Example 9-19 SYSACCEL.SYSACCELERATORS

ACCELERATORNAME	LOCATION
-----	-----
DZA1IDAA	DZA1IDAA
DZA1STPR	DZA1STPR
RAVIRTUE	?

Each physical Accelerator is associated to a remote location in SYSIBM.LOCATIONS, as shown in Example 9-20.

Example 9-20 SYSIBM.LOCATIONS

LOCATION	LINKNAME	IBMREQD	PORT	TPN	DBALIAS	TRUSTED	SECURE
-----	-----	-----	-----	-----	-----	-----	-----
DZA1IDAA	DZA1IDAA	N	1400			Y	N
DZA1STPR	DZA1STPR	N	1400			Y	N

The relationship between a LOCATION and an IP address can be found in the DB2 catalog table, SYSIBM.IPNAMES, as shown in Example 9-21.

Example 9-21 SYSIBM.IPNAMES

LINKNAME	SECURITY_OUT	USERNAMES	IBMREQD	IPADDR
-----	-----	-----	-----	-----
DZA1IDAA	P	S	N	135.25.80.210
DZA1STPR	P	S	N	135.25.80.250

Example 9-22 shows the output of the **-DIS ACCEL(DZA1IDAA)DETAIL** command in our environment.

Example 9-22 -DIS ACCEL DETAIL command output example

```

DSNX810I  -DZA1 DSNX8CMD DISPLAY ACCEL FOLLOWS -
DSNX830I  -DZA1 DSNX8CDA
ACCELERATOR                                MEMB  STATUS  REQUESTS  ACTV  QUED  MAXQ
-----
DZA1IDAA                                DZA1  STARTED      5285    0    0    14
LOCATION=DZA1IDAA HEALTHY
DETAIL STATISTICS
  LEVEL  = AQT03010
  STATUS = ONLINE
  FAILED QUERY REQUESTS                        =      96
  AVERAGE QUEUE WAIT                          =     59 MS
  MAXIMUM QUEUE WAIT                          =    187 MS
  TOTAL NUMBER OF PROCESSORS                  =      96
  AVERAGE CPU UTILIZATION ON COORDINATOR NODES =     2.00%
  AVERAGE CPU UTILIZATION ON WORKER NODES    =     1.00%
  NUMBER OF ACTIVE WORKER NODES               =      12
  TOTAL DISK STORAGE AVAILABLE                 = 33557184 MB
  TOTAL DISK STORAGE IN USE                   =    12.40%
  DISK STORAGE IN USE FOR DATABASE            = 3534847 MB
DISPLAY ACCEL REPORT COMPLETE
DSN9022I  -DZA1 DSNX8CMD '-DISPLAY ACCEL' NORMAL COMPLETION
***

```

Under this configuration and status, we simulated a network communication problem between DB2 and the Accelerator.

Example 9-23 illustrates how the communication problem is reported in the system log.

Example 9-23 TCP/IP CONVERSATION FAILED reported in system log

```

14.17.42 STC05580 DSNL511I -DZA1 DSNLIENO TCP/IP CONVERSATION FAILED 841
841 TO LOCATION DZA1IDAA
841 IPADDR=::135.25.80.210 PORT=1400
841 SOCKET=READ RETURN CODE=0 REASON CODE=00000000
14.17.42 STC05580 DSNL511I -DZA1 DSNLIENO TCP/IP CONVERSATION FAILED 842
842 TO LOCATION DZA1IDAA
842 IPADDR=::135.25.80.210 PORT=1400
842 SOCKET=CONNECT RETURN CODE=1128 REASON CODE=76630291
14.17.42 STC05580 DSNX880I -DZA1 DSNX8EKG DDF CONNECT FAILED WITH 843
843 RETURN CODE=12
843 SQLCODE = -30081
843 SQLERRMT =
843 TCP/IP SOCKETS ::135.25.80.210 CONNECT 1128 76630291 0000
843 SQLERRP = DSNLIENO
843 SQLERRD = 00000009 00000000 00000000 FFFFFFFF 00000000
843 00000000
843 SQLWARN 0= ,1= ,2= ,3= ,4= ,5= ,6= ,7= ,8= ,9= ,A=
843 SQLSTATE = 08001

```

Example 9-24 shows how this error is reported in the DB2 MSTR address space.

Example 9-24 TCP/IP CONVERSATION FAILED reported in DB2 MSTR log

```

DSNL511I -DZA1 DSNLIENO TCP/IP CONVERSATION FAILED 841
TO LOCATION DZA1IDAA
IPADDR=::135.25.80.210 PORT=1400
SOCKET=READ RETURN CODE=0 REASON CODE=00000000
DSNL511I -DZA1 DSNLIENO TCP/IP CONVERSATION FAILED 842
TO LOCATION DZA1IDAA
IPADDR=::135.25.80.210 PORT=1400
SOCKET=CONNECT RETURN CODE=1128 REASON CODE=76630291
DSNX880I -DZA1 DSNX8EKG DDF CONNECT FAILED WITH 843
RETURN CODE=12
SQLCODE = -30081
SQLERRMT =
TCP/IP SOCKETS ::135.25.80.210 CONNECT 1128 76630291 0000
SQLERRP = DSNLIENO
SQLERRD = 00000009 00000000 00000000 FFFFFFFF 00000000
00000000
SQLWARN 0= ,1= ,2= ,3= ,4= ,5= ,6= ,7= ,8= ,9= ,A=
SQLSTATE = 08001

```

DSNX880I indicates that a DDF operation failed. The Accelerator server is placed in the STOPERR state. As a result, the Accelerator server is not active, and new requests for the Accelerator server are rejected.

Refer to the documentation of the SQLCODE that is specified in the message. Use the suggested actions in that documentation to correct the problem. Then, issue the **-START ACCEL** command for the Accelerator server.

If the operation is **CONNECT** and the return-code is 8, ensure that the **LOCATION** column of the row that defines the Accelerator in **SYSACCEL.SYSACCELERATORS** is a valid Accelerator location name.

Example 9-25 shows how to verify the status of the Accelerator. using the **-DIS ACCEL(*)** command after problem resolution.

Example 9-25 DIS ACCEL output example

DSNX810I -DZA1 DSNX8CMD DISPLAY ACCEL FOLLOWS -							
DSNX830I -DZA1 DSNX8CDA							
ACCELERATOR	MEMB	STATUS	REQUESTS	ACTV	QUED	MAXQ	
-----	----	-----	-----	----	-----	-----	
DZA1IDAA	DZA1	STARTED	0	0	0	14	
LOCATION=DZA1IDAA HEALTHY							
DETAIL STATISTICS							
LEVEL = AQT03010							
STATUS = ONLINE							
FAILED QUERY REQUESTS = 0							
AVERAGE QUEUE WAIT = 60 MS							
MAXIMUM QUEUE WAIT = 187 MS							
TOTAL NUMBER OF PROCESSORS = 96							
AVERAGE CPU UTILIZATION ON COORDINATOR NODES = 3.00%							
AVERAGE CPU UTILIZATION ON WORKER NODES = 2.00%							
NUMBER OF ACTIVE WORKER NODES = 12							
TOTAL DISK STORAGE AVAILABLE = 33557184 MB							
TOTAL DISK STORAGE IN USE = 12.40%							
DISK STORAGE IN USE FOR DATABASE = 3534847 MB							
ACCEL REPORT COMPLETE							
DSN9022I -DZA1 DSNX8CMD '-DISPLAY ACCEL' NORMAL COMPLETION							

Sample scenario: reaching the concurrency limit

The maximum level of concurrency is 100 queries in the DB2 Analytics Accelerator. Not all of them can be executed concurrently by the Netezza appliance, and the DB2 Analytics Accelerator queues them until they can be executed. Queuing increases the total response time.

If many subsystems are connecting to the same DB2 Analytics Accelerator, for instance a production environment sharing the resource with a test environment, the queries from all the connected subsystems add up to the limit of 100.

See 4.4, “Allocating resources with workload isolation for a shared Accelerator” on page 106 for information and examples of workload isolation.

Example 9-26 illustrates the **-DIS ACCEL** command output when the system reaches the concurrency level of 100 queries.

Example 9-26 DIS ACCEL command showing 100 active requests

14.15.13	STC05580	DSNX830I -DZA1 DSNX8CDA	862				
862		ACCELERATOR		MEMB	STATUS	REQUESTS	ACTV QUED MAXQ
862		-----		----	-----	-----	
862		DZA1STPR		DZA1	STARTED	951	100 0 36
862		LOCATION=DZA1STPR HEALTHY					
862		DETAIL STATISTICS					
862		LEVEL = AQT03012					
862		STATUS = ONLINE					
862		FAILED QUERY REQUESTS				=	18

862	AVERAGE QUEUE WAIT	=	1947 MS
862	MAXIMUM QUEUE WAIT	=	254859 MS
862	TOTAL NUMBER OF PROCESSORS	=	224
862	AVERAGE CPU UTILIZATION ON COORDINATOR NODES	=	7.00%
862	AVERAGE CPU UTILIZATION ON WORKER NODES	=	34.00%
862	NUMBER OF ACTIVE WORKER NODES	=	7
862	TOTAL DISK STORAGE AVAILABLE	=	48000960 MB
862	TOTAL DISK STORAGE IN USE	=	8.18%
862	DISK STORAGE IN USE FOR DATABASE	=	3507285 MB
862	DISPLAY ACCEL REPORT COMPLETE		
14.15.13 STC05580 DSN9022I -DZA1 DSNX8CMD '-DISPLAY ACCEL' NORMAL COMPLETION			

Example 9-27 shows the job output of an application reaching the limit of 100 concurrent queries.

Example 9-27 Job output

```

***INPUT STATEMENT:
  SELECT COUNT_BIG(*) FROM BID1TB.GOSLDW_SALES_FACT;
SQLERROR ON  SELECT  COMMAND, OPEN  FUNCTION
RESULT OF SQL STATEMENT:
DSNT408I SQLCODE = -30040, ERROR: EXECUTION FAILED DUE TO UNAVAILABLE RESOURCES THAT WILL NOT AFFECT THE SUCCESSFUL
EXECUTION OF SUBSEQUENT COMMANDS OR SQL STATEMENTS. REASON 00001304 TYPE OF RESOURCE 00001409
DSNT418I SQLSTATE = 57012 SQLSTATE RETURN CODE
DSNT415I SQLERRP = DSNLZRSQ 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

```

Example 9-28 shows the corresponding error in the DB2 MSTR address space output.

Example 9-28 DB2 MSTR address space

14.13.53	STC05580	DSNL031I -DZA1 DSNLZDJN DRDA EXCEPTION CONDITION IN 605
605		RESPONSE FROM SERVER LOCATION=DZA1STPR FOR THREAD WITH
605		LUWID=USIBMT6.DDFDZA1.CB5D5FF22832=104142
605		REASON=00D351FF
605		ERROR ID=DSNLZRPA0001
605		CORRELATION ID=JGJOB55
605		CONNECTION ID=BATCH
605		IFCID=0191
605		SEE TRACE RECORD WITH IFCID SEQUENCE NUMBER=00000001

The description of REASON=00D351FF is that DB2 received a DDM reply message from the remote server in response to a DDM command. The reply message, although valid for the DDM command, indicates that the DDM command and thus the SQL statement, was not successfully processed. The application is notified of the failure through the architected SQLCODE (-300xx) and associated SQLSTATE. An alert is generated and message DSNL031I is written to the console.

Using DB2 traces, there is no direct way to know if a specific thread has been rejected by the DB2 Analytics Accelerator. OMPE ACCOUNTING REPORT LAYOUT(LONG) or LAYOUT (ACCEL) shows information about Accelerators only if the threads communicate with the Accelerator. This is applicable also to threads that are rejected by the Accelerator. Such a thread shows an Accelerator section in the report. For a thread running with the special register CURRENT QUERY ACCELERATION set to ENABLE, look for a ROLLBACK in the Accounting report.

IBM Data Studio can be used for monitoring failing queries in this scenario, as shown in Figure 9-3.

Accelerator: DZA1STPR @ DZA1

Acceleration: Started [Stop](#)

Status: Online

Used space: 3.34 TB of 45.8 TB

Replication: Error [Start](#)

Credentials valid since: 2/21/13 8:29 PM [Update](#)

Trace: DEFAULT / OFF [Configure](#) [Save](#) [Clear](#)

Active queries: 0

Replication latency: n/a [Show events](#)

▼ Query Statistics

Statistics collected since accelerator server process was started.

Successful queries: 1,008
Maximum queue wait time: 622.385 seconds
Maximum number of queries in queue: 36

Failed queries: 19
Average queue wait time: 7.681 seconds

► About

► Tables (163 of 246 loaded / 164 of 246 enabled for acceleration)

▼ Query Monitoring - Currently retrieving queries .

[Show SQL...](#)
[Show Plan...](#)
[Re-Run](#)
[Cancel](#)

SQL Text like: type filter text

SQL Text	User ID	Start Time	State	Queue Wait ...	Execution Time
	IBM	5/15/13 2:14:35 PM	Failed (57012)	0 seconds	0 seconds
	IBM	5/15/13 2:14:31 PM	Failed (57012)	0 seconds	0 seconds
SELECT COUNT_BIG(*) FROM BID1TB.GOSLDW_SALES_FACT	IBM	5/15/13 2:14:27 PM	Successful	0 seconds	827 seconds
	IBM	5/15/13 2:14:23 PM	Failed (57012)	0 seconds	0 seconds
	IBM	5/15/13 2:14:19 PM	Failed (57012)	0 seconds	0 seconds
	IBM	5/15/13 2:14:15 PM	Failed (57012)	0 seconds	0 seconds
SELECT COUNT_BIG(*) FROM BID1TB.GOSLDW_SALES_FACT	IBM	5/15/13 2:14:11 PM	Successful	0 seconds	842 seconds
SELECT COUNT_BIG(*) FROM BID1TB.GOSLDW_SALES_FACT	IBM	5/15/13 2:14:07 PM	Successful	0 seconds	846 seconds
SELECT COUNT_BIG(*) FROM BID1TB.GOSLDW_SALES_FACT	IBM	5/15/13 2:14:03 PM	Successful	0 seconds	850 seconds
SELECT COUNT_BIG(*) FROM BID1TB.GOSLDW_SALES_FACT	IBM	5/15/13 2:13:59 PM	Successful	0 seconds	854 seconds
SELECT COUNT_BIG(*) FROM BID1TB.GOSLDW_SALES_FACT	IBM	5/15/13 2:13:55 PM	Successful	0 seconds	857 seconds
SELECT COUNT_BIG(*) FROM BID1TB.GOSLDW_SALES_FACT	IBM	5/15/13 2:13:51 PM	Successful	0 seconds	861 seconds
SELECT COUNT_BIG(*) FROM BID1TB.GOSLDW_SALES_FACT	IBM	5/15/13 2:13:47 PM	Successful	0 seconds	865 seconds
SELECT COUNT_BIG(*) FROM BID1TB.GOSLDW_SALES_FACT	IBM	5/15/13 2:13:43 PM	Successful	0 seconds	869 seconds

Figure 9-3 IBM Data Studio monitoring queries

9.1.2 DB2 commands

DB2 Analytics Accelerator support introduces new DB2 commands and modifies existing ones to help monitor the DB2 Analytics Accelerator activity in almost real time. The **DISPLAY ACCEL (*)** command displays information about Accelerator servers.

Refer to *DB2 10 for z/OS Command Reference*, SC19-2972, for details about the options of this command. Example 9-29 shows the syntax that we used to list the Accelerators and their status, as available in our systems.

Example 9-29 Displaying Accelerators status using commands

```
DISPLAY ACCEL (*)
```

Example 9-30 on page 227 shows the output of this command.

Example 9-30 DISPLAY ACCEL() output example*

```

DSNX810I  -DA12 DSNX8CMD DISPLAY ACCEL FOLLOWS -
DSNX830I  -DA12 DSNX8CDA
ACCELERATOR          MEMB  STATUS  REQUESTS  ACTV  QUED  MAXQ
-----
IDAATF3             DA12  STARTED           0    0    0    0
SAMPLE             DA12  STARTEXP          0    0    0    0
SAMP2              DA12  STARTEXP          0    0    0    0
VIRTUAL4           DA12  STARTEXP          0    0    0    0
DISPLAY ACCEL REPORT COMPLETE
DSN9022I  -DA12 DSNX8CMD '-DISPLAY ACCEL' NORMAL COMPLETION
***

```

The message DSNX830I provides the following information:

ACCELERATOR	The name of the Accelerator server.
MEMB	The name of the DB2 data sharing member.
STATUS	The status of the Accelerator server. The status can be any of the following values:
STARTED	The Accelerator server is able to accept requests.
STARTEXP	The Accelerator server was started with the EXPLAINONLY option and is available only for EXPLAIN requests.
STOPPEND	The Accelerator server is no longer accepting new requests. Active Accelerator threads are allowed to complete normally, and queued Accelerator threads are terminated. The Accelerator server was placed in this status by the STOP ACCEL MODE(QUIESCE) command.
STOPPED	The Accelerator server is not active. New requests for the Accelerator are rejected. The Accelerator server was placed in this status by the STOP ACCEL command.
STOPERR	The Accelerator server is not active. The Accelerator server was placed in this status by an error condition. New requests for the Accelerator are rejected.
REQUESTS	The number of query requests that have been processed.
ACTV	The current number of active, accelerated queries.
QUED	The current number of queued requests.
MAXQ	The highest number of queued requests that are reached since the Accelerator was started.

The **DISPLAY ACCEL** command can be submitted with the **DETAIL** option to obtain more information:

```
-DIS ACCEL(IDAATF3) DETAIL
```

The results of this command are illustrated in Example 9-31.

Example 9-31 DISPLAY ACCEL DETAIL output example

```

DSNX810I  -DA12 DSNX8CMD DISPLAY ACCEL FOLLOWS -
DSNX830I  -DA12 DSNX8CDA
ACCELERATOR          MEMB  STATUS  REQUESTS  ACTV  QUED  MAXQ
-----
IDAATF3             DA12  STARTED        1682    0    0    62
LOCATION=IDAATF3  HEALTHY

```

```

DETAIL STATISTICS
LEVEL = AQT02012
STATUS = ONLINE
FAILED QUERY REQUESTS = 6765
AVERAGE QUEUE WAIT = 4570 MS
MAXIMUM QUEUE WAIT = 246244 MS
TOTAL NUMBER OF PROCESSORS = 24
AVERAGE CPU UTILIZATION ON COORDINATOR NODES = .00%
AVERAGE CPU UTILIZATION ON WORKER NODES = 1.00%
NUMBER OF ACTIVE WORKER NODES = 3
TOTAL DISK STORAGE AVAILABLE = 8024544 MB
TOTAL DISK STORAGE IN USE = 7.12%
DISK STORAGE IN USE FOR DATABASE = 354205 MB
DISPLAY ACCEL REPORT COMPLETE
DSN9022I -DA12 DSNX8CMD '-DISPLAY ACCEL' NORMAL COMPLETION
***

```

In this example, the number of REQUESTS and ACTIVE requests is 0. This was the status before the execution of one of our workload scenarios.

To illustrate how you can use this command to monitor DB2 Analytics Accelerator activity, consider the contents of Example 9-32. It shows a high number of requests and 100 concurrent active queries being executed in the DB2 Analytics Accelerator.

Example 9-32 DISPLAY ACCEL DETAIL output sample showing activity

```

DSNX810I -DA12 DSNX8CMD DISPLAY ACCEL FOLLOWS -
DSNX830I -DA12 DSNX8CDA
ACCELERATOR MEMB STATUS REQUESTS ACTV QUED MAXQ
-----
IDAATF3 DA12 STARTED 1690 100 0 62
LOCATION=IDAATF3 HEALTHY
DETAIL STATISTICS
LEVEL = AQT02012
STATUS = ONLINE
FAILED QUERY REQUESTS = 6770
AVERAGE QUEUE WAIT = 1443 MS
MAXIMUM QUEUE WAIT = 236080 MS
TOTAL NUMBER OF PROCESSORS = 24
AVERAGE CPU UTILIZATION ON COORDINATOR NODES = 2.00%
AVERAGE CPU UTILIZATION ON WORKER NODES = 84.00%
NUMBER OF ACTIVE WORKER NODES = 3
TOTAL DISK STORAGE AVAILABLE = 8024544 MB
TOTAL DISK STORAGE IN USE = 7.12%
DISK STORAGE IN USE FOR DATABASE = 354205 MB
DISPLAY ACCEL REPORT COMPLETE
DSN9022I -DA12 DSNX8CMD '-DISPLAY ACCEL' NORMAL COMPLETION
***

```

Note: The **DETAIL STATISTICS** information is obtained from the DB2 Analytics Accelerator heartbeat and is refreshed every 20 seconds.

9.1.3 Monitoring the Accelerator using SYSPROC.ACCEL_GET_QUERY_DETAILS stored procedure

The stored procedure SYSPROC.ACCEL_GET_QUERY_DETAILS retrieves information about past or active queries from an Accelerator. The output is returned in a result set and the amount of data can be large.

Note: For details about this and all Accelerator-provided stored procedures, refer to the IBM publication: *IBM DB2 Analytics Accelerator for z/OS Version 3.1.0 Stored Procedures Reference*, SH12-6984.

Example 9-33 shows the syntax for calling the SYSPROC.ACCEL_GET_QUERY_DETAILS stored procedure.

Example 9-33 Syntax for calling SYSPROC.ACCEL_GET_QUERY_DETAILS procedure

```
CALL PROCEDURE SYSPROC.ACCEL_GET_QUERY_DETAILS  
(accelerator_name,  
 plan_id,  
 message);
```

Following are the options:

accelerator_name	The unique name of the Accelerator.
plan_id	The identifier (plan ID) of the query that you want to obtain details about. You can run the SYSPROC.ACCEL_GET_QUERIES stored procedure to obtain a list of these identifiers. The plan IDs of past or currently active queries are shown in the query_listoutput XML string.

Important: The input string must conform to the structure of the **messageControl1** element in the *hlq.SAQTSAMP(AQTSXSD1)* data set. For more details about running the Accelerator stored procedures from a batch job and how to specify the input XML string, refer to the section “messageControl: of XML content for the messageinput parameter” of the publication: *IBM DB2 Analytics Accelerator for z/OS Version 3.1.0 Stored Procedures Reference*, SH12-6984.

Calling the Accelerator stored procedures using JCL

The Accelerator provides a C++ sample program that can be used for calling some Accelerator stored procedures. The sample code and job control language (JCL) can be found in members AQTSJI03 (JCL), and AQTSCALL (sample code) of library *hlq.SAQTSAMP*.

Calling an Accelerator stored procedure from a REXX program

You can also call the Accelerator provided stored procedures from a REXX program.

Example 9-34 on page 230 shows the REXX code that we used for calling the Accelerator stored procedure SYSPROC.ACCEL_GET_QUERIES. This sample program includes some XML parsing capabilities to show on a table format the last top 10 queries by elapsed time in the Accelerator.

Example 9-34 Calling SYSPROC.ACCEL_GET_QUERIES from REXX

```
/* REXX */
/*-----*/
/* Accel redbook SG24-8151. 9 MAY 2013. Cristian Molaro. */
/* Sample program to call an Accel stored procedure. */
/* This code includes some XML parsing code for tabulating results. */
/*-----*/

/*-----*/
/* Initialization of working variables */
/*-----*/

debug      = 1                /* Set to 1 to get debug information */
db2ssid    = "DZA1"           /* Target DB2 subsystem */
AccelName  = 'DZA1STPR'       /* Target accelerator name */

/* Query selection criteria */
QuerySelection = ,
    '<?xml version="1.0" encoding="UTF-8" ?> ',
    '<dwa:querySelection ',
    'xmlns:dwa="http://www.ibm.com/xmlns/prod/dwa/2011" ',
    'version="1.0"> ',
    '<!-- filter conditions are combined with AND --> ',
    '<filter scope="all" /> ',
    '<result order="elapsedTime" maxRows="10"/> ',
    '</dwa:querySelection> '

/* Query list, the variable that will receive the answer from Accel*/
QueryList = left(' ',1000000)
QueryListInd = 1              /* Null value indicator */

/* Message specification */
MsgSpec = '<?xml version="1.0" encoding="UTF-8" ?> ',
    '<spctrl:messageControl ',
    'xmlns:spctrl="http://www.ibm.com/xmlns/prod/dwa/2011" ',
    'version="1.0" versionOnly="false" > ',
    '</spctrl:messageControl>'copies(' ',2000)
MsgInd = 1                    /* Null value indicator */

/*-----*/
/* Connect to DB2 */
/*-----*/

'SUBCOM DSNREXX'
If rc then
    S_RC = RXSUBCOM('ADD','DSNREXX','DSNREXX')

Address DSNREXX "CONNECT" db2ssid

If SQLCODE <> "0" then
    Do
        say "Failure to connect to database" db2ssid
        say "SQLCODE --> " SQLCODE
        exit 8
    End
Address DSNREXX

/*-----*/
/* Call SP SYSPROC.ACCEL_GET_QUERIES */
/*-----*/
```

Call GETQUERIES

```
/*-----*/
/* Process results, QueryList */
/*-----*/
```

Call PROCESSQUERIES

```
/*-----*/
/* End program */
/*-----*/
Exit /* bye */
```

```
/*-----*/
/* Routines */
/*-----*/
```

PROCESSQUERIES:

```
/*-----*/
/* This routine process the results of SYSPROC.ACCEL_GET_QUERIES */
/*-----*/
```

QueryList = strip(QueryList,'t')

cnt = 0

Do until length(QueryList) = 0

```
  parse value QueryList with . '<query user=' user '' QueryList
  parse value QueryList with . 'planID=' planid '' QueryList
  parse value QueryList with . 'ntInfo productID=' pid '' QueryList
  parse value QueryList with . 'user=' user '' QueryList
  parse value QueryList with . 'workstation=' wrkst '' QueryList
  parse value QueryList with . 'application=' appl '' QueryList
  parse value QueryList with . 'networkID=' netid '' QueryList
  parse value QueryList with . 'locationName=' locnm '' QueryList
  parse value QueryList with . 'luName=' lunam '' QueryList
  parse value QueryList with . 'connName=' connnm '' QueryList
  parse value QueryList with . 'connType=' conntyp '' QueryList
  parse value QueryList with . 'corrID=' corrid '' QueryList
  parse value QueryList with . 'authID=' authid '' QueryList
  parse value QueryList with . 'planName=' plannm '' QueryList
  parse value QueryList with . 'subSystemID=' ssid '' QueryList
  parse value QueryList with . 'state=' state '' QueryList
  parse value QueryList with . 'mitTimestamp=' subts '' QueryList
  parse value QueryList with . 'apsedTimeSec=' elap '' QueryList
  parse value QueryList with . 'utionTimeSec=' exects '' QueryList
  parse value QueryList with . 'priority=' prior '' QueryList
  parse value QueryList with . 'resultRows=' resrows '' QueryList
  parse value QueryList with . 'resultBytes=' resbyts '' QueryList
  parse value QueryList with . 'sqlCode=' sqlcode '' QueryList
  parse value QueryList with . 'rDescription=' erdesc '' QueryList
  parse value QueryList with . 'task id=' taskid '' QueryList
  parse value QueryList with . '<sql>' sql '</sql>' QueryList
```

if length(sql) > 0 then Do

```
  subts = substr(subts,1,10) || '-' || substr(subts,12,15)
  subts = translate(subts,',' ,':')
```

```
  sql = substr(sql,10,(length(sql)))
  sql = strip(sql,'T')
  sql = substr(sql,1,(length(sql) - 3))
  cnt = cnt + 1
```

```

/* say cnt quser planid pid user wrkst ,
   appl netid locnm lunam connnam conntyp corrid ,
   authid plannm ssid state subts elap exects prior ,
   resrows resbyts sqlcode erdesc taskid sql */

/* insert into DB2 */
sqlstmt1 = "insert into accel_queries values ( " ,
"" || AcclName || " , current timestamp , " || ,
"" || quser || " , " || planid || " , " || pid || " , " ,
"" || user || " , " || wrkst || " , " ,
"" || appl || " , " || netid || " , " || locnm || " , " ,
"" || lunam || " , " || connnam || " , " || conntyp || " , " ,
"" || corrid || " , " ,
"" || authid || " , " || plannm || " , " || ssid || " , " ,
"" || state || " , " || subts || " , " || elap || " , " ,
"" || exects || " , " || prior || " , " ,
"" || resrows || " , " || resbyts || " , " || sqlcode ,
"" || " , " || erdesc || " , " || taskid || " , " || sql || " )"

Address DSNREXX "EXECSQL " sqlstmt1
Address DSNREXX "EXECSQL COMMIT"

end /* if length(sql) > 0 then Do */
End /* Do while length(QueryList) >0 */

Return

GETQUERIES:
/*-----*/
/* This routine calls SYSPROC.ACCEL_GET_QUERIES */
/*-----*/
Address DSNREXX "EXECSQL CALL SYSPROC.ACCEL_GET_QUERIES ( " ,
               ":AcclName," ,
               ":QuerySelection," ,
               ":QueryList    INDICATOR :QueryListInd," ,
               ":MsgSpec      INDICATOR :MsgInd ) "

If SQLCODE <> 0 then do
  SQLERRORPOSITION = 'Call Stored Procedure'
  call SQLERRORROUTINE
End
Else do /* Stored Procedure completed successfully */
  If MsgInd >= 0 then Do
    say "+Message:" MsgSpec
    If (pos('AQT10000I',MsgSpec) = 0) then Exit 4
  End
  Else say "No message available"
End

Return

SQLERRORROUTINE:
/*-----*/
/* SQL error handling */
/*-----*/
say 'POSITION    = ' SQLERRORPOSITION
say 'SQLCODE     = ' SQLCODE
say 'SQLSTATE    = ' SQLSTATE
say 'SQLERRP     = ' SQLERRP

```



```

say 'TOKENS      = ' TRANSLATE(SQLERRMC,' ','FF'X)
say 'SQLERRD.1   = ' SQLERRD.1
say 'SQLERRD.2   = ' SQLERRD.2
say 'SQLERRD.3   = ' SQLERRD.3
say 'SQLERRD.4   = ' SQLERRD.4
say 'SQLERRD.5   = ' SQLERRD.5
say 'SQLERRD.6   = ' SQLERRD.6
say 'SQLWARN.0   = ' SQLWARN.0
say 'SQLWARN.1   = ' SQLWARN.1
say 'SQLWARN.2   = ' SQLWARN.2
say 'SQLWARN.3   = ' SQLWARN.3
say 'SQLWARN.4   = ' SQLWARN.4
say 'SQLWARN.5   = ' SQLWARN.5
say 'SQLWARN.6   = ' SQLWARN.6
say 'SQLWARN.7   = ' SQLWARN.7
say 'SQLWARN.8   = ' SQLWARN.8
say 'SQLWARN.9   = ' SQLWARN.9
say 'SQLWARN.10  = ' SQLWARN.10
EXIT 8

```

Note: The query_list XML file might be large. We observed data truncation when the receiving REXX variable was dimensioned small. Check for SQLCODE=0 and SQLSTATE=01004 for detecting this situation.

We created a custom DB2 table to store the parsed information. Example 9-35 shows the DDL of this table, ACCEL_QUERIES. This example table was created in a 32 KB page table space.

Example 9-35 ACCEL_QUERIES

```

CREATE TABLE CRISTIA.ACCEL_QUERIES
  (ACCELERATOR          CHAR(8)
  ,INSERT_TS           TIMESTAMP
  ,QUERY_USER          CHAR(8)
  ,PLANID              CHAR(8)
  ,PRODUCTID          VARCHAR(50)
  ,"USER"              CHAR(8)
  ,WORKSTATION         VARCHAR(50)
  ,APPLICATION         VARCHAR(50)
  ,NETWORKID           VARCHAR(50)
  ,LOCATIONNAME         VARCHAR(50)
  ,LUNAME              VARCHAR(50)
  ,CONNNAME            VARCHAR(50)
  ,CONNTYPE            VARCHAR(50)
  ,CORRID              VARCHAR(50)
  ,AUTHID              VARCHAR(50)
  ,PLANNAME            VARCHAR(50)
  ,SUBSYSTEMID         VARCHAR(50)
  ,EXECUTION_STATE     CHAR(8)
  ,SUBMITTIMESTAMP     TIMESTAMP
  ,ELAPSEDTIMESEC      INTEGER
  ,EXECUTIONTIMESEC    INTEGER
  ,PRIORITY             CHAR(25)
  ,RESULTROWS          INTEGER
  ,RESULTBYTES         INTEGER
  ,SQLCODE              CHAR(10)
  ,ERRORDescription    VARCHAR(50)
  ,TASK_ID             INTEGER
  ,SQL                 VARCHAR(30000)

```

```

)
IN CRIS.CRISTS1 ;
COMMIT;

```

Using query tools, such as QMF, you are able to create custom reports based on this information. This method can be easier to work with than using XML as input.

9.2 Monitoring low latency updates

For a detailed description of the low latency function of the Accelerator, see Chapter 7, “Incremental update” on page 161.

Monitoring low latency updates can be achieved by using the **waitForReplication** parameter of the **SYSPROC.ACCEL_CONTROL_ACCELERATOR** stored procedure. The incremental update function provides an estimate of the current latency time in the **latencyInSecondsXML** element. This element is included in the result XML code returned by the **ACCEL_CONTROL_ACCELERATOR** stored procedure if the **<getAcceleratorInfo/>** element was specified as the command input parameter.

A way of executing this stored procedure is via IBM Data Studio. Figure 9-4 shows how to select this stored procedure in Data Studio.

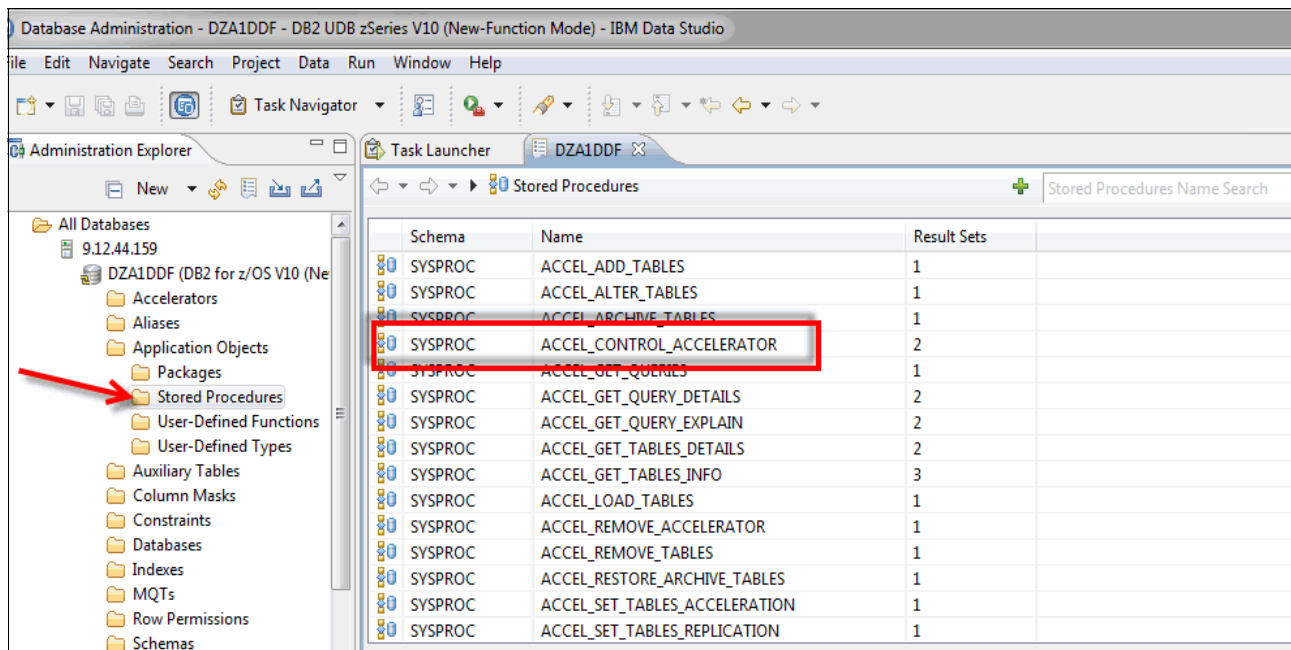


Figure 9-4 Selecting **ACCEL_CONTROL_ACCELERATOR** stored procedure in DataStudio

Right-click the stored procedure name in the list and select **Run**. Figure 9-5 on page 235 shows an example.

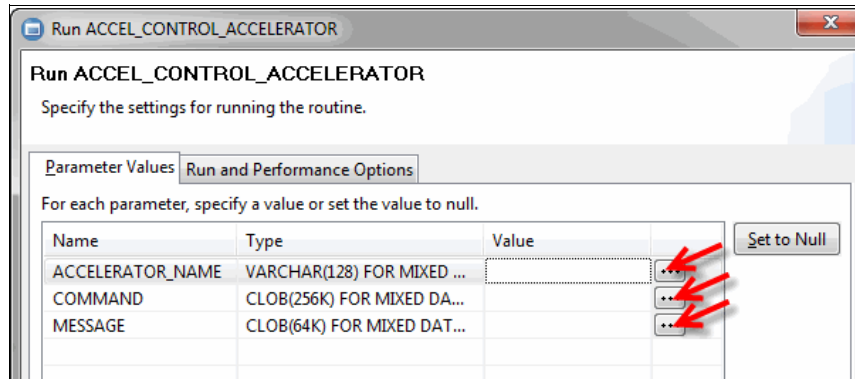


Figure 9-5 Run ACCEL_CONTROL_ACCELERATOR stored procedure windows

Example 9-36 shows how to create a command file including the `<getAcceleratorInfo/>` directive.

Example 9-36 ACCEL_CONTROL_ACCELERATOR command file

```
<?xml version="1.0" encoding="UTF-8" ?>
<aqttables:controlCommand
xmlns:aqttables="http://www.ibm.com/xmlns/prod/dwa/2011" version="1.2" >
<getAcceleratorInfo/>
</aqttables:controlCommand>
```

After this information is entered, you can browse a parsed version of the command to check its correctness. This is shown in Figure 9-6.

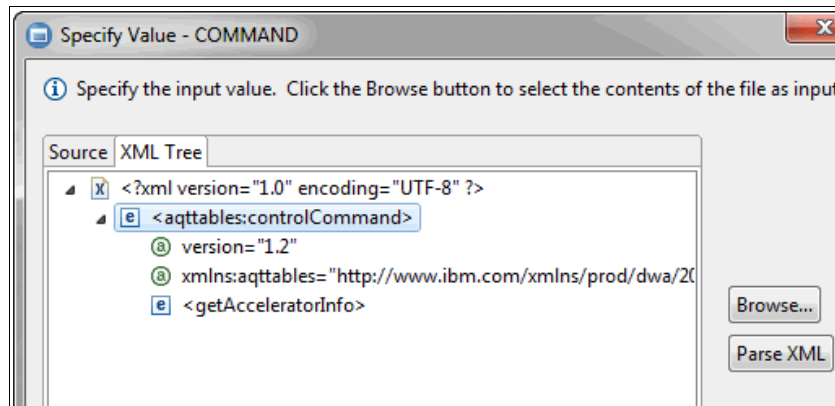


Figure 9-6 Browsing the command XML tree

Example 9-37 shows the syntax example of the ACCEL_CONTROL_ACCELERATOR message file.

Example 9-37 ACCEL_CONTROL_ACCELERATOR message file

```
<?xml version="1.0" encoding="UTF-8" ?>
<spctrl:messageControl
xmlns:spctrl="http://www.ibm.com/xmlns/prod/dwa/2011"
version="1.0" versionOnly="false" >
</spctrl:messageControl>
```

Figure 9-7 illustrates how to browse the XML tree of the message file within Data Studio. Use this technique to verify the correctness of the information.

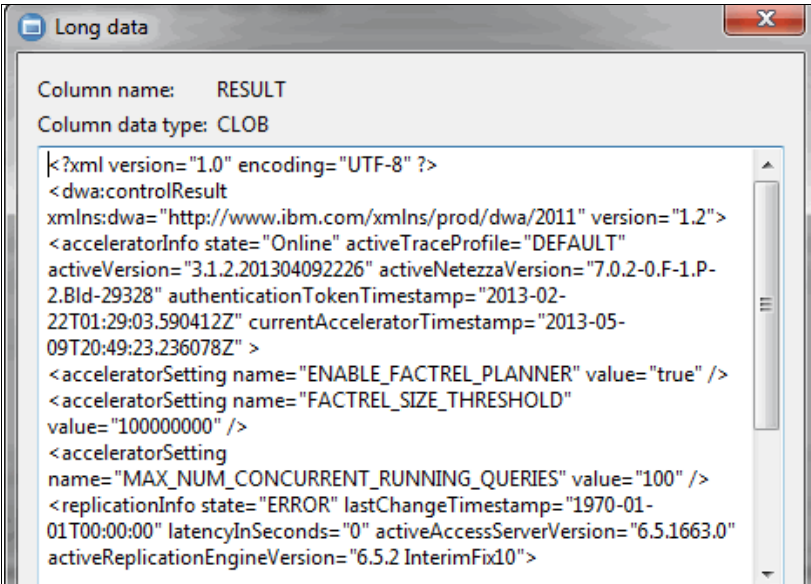


Figure 9-7 Browsing the ACCEL_CONTROL_ACCELERATOR XML message file

To execute the stored procedure, press **Run**. Figure 9-8 shows how to monitor the execution in Data Studio.

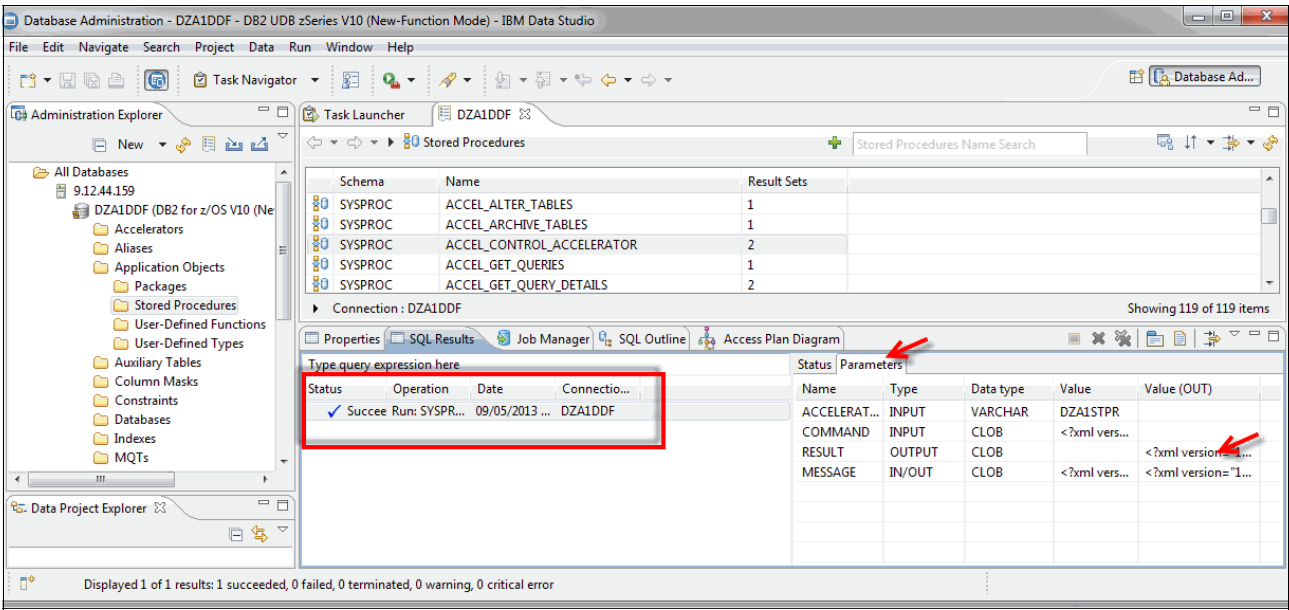


Figure 9-8 Monitoring the execution of ACCEL_CONTROL_ACCELERATOR in Data Studio

You can see the execution output by selecting the RESULT XML file. This is illustrated in Figure 9-9 on page 237.

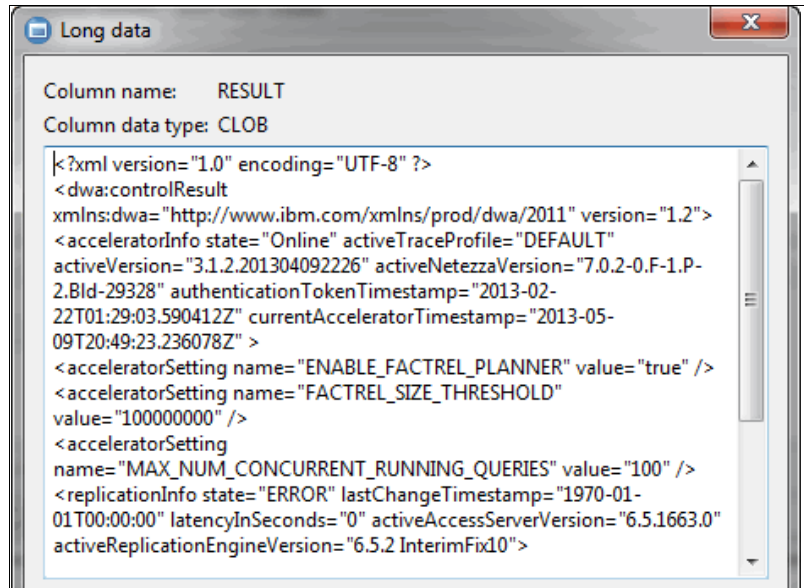


Figure 9-9 ACCEL_CONTROL_ACCELERATOR_RESULT example

Example 9-38 shows the complete RESULT text and highlights the **latencyInSeconds** parameter. In this example, latency of changes is 11 seconds.

Example 9-38 ACCEL_CONTROL_ACCELERATOR_RESULT showing latencyInSeconds

```
<?xml version="1.0" encoding="UTF-8" ?>
<dwa:controlResult xmlns:dwa="http://www.ibm.com/xmlns/prod/dwa/2011" version="1.2">
<acceleratorInfo state="Online" activeTraceProfile="DEFAULT" activeVersion="3.1.2.201304092226"
activeNettezzaVersion="7.0.2-0.F-1.P-2.B1d-29328" authenticationTokenTimestamp="2013-02-22T01:29:03.590412Z"
currentAcceleratorTimestamp="2013-05-09T20:40:00.558737Z" >
<acceleratorSetting name="ENABLE_FACTREL_PLANNER" value="true" />
<acceleratorSetting name="FACTREL_SIZE_THRESHOLD" value="100000000" />
<acceleratorSetting name="MAX_NUM_CONCURRENT_RUNNING_QUERIES" value="100" />
<replicationInfo state="STARTED" lastChangeTimestamp="2013-05-09T20:35:04.296083Z" latencyInSeconds="11"
activeAccessServerVersion="6.5.1663.0" activeReplicationEngineVersion="6.5.2 InterimFix10">
  <sourceAgent insertCount="0" updateCount="0" deleteCount="0" />
  <targetAgent insertCount="0" updateCount="0" deleteCount="0" />
</replicationInfo>
</acceleratorInfo>
</dwa:controlResult>
```

Example 9-39 shows the RESULT XML file when the replication is in ERROR status.

Example 9-39 ACCEL_CONTROL_ACCELERATOR_RESULT replicationInfo state="ERROR"

```
<?xml version="1.0" encoding="UTF-8" ?>
<dwa:controlResult xmlns:dwa="http://www.ibm.com/xmlns/prod/dwa/2011" version="1.2">
<acceleratorInfo state="Online" activeTraceProfile="DEFAULT" activeVersion="3.1.2.201304092226"
activeNettezzaVersion="7.0.2-0.F-1.P-2.B1d-29328" authenticationTokenTimestamp="2013-02-22T01:29:03.590412Z"
currentAcceleratorTimestamp="2013-05-09T20:49:23.236078Z" >
<acceleratorSetting name="ENABLE_FACTREL_PLANNER" value="true" />
<acceleratorSetting name="FACTREL_SIZE_THRESHOLD" value="100000000" />
<acceleratorSetting name="MAX_NUM_CONCURRENT_RUNNING_QUERIES" value="100" />
<replicationInfo state="ERROR" lastChangeTimestamp="1970-01-01T00:00:00" latencyInSeconds="0"
activeAccessServerVersion="6.5.1663.0" activeReplicationEngineVersion="6.5.2 InterimFix10">
  <sourceAgent insertCount="-1" updateCount="-1" deleteCount="-1" />
  <targetAgent insertCount="-1" updateCount="-1" deleteCount="-1" />
</replicationInfo>
</acceleratorInfo>
</dwa:controlResult>
```

```
</replicationInfo>
</acceleratorInfo>
</dwa:controlResult>
```

In our tests, we forced a replication error for documentation purposes. Example 9-40 illustrates the feedback in the system log.

Example 9-40 Communication errors during replication failure

```
+CHC6503E (CHCPROC) Communications link to NOTHING.ATTDNS.COM:0
abnormally terminated by remote system (link ...
+CHC6503E (CHCPROC) ... 1, process 1)
+CHC6448W (CHCPROC) Communications link in link slot 1 has terminated
abnormally
+CHC6503E (CHCPROC) Communications link to NOTHING.ATTDNS.COM:0
abnormally terminated by remote system (link ...
+CHC6503E (CHCPROC) ... 1, process 2)
+CHC6448W (CHCPROC) Communications link in link slot 1 has terminated
abnormally
+CHC6503E (CHCPROC) Communications link to 135.25.80.210:11301
abnormally terminated by remote system (link 6 ...
+CHC6503E (CHCPROC) ... , process 3)
+CHC6503E (CHCPROC) Communications link to 135.25.80.210:11301
abnormally terminated by remote system (link 5 ...
+CHC6503E (CHCPROC) ... , process 1)
+CHC6448W (CHCPROC) Communications link in link slot 6 has terminated
abnormally
+CHC6448W (CHCPROC) Communications link in link slot 5 has terminated
abnormally
+CHC6503E (CHCPROC) Communications link to 135.25.80.210:11301
abnormally terminated by remote system (link 6 ...
```

Example 9-41 shows the error feedback as received in the DB2 MSTR log

Example 9-41 Replication failures in DB2 MSTR log

```
16.34.58 STC05580 DSNX881I -DZA1 20 W 113 351
351 (2013-05-09T16:34:32.000-04:00) Id: 12 Subscription:
351 ACCEL_DZA1STPR_DZA1DDF Message: CHC0012W Replication functions are
351 already running for subscription DZA1IDAA. The request has been
351 rejected Originator: DSC
16.37.58 STC05580 DSNX881I -DZA1 20 W 114 401
401 (2013-05-09T16:37:37.000-04:00) Id: 6436 Subscription:
401 ACCEL_DZA1STPR_DZA1DDF Message: CHC6436W Communications session
401 NOTHING.ATTDNS.COM:38177 in session slot 3 and state 8 has lost a
401 resource Originator: CMO
```

Tip: Refer to “Synchronizing data in IBM DB2 Analytics Accelerator for z/OS” at: <http://www.ibm.com/support/docview.wss?uid=swg27038501> for more details about the SYSPROC.ACCEL_CONTROL_ACCELERATOR stored procedure.

9.3 Reporting and monitoring using OMPE

This section describes OMPE support for the Accelerator-enabled environments. OMEGAMON XE for DB2 PE/PM on z/OS V510 or V511 provides support for DB2 Analytics

Accelerator. The OMPE reports shown in this publication were generated using OMEGAMON XE for DB2 PE/PM on z/OS V511.

We describe the following:

- ▶ Batch reports
- ▶ OMPE classic interface
- ▶ Online monitoring using OMPE GUI
- ▶ The Accelerator support in the OMPE Performance Database

9.3.1 Batch reports

The OMPE system parameters report provides information about the configuration of the DB2 system being monitored. A report entry is produced for each location present in the input data. An entry is also produced if DB2 was restarted with changed system parameters or a change to the system parameters was detected when the Statistics interval was reached. Some parameters, such as buffer pool and group buffer pool attributes, can be changed while a system is active. If the appropriate DB2 trace class is active, the changes are recorded in the System Parameters report in the order of occurrence.

Example 9-42 shows an example of invocation of this report.

Example 9-42 OMPE SYSPARMS command example

```
FPEC2001I  COMMAND INPUT FROM DDNAME SYSIN
0          SYSPARMS TRACE
0          EXEC
FPEC1999I  SYSTEM INITIALIZATION COMPLETE. RETURN CODE 0
```

An output example is illustrated in Example 9-43.

Example 9-43 OMPE SYSPARMS report

1 LOCATION: DZAIDDF GROUP: N/P MEMBER: N/P SUBSYSTEM: DZA1 DB2 VERSION: V10	OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V5R1M1) SYSTEM PARAMETERS REPORT	PAGE: 1-4 ACTUAL FROM: 04/12/13 14:00:33.00
---	---	--

OTHER SYSTEM PARAMETERS ----- DUAL BSDS MODE (TWOBSDS).....YES ROLL UP PARALLEL TASK ACCOUNTING (PTASKROL).....YES NO. PAGES SMALL TABLE THRESHOLD (NPGTHRS).....0 SMS DATACLASS NAME FOR TS (SMSDCFL).....N/P SMS DATACLASS NAME FOR IS (SMSDCIX).....N/P OFFLOAD OPTION (OFFLOAD).....NO SU CONVERSION FACTOR......258 OUTER JOIN PERFORMANCE ENHANCEMENTS (OJPERFEH).....YES MINIMUM DIVIDE SCALE (MINDVSCL).....NONE STAR JOIN THRESHOLD (SJTABLES)......4 ONLINE SYSTEM PARM USER ID MONITOR.....N/P ONLINE SYSTEM PARM CORREL ID MONITOR.....N/P ONLINE SYSTEM PARM TIME CHANGED.....N/P ONLINE SYSTEM PARM TYPE.....N/P DB2-SUPPLIED DECP INDICATOR.....X'D5' MAX CONCURRENT PKG OPS (MAX_CONCURRENT_PKG_OPS).....10 MAX TEMP STORAGE PER AGENT IN MB (MAXTEMPS).....0 MAX TEMP RID (MAXTEMPS_RID).....NOLIMIT ADMIN SCHEDULER JCL PROC NAME (ADMTPROC).....N/P FREE LOCAL CACHED STATEMENTS (CACHEDYN_FREELocal).....1 INDEX I/O PARALLELISM (INDEX_IO_PARALLELISM).....YES REBIND PLANMGMT DEFAULT (PLANMGMT).....EXTENDED PLANMGMTSCOPE DEFAULT (PLANMGMTSCOPE).....STATIC ZOSMETRICS.....YES QUERY ACCELERATION (QUERY_ACCELERATION).....NO QUERY ACCEL OPT.....NONE ACCEL ARCHIVED DATA.....ENABLE	DATABASES AND SPACES STARTED AUTOMATICALLY (DSNTIPS) ----- ALL ICF CATALOG QUALIFIERS ----- DB2ZA SIZES PANEL 1 (DSNTIPD) ----- USER LOB VALUE STORAGE IN KB (LOBVALA).....10,240 SYSTEM LOB VALUE STORAGE IN MB (LOBVALS).....2,048 MAXIMUM NUMBER OF LE TOKENS (LEMAX).....20 USER XML VALUE STG IN KB (XMLVALA).....204,800 SYSTEM XML VAL STG IN MB (XMLVALS).....10,240 PROTECTION PANEL (DSNTIPP1) ----- SECURITY ADMINISTRATOR 1 AUTHORIZATION ID (SECADM1).....SECADM SECURITY ADMINISTRATOR 1 TYPE (SECADM1_TYPE).....AUTHID SECURITY ADMINISTRATOR 2 AUTHORIZATION ID (SECADM2).....SECADM SECURITY ADMINISTRATOR 2 TYPE (SECADM2_TYPE).....AUTHID SEPARATE SECURITY DUTIES (SEPARATE_SECURITY).....NO INCLUDE DEPENDENT PRIVILEGES (REVOKE_DEP_PRIVILEGES).....S
--	---

This information is of particular interest for the scope of this publication:

- ▶ **QUERY ACCELERATION (QUERY_ACCELERATION):** The initial value of **CURRENT QUERY ACCELERATION** is determined by the value of DB2 subsystem **QUERY_ACCELERATION** parameter. The default for the initial value of that subsystem parameter is **NONE** unless your installation has changed the value.
- ▶ **QUERY ACCEL OPT**
- ▶ **ACCEL ARCHIVED DATA**

9.3.2 OMPE classic interface

APAR PM80739 added real-time monitoring of Analytics Accelerators to the OMPE classic user interface. This update includes these changes:

- ▶ In the panel **Threads Summary Excluding Idle Threads (ZALLT)**, the new thread status **"IN-ACCEL"** is added.
- ▶ In the panel **Thread Accelerator Detail (ZTACEL)**, the new option **"W"** provides details about using the thread accelerator. Option **"W"** is available after you zoomed into one of the threads by pressing **F11 (Zoom)** from most of the panels. You can also display the **IN-ACCEL** status by using option **"A-THREAD DETAIL"**.
- ▶ In the panel **Resource Managers and Other DB2 Subsystem Information (ZRMMENU)**, the new option **"N" - ACCELERATOR** is available. You can display a list of the available Accelerators by selecting option **N**. You can sort the list of Accelerators by each of the fields.
- ▶ The panel **Accelerator Summary (ZACST)** shows a list of Accelerators that are available in the current DB2 subsystem. By zooming into an individual Accelerator, the panel **Accelerator Statistics Detail (ZACSD)** is displayed. It shows detailed statistics information about the selected Accelerator.

Example 9-44 shows an example of the OMPE classic interface real-time menu.

Example 9-44 OMPE classic interface, real-time menu

```

_____ ZMENU   VTM    02      V511.#P DZA1 S 05/10/13 10:33:18 2
> Help/News/Index PF1          Exit PF3          PF Keys PF5
>   Type a selection letter at the left end of the top line and press ENTER.
=====
MENU      OMEGAMON CLASSIC INTERFACE -- REALTIME MAIN MENU
_ S SUMMARY ..... Summary of DB2 activity
_ E EXCEPTIONS ..... Current or potential system problems
_ T THREAD ACTIVITY ..... Thread activity information
_ U THREAD ACTIVITY ..... Thread activity information by package
_ L LOCKING CONFLICTS .... Locking conflict information
_ R RESOURCE MANAGERS .... Resource manager, other DB2 subsystem information
_ A APPLICATION TRACE .... Trace and view application activity
_ D DISTRIBUTED DATA ..... Distributed database system information
_ O OBJECT ANALYSIS ..... Object and volume information
_ G DB2 CONNECT SERVER ... DB2 Connect/Gateways with connection to DB2
_ C MVS CONSOLE ..... MVS console to issue commands and view messages
_ B DB2 CONSOLE ..... DB2 console to issue commands and view messages
_ M MISCELLANEOUS ..... Address space information, OMEGAMON commands, etc.
_ P PROFILE ..... Customize OMEGAMON session and exception settings
_ H HISTORY ..... Near-Term History information
_ V SQL PA REPORTS..... View SQL PA reports
_ Z OTHER DB2 ..... Redirect monitoring to another DB2
=====

```

Use option T of this panel to have access to the Thread activity. This panel is shown in Example 9-45.

Example 9-45 OMPE Classic interface, Thread activity panel

```

_____ ZALLT   VTM   02       V511.#P DZA1 S 05/10/13 10:50:19 2
> Help PF1      Back PF3          Up PF7       Down PF8       Sort PF10      Zoom PF11
> T.A
>          Thread Activity: Enter a selection letter on the top line.

> *-All-Idle   B-TSO      C-CICS      D-IMS      E-Background  F-Dist Allied
> G-Dist DBAC   H-Util    I-Inact     J-Filter   K-Functions   L-Stored Proc
> M-Triggers    N-Sysplex  O-Enclaves P-Worksta Q-All+Idle
=====
>          Threads Summary Excluding Idle Threads
THDA
+ *
+ Elapsed      Planname  CPU    Status      GetPg  Update Commit CORRID/JOBN
+ -----
+ 00:32:33.1   K02PLAN  00.0%  NOT-IN-DB2    0      0      2 D51102
+ 00:32:33.1   K02PLAN  00.0%  NOT-IN-DB2   285    0      7 D51102
+ 00:32:03.0   K02PLAN  00.0%  NOT-IN-DB2    0      0      0 D51102
+ 00:05:51.4   DSNTPE10 00.0%  IN-ACCEL     113    0      0 BENCH01N
+ 00:05:51.4   DSNTPE10 00.0%  IN-ACCEL     113    0      0 BENCH02N
+ 00:05:51.4   DSNTPE10 00.0%  IN-ACCEL     113    0      0 BENCH03N
+ 00:05:51.4   DSNTPE10 00.0%  IN-ACCEL     113    0      0 BENCH04N
+ 00:05:51.4   DSNTPE10 00.0%  IN-ACCEL     113    0      0 BENCH05N
+ 00:05:51.4   DSNTPE10 00.0%  IN-ACCEL     113    0      0 BENCH06N
+ 00:04:16.0   DISTSERV 00.0%  WAIT-REMREQ    0      0      0 db2jcc_appli
+ 00:03:08.0   DISTSERV 00.0%  WAIT-REMREQ    0      0      0 db2jcc_appli
+ 00:01:04.6   DISTSERV 00.0%  WAIT-REMREQ    0      0      0 db2jcc_appli
+ 00:00:00.0   DISTSERV 00.0%  WAIT-REMREQ    0      0      0 db2jcc_appli

```

By using this panel, you can identify the threads that are currently busy in the Accelerator. This is indicated under the column Status as IN_ACCEL. Doing zoom, by using F11, on a thread using the Accelerator gives you the thread details as illustrated in Example 9-46

Example 9-46 OMPE Classic Thread detail

```

_____ ZTDTL   VTM   02       V511.#P DZA1 05/10/13 10:51:23 2
> Help PF1                                     Back PF3

>          THREAD INFORMATION: Enter a selection letter on the top line.

> *-THREAD DETAIL B-LOCK COUNTS C-LOCK WAITS      D-LOCKS OWNED  E-GLOBAL LOCKS
> F-CURRENT SQL   G-SQL COUNTS  H-DISTRIBUTED  I-BUFFER POOL  J-GROUP BP
> K-PACKAGES      L-RES LIMIT   M-PARALLEL TASKS N-UTILITY    O-OBJECTS
> P-CANCEL THREAD Q-DB2 CONSOLE R-DSN ACTIVITY   S-APPL TRACE   T-ENCLAVE
> U-LONG NAMES          W-ACCEL ACTIVITY
=====
>          THREAD DETAIL
PLAN
+ Thread: Plan=DSNTEP10 Connid=BATCH   Corrid=BENCH01N   Authid=CRISTIA
+ Attach:
+ Dist : Type=DISTRIBUTED ALLIED, Luwid=USIBMT6.DDFDZA1.CB56E93193C3=24190
+ Location : DZA1IDAA
act
+ Thread Activity                               User Defined Functions
+ -----
+ DB2 Status = IN-ACCEL   TCB Time (SQL) = 00:00:00.000

```

+ MVS Status	=	N/A	Wait for TCB Time	=	00:00:00.000
+ Total Elapsed Time	=	00:00:44.255	Elapsed Time	=	00:00:00.000
+ CP CPU Utilization	=	00.0%	Elapsed Time (SQL)	=	00:00:00.000
+ Total CP CPU Time	=	00:00:00.002	SQL Events	=	0
+ IIP CPU Time	=	00:00:00.000			
+ Total Parallel Tasks	=	0			
+ Current Parallel Tasks	=	0			

Selecting option **W-ACCEL ACTIVITY** provides the **THREAD ACCELERATOR DETAIL** panel as shown in Example 9-47.

Example 9-47 OMPE Classic Thread Accelerator Detail panel

```

===== ZTACEL  VTM    02      V511.#P DZA1 05/10/13 10:53:36 11
=====
>
      THREAD ACCELERATOR DETAIL
PLAN
+ Thread: Plan=DSNTEP10 Connid=BATCH Corrid=BENCH01N Authid=CRISTIA
+ Attach:
+ Dist : Type=DISTRIBUTED ALLIED, Luwid=USIBMT6.DDFDZA1.CB56E93193C3=24190
+ Location : DZA1IDAA
      accel
+ Accelerator Data for this thread follows
+ Number of accelerators accessed: 1
+
+ Name:DZA1IDAA
+
+           CPU Times           Elapsed Times
+           -----
+ In DB2      00:00:00.000      00:00:00.000
+ Accelerator 00:00:00.000      00:00:00.000
+ Accumulated wait      00:00:00.000
+
+           Counters
+           -----
+ Connects:      1 Requests:      1
+ Timeouts:      0 Failures:      0
+
+           Data Transfer
+           -----
+           Sent           Received
+           -----
+ Bytes:      2099           656
+ Messages:      11           0
+ Blocks:      0           0
+ Rows:      0           0
=====

```

Example 9-48 shows the **RESOURCE MANAGERS** main menu panel highlighting the option, **N - ACCELERATOR**.

Example 9-48 OMPE Classic resource managers main menu

```

===== ZRMMENU  VTM    02      V511.#P DZA1 05/10/13 10:39:01 2
=====
> Help PF1 Back PF3
> R.

> Enter a selection letter on the top line.
=====
> RESOURCE MANAGERS AND OTHER DB2 SUBSYSTEM INFORMATION

_ A BUFFER MANAGER ..... Buffer Manager Information
_ B LOG MANAGER ..... DB2 Log Manager Information

```

```

- C EDM POOL ..... EDM Pool Information
- D BIND STATISTICS ..... Bind Statistics
- E SUBSYSTEM MANAGER ..... DB2 Subsystem Support Manager Statistics
- F ACTIVE TRACES ..... Current Trace Activity
- G START-UP OPTIONS..... IRLM and Stored Procedures Start-Up Options
- H DSNZPARM ..... DB2 Installation Parameters
- I LOCK/CLAIM/DRAIN..... Lock Manager/Claim/Drain Statistics
- J SQL/RID POOL/PARALLEL... SQL/RID Pool/Parallelism/Stored Proc. Information
- K OPEN/CLOSE STATISTICS... Dataset Open and Close Statistics
- L DB2 COMMANDS ..... DB2 Command Statistics
- M DB2 Storage ..... Storage Management Pool Summary
- N ACCELERATOR ..... Accelerator Information

```

```
=====
```

Selecting this option leads to the Accelerator Summary panel illustrated in Example 9-49.

Example 9-49 OMPE Classic Accelerator Summary panel

```

_____ ZACST   VTM    02      V511.#P DZA1 05/10/13 10:40:10  2
> Help PF1   Back PF3      Up PF7      Down PF8      Sort PF10      Zoom PF11
> R.N
=====
>
                                     Accelerator Summary

ACST
+   Number of accelerators defined:      1
+
+
+ *
+ Name           State           Requests   Active   Max Actv
+ -----
+ DZA1IDAA       ONLINE           122       0        12
=====

```

Zooming into an Accelerator gives the panel that is shown in Example 9-50.

Example 9-50 OMPE Classic Accelerator Detail panel

```

_____ ZACSD   VTM    02      V511.#P DZA1 05/10/13 10:45:43  2
> Help PF1   Back PF3      Up PF7      Down PF8
>
=====
>
                                     Accelerator Detail

ACSD
+ Collection Interval: REALTIME           Start: 05/10 10:45:42
+ Report Interval:    1 sec               End:   05/10 10:45:43
+
+ Name                = DZA1IDAA
+ State               = ONLINE
+ Product ID          = AQT03012
+ Curr Active Requests = 6
+ Max Active Requests = 12
+ Avg Coord CPU       = 4.00%
+ Avg Worker CPU      = 5.00%
+ Total Num Processors = 96
+ Processing Capacity = 0
+ Act Worker Nodes    = 12
+ Avg Queue Wait Time (MS) = 4726
+ Max Queue Wait Time (MS) = 1276823

```

```

+ Query Queue Len 3 HR Avg =      0
+ Query Queue Len 24 HR Avg =      0
+ Max Query Queue Len      =      8
+ Avail Disk (MB)          = 33557184
+ DB Disk In Use (MB)      = 3534789
+ In-use Disk              = 12.40%

```

In the same panel, you can get more details by using PF8 to go down, as shown in Example 9-51.

Example 9-51 OMPE Classic Accelerator Detail panel, part 2

	ZACSD	VTM	02	V511.#P	DZA1	05/10/13	10:45:43	29
+			TOTAL	INTERVAL				
+			QUANTITY	QUANTITY	(1)		
+			-----	-----		-----		
+ Query Reqs Since Start			122	0		.00		
+ Failed Reqs Since Start			19	0		.00		
+ Failed Reqs Inv State			0	0		.00		
+ Total Num Accel Connects			221	0		.00		
+ Total Num Accel Requests			419	0		.00		
+ Total Timed out Reqs			0	0		.00		
+ Total Failed Reqs			0	0		.00		
+ Num Bytes Sent			1902976	0		.00		
+ Num Bytes Received			1488731	0		.00		
+ Num Msgs Sent			2431	0		.00		
+ Num Msgs Received			2180	0		.00		
+ Num Blocks Sent			0	0		.00		
+ Num Blocks Received			136	0		.00		
+ Num Rows Sent			0	0		.00		
+ Num Rows Received			10922	0		.00		

Further information is available by pressing PF8 again, as shown in Example 9-52.

Example 9-52 OMPE Classic Accelerator Detail panel, part 3

	ZACSD	VTM	02	V511.#P	DZA1	05/10/13	10:45:43	49
+			Total	Interval				
+			Times	Times				
+			-----	-----				
+ In DB2 CPU Time			00:00:00.050	00:00:00.000				
+ DB2 Elapsed Time			02-04:49	00:00:00.000				
+ Accel CPU Time			00:00:00.440	00:00:00.000				
+ Accel Elapsed Time			01-13:46	00:00:00.000				
+ Accum Wait Time			05:34:05.684	00:00:00.000				
=====								

9.3.3 Online monitoring using OMPE GUI

We downloaded and installed the IBM DB2 Performance Expert Client V5 from the web page “Service updates for Tivoli OMEGAMON XE for DB2 Performance Expert” that is available at <http://www.ibm.com/support/docview.wss?rs=434&uid=swg27013147>. Instructions and requirements for the installation are available in the section “Installing the program files of Performance Expert Client” of the IBM publication *IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS Version 5.1.1 Configuration and Customization*, GH12-6970.

In our tests, we used the GUI Version 5.1.1. More version detail information can be obtained from the menu sequence **Help** → **Product Information**, as illustrated in Figure 9-10.

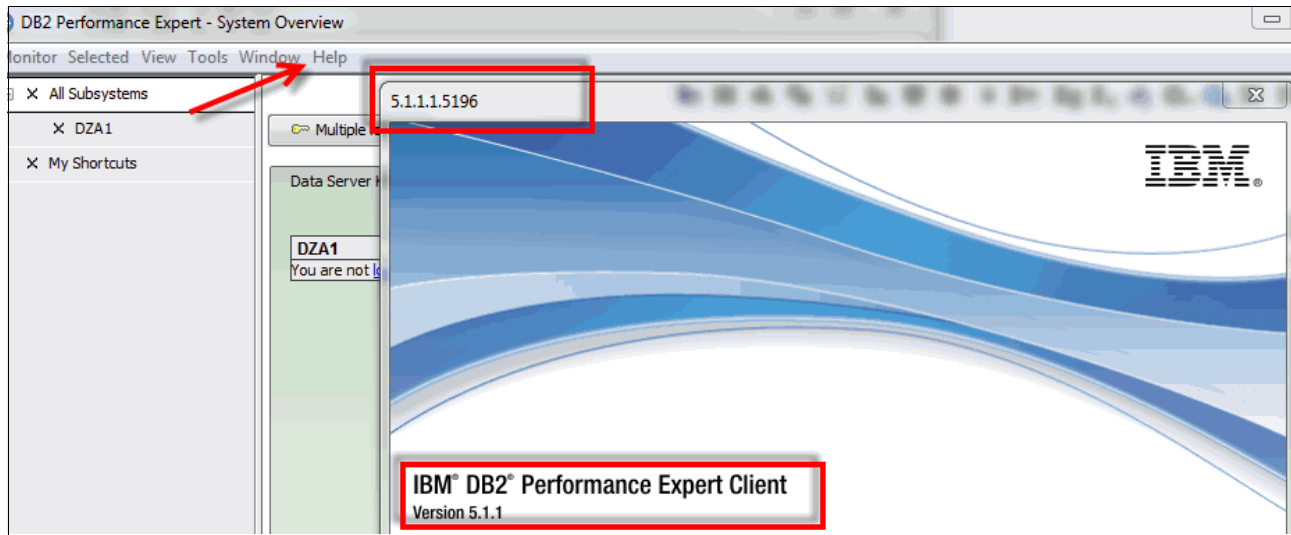


Figure 9-10 OMPE Client product version information

To monitor a DB2 system from Performance Expert Client, you must first configure a connection between the Performance Expert Client and the Performance Expert Server, which collects the performance data of this DB2 system. Example 9-53 shows the output of the **-DIS GROUP** command that is issued in our target DB2 subsystem.

Example 9-53 -DIS GROUP output example

```

DSN7100I  -DZA1 DSN7GCMD
*** BEGIN DISPLAY OF GROUP(.....) CATALOG LEVEL(101) MODE(NFM )
          PROTOCOL LEVEL(2)  GROUP ATTACH NAME(....)

-----
DB2      DB2 SYSTEM  IRLM
MEMBER  ID  SUBSYS  CMDPREF  STATUS  LVL NAME  SUBSYS  IRLMPROC
-----
.....   0  DZA1    -DZA1    ACTIVE  101 P59   IZA1    DZA1IRLM
-----

SPT01 INLINE LENGTH:      32138
*** END DISPLAY OF GROUP(.....)
DSN9022I  -DZA1 DSN7GCMD 'DISPLAY GROUP ' NORMAL COMPLETION
***
  
```

For our tests, we established a direct connection to our DB2 for z/OS. Alternatively, you can establish a connection to a target DB2 subsystem through a DB2 Connect server.

Figure 9-11 shows how to open the Statistics Details panel. The access is available under the Monitor contextual menu.

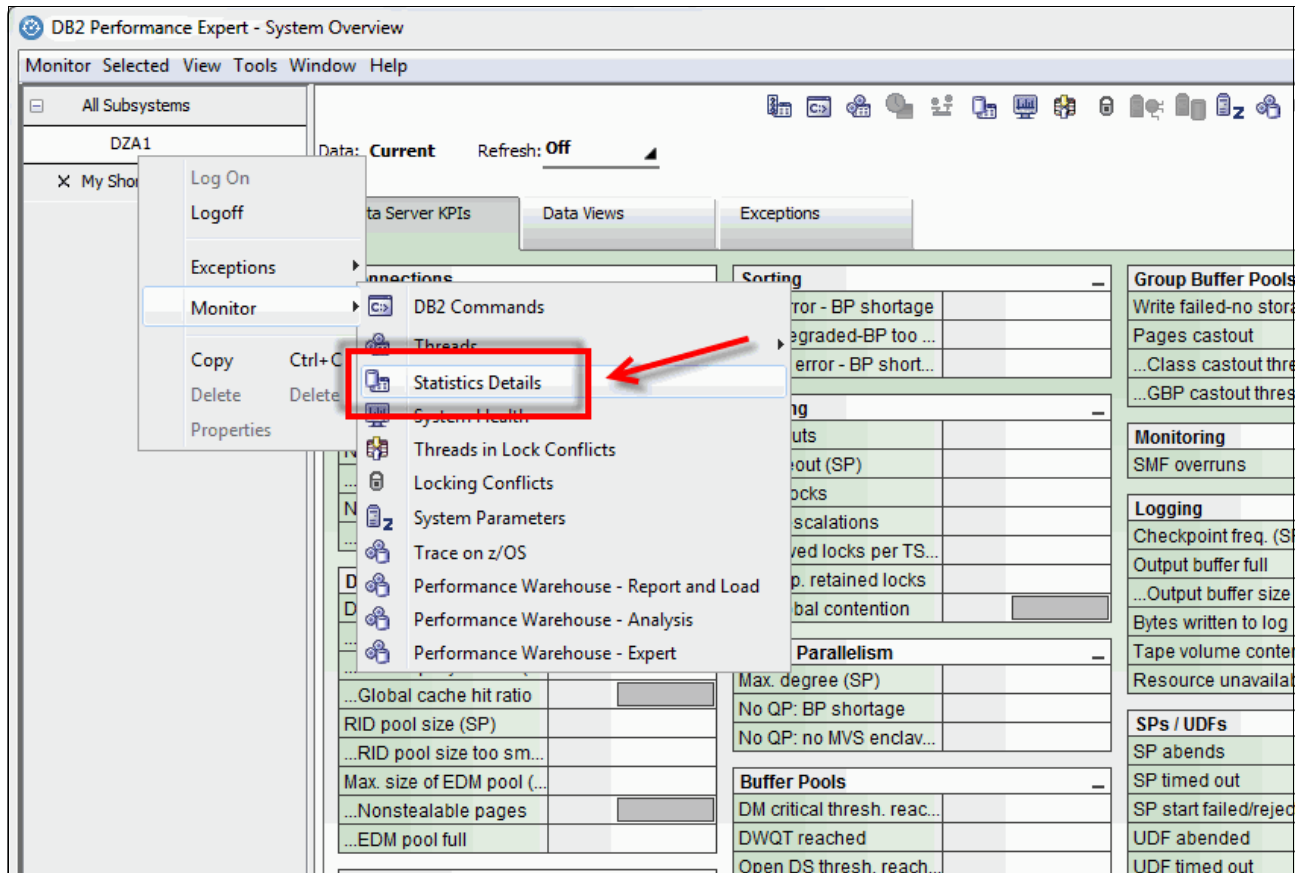


Figure 9-11 OMPE GUI Statistics Details option

The OMPE GUI 5.1.1 includes a new Accelerators section under the Statistics Details panel. An illustration of this section is shown in Figure 9-12.

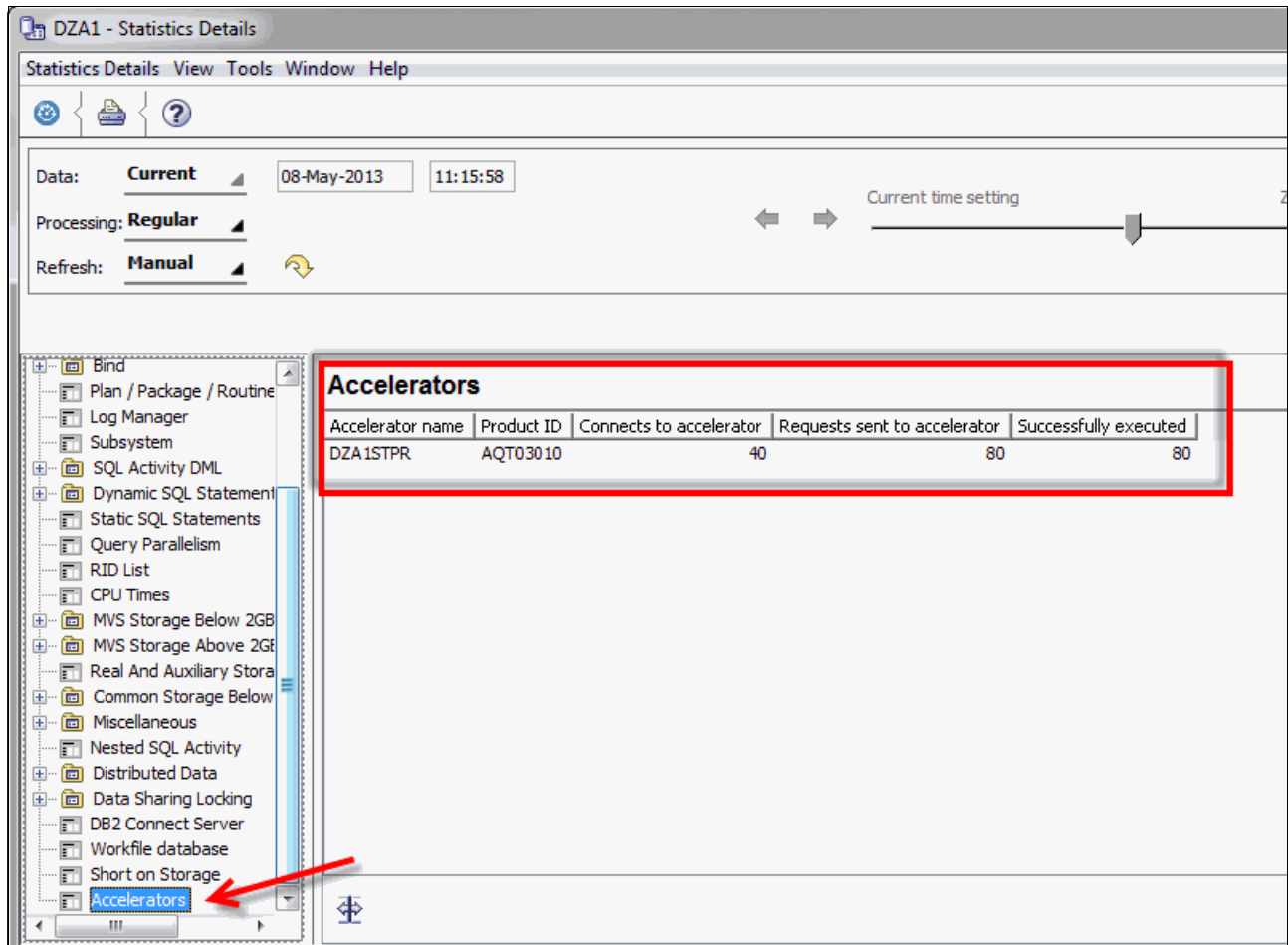


Figure 9-12 OMPE GUI Accelerators section of Statistics Details panel

Use the Accelerator Details pane to view the number of connects and responses sent to, and received from, each Accelerator that provided services to a DB2 subsystem within the reported interval.

Figure 9-13 shows an example of the Accelerator Details panel.

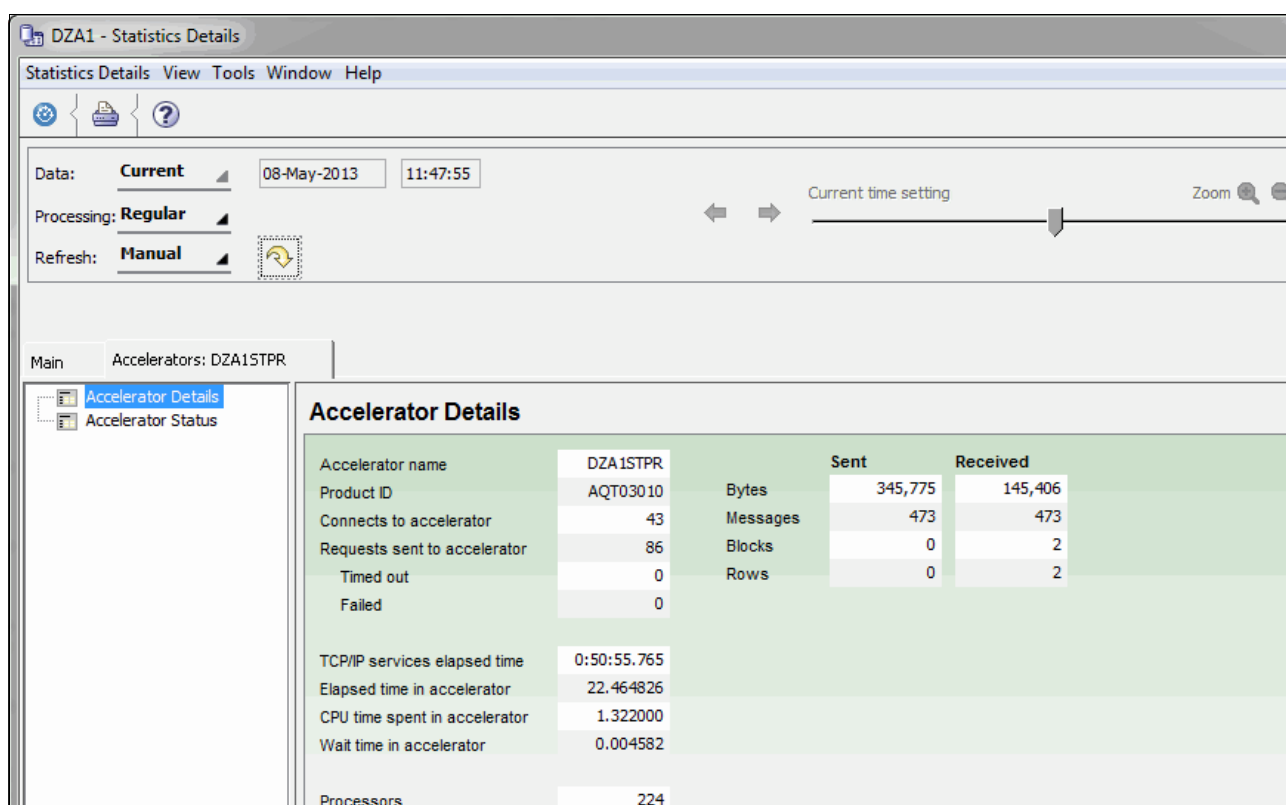


Figure 9-13 OMPE GUI Accelerator Details panel

The DB2 subsystem is connected with an Accelerator via DRDA when submitting SQL queries. The performance counters of the DRDA connection (on the upper part of the pane starting with counter Connects to Accelerator to Requests sent to Accelerator and from Bytes Sent/Received to Rows Sent/Received) are collected during the termination of the associated thread. The other Statistics counters are collected periodically from the Accelerator.

Example 9-54 shows the output of the **-DIS ACCEL DETAIL** DB2 command. Use this command to get detailed information about the status and activity of an Accelerator.

Example 9-54 -DIS ACCEL DETAIL command

```

DSNX810I  -DZA1 DSNX8CMD DISPLAY ACCEL FOLLOWS -
DSNX830I  -DZA1 DSNX8CDA
ACCELERATOR          MEMB  STATUS  REQUESTS ACTV  QUED  MAXQ
-----
DZA1STPR             DZA1  STARTED    2395    0    0    12
LOCATION=DZA1STPR HEALTHY
DETAIL STATISTICS
  LEVEL = AQT03010
  STATUS = ONLINE
  FAILED QUERY REQUESTS          =      17
  AVERAGE QUEUE WAIT             =     23 MS
  MAXIMUM QUEUE WAIT             =    107 MS
  TOTAL NUMBER OF PROCESSORS     =     224
  AVERAGE CPU UTILIZATION ON COORDINATOR NODES =    2.00%
  AVERAGE CPU UTILIZATION ON WORKER NODES    =    1.00%
  NUMBER OF ACTIVE WORKER NODES  =       7
  TOTAL DISK STORAGE AVAILABLE   = 48000960 MB

```



```

TOTAL DISK STORAGE IN USE                =      8.18%
DISK STORAGE IN USE FOR DATABASE          =  3507153 MB
DISPLAY ACCEL REPORT COMPLETE
DSN9022I  -DZA1 DSNX8CMD '-DISPLAY ACCEL' NORMAL COMPLETION
***

```

The same information can be obtained from the OMPE GUI. Figure 9-14 shows an example of how the information obtained by the **-DIS ACCEL** command is displayed in the OMPE GUI.

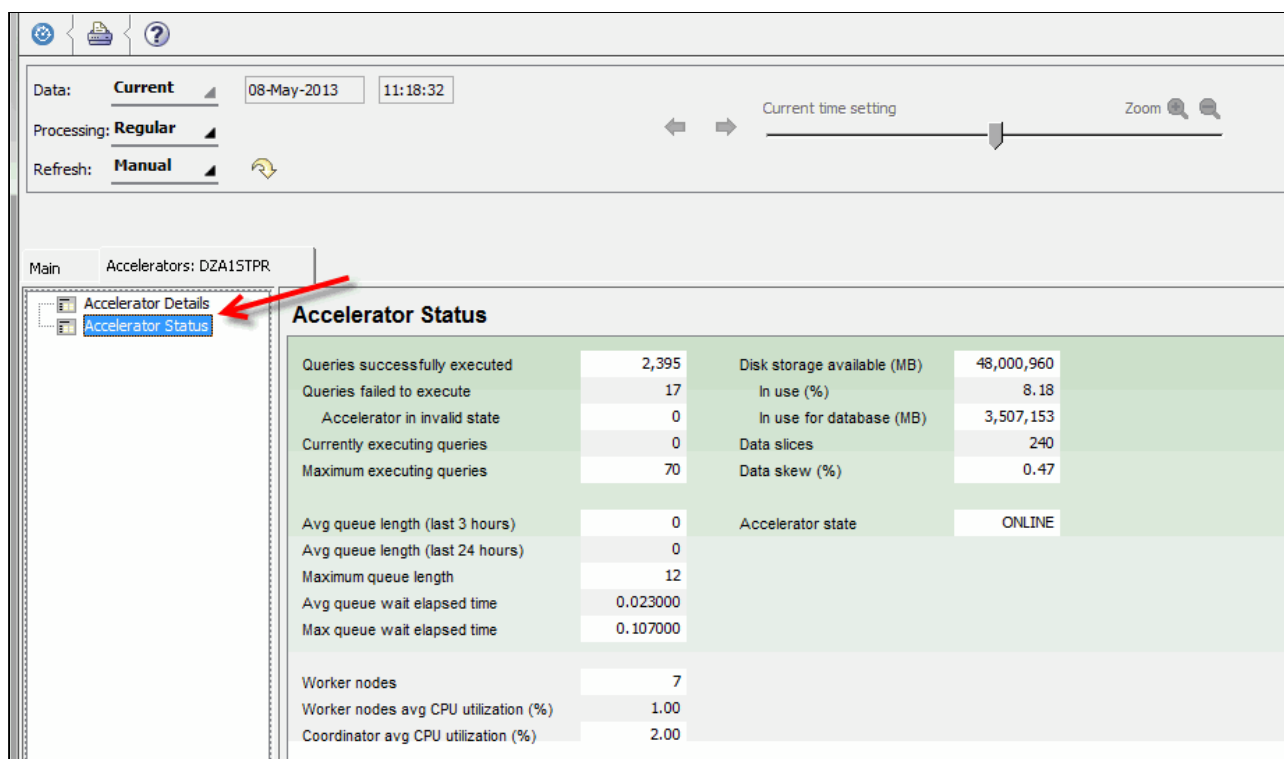


Figure 9-14 OMPE GUI Accelerator Status panel

9.3.4 The Accelerator support in the OMPE Performance Database

The OMPE Performance Database is a DB2 database, which can hold aggregated DB2 activity information spanning a long period of time. You can store performance data from the following data groups:

- ▶ Accounting
- ▶ Audit
- ▶ Locking
- ▶ Record traces (IFCIDs 22, 63, 96, and 125)
- ▶ Statistics and system parameters

You must build, load, and maintain the DB2 tables for the Performance Database manually. The product provided sample library *hlq.RKO2SAMP* contains the following items:

- ▶ The definitions of the Data Definition Language (DDL)
- ▶ The definitions of the Data Manipulation Language (DML)
- ▶ The DB2 LOAD Statements

For implementation information, refer to the section “Adding a Performance Database” of the IBM publication *IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS Version*

5.1.1 Configuration and Customization, GH12-6970. Figure 9-15 illustrates how performance data is formatted and loaded into the OMPE Performance Database.

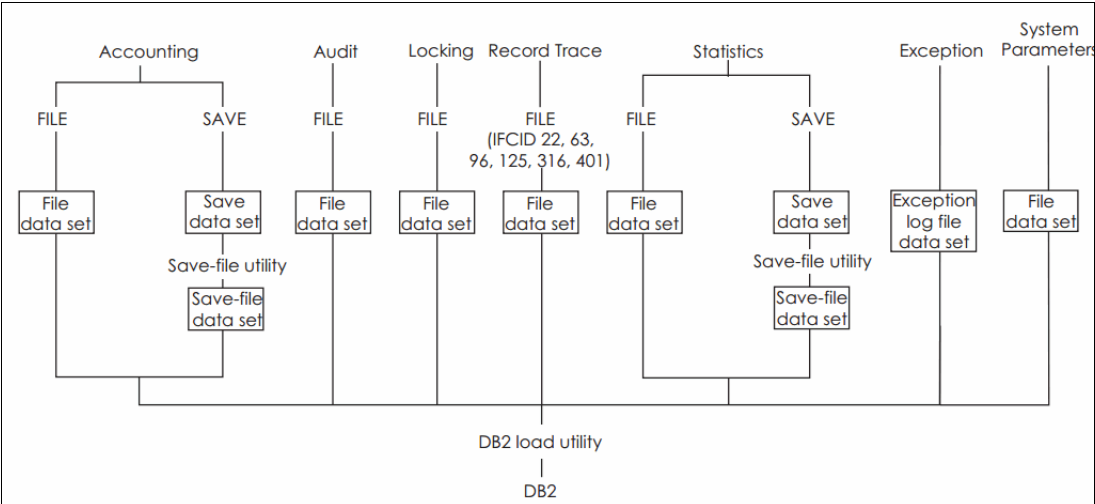


Figure 9-15 How performance data is formatted and loaded into the OMPE database

Figure 9-16 illustrates how OMPE stores accounting records on tables. This figure situates the Accelerator data table on its context. The Accelerator data table contains one row per thread-related Accelerator activity.

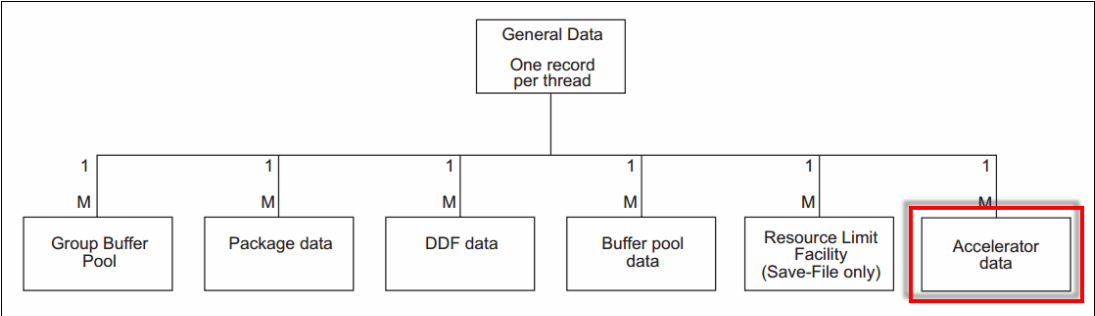


Figure 9-16 The OMPE Accounting tables

Table 9-1 lists the *hlq.TKO2SAMP* partitioned data set (PDS) members that are used to create OMPE Performance Database tables.

Table 9-1 *hlq.TKO2SAMP* PDS members used to create OMPE database tables

Type of data	CREATE TABLE statements	LOAD control statements	Table description
General	DGOACFGE	DGOALFGE	DGOABFGE
Group Buffer Pool	DGOACFGP	DGOALFGP	DGOABFGP
Buffer Pool	DGOACFBU	DGOALFBU	DGOABFBU
DDF Records	DGOACFDF	DGOALFDF	DGOABFDF
Package Records	DGOACFPK	DGOALFPK	DGOABFPK
Accelerator	DGOACFXC	DGOALFXC	DGOABFXC

These PDS members are of interest for the scope of this chapter:

- ▶ DGOABFXC: Contains column description for the table Accounting Accelerator Data
- ▶ DGOABSXC: Accounting Accelerator Save-File Data
- ▶ DGOADFXC: FILE Record Layout Description: Accounting Accelerator Data
- ▶ DGOADSXC: Save-File Record Layout Description Accounting Accelerator Data

Creating the OMPE Performance Warehousing tables

These *hlq*.RKO2SAMP PDS members contain the DDL that is required for the creation of the Accelerator-related tables:

- ▶ DGOACFXC: SQL for creating Accounting Table for data of category “ACCELERATOR”
- ▶ DGOACSXC: SQL for creating Accounting Table for SAVE data of category “ACCELERATOR”
- ▶ DGOSCXCL: SQL for creating Statistics Accelerator Table

Example 9-55 shows the DDL for creating the table, DB2PMFACCT_ACCEL.

Example 9-55 DDL in DGOACFXC, Accounting Table for data of category “ACCELERATOR”

```
CREATE TABLE DB2PMFACCT_ACCEL
(DB2PM_REL          SMALLINT          NOT NULL WITH DEFAULT,
 DB2_REL            CHAR(2)           NOT NULL WITH DEFAULT,
 LOCAL_LOCATION     VARCHAR(128)      NOT NULL WITH DEFAULT,
 GROUP_NAME         CHAR(8)           NOT NULL WITH DEFAULT,
 SUBSYSTEM_ID       CHAR(4)           NOT NULL WITH DEFAULT,
 MEMBER_NAME        CHAR(8)           NOT NULL WITH DEFAULT,
 TIMESTAMP          TIMESTAMP         NOT NULL,
 NET_ID             CHAR(8)           NOT NULL WITH DEFAULT,
 LUNAME             CHAR(8)           NOT NULL WITH DEFAULT,
 INSTANCE_NBR       CHAR(12)          NOT NULL WITH DEFAULT,
 PLAN_NAME          CHAR(8)           NOT NULL WITH DEFAULT,
 MVS_ID             CHAR(4)           NOT NULL WITH DEFAULT,
 ACCEL_NAME         VARCHAR(128)      NOT NULL WITH DEFAULT,
 PRODUCT_ID         CHAR(8)           NOT NULL WITH DEFAULT,
 CONNECTS           INTEGER           NOT NULL WITH DEFAULT,
 REQUESTS           INTEGER           NOT NULL WITH DEFAULT,
 REQUESTS_TIMED_OUT INTEGER           NOT NULL WITH DEFAULT,
 REQUESTS_FAILED    INTEGER           NOT NULL WITH DEFAULT,
 BYTES_SENT         DECIMAL(19,0)     NOT NULL WITH DEFAULT,
 BYTES_RCD          DECIMAL(19,0)     NOT NULL WITH DEFAULT,
 MSGS_SENT          INTEGER           NOT NULL WITH DEFAULT,
 MSGS_RCD           INTEGER           NOT NULL WITH DEFAULT,
 BLOCKS_SENT        INTEGER           NOT NULL WITH DEFAULT,
 BLOCKS_RCD         INTEGER           NOT NULL WITH DEFAULT,
 ROWS_SENT          DECIMAL(19,0)     NOT NULL WITH DEFAULT,
 ROWS_RCD           DECIMAL(19,0)     NOT NULL WITH DEFAULT,
 TCPIP_SRVS_CPU     DECIMAL(13,6)     NOT NULL WITH DEFAULT,
 TCPIP_SRVS_ELAPSED DECIMAL(13,6)     NOT NULL WITH DEFAULT,
 ACCEL_CPU_TIME     DECIMAL(13,6)     NOT NULL WITH DEFAULT,
 ACCEL_ELAPSED_TIME DECIMAL(13,6)     NOT NULL WITH DEFAULT,
 ACCEL_WAIT_TIME    DECIMAL(13,6)     NOT NULL WITH DEFAULT)
```

OMPE also provides support for Accelerators and statistics traces. Figure 9-17 on page 252 is a schema of the OMPE Performance Database Statistics tables showing the Accelerator data table in its context.

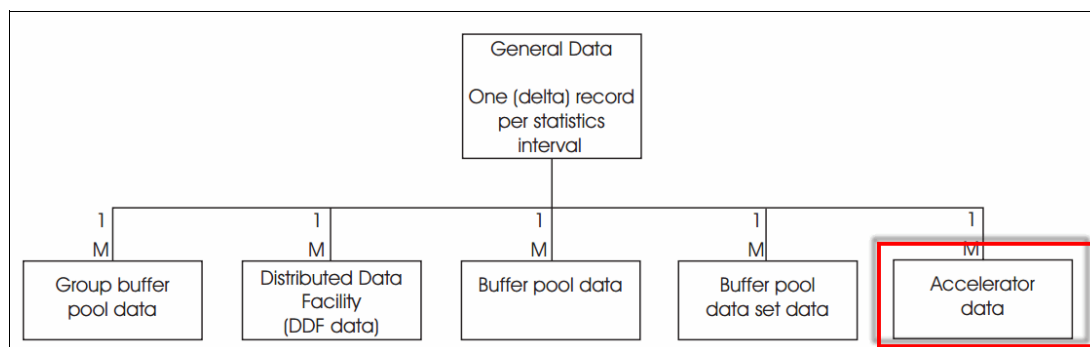


Figure 9-17 The OMPE Statistics tables

The Accelerator data contains one row per active Accelerator that is attached to the DB2 subsystem. Table 9-2 lists the PDS members that are used for creating the OMPE Performance Database Statistics tables.

Table 9-2 hlq.TKO2SAMP PDS members used to create OMPE Performance Database Statistics Tables

Type of data	CREATE TABLE statements	LOAD control statements	Table description
General Data	DGOSCGEN	DGOSLGEN	DGOSBGEN
Buffer Pool	DGOSCBUF	DGOSLBUF	DGOSBBUF
DDF	DGOSCDDF	DGOSLDDF	DGOSBDDF
Group BPool	DGOSCGBP	DGOSLGBP	DGOSBGBP
BPool Data Set	DGOSCSET	DGOSLSET	DGOSBSET
Accelerator	DGOSCXCL	DGOSLXCL	DGOSBXCL

Example 9-56 shows the DDL that is used for creating the Statistics Accelerator Table, DB2PM_STAT_ACCEL.

Example 9-56 DDL in DGOSCXCL, Statistics Accelerator Table

```

CREATE TABLE DB2PM_STAT_ACCEL
(DB2PM_REL          SMALLINT,
 DB2_REL            CHAR(2),
 LOCAL_LOCATION     CHAR(18),
 GROUP_NAME         CHAR(8),
 SUBSYSTEM_ID       CHAR(4),
 MEMBER_NAME        CHAR(8),
 INTERVAL_TSTAMP    TIMESTAMP,
 INTERVAL_ELAPSED   DECIMAL(15,6),
 BEGIN_REC_TSTAMP   TIMESTAMP NOT NULL,
 END_REC_TSTAMP     TIMESTAMP NOT NULL,
 ACCEL_NAME         VARCHAR(128) NOT NULL WITH DEFAULT,
 PRODUCT_ID         CHAR(8) NOT NULL WITH DEFAULT,
 CONNECTS           DOUBLE PRECISION NOT NULL WITH DEFAULT,
 REQUESTS           DOUBLE PRECISION NOT NULL WITH DEFAULT,
 REQUESTS_TIMED_OUT DOUBLE PRECISION NOT NULL WITH DEFAULT,
 REQUESTS_FAILED    DOUBLE PRECISION NOT NULL WITH DEFAULT,
 BYTES_SENT         DOUBLE PRECISION NOT NULL WITH DEFAULT,
 BYTES_RCD          DOUBLE PRECISION NOT NULL WITH DEFAULT,
 MSGS_SENT          DOUBLE PRECISION NOT NULL WITH DEFAULT,
 MSGS_RCD           DOUBLE PRECISION NOT NULL WITH DEFAULT,

```

BLOCKS_SENT	DOUBLE PRECISION	NOT NULL WITH DEFAULT,
BLOCKS_RCD	DOUBLE PRECISION	NOT NULL WITH DEFAULT,
ROWS_SENT	DOUBLE PRECISION	NOT NULL WITH DEFAULT,
ROWS_RCD	DOUBLE PRECISION	NOT NULL WITH DEFAULT,
TCPIP_SRVS_ELAPSED	DECIMAL(15,6)	NOT NULL WITH DEFAULT,
ACCEL_CPU_TIME	DECIMAL(15,6)	NOT NULL WITH DEFAULT,
ACCEL_ELAPSED_TIME	DECIMAL(15,6)	NOT NULL WITH DEFAULT,
ACCEL_WAIT_TIME	DECIMAL(15,6)	NOT NULL WITH DEFAULT,
AVG_QUEUED_3HRS	DOUBLE PRECISION	NOT NULL WITH DEFAULT,
AVG_QUEUED_24HRS	DOUBLE PRECISION	NOT NULL WITH DEFAULT,
MAX_QUEUED	DOUBLE PRECISION	NOT NULL WITH DEFAULT,
AVG_QUEUE_WAIT	DECIMAL(15,6)	NOT NULL WITH DEFAULT,
MAX_QUEUE_WAIT	DECIMAL(15,6)	NOT NULL WITH DEFAULT,
DISK_AVAIL_STORAGE	DOUBLE PRECISION	NOT NULL WITH DEFAULT,
DISK_STOR_IN_USE	DOUBLE PRECISION	NOT NULL WITH DEFAULT,
DISK_STOR_INUSE_DB	DOUBLE PRECISION	NOT NULL WITH DEFAULT,
DATA_SLICES	DOUBLE PRECISION	NOT NULL WITH DEFAULT,
DATA_SKEW	DOUBLE PRECISION	NOT NULL WITH DEFAULT,
ACCEL_PROCESSORS	DOUBLE PRECISION	NOT NULL WITH DEFAULT,
QUERIES_SUCCEEDED	DOUBLE PRECISION	NOT NULL WITH DEFAULT,
QUERIES_FAILED	DOUBLE PRECISION	NOT NULL WITH DEFAULT,
QUERIES_FAILED_INV	DOUBLE PRECISION	NOT NULL WITH DEFAULT,
CUR_EXEC_QUERIES	DOUBLE PRECISION	NOT NULL WITH DEFAULT,
MAX_EXEC_QUERIES	DOUBLE PRECISION	NOT NULL WITH DEFAULT,
WORKER_NODES	DOUBLE PRECISION	NOT NULL WITH DEFAULT,
CP_UTIL_WORKER	DOUBLE PRECISION	NOT NULL WITH DEFAULT,
CP_UTIL_COORD	DOUBLE PRECISION	NOT NULL WITH DEFAULT)

Example 9-57 shows the JCL that we used in our environment for loading the Accounting and Statistics tables.

Example 9-57 Loading the OMPE Performance Warehousing tables

```
//DB2CONCU JOB CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),REGION=OM,
//          NOTIFY=&SYSUID
//* -----
/* CRIS MOLARO - Acce1 3.1 REDBOOK
/* -----
//DB2PM EXEC PGM=DB2PM,REGION=OM
//STEPLIB DD DSN=OMEGAMON.V5R1M1.D120921.TKANMOD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
GLOBAL TIMEZONE(+5)
ACCOUNTING
FILE
INCLUDE(SUBSYSTEM(DZA1))
STATISTICS
FILE
INCLUDE(SUBSYSTEM(DZA1))
EXEC
//INPUTDD DD DSN=PDBPE.ISAS.J30513B1.SMFDUMP,DISP=SHR
//ACFILDD1 DD DISP=(NEW,CATLG),DSN=CRISTIA.ISAS.J30513B1.ACFILDD1,
//          SPACE=(CYL,(500,500),RLSE),RECFM=VB,LRECL=9072,
//          BLKSIZE=9076
//STFILDD1 DD DISP=(NEW,CATLG),DSN=CRISTIA.ISAS.J30513B1.STFILDD1,
//          SPACE=(CYL,(500,500),RLSE),RECFM=VB,LRECL=9072,
//          BLKSIZE=9076
//* -----
/* LOAD ACCOUNTING TABLES
/* -----
```

```

//LOADACC EXEC DSNUPROC,SYSTEM=DZA1,
//          LIB='DB2A10.ISAS.SDSNLOAD',
//          UID='CRIS.LOADIDAA'
//DSNUPROC.SYSPUNCH DD SYSOUT=*
//DSNUPROC.SYSUT1 DD DSN=PDBPE.ISAS.SYSUT1,
//          DISP=(MOD,DELETE,CATLG),
//          SPACE=(TRK,(180,90),RLSE),
//          UNIT=SYSDA
//DSNUPROC.SORTOUT DD DSN=PDBPE.ISAS.SORTOUT,
//          DISP=(MOD,DELETE,CATLG),
//          SPACE=(CYL,(500,500),RLSE),
//          UNIT=SYSDA
//DSNUPROC.SYSREC DD DISP=OLD,DSN=CRISTIA.ISAS.J30513B1.ACFILDD1
//DSNUPROC.SYSIN DD DISP=SHR,DSN=CRISTIA.UTIL.CNTL(DGOALFGE)
//          DD DISP=SHR,DSN=CRISTIA.UTIL.CNTL(DGOALFXC)
/* -----
/* LOAD STATISTICS TABLES
/* -----
//LOADSTS EXEC DSNUPROC,SYSTEM=DZA1,
//          LIB='DB2A10.ISAS.SDSNLOAD',
//          UID='CRIS.LOADSTATS'
//DSNUPROC.SYSPUNCH DD SYSOUT=*
//DSNUPROC.SYSUT1 DD DSN=PDBPE.ISAS.SYSUT1,
//          DISP=(MOD,DELETE,CATLG),
//          SPACE=(TRK,(180,90),RLSE),
//          UNIT=SYSDA
//DSNUPROC.SORTOUT DD DSN=PDBPE.ISAS.SORTOUT,
//          DISP=(MOD,DELETE,CATLG),
//          SPACE=(CYL,(500,500),RLSE),
//          UNIT=SYSDA
//DSNUPROC.SYSREC DD DISP=OLD,DSN=CRISTIA.ISAS.J30513B1.STFILDD1
//DSNUPROC.SYSIN DD DISP=SHR,DSN=CRISTIA.UTIL.CNTL(DGOSLGEN)
//          DD DISP=SHR,DSN=CRISTIA.UTIL.CNTL(DGOSLXCL)

```

Querying the OMPE Performance Warehousing tables

The OMPE Performance Database tables offer a great way of querying DB2 accounting and statistics traces using SQL. Because the traditional and Accelerator accounting data is stored in different tables, you have to join DB2PMFACCT_GENERAL and DB2PMFACCT_ACCEL to get the full activity picture for a thread. Example 9-58 shows a way of achieving this.

Example 9-58 Querying DB2PMFACCT_GENERAL and DB2PMFACCT_ACCEL tables

```

SELECT GEN.PRIMAUTH, GEN.PLAN_NAME,
       GEN.CLASS1_TIME_BEG,
       GEN.CLASS1_TIME_END,
       GEN.CLASS2_ELAPSED,
       ACC.TCPIP_SRVS_CPU,
       ACC.TCPIP_SRVS_ELAPSED,
       ACC.ACCEL_CPU_TIME,
       ACC.ACCEL_ELAPSED_TIME,
       ACC.ACCEL_WAIT_TIME,
       GEN.CLASS1_CPU_TOTAL,
       GEN.CLASS1_CPU_TOTAL,
       GEN.CLASS3_SYNC_IO, GEN.CLASS3_LOCK_LATCH
FROM   CRISTIA.DB2PMFACCT_GENERAL GEN, CRISTIA.DB2PMFACCT_ACCEL ACC
WHERE  GEN.LOCAL_LOCATION = ACC.LOCAL_LOCATION AND
       GEN.GROUP_NAME     = ACC.GROUP_NAME AND

```

```

GEN.SUBSYSTEM_ID = ACC.SUBSYSTEM_ID AND
GEN.MEMBER_NAME = ACC.MEMBER_NAME AND
GEN.TIMESTAMP = ACC.TIMESTAMP
;

```

We found that creating a view containing the full set of General and Accelerator columns was handy for analyzing test results. A partial copy of the DDL is shown in Example 9-59.

Example 9-59 Creating a view with General and Accelerator accounting information

```

CREATE VIEW PATADM.DB2PMFACCT_GENERAL_ACCELL AS
(
SELECT
GEN.DB2PM_REL
,GEN.DB2_REL
,GEN.LOCAL_LOCATION
,GEN.GROUP_NAME
....
,GEN.CSWL_STMTS_PARSED
,GEN.CSWL_LITS_REPLACED
,GEN.CSWL_MATCHES_FOUND
,GEN.CSWL_DUPLS_CREATED
,GEN.LOCK_SUSP
,GEN.CLASS3_LOCK
,GEN.LATCH_SUSP
,GEN.CLASS3_LATCH
,GEN.MVS_ID
,GEN.FALSE_CONTENTIONS
,GEN.PAR_COUNT_OR_INTV
,ACC.ACCEL_NAME
,ACC.PRODUCT_ID
,ACC.CONNECTS
,ACC.REQUESTS
,ACC.REQUESTS_TIMED_OUT
,ACC.REQUESTS_FAILED
,ACC.BYTES_SENT
,ACC.BYTES_RCD
,ACC.MSGS_SENT
,ACC.MSGS_RCD
,ACC.BLOCKS_SENT
,ACC.BLOCKS_RCD
,ACC.ROWS_SENT
,ACC.ROWS_RCD
,ACC.TCPIP_SRVS_CPU
,ACC.TCPIP_SRVS_ELAPSED
,ACC.ACCEL_CPU_TIME
,ACC.ACCEL_ELAPSED_TIME
,ACC.ACCEL_WAIT_TIME
FROM
PATADM.DB2PMFACCT_GENERAL GEN, PATADM.DB2PMFACCT_ACCEL ACC
WHERE
GEN.LOCAL_LOCATION = ACC.LOCAL_LOCATION AND
GEN.GROUP_NAME = ACC.GROUP_NAME AND
GEN.SUBSYSTEM_ID = ACC.SUBSYSTEM_ID AND
GEN.MEMBER_NAME = ACC.MEMBER_NAME AND
GEN.TIMESTAMP = ACC.TIMESTAMP
) ;

```

A very practical way of exploiting the table information is to create custom charts using DB2 for z/OS as an Open Database Connectivity (ODBC) data source and an Excel spreadsheet. After defining the DB2 subsystem that contains the OMPE Performance Database tables as an ODBC data source, create a pivot table as follows: In Excel, **Data → PivotTable and PivotChart Report**. Select **External data source** and then the **Get Data** option. This process is illustrated in Figure 9-18.

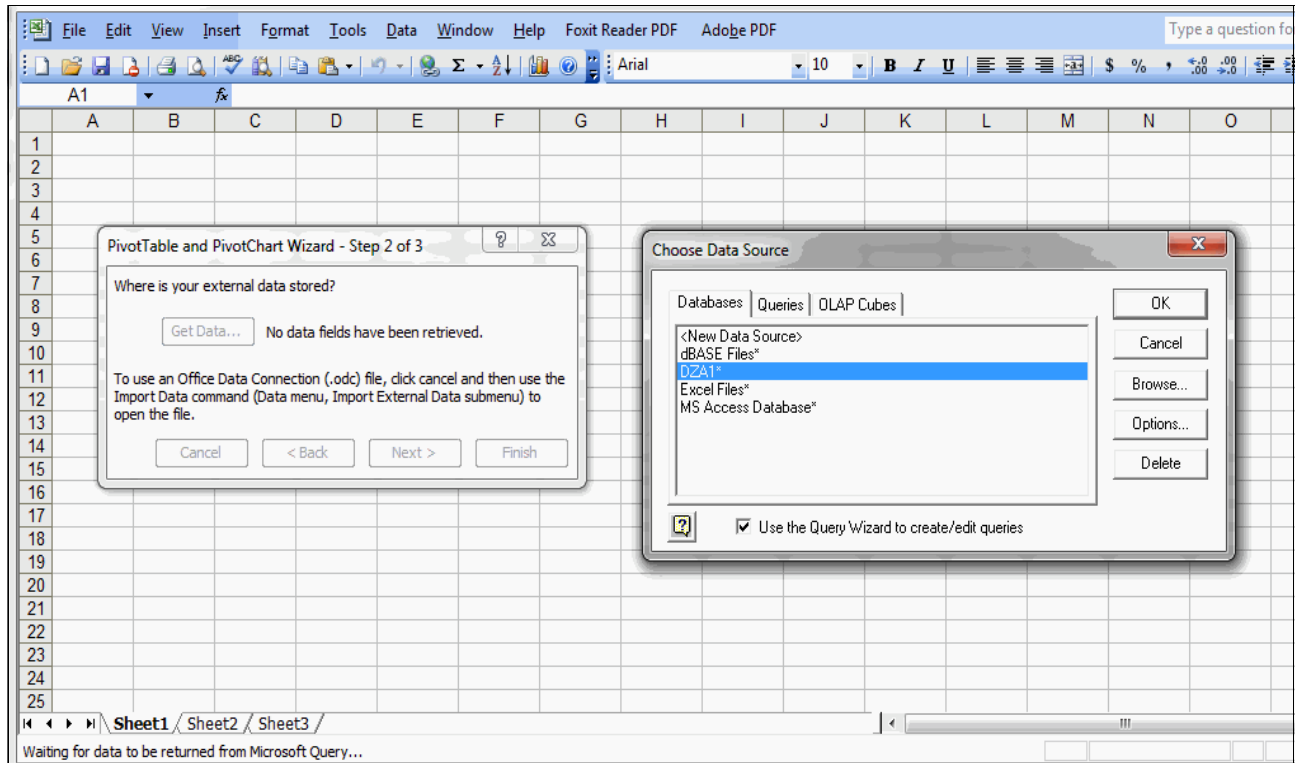


Figure 9-18 Selecting a DB2 ODBC data source

Figure 9-19 shows the prompt that you receive upon connection. Here, you enter the mainframe user ID and password.

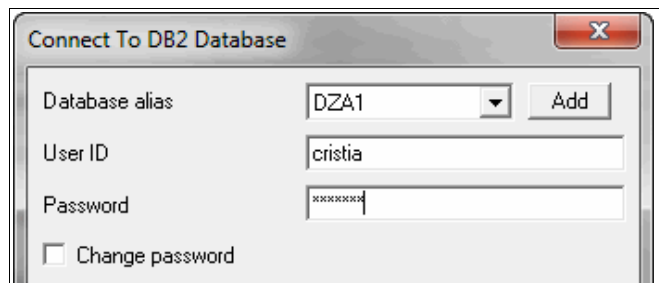


Figure 9-19 Connect to DB2 for z/OS ODBC data source

Figure 9-20 illustrates the select table dialog. In this example, we use the DB2PM_STAT_ACCEL view that was created by using the DDL in Example 9-59 on page 255.

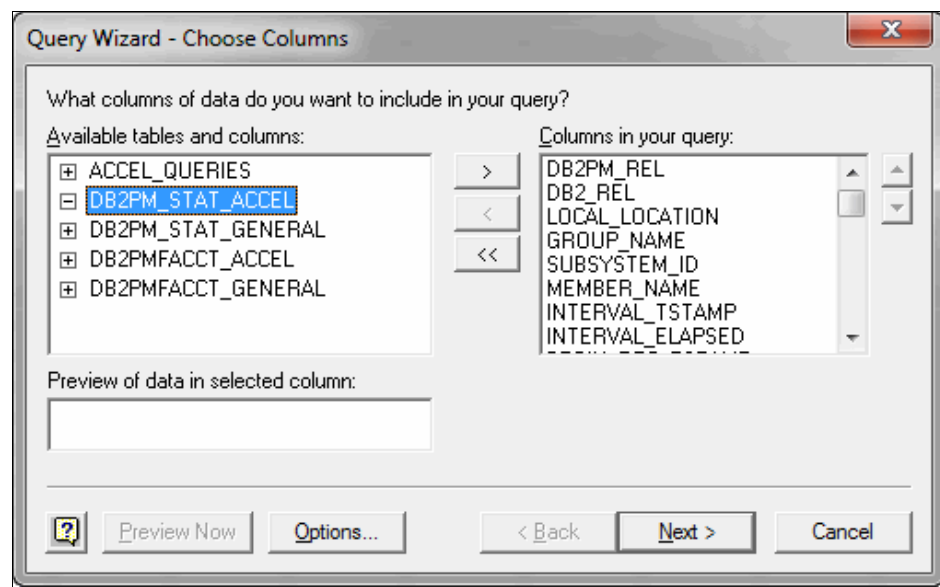


Figure 9-20 Select source table dialog

Figure 9-21 shows an example of dynamically reporting Accelerator activity using this setup. In this case, we collect BYTES that are sent and received at 1-minute intervals.

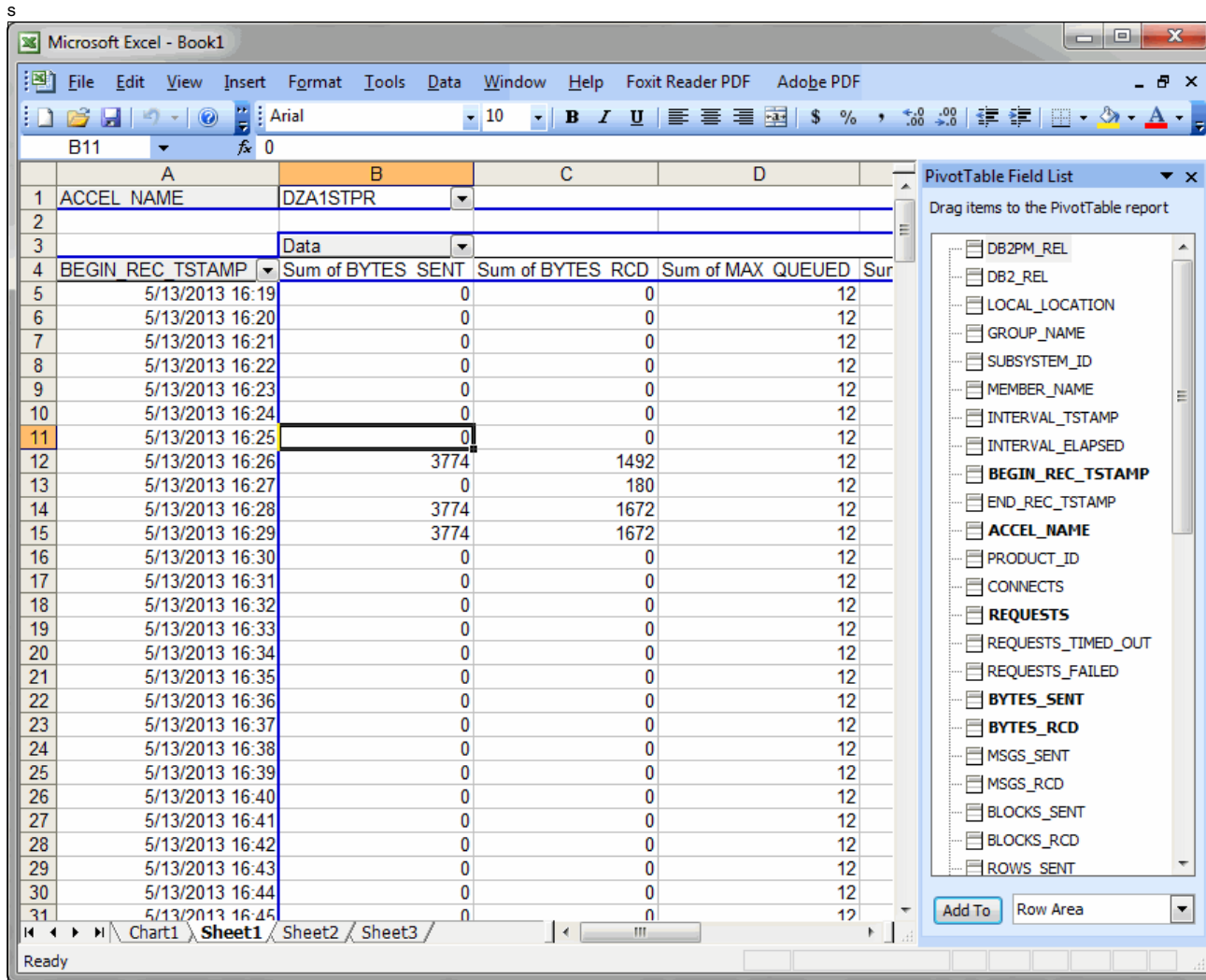


Figure 9-21 Reporting accounting data using a pivot table

Drop, load, and refresh scenario

As a practical example of using this reporting infrastructure, consider the following scenario:

1. The initial environment is composed by several tables that are accelerated, thus copied in the Accelerator
2. A table is removed from the Accelerator
3. The same table is loaded again into the Accelerator
4. After a while, the same table is refreshed

Figure 9-22 displays a chart that shows the changes on disk storage utilization as reported by the statistics tables in the OMPE Performance Database.

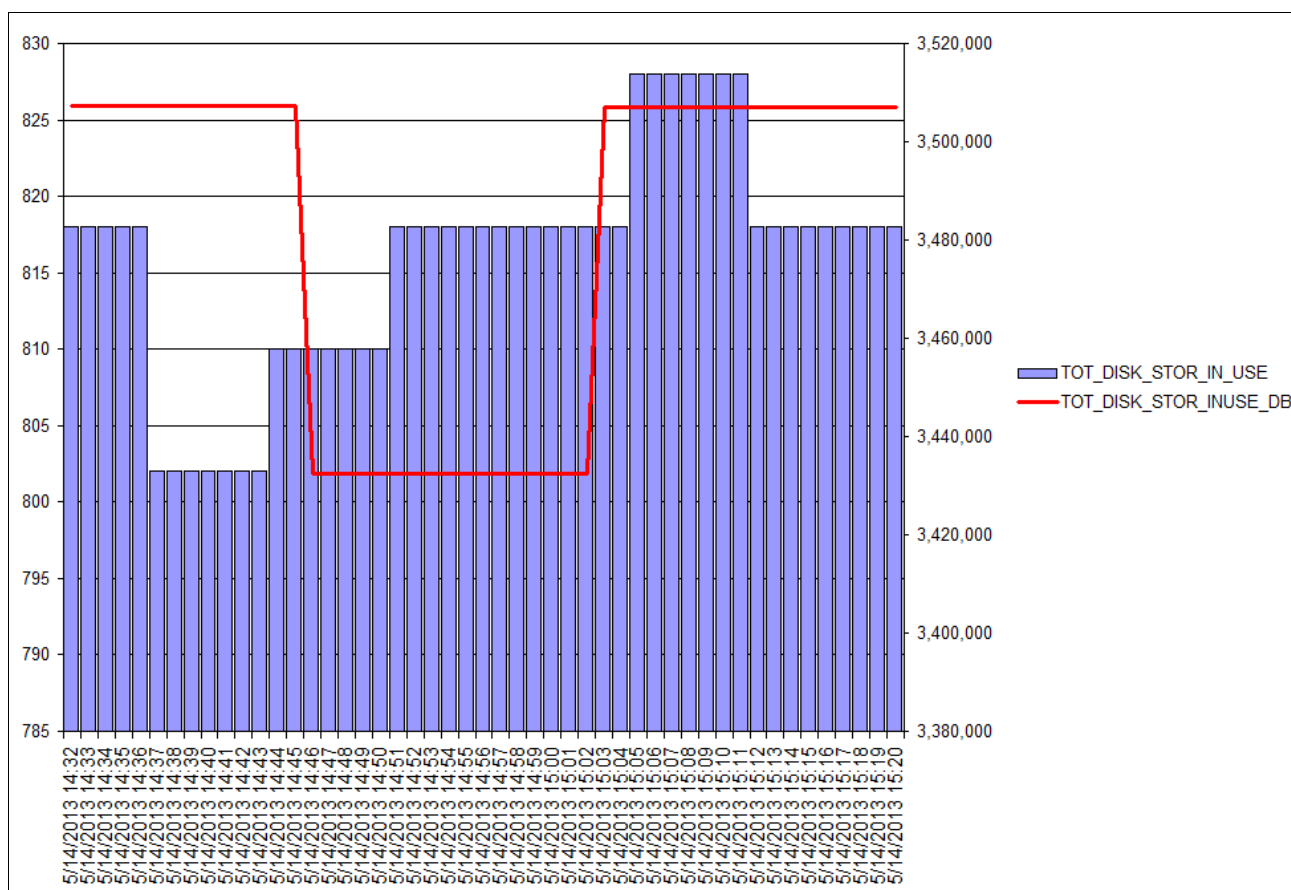


Figure 9-22 Drop, load, and refresh an accelerated table scenario

9.3.5 The Accelerator support using the new OMPE Spreadsheet Data Generator

In section 9.3.4, “The Accelerator support in the OMPE Performance Database” on page 249, we described how to take advantage of the OMPE Performance Database while running SQL queries in order to load the data into spreadsheet tools and using Pivot tables.

During our project, OMPE introduced a new feature that allows you to directly generate Statistics and Accounting data into CSV data files without going through loading the data into the Performance Database and running SQL queries.

The new OMPE feature has been introduced by APAR PM73732 (PTF UK90267 for OMPE V510, and UK90268 for OMPE V511) and follow on APAR PM87042 (PTFs UK93562 and UK93563.) A new job step takes the generated SAVE or FILE data set as input, uses a new user customizable field selection list, and generates as output the CSV data set according to the user-customized field selection list. The field selection lists for all Statistics and Accounting fields are provided by OMPE so that it can also be ready for immediate use.

For example, with this new feature, it is easy to produce reports on workload with an activated (first hour) and deactivated Accelerator (second hour), as shown in Figure 9-23.

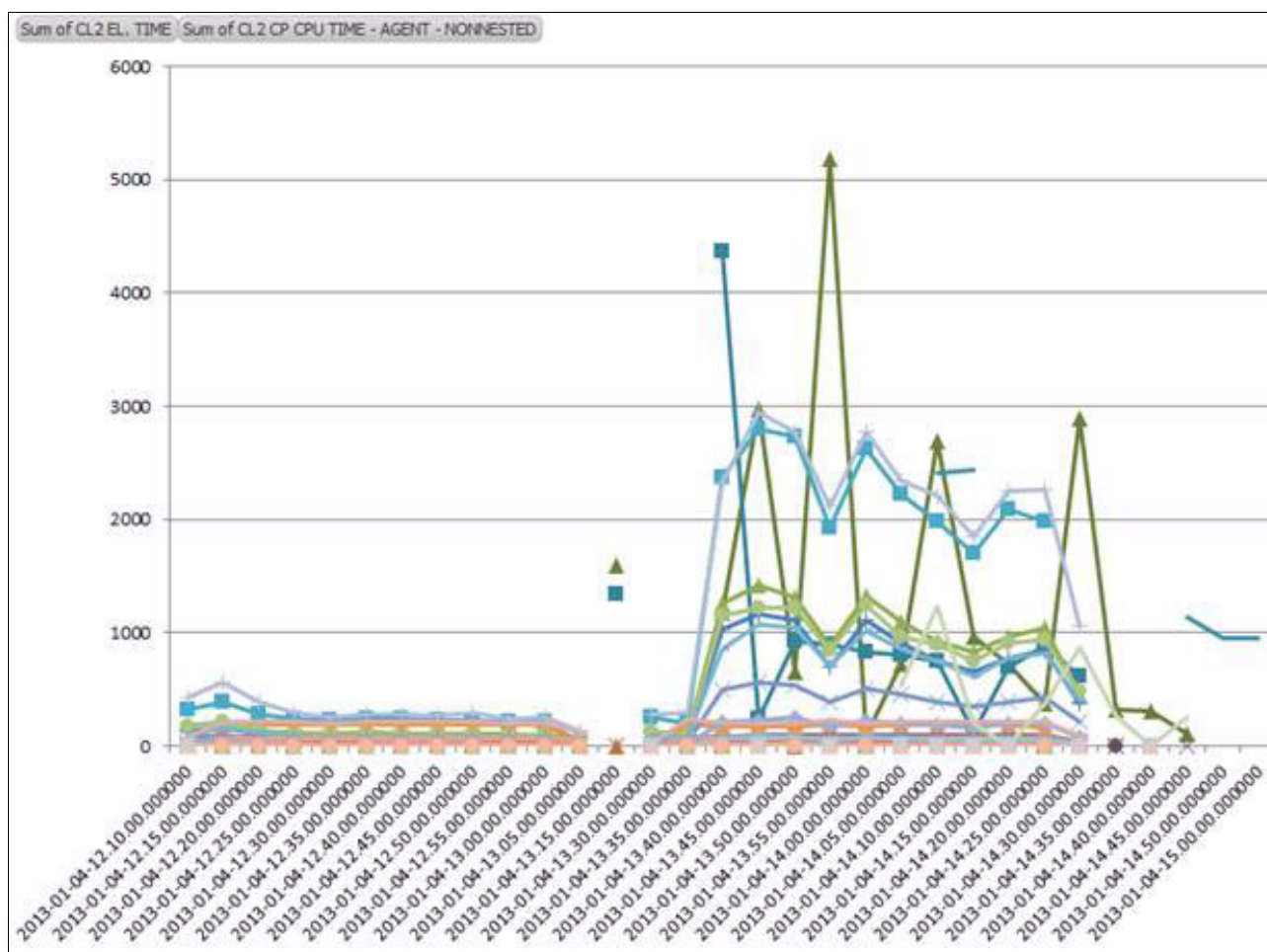


Figure 9-23 Reported CPU by hour

In this example, the non-accelerated SQL queries in the second hour consume much of the general central processor (GCP) CPU so that other transactions have to wait and show longer elapsed times.

9.4 Cataloging a DB2 for z/OS database as an ODBC data source

The DB2 Client Version 10 does not come with the Configuration Assistant. In previous versions, the Configuration Assistant, a GUI tool, could be used to catalog a DB2 for z/OS database as an ODBC data source in a Windows machine. With Version 10, the configuration has to be done by using commands. In this section, we describe the steps that are involved in the process.

Start by getting the host database configuration information by issuing the **-DIS DDF** command. The output of this command in our environment is shown in Example 9-60.

Example 9-60 -DIS DDF output example

```
DSNL080I  -DZA1 DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
DSNL081I  STATUS=STARTD
DSNL082I  LOCATION          LUNAME          GENERICLU
DSNL083I  DZA1DDF          USIBMT6.DDFDZA1  -NONE
DSNL084I  TCPPORT=446      SECPOR=448      RESPORT=4461  IPNAME=-NONE
DSNL085I  IPADDR=::135.25.80.5
DSNL086I  SQL      DOMAIN=PATP59
DSNL105I  CURRENT DDF OPTIONS ARE:
DSNL106I  PKGREL = BNDOPT
DSNL099I  DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

Note this information:

- ▶ Location: Provided in message DSNL083I. The location in this example is DZA1DDF.
- ▶ TCP/IP port: Provided in message DSNL084I. The port in this example is 446.

The IP address that is provided in message DSNL085I might not work as a target IP from your workstation depending from where it was obtained. Refer to the documentation of message DSNL085I for more details. In our environment, we can reach the DB2 server by using the IP address 9.12.44.159.

This information will be used in commands within the DB2 Command Line Processor (CLP). The CLP is part of the DB2 Client and is included in other DB2 offerings as well. Figure 9-24 shows how to open the DB2 CLP in a Windows 7 workstation.

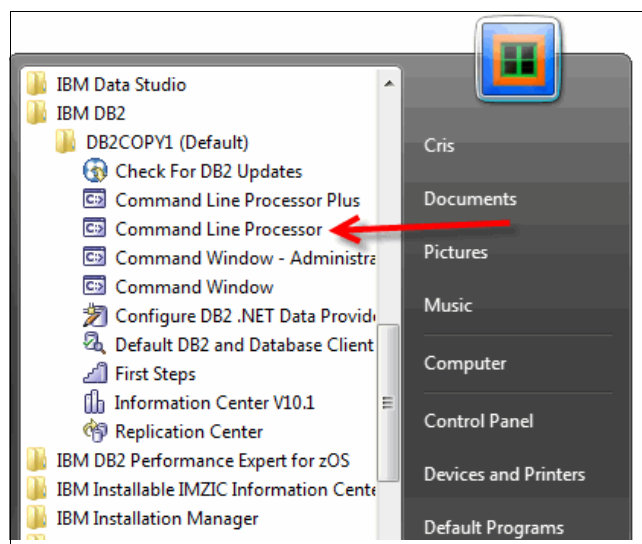


Figure 9-24 Opening the DB2 Command Line Processor in Windows

Figure 9-25 shows the contents of the DB2 CLP when opened.

```
(C> Copyright IBM Corporation 1993,2007
Command Line Processor for DB2 Client 10.1.0

You can issue database manager commands and SQL statements from the command
prompt. For example:
    db2 => connect to sample
    db2 => bind sample.bnd

For general help, type: ?.
For command help, type: ? command, where command can be
the first few keywords of a database manager command. For example:
    ? CATALOG DATABASE for help on the CATALOG DATABASE command
    ? CATALOG          for help on all of the CATALOG commands.

To exit db2 interactive mode, type QUIT at the command prompt. Outside
interactive mode, all commands must be prefixed with 'db2'.
To list the current command option settings, type LIST COMMAND OPTIONS.

For more detailed help, refer to the Online Reference Manual.

db2 =>
```

Figure 9-25 DB2 CLP example

The next step is to catalog a TCP/IP node using the CLP. Cataloging a TCP/IP node adds an entry to the Data Server Client node directory that describes the remote node. This entry specifies the chosen alias (node_name), the host name (or ip_address), and the svcname (or port_number) that the client uses to access the remote host. Example 9-61 shows the commands that are used in our environment.

Example 9-61 DB2 catalog TCP/IP node example

```
catalog tcpip node P59 remote 9.12.44.159 server 446 ostype mvs
```

Example 9-62 illustrates the output of this command.

Example 9-62 DB2 catalog TCP/IP node output example

```
db2 => catalog tcpip node P59 remote 9.12.44.159 server 446 ostype mvs
DB20000I The CATALOG TCPIP NODE command completed successfully.
DB21056W Directory changes may not be effective until the directory cache is
refreshed.
db2 =>
```

Tip: To refresh the CLP directory cache, issue a DB2 **TERMINATE**. To refresh the directory information for another application, stop and restart that application. To refresh the directory information for the database, stop (**db2stop**) and restart (**db2start**) the database.

Example 9-63 shows the output of executing a DB2 **TERMINATE** command.

Example 9-63 DB2 Terminate output example

```
db2 => quit
DB20000I The QUIT command completed successfully.
C:\Program Files\IBM\SQLLIB\BIN>db2 terminate
DB20000I The TERMINATE command completed successfully.

C:\Program Files\IBM\SQLLIB\BIN>db2
```

The node directory is created and maintained on each database client. The directory contains an entry for each remote database partition server having one or more databases that the client can access. The DB2 client uses the communication end-point information in the node directory whenever a database connection or instance attachment is requested. The entries

in the directory also contain information about the type of communication protocol to be used to communicate from the client to the remote database partition. Cataloging a local database partition creates an alias for an instance that resides on the same computer.

Example 9-64 illustrates the execution of a DB2 **list node directory** command in our environment. Use this command to verify the successful addition of the node.

Example 9-64 DB2 list node directory command

```
db2 => list node directory

Node Directory

Number of entries in the directory = 1

Node 1 entry:

Node name           = P59
Comment             =
Directory entry type = LOCAL
Protocol            = TCPIP
Hostname            = 9.12.44.159
Service name        = 446

db2 =>
```

The **CATALOG DATABASE** command stores database location information in the system database directory. The database can be located either on the local workstation or on a remote database partition server. Example 9-65 illustrates the command that is used in our environment.

Example 9-65 DB2 catalog database command example

```
catalog db DZA1DDF as DZA1 at node P59 authentication SERVER_ENCRYPT
```

Example 9-66 shows the output execution.

Example 9-66 DB2 catalog database command output example

```
db2 => catalog db DZA1DDF as DZA1 at node P59 authentication SERVER_ENCRYPT
DB20000I The CATALOG DATABASE command completed successfully.
DB21056W Directory changes may not be effective until the directory cache is
refreshed.
db2 =>
```

The **LIST DATABASE DIRECTORY** command lists the contents of the system database directory. Use this command to verify the addition of a database, as illustrated in Example 9-67.

Example 9-67 DB2 list database directory command output example

```
db2 => list db directory

System Database Directory

Number of entries in the directory = 2

Database 1 entry:

Database alias           = DZA1
Database name            = DZA1DDF
```

```

Node name                = P59
Database release level   = f.00
Comment                  =
Directory entry type     = Remote
Authentication           = SERVER_ENCRYPT
Catalog database partition number = -1
Alternate server hostname =
Alternate server port number =

```

Database 2 entry:

```

Database alias           = SAMPLE
Database name            = SAMPLE
Local database directory = C:
Database release level   = f.00
Comment                  =
Directory entry type     = Indirect
Catalog database partition number = 0
Alternate server hostname =
Alternate server port number =

```

db2 =>

Finally, connect to the target database by using the CLP for verification, as shown in Example 9-68.

Example 9-68 Connect to a DB2 for z/OS database using the CLP

```

db2 => connect to DZA1 user cristia
Enter current password for cristia:

```

Database Connection Information

```

Database server      = DB2 z/OS 10.1.5
SQL authorization ID = CRISTIA
Local database alias = DZA1

```

db2 =>

A data source, in ODBC terminology, is a user-defined name for a specific database or file system. That name is used to access the database or file system through ODBC application programming interfaces (APIs). Either user or system data sources can be cataloged. A user data source is only visible to the user who cataloged it, whereas a system data source is visible to and can be used by all other users. The **CATALOG ODBC DATA SOURCE** command is used to catalog a user or system ODBC data source. Example 9-69 shows the command to be executed in our environment.

Example 9-69 DB2 catalog ODBC data source command example

```

catalog odbc data source DZA1

```

Example 9-70 illustrates the execution results in our environment.

Example 9-70 DB2 catalog ODBC data source command output example

```

db2 => catalog odbc data source DZA1
DB20000I The CATALOG USER ODBC DATA SOURCE command completed successfully.

```

Use the **LIST ODBC DATA SOURCES** command to confirm the changes, as shown in Example 9-71.

Example 9-71 DB2 LIST ODBC DATA SOURCES command example

```
db2 => list odbc data sources
      User ODBC Data Sources
```

Data source name	Description

MS Access Database	Microsoft Access Driver (*.mdb)
Excel Files	Microsoft Excel Driver (*.xls)
dBASE Files	Microsoft dBase Driver (*.dbf)
DZA1	IBM DB2 ODBC DRIVER - DB2COPY1
db2 =>	



IBM zEnterprise Analytics System 9700

The IBM zEnterprise Analytics System 9700, formerly known as IBM Smart Analytics System 9700, is a fully integrated and scalable data warehouse and analytics System z solution designed to give organizations the insight needed to work smarter today and into the future. A 9700 implementation takes advantage of a System z hardware and operating system combined with an integrated software stack solving the needs of an Enterprise Data Warehouse (EDW) and Business Analytics (BA) environment.

In this chapter, an overview of the IBM zEnterprise Analytics System 9700 and 9710 are provided. The architecture and the special requirements and conditions of the IBM zEnterprise Analytics System 9700 and 9710 are described. The architectural overview discusses the components of the hardware and software configuration and how the solution fits together. The rest of the chapter examines the content and functionality of the zEnterprise Analytics System 9700 and 9710 solution.

The following topics are covered:

- ▶ Introduction
- ▶ Architectural overview
- ▶ Hardware specification
- ▶ Software overview
- ▶ Network specification
- ▶ Optional software components overview

10.1 Introduction

The IBM zEnterprise Analytics System 9700 and 9710 are a set of fully integrated and scalable data warehouse and analytics solutions designed to give an organization the insight it needs to work smarter. They combine software, server, and storage resources to put the right answers in the hands of the corporate decision makers today, placing your business in the best possible position to answer the questions of tomorrow.

The zEnterprise Analytics System 9700 and 9710 consist of a set of components that have been selected, tested, optimized, and priced to enable customers to deploy a next generation decision system. It has been sized for a wide variety of configurations, ranging from the modest of database sizes and users up through the petabyte systems with hundreds to thousands of users.

Deeply optimized and ready to use, the zEnterprise Analytics System 9700 and 9710 can help quickly turn information into insight and deliver that insight where and when it is needed. As a result, corporations can quickly respond to ever-changing business conditions and uncover and capture new revenue opportunities for their organization. Based on a proven infrastructure of IBM software, servers, and storage, the IBM zEnterprise Analytics System 9700 and 9710 are designed to be simply and flexibly deployed with the ability to expand to fit the evolving corporate business needs.

The zEnterprise Analytics System 9700 and 9710 is made up of two hardware solutions: the 9700 uses the IBM zEnterprise EC12 for larger enterprises, and the 9710 for mid-sized to smaller businesses by taking advantage of the zBC12. Each is described in the sections that follow.

An integral part of the zEnterprise Analytics System 9700 and an available option with the zEnterprise Analytics System 9710 is the DB2 Analytics Accelerator for z/OS.

10.1.1 zEnterprise Analytics System 9700 configuration

The zEnterprise Analytics System 9700 is based on matured, time-tested hardware and software that has set the standards for others to follow. It is the leader in security, availability, recoverability, and virtualization.

Figure 10-1 on page 272 highlights the 9700 core components, which include:

- ▶ **IBM DB2 10 for z/OS Value Unit Edition (VUE)**, the world-class leading enterprise data server designed and tightly integrated with the IBM System z mainframe to leverage the strengths of IBM System z and to reduce total cost of ownership through process enhancements and productivity improvements for database administrators and application developers.

DB2 10 for z/OS VUE provides the same robust DB2 for z/OS data server only at a one-time charge (OTC) price. DB2 VUE is also only available for eligible net new applications or workloads. Eligible applications and workloads are strictly defined and have specific requirements.

DB2 10 for z/OS is optionally available with monthly license charge (MLC) pricing.

- ▶ **IBM DB2 Analytics Accelerator for z/OS V3.1**, which blends System z and Netezza technologies, delivers mixed workload performance for complex queries at up to 2000x faster while retaining single record lookup speed, all while eliminating costly query tuning for routed query processing and providing full transparency to applications.

As just mentioned, part of the performance impact of the DB2 Analytics Accelerator comes in the form of Netezza technology. That technology is available in two different models: the IBM PureData System for Analytics N1001 (available as of October 2012) and the IBM PureData System for Analytics N2001 (available as of February 2013). The initial, earlier configurations of the DB2 Analytics Accelerator included the IBM Netezza 1000 (immediately prior to the N1001) and IBM Netezza TwinFin (predecessor to the Netezza 1000).

The IBM PureData System for Analytics N1001 delivers greater concurrency and throughput than previous generations. It delivers more than 20 times greater tactical query throughput than previous generations through improved overall I/O scan speeds and faster FPGA and CPU processor speeds. The host has 2 x 4-core Intel CPUs and the S-Blades are Intel 8-core 2.4 GHz CPUs and FPGA engines and 24 GB of RAM with a 1:1:1 ratio between the disks, processors, and FPGA.

The total capacity of the N1001 is 96 3.5 in. SAS drives per rack, 92 active, with an effective raw capacity of 96 TB. The N1001 has more rack choices with ¼, ½, 1, 1 ½, and 2 - 10 available. As you can see, the N2001 currently has far fewer.

The N2001 has some significant improvements over the still available N1001. The N2001 system even has some similarities to the System z with three attributes of particular interest: performance, energy efficiency, and reliability.

Looking at performance first, the N2001 is roughly three times faster than the previous PureData System for Analytics model, the N1001, through improved overall I/O scan speeds and faster FPGA and CPU processor speeds. The host processors now have 6-core Intel 3.46 GHz CPUs and the S-Blades are Intel 8-core 2+ GHz CPUs. However, that is not all that contributes to the N2001's performance improvement. It also boosts the overall disk scan rate to 128 GBps per rack assuming four times compression.

Next, the N2001 is energy efficient. Like so many of the systems and processors that IBM delivers today, once again more is being accomplished with less. The N2001 provides about 50% more capacity than the previous processor's model without increasing the power, cooling, or floor space requirements. All three energy efficiency improvements are important in the constant and ongoing corporate battle with the cost of doing business.

The third and last attribute is how the N2001 delivers improved reliability with more spare drives and faster disk regeneration along with the same disk mirroring as before. The N2001 has a ratio of 1 spare for every 7 drives using 34 hot spares and 240 active drives. This is more than three times the number of spares per active drive ratio than the N2001's predecessor, the N1001. The drives used also have a capacity of 600 GB per drive, down from 1 TB per drive. In a failover to a spare drive, the process takes less time because the drives are smaller and faster. However, the total capacity of the N2001 should not be of concern. The N2001 has 288 2.5 in. SAS2 drives per rack, 240 active, with an effective raw capacity of 172.8 TB. Its predecessor had 96 3.5 in. SAS drives per rack, and 92 active, with a raw capacity of 96 TB, far less than the N2001.

- **DB2 Accessories Suite for z/OS V2.2** is a bundle of components that are designed to enhance your use of DB2 for z/OS. The Accessories Suite is a download that is packaged at no charge with the zEnterprise Analytics System 9700 that consists of six separate components designed to enhance DB2. Following are the components:
 - IBM Data Studio provides health monitoring, single query tuning, and application development and database administration tools for DB2 for z/OS. Data Studio is also the base for the IBM DB2 Analytics Accelerator Studio V3.1 plug-in.
 - IBM SPSS Modeler Server Scoring Adapter for DB2 on z/OS enables customers to score predictive models built by IBM SPSS Modeler directly within a specific online transaction processing (OLTP) transaction that is running on DB2 for z/OS. This

component is essential if there are plans to add the SPSS Modeler V15 portion of the Data Analytics Pack, a zEnterprise Analytics System 9700 add-on.

- 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. The International Components for Unicode for DB/2 for z/OS is a set of C/C++ and Java libraries for Unicode support and software internalization.
 - DB2 for z/OS uses the IBM Text Search for DB2 for z/OS, which provides excellent query performance and scalability by combining optimization and state-of-the-art search technologies.
 - The last of the six components is the IBM Installable Information Management Software for z/OS Solutions Information Center, which allows access to the Information Center without the necessity of Internet connectivity.
- **DB2 Utilities Suite for z/OS V10** is a comprehensive set of tools for managing all DB2 data maintenance tasks. The Utility Suite consists of 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, STOSPACE, and UNLOAD.

The DB2 utilities operate on DB2 catalog and directory, as well as any user-defined objects.

- **z/OS 1.13** operating system stack

z/OS V1 Base (5694-A01) with the following features:

- C/C++ without Debug
- DFSMS dsshsm

DFSMSShsm is a functional component of DFSMS that is used as an automated hierarchical storage manager and productivity tool. A storage device hierarchy consists of a set of storage devices that have differing costs for storing data, differing storage amounts, and differing speeds for accessing the data. Storage devices at the lower level typically provide lower cost per byte storage and might have slower access time.

- DFSMSrmm

DFSMSrmm is also a functional component of DFSMS for managing all removable media resources including automatic, manual, and virtual tape libraries for the z/OS environment.

- DFSORT

DFSORT is the IBM high-performance sort, merge, copy, analysis, and reporting product.

- InfoPrint Server for z/OS

Infoprint Server is the basis for a total output solution for the z/OS environment. It consolidates print workload from multiple servers onto a central z/OS print server, which allows the leverage of power, scale, and resilience of z/OS. It can help deliver improved efficiency, manageability, and data usability, as well as lower overall printing cost by giving you the flexibility of using high-volume, high-speed printers, local network printers, or electronic delivery, anywhere and anytime.

- JES3 (Optional)

JES3 performs resource management and workflow management before and after job execution, while z/OS performs resource and workflow management during job execution. It considers job priorities, device and processor alternatives, and installation-specified preferences in preparing jobs for processing job output.

- RMF

RMF is the IBM strategic product for z/OS performance measurement and management. It is the base product to collect performance data for z/OS and sysplex environments to monitor systems' performance behavior and allows you to optimally tune and configure your system according to your business needs.

- SDSF

SDSF provides an easy and secure way to monitor and manage a z/OS Sysplex. It allows for the control of jobs and job output, control of devices (such as printers, readers, lines, and spool offloaders), control of system resources (such as Workload Manager (WLM) scheduling environments and Job Entry Subsystem (JES) job classes), and access and control of system log and action messages.

- z/OS Security Server

IBM Security Server is a set of features in z/OS that provides security. The most well known component is Resource Access Control Facility (RACF).

- ▶ **IBM System Storage® DS8870**, the most advanced model in the IBM high-end disk portfolio. The DS8700, with its 2304 TB maximum physical storage capacity, is designed to manage a broad scope of storage workloads that exist in today's complex data center, doing it effectively and efficiently.
- ▶ **zEnterprise EC12** (zEC12) system is the result of an investment by IBM Systems and Technology Group of more than \$1 billion in IBM research and development primarily in Poughkeepsie, New York, as well as 17 other IBM labs around the world and in collaboration with some top clients of IBM. The zEC12 increases performance of analytic workloads by 30% compared to its IBM predecessor (z196) with its 25% more performance per core, over 100 configurable cores, and 50% greater total system capacity that is again compared to its predecessor as well as taking advantage of the world's fastest chip, which is running at 5.5 GHz.

Figure 10-1 on page 272 shows a diagram of the zEnterprise Analytics System 9700 core configuration.



Figure 10-1 9700 core configuration

The surrounding add-on packs (outer circle) are optional add-ons, which can be added to the core offering. These packs extend the “out-of-the-box” analytics and data warehousing solution. The Data Analytics Pack includes Cognos 10.2 BI and SPSS Modeler 15, while the Data Integration Pack consists of the InfoSphere Information Server for Data Integration 9.1. The Fast Start Services Pack with selectable options such as predefined services, optimization services, and uniquely created customer-specific services can be tailored to meet the unique needs of each customer.

For the zEnterprise Analytics System 9700, the IBM zEnterprise EC12 (zEC12) is the foundation either as a stand-alone system or as an additional logical partition (LPAR). The zEC12 can support up to 120 processors running at 5.5 GHz and is capable of executing more than 78,000 millions of instructions per second (MIPS). This is up to 50% more total capacity than its predecessor, the z196. The zEC12 has 101 cores that can be configured as general purpose central processors (CPs), Integrated Facilities for Linux (IFLs), System z Application Assist Processors (zAAPs), System z Integrated Information Processors (zIIPs), Internal Coupling Facilities (ICFs), or system assist processors (SAPs). The zEC12 can also be configured with up to 3 terabytes of real memory. In addition, it has new features such as Flash Express, internal solid-state disk. All of the details of the zEC12 are available with the download of “zEnterprise EC12 (zEC12) data sheet” at the following site:

<http://public.dhe.ibm.com/common/ssi/ecm/en/zsd03029usen/ZSD03029USEN.PDF>

A customer can request z196 in lieu of zEC12. If this rare case arises, the customer can work with IBM to make the desired switch.

Additional details about the IBM zEnterprise Analytics System 9710 can be found at the following site:

<http://www.ibm.com/software/data/infosphere/zenterprise-analytics-system/9700>

10.1.2 System 9710 configuration

For customers looking for a more modest solution than the zEnterprise Analytics 9700, yet with the availability, security, and virtualization of a zEnterprise Analytics 9700, there is a more cost-sensitive solution called the IBM zEnterprise Analytics 9710. It is based on the IBM zEnterprise BC12 (zBC12), delivering the quality of service of System z at an entry-level cost. Customers can now deploy an IBM z/OS solution that can scale to meet the requirements for data marts and full-size data warehouses for entry-level customers.

The IBM zEnterprise Analytics 9710 is an excellent augmentation to an operational environment, providing analytic processing or an entry-level freestanding system capable of OLTP, consolidation, and analytics. It can support operational analytics, deep mining, and analytical reporting with minimal data movement, performing operational analytics with the quality of service that organizations expect from System z.

Figure 10-2 highlights the 9710 core components. The 9710 core components include DB2 10 for z/OS VUE, zBC12, z/OS 1.13 operating system stack, DB2 Accessories Suite 2.2, and DB2 Utilities Suite 10.1.



Figure 10-2 9710 core configuration

Customers have the option of adding the DS8870 Storage System and the DB2 Analytics Accelerator 3.1 to their 9710 core offering configuration. Like the 9700, the packs that are listed in the circle are optional add-ons, which can be added to the core offering.

Additional details about the IBM zEnterprise Analytics System 9710 can be found at the following site:

<http://www.ibm.com/software/data/infosphere/zenterprise-analytics-system/9710>

10.2 Architectural overview

In this section, we describe the zEnterprise Analytics System 9700 and 9710 determining the processor and memory size that is required by different query workload types and data warehouse size.

There are options that are available for configuring the hardware for both the 9700 and 9710 solutions: stand-alone, and bolt-on.

The stand-alone option ships a new System z Enterprise server that runs nothing but the zEnterprise Analytics System 9700 or 9710 solution. The bolt-on option takes the same capacity (CPs, zIIPs, IFLs, and memory) that would be shipped on a corresponding stand-alone system, and adds that capacity to an existing System z server that the customer already owns. A new LPAR is then created to use the additional capacity that is added to support the zEnterprise Analytics System 9700 or 9710.

There are a few other considerations when accepting a bolt-on solution. Capacity is added to the current server with the understanding that:

- ▶ The total amount of required capacity is split between CPs and zIIPs.
- ▶ It is possible that the SMP effect caused by adding additional CPs and zIIPs to an existing configuration might result in a lower capacity than required for a particular 9700 or 9710 configuration. It could be necessary to add additional CPs and zIIPs to compensate for this capacity reduction.
- ▶ The total number of zIIPs cannot exceed twice the number of CPs on the zEC12 and zBC12 only.

10.2.1 zEnterprise Analytics System 9700

The IBM zEnterprise Analytics System 9700 (9700) is built upon the zEnterprise EC12 (zEC12) platform. Configurations for the zEnterprise Analytics System follow a spectrum from basic to advance. One of the more significant features of the zEnterprise Analytics System 9700 is that configurations are sized based on testing.

In general, basic configurations are built to service workloads that are primarily driven by queries, with a relatively simple database and accelerator refresh process. The refresh process should not interfere with a customer's real-time processing, sometimes referred to as "fitting within a customer's batch window". The basic configuration also assumes only light, if any, INSERT/UPDATE/DELETE activity with minimal extract, load, and transfer (ELT) and ETL (extract, transfer, and load) or other processing against the warehouse data that could affect a warehouse's capacity consumption or availability.

The more advanced configurations are intended for workloads that go beyond query and basic refresh processing. Queries for these workloads tend to be more complex, and have a higher likelihood of change activity in the form of INSERT/UPDATE/DELETE activity or ELT and ETL processing. These workloads also will usually have stricter service level agreement (SLA) requirements.

Determining the size of a zEnterprise Analytics System 9700 or 9710 starts with an assumption of the complexity of the query workload and the size, or potential size, of the data warehouse. To ensure like comparisons, the database size is always stated in the amount of uncompressed data that the customer expects their warehouse will contain. The 9700's basic configurations range from a data warehouse "starter" size that contains anything 4 terabytes (TB) of size or less, through 50 TB, with special handling for anything larger than 50 TB. The

data sizes used all overlap to account for more or less SQL complexity for different amounts of warehouse data.

Different database sizes are associated to a different processor size. The starter size, the smallest configuration available, is associated with a 2827-701 (the zEC12 base model) consisting of one CP and one zIIP specialty engine with the number of CPs and zIIPs increasing by one for each grouping of data up through 50 TB. When the data warehouse size exceeds 50 TB, these assumptions are avoided and detailed sizing must be performed.

Note: Although MIPS are no longer used when describing capacity, they are used here as more of an index to compare the different workload sizes using one consistent value.

A possible scenario that might be used could involve a query workload whose complexity has already been determined that will process a certain size data warehouse. Using the potential MIPS that could be consumed, a CP/zIIP configuration is determined. The amount of memory necessary is predetermined based on the number of processors being selected. All 9700 configurations assume the use of the DB2 Analytics Accelerator.

To this point, it is assumed that the query workload falls into a more basic category as defined earlier. Should the workload take on the more complex or “advanced” workload, a different set of values are used for CPs, zIIPs, and memory. For example, the smallest configuration starts at four CPs and two zIIPs with 192 GB of memory. With a more advanced configuration, a much higher capacity requirement would be necessary. Not only do the advanced configurations assume a more complex query and more vigorous performance requirement, they also assume the additional complexity of adding maintenance into the processing stream.

Regardless of the configuration model, an increase in the capacity requirement (or an increase in query workload cost), the number of processors (both CP and zIIP), and memory requirements all increase to accommodate the larger data volumes. The more data, the more rows need to be “touched” by a query, and the more capacity will be required to meet SLA expectations. Any potential configuration over 50 TB is addressed on an individual case-by-case basis.

There are several key assumptions common to the 9700 stock configuration:

- ▶ 80% of the Mixed Query workload processing will reside on the DB2 Analytics Accelerator for z/OS.
- ▶ All of the DS8870 storage configurations are complete, new adds.
- ▶ The System Storage DS8870 configurations assume no exploitation of DB2 Analytics.
- ▶ Accelerator High Performance Storage Saver (HPSS) function is delivered in V3 of the Accelerator.
- ▶ The stock configurations are targeted for a customer’s “production” workload. It does not necessarily include resources to support a test, QA, or high availability disaster recovery (HADR) environments.
- ▶ Unlike the advanced configurations, the basic configurations all have an equal number of zIIPs configured, as the general-purpose processors. The assumption in this difference is the expectation that distributed (DDF) query workload will be the peak workload in the basic configurations, yielding quality zIIP exploitation. The anticipated peak workload in the advanced configuration will be a combination of query and other workloads that are less apt to yield quality zIIP offload.

10.2.2 zEnterprise Analytics System 9710

The zEnterprise Analytics System 9710 (9710) is built upon the zEnterprise BC12 (zBC12) or zEnterprise 114 (z114) platforms. For customers with a smaller System z footprint, the 9710 can provide them with the only integrated operational and data warehouse platform in the industry along with the best price performance in the marketplace. In order to provide a lower price point, the 9710 is a slightly different offering from the 9700 that is described above (besides the platform difference zBC12 versus EC12).

The default 9710 comes in two alternative configurations. The first configuration offers traditional System z CPs along with zIIP specialty engines. The second configuration offers the traditional System z CPs and zIIP along with an IBM DB2 Analytics Accelerator. With the 9710, both configurations offer DS8870 disk as an option. The configuration that is selected will depend upon the performance requirements.

As with the zEnterprise Analytics System 9700, there is an optional analytics pack, data transformation pack, and fast start services pack.

The 9710 can be customized in the same way as the 9700. One alternative is to spread the MIPS across more, but, smaller capacity CPs. There is a five core limit on the z114 and a six core limit for the zBC12 (up to 58% more capacity as compared to the previous z114 with double memory at 496 GB) that must be dealt with. Typically, the more CPs, the better the parallel execution of queries within DB2 for z/OS. However, with the Accelerator there could be less need to take advantage of parallel performance within DB2 for z/OS because the queries that require parallel execution can often be offloaded to the Accelerator. The Accelerator however, does not negate the need for a zIIP specialty engine. Workload could still exist that can take advantage of a zIIP. Typically, a zIIP processor is deployed along with a general processor in order to provide greater capacity at a lower cost.

The z114 (Model M10) can be configured with a total of 10 processors, from one - five CPs, and from zero - five specialty processors. The zBC12 (Model H13) can be configured with a total of 13 processors, from one - six CPs, and from zero - six specialty processors. The 9710 can also be configured with up to 10 IFLs, or 10 Integrated Coupling Facilities (ICFs) for the z114, and 13 IFLs or 13 ICFs for a zBC12.

10.2.3 IBM System Storage DS8870

The suggested disk configurations for the zEnterprise Analytics System 9700 are based on the IBM System Storage DS8870 Model 961 (base model) and 96E (expansion model). The DS8870 offers up to three times faster performance and 20% higher energy efficiency than its direct predecessor, the DS8800. The DS8870 excels in supporting the IBM zEnterprise EC12 server environment, is tremendously scalable, and has virtualization capabilities.

The physical storage capacity for the DS8870 is contained in the disk drive sets. A disk drive set contains 16 disk drive modules (DDMs), the installed disk, which each has the same capacity and the same drive speed or revolutions per minute (rpm). Each DDM in the 9700 configuration has a 300 GB capacity and a 15,000 rpm drive speed. In addition, the 9700 DS8870 storage capacity is configured as RAID 5 and each DDM is an industry-standard serial-attached SCSI (SAS) disk. There is also an IBM POWER7® processor-based server, which contains the processor and memory that drive all functions within the DS8870.

In descriptions about disk requirements and the determination of the capacity that is necessary for one of the sized data warehouses, the terms *raw user data*, *usable storage goals*, and *total usable space* are used:

- ▶ **Raw user data**

Raw user data is the amount of raw uncompressed user data in (or proposed for) the data warehouse.

- ▶ **Usable storage goals**

The usable storage goal is storage estimation that is based on 1.5x “raw user data”. This amount is an attempt to estimate the entire disk storage requirement that not only includes the raw user data that is necessary to create the data warehouse, but all the supporting data structures such as indexes and data sets that are required in support of DB2 for z/OS. Some examples of the objects making up the 1.5 multiple are table space, indexes, database overhead, metadata (metadata can include views, catalog information, logs, and so on), MQTs (MQTs and other physical representation of data in use), sort space, and utility work space.

Usable storage is an approximation for planning purposes based on the device type. The usable storage goal in most cases is going to be slightly less than the total usable space on the device.

- ▶ **Total usable space**

Total usable space is the total calculated available storage that is based on the device type after applying a device overhead percentage factor.

- ▶ **Cache**

Cache represents a value that will satisfy the performance requirement with a reasonable amount of cache that is specified to allow the cache prefetch algorithms to optimally perform. The cache value should scale up linearly. For example, twice the cache of the 5 TB configuration should be used for a 10 TB warehouse configuration, or four times the 5 TB configuration for a 20 TB warehouse configuration.

Any disk information that is supplied should be considered only as approximations and only used for planning purposes.

10.3 Hardware specification

The following hardware components are included:

- ▶ IBM zEnterprise EC12 (2827-702)
- ▶ IBM Storage System DS8870 (2421-951)
- ▶ PureData System for Analytics N1001-010 for System z Appliance Install Annual Appliance; Maintenance + Subscription and Support Renewal 12 Months (E0CW7LL)
- ▶ PureData System for Analytics N1001-010 or System z Appliance Install Appliance + Subscription and Support 12 Months (D0LHHLL)
- ▶ (optional) IBM PureData System for Analytics N1001-010 Appliance Install Initial Appliance Business Critical Service Upgrade 12 Months (D0LH0LL)
- ▶ Upgrade of warranty to business critical level for the Netezza appliance. This moves the HW warranty from being a 9 - 5, Monday - Friday deal to a 24x7 warranty
- ▶ IBM PureData System for Analytics N1001-010 Appliance Install Subsequent Appliance Business Critical Service Upgrade 12 Months (E0CW8LL)

- Upgrade of support to business critical level for the Netezza appliance. This moves the HW support from being a 9 - 5, Monday - Friday deal to a 24x7 support

Note: Additional details for configuration of the DB2 Analytics Accelerator as well as prerequisites that are required for the DB2 Analytics Accelerator can be found in Appendix B, “Notes on maintenance” on page 307.

10.4 Software overview

In this section, we provide a general overview of the software components.

10.4.1 System z software for 9700

The predetermined software stack provides program products that are required to deliver a data warehousing solution. Unlike Solution Editions, this stack is fixed and cannot be modified. The following products are included in the z/OS software stack for Core 9700 LPAR for the database offering:

- MLC products
 - z/OS V1 Base (5694-A01) with the following features:
 - C/C++ without Debug
 - DFSMS dsshsm
 - DFSMSrmm
 - DFSORT
 - InfoPrint Server
 - JES3 (Optional)
 - RMF
 - SDSF
 - z/OS Security Server
- OTC products
 - DB2 10 for z/OS VUE (5697-P31)
 - Utilities Suite V10 (5655-V41)
 - DB2 Accessories Suite for z/OS v2.2 (5697-Q02)
 - IBM SDK for z/OS, V5 (31-bit) (5655-N98)
 - IBM SDK for z/OS, V5 (64-bit) (5655-N99)
 - DB2 Analytics Accelerator for z/OS V3.1 (5697-DAA)

10.5 Network specification

The IBM Smart Analytics System 9700 uses the following networks:

- z/OS
 - The z/OS network uses the z/OS Communications Server (TCP/IP), which is connected to the z/VM LPAR via IBM HiperSockets™.
 - The z/OS Communications Server provides a secure platform for developing and sharing mainframe workloads in a TCP/IP environment.

- ▶ **z/VM**

TCP/IP has been configured according to specifications provided by the customer. For example, the z/VM user ID of the z/VM TCP/IP stack virtual machine is TCPIP. The host name, domain name, domain IP address, device number, and IP address have already been preconfigured according to the installation specifications. Path maximum transmission unit (MTU) discovery is enabled and queued direct I/O (QDIO) (layer 3) has been selected. The network type is Ethernet and the MTU size is set to 1500.

- ▶ **Linux on System z network**

TCP/IP connectivity for Linux guests. Virtual network interfaces allow the real connections to be shared. The virtual network connection that is used here is via VSWITCH. In addition to providing a network of virtual adapters, the switch is connected directly to an OSA-Express QDIO adapter

10.6 Optional software components overview

Three different and separate optional packs are available to both the 9700 and 9710 core configurations. They include:

- ▶ Data Analytics Pack
- ▶ Data Integration Pack
- ▶ Fast Start Services Pack

It is your choice to include none, one, two, or all three to customize the 9700 and 9710 configurations to meet your needs. These optional packs are only available as add-ons to the core components.

10.6.1 Data Analytics Pack

The Data Analytics Pack provides the business analytics software. These software products run on Linux for System z and require the addition of an IFL specialty processor. An IFL is dedicated to Linux workloads.

The additional software products included are IBM Cognos Business Intelligence V10.2 for Linux on System z and IBM SPSS Modeler V15 for Linux on System z Server and Client.

- ▶ **Cognos Business Intelligence V10.2 for Linux on System z (5724-W12)**, along with IBM Cognos BI Search and IBM Cognos for Microsoft Office, provides the business user with the ability to interact, search, and assemble all perspectives of a business. It also provides a limitless business intelligence workspace to support how people think and work. It delivers the complete range of business intelligence capabilities including query and reporting, analysis, dashboarding, and scorecarding on a single, service-oriented architecture (SOA). It expands traditional business intelligence with planning, scenario modeling, real-time monitoring, collaboration, and predictive analytics that extend your business intelligence capabilities.

Cognos Business Intelligence V10.2 includes the following capabilities:

- Enhanced visualization
 - Full fidelity publish allows a user to publish and share content from IBM Cognos Insight to IBM Cognos Workspace (previously named IBM Business Insight), providing greater insight by incorporating personal analytics alongside enterprise content.

- The visualization coach in IBM Cognos Workspace provides automatic chart recommendation that analyses data and produces the most suitable display type, helping users save time in determining the best visualization for any given data set.
- Tabbed workspaces with global filtering in IBM Cognos Workspace allow users to gain greater context around information by getting a consistent view across a tabbed workspace.
- Graduated capabilities in IBM Cognos Workspace can help increase user acceptance by providing just the right amount of functionality within the workspace.
- Enhanced integration
 - IBM Cognos Workspace integration with IBM WebSphere Business Process Management expands the scope of information for decision making by enabling users to pass context from IBM Cognos Workspace to the WebSphere Business Process engine to allow human interaction for review purposes.
 - Support for IBM InfoSphere BigInsights™ provides access to massive amounts of data, including unstructured data with linear scale.
- Enhanced performance
 - Multipage report trickle feed for IBM Cognos Mobile reports (both native application and web application) provides faster time to insight as mobile users can now access and interact with each page as the report loads rather than wait for the entire report to download to the mobile device.
 - IBM Cognos Workspace provides a 30% increase in search performance, which ensures that users can search and find even more relevant results faster.
 - Dynamic Cubes provide low latency, high performance, in-memory online analytical processing analytics over large volumes of data (greater than 1 TB), which reduce data movement and computation, and improve responsiveness.
- **SPSS Modeler V15 Premium** for Linux on System z Server (5725-A65) and Client (5725-A64) IBM SPSS Modeler V15.0 is a high-performance predictive and text analytics workbench that enables organizations to gain unprecedented insight from their data and helps deliver a positive return on investment by embedding predictive analytics into business processes.

SPSS Modeler Premium goes beyond the analysis of structured numerical data alone and includes information from unstructured data such as web activity, blog content, customer feedback, emails, articles, and more to create more accurate predictive models. New features include entity analytics that further extends capabilities to resolve relationships between data originating from different sources and address issues with data quality and matching. Social network analysis is a new capability that facilitates the discovery of relationships among social entities and the implications of these relationships on an individual's behavior.

IBM SPSS Modeler Server Scoring Adapters allow data to be scored by the generated models within the database, which provides significant performance improvements.

Scoring data is defined as deploying a predictive model on new data with an unknown outcome. The scoring adapters allow this process of evaluating each record and producing a score (or prediction) to be done within the database without having to export the data from it, run it through the model, and import it back again.

SPSS Modeler 15 ported scoring functions into DB2 on z/OS to take advantage of zEC12 performance. Scores can be invoked by SQL statements and run in user-defined functions (UDFs). Rather than extracting data from DB2 and sending it to SPSS on another platform, it can be scored within the scope of a DB2 transaction, against the primary data for faster, more accurate results. Lab measurements demonstrate the ability to drive

10,000 scoring transactions per second with 15-millisecond response times. Moving the computation to the OLTP data includes the following benefits:

- Reduces latency
- Minimizes data movement
- Enables more complex scores to be run within SLAs
- Lessens complexity
- Benefits from the high qualities of service of System z

The Data Analytics Pack does require a sizing recommendation for the optional add-on in support of business analytics software: Cognos Business Intelligence V10.2 for Linux on System z, and SPSS Modeler for Linux on System z V15. These sizings are in support of the Cognos pricing model.

Based on the number of proposed Cognos concurrent users, the number of IFL specialty engines and memory size can be determined. As the number of concurrent users increase, the number of IFL and the amount of memory increase accordingly. These configurations are based on the mixed market workload. A Cognos approved sizing can be substituted.

Note: The pool of IFLs will be shared by all products.

It should also be noted that the SPSS Modeler for z/Linux is limited to just one IFL.

10.6.2 Data Integration Pack

The Data Integration Pack provides data movement and transformation, data discovery, and real-time delivery. This comprehensive package of data integration and delivery components is complemented with full metadata management and lineage management. This metadata stores glossary terms, data structure definitions, and tracks the source and all the modifications of data that are being analyzed to provide user confidence with the trusted data.

A Linux for System z LPAR (and IFL) is required for the Data Integration Pack. If the Data Integration Pack is being installed with the Data Analytics Pack, they can be installed in the same LPAR or in separate LPARs.

InfoSphere Information Server for Data Integration 9.1 is the software that is included in the Data Integration Pack.

IBM Information Server for System z is a fully integrated software platform that profiles, cleanses, transforms, and delivers information from mainframe and distributed data sources alike to drive greater insight for the business without added IBM z/OS operational costs. It can help you derive more value from the complex, heterogeneous information that is spread across your systems. It enables your organization to integrate disparate data and deliver trusted information wherever and whenever it is needed, in line and in context, to specific people, applications, and processes. It helps business and IT personnel collaborate to understand the meaning, structure, and content of any information across any source.

InfoSphere Information Server for Data Integration 9.1 in turn includes:

- InfoSphere Blueprint Director

Blueprint Director is used to define and manage a blueprint of your information project landscape from initial sketches through delivery. It provides consistency and connectivity to your information architecture solution by linking the solution overview and detailed design documents together. This linkage allows all team members to understand the project as it evolves.

- ▶ **InfoSphere Change Data Delivery (cannot be used with DB2 for z/OS)**
Change Data Delivery helps provide rapid and timely delivery of data changes, which ensures that critical information is readily available to lines of business, thereby increasing visibility and productivity. It integrates directly with InfoSphere DataStage, enabling real-time ETL processes and event-driven data cleansing.
- ▶ **InfoSphere Data Architect**
Data Architect is a collaborative data design solution that enables you to discover, model, relate, standardize, and integrate diverse and distributed data assets throughout an enterprise. It includes support for column-organized tables and can offer a better understanding of current data assets to help increase efficiency and reduce time to market.
- ▶ **InfoSphere Data Click**
Data Click simplifies data movement and eases data placement. It can be used to offload warehouse databases or offload select schemas and tables within warehouse databases by retrieving data so that it can be worked on in a test environment. For example, an operational database can be moved to a private sandbox so that data can be isolated for experimenting with data transformations, or for creating reports from subsets of the data. By isolating and analyzing the data in a test environment, the integrity of the business information in the production environment is not placed in jeopardy.

InfoSphere Data Click relies on InfoSphere DataStage and can also use InfoSphere Change Data Capture to provide efficient extraction, high throughput, and with minimum risk to your production system.
- ▶ **InfoSphere DataStage**
DataStage is built for the simplest to the most sophisticated data transformations performed every day that require ease of use and is designed to work the way that you think.
- ▶ **InfoSphere DataStage and IBM QualityStage® Designer**
DataStage and QualityStage provide a graphical framework that you use to design and run the jobs that transform and cleanse your data.

QualityStage is a data cleansing, data standardization, data matching, and data validation solution. When information is coming in from various sources, there is typically often a need to synthesize all of the data so that it has a common format or standard for the target environment. QualityStage performs that business purpose.
- ▶ **InfoSphere Discovery for Information Integration**
InfoSphere Discovery provides the capabilities to automate the identification and definition of data relationships across the complex, heterogeneous environments that are prevalent in IT organizations today.
- ▶ **InfoSphere FastTrack**
FastTrack automates multiple data integration tasks from analysis to code generation, while incorporating the business perspective and maintaining lineage and documented requirements.
- ▶ **InfoSphere Information Services Director**
Information Services Director provides an integrated environment that enables you to rapidly deploy information server logic as services. It utilizes other components of IBM InfoSphere Information Server for understanding, cleansing, and transforming information and deploys those integration tasks as consistent and reusable information services.

- **InfoSphere Metadata Workbench**

Metadata Workbench provides a visual, web-based exploration of metadata that is generated, used, and imported by IBM InfoSphere Information Server. InfoSphere Metadata Workbench improves transparency of IT to the business, and increases IT responsiveness by tracing and maintaining the relationship paths of information throughout a data integration project.

Users of the different product modules of InfoSphere Information Server can use the metadata workbench to view the metadata and the data assets in the InfoSphere Information Server metadata repository. It provides end-to-end data flow reporting and impact analysis of the data assets that are used by InfoSphere Information Server product modules.

10.6.3 Fast Start Services Pack

Fast Start Services Pack (FSSP) is an optional add-on for a basic installation of both the hardware and software that is included in the 9700 and 9710 core offerings. There is also an option to add additional services to the FSSP to include the installation of the other add-on packs, Data Analytics, and Data Integration packs.

The Fast Start Services Pack is tailored to meet the unique needs of each customer. The customer selects from a set of predefined services for installation and configuration to speed implementation and minimize impact on site staff. Optimization services are also available for custom tuning to your data and workload requirements.

This pack is uniquely created for each customer, which allows services and training to augment customer skills that are available while allowing skills transfer to be performed.

10.6.4 DB2 Connect

DB2 Connect is the fourth component that is required when queries connect from a distributed source (Windows, UNIX, Linux, Linux for System z) to DB2 for z/OS, as with the tools that are available in the Data Analytics Pack and Data Integration Pack. DB2 Connect Application Server Edition allows unlimited users of an application that is deployed on the specific application server using direct connection or unlimited installations of DB2 Connect servers to access unlimited DB2 data servers on host systems.



High availability considerations

If the queries routed to the Accelerator cannot be run on the DB2 data server because of resource constraints on the z/OS platform, the availability of the Accelerator gets critical.

For an application to be truly available, all of the dependent components must be available. In addition to capitalizing on the availability of the System z platform, the IBM Analytics Accelerator provides many hardware level redundancies to improve the availability. The Netezza High Availability Cluster has two hosts (active and standby), which have file systems mirrored between them for failover purposes. The file system is only active on one host at any point in time, thereby providing 100% redundancy for the SMP host processors, and the stored data. In addition, each N2001 rack contains 34 hot spare drives and 240 active drives for a ratio of one spare per seven drives. Each N1001 rack contains four hot spare drives and 92 active drives for a ratio of one spare per 23 drives. The N2001 has 3.3x more spares per active drive than the N1001 rack.

In this chapter, the following topics are described to show some sample scenarios about how you can configure for high availability at the Accelerator level to improve the availability of accelerated queries:

- ▶ Loading data on multiple Accelerators for high availability
- ▶ High availability configuration with a hot standby Accelerator
- ▶ CDC considerations for high availability
- ▶ HPSS considerations for high availability

11.1 Loading data on multiple Accelerators for high availability

Starting from Version 3.1 of the IBM DB2 Analytics Accelerator for z/OS, a given table can be enabled for acceleration on multiple Accelerators concurrently. This feature forms the basis for the high availability configuration on the Accelerator side.

High availability of the IBM DB2 Analytics Accelerator can be achieved by pairing two Accelerators (in parallel) to the DB2 for z/OS environment.

There are many high availability configuration possibilities. There might be some special considerations that are unique to each business environment and your high availability configuration should be designed to accommodate your specific needs. The following are some possible scenarios that are applicable to both data sharing and non-data sharing environments and described in this chapter:

- ▶ High availability configuration without High Performance Storage Saver (HPSS). Refer to the descriptions in section 11.2, “High availability configuration with a hot standby Accelerator” on page 286
- ▶ High availability considerations for Change Data Capture (CDC). Refer to section 11.3, “CDC considerations for high availability” on page 292

11.1.1 Shared Accelerator considerations

During high availability design, ensure that the workload of one DB2 subsystem does not monopolize the resources of a shared Accelerator under any circumstances. See Chapter 4, “Accelerator resource management” on page 87.

For instance, a development subsystem that is attached to the same Accelerator as a production subsystem, should not be allowed to consume all Accelerator resources during normal operation or during a failover scenario.

11.1.2 Disaster recovery considerations

Necessary building blocks are available for designing a simple disaster recovery solution with the Accelerator in the mix. A simple disaster recovery scenario is described in section 11.2.2, “Option 2: Accelerators not in the same physical location” on page 289.

Note: As in the case of the primary Accelerator, the standby Accelerator would function without any need for Application code changes, that is, the failover would be transparent to your Applications. All of the subsequent query requests would flow as normal through DB2 for z/OS to the active Accelerator.

11.2 High availability configuration with a hot standby Accelerator

This section provides a simple high availability scenario. Whether or not this scenario is applicable to your environment is to be determined by carefully studying the underlying business need for high availability. This description would be applicable to a similar scenario with two stand-alone non-data sharing DB2 subsystems also.

Figure 11-1 provides a simple high availability schema in a 2-way DB2 data sharing environment using two Accelerators.

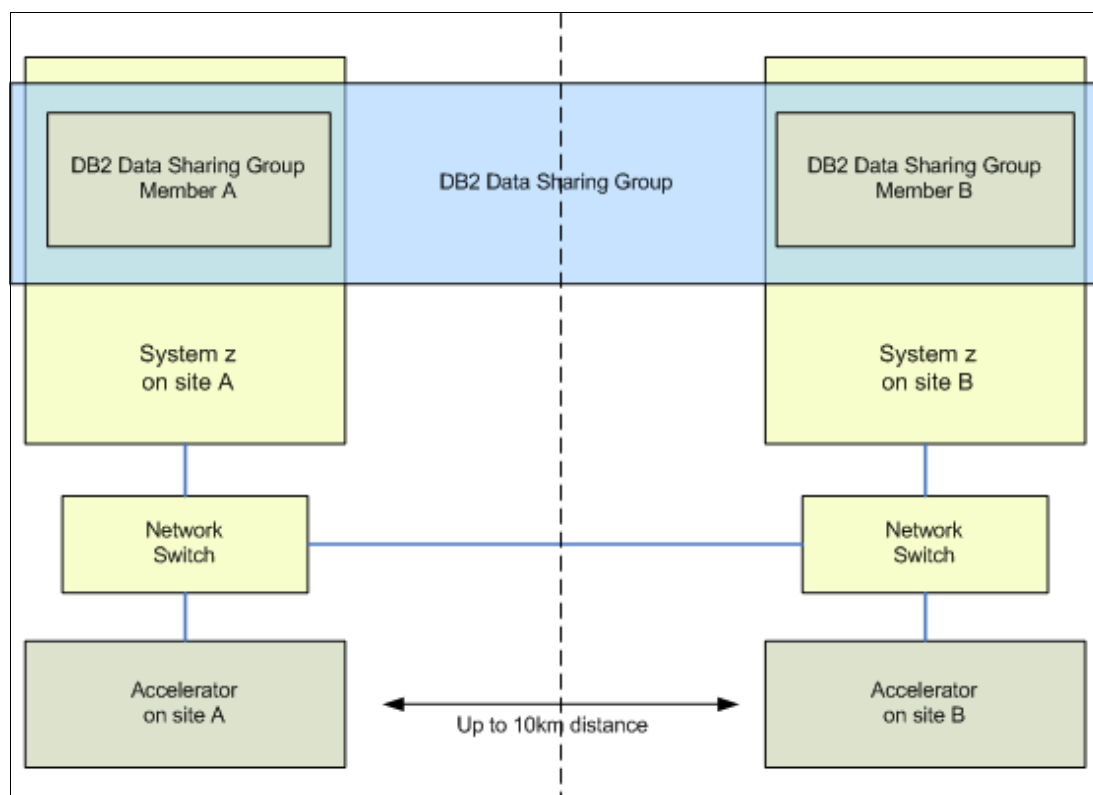


Figure 11-1 High availability configuration using two Accelerators in 2-way data sharing group

In this scenario, we ensure the following:

- ▶ Only one of the Accelerators is kept active at any given time. Only the primary Accelerator is started and ready to accelerate the queries. The backup Accelerator is maintained in a stopped state.
- ▶ The entire set of tables exist on both Accelerators.
- ▶ The entire set of tables are enabled on both the Accelerators.
- ▶ The entire set of tables have been refreshed with data from DB2 to a point that is acceptable to the business.
- ▶ Incremental updates might or might not exist on the tables under consideration.
- ▶ HPSS partitions must not exist.

In case an Accelerator fails, DB2 master address space receives a DSNX88* message. This message can be used to detect the failure and stop the primary Accelerator that failed, and immediately start the backup Accelerator. Then, the surviving Accelerator would start accepting the eligible routed queries for acceleration.

In most of the practical failure scenarios, the failed Accelerator would go to a stopped status automatically. Therefore, there is no need to stop it if DSNX88I messages are not found on the syslog. So, both the primary and the backup Accelerators may be kept active at all times.

Even though Figure 11-1 shows a limit of 10 km (6.2 mi) between the Accelerators, this is not a hard limit. The two Accelerators can be more than 10 km apart.

Note: If both of the Accelerators are active and ready to accelerate the incoming queries, there is no way to control which Accelerator would be picked by DB2 for routing the queries.

You have at least two options during physical implementation of this simple high availability configuration depending on where the primary and backup Accelerators are located. Both options are described here along with some disaster recovery considerations. Components that are required for a fully redundant (high availability) network between the IBM zEnterprise System and the Accelerator (PureData System for Analytics appliance) are documented in “Prerequisites and Maintenance for IBM DB2 Analytics Accelerator for z/OS Version 3.1”, at the following URL:

<http://www.ibm.com/support/docview.wss?uid=swg27035960>

11.2.1 Option 1: Both the Accelerators in the same physical location

This option is depicted in Figure 11-2.

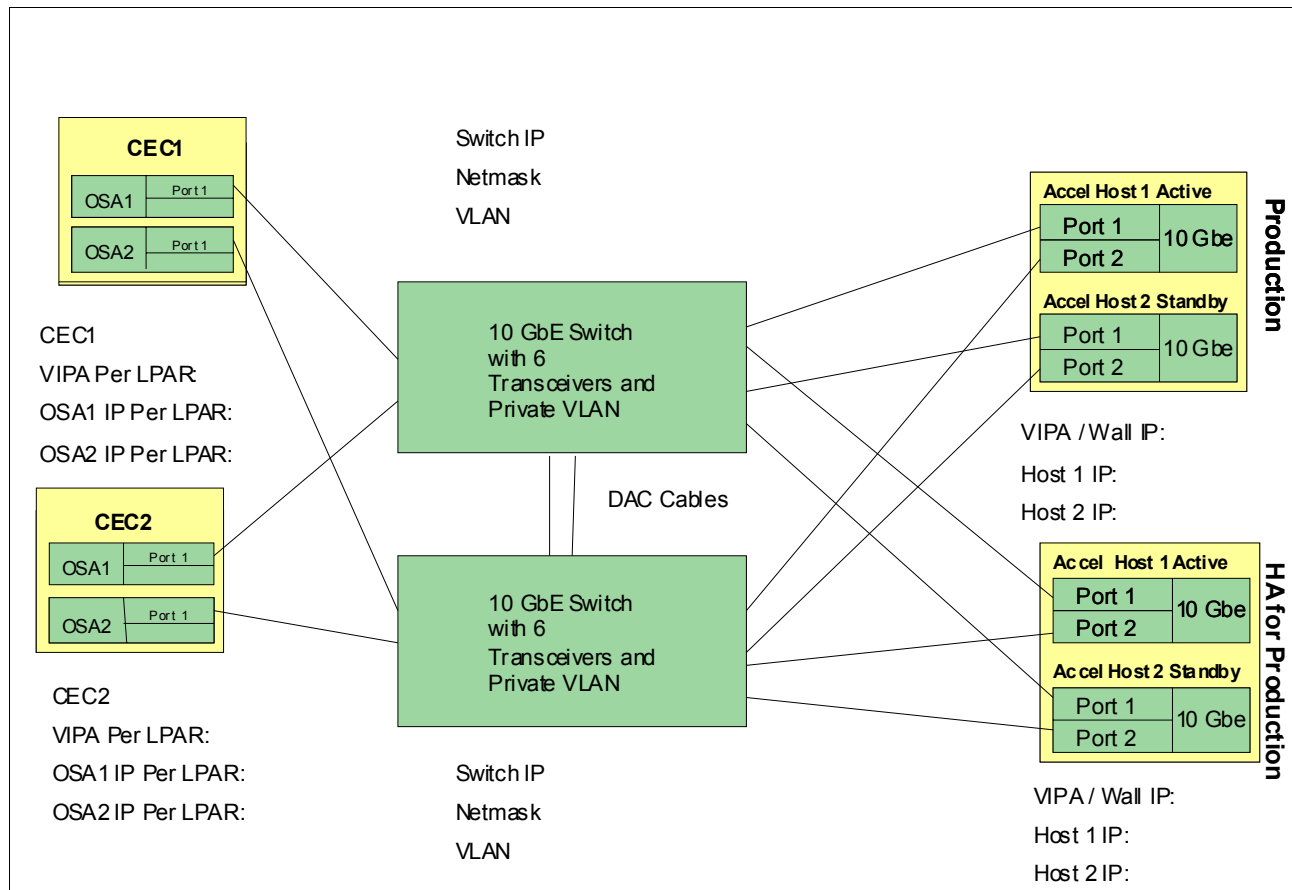


Figure 11-2 High availability configuration with both the Accelerators in the same location

In this option, because both Accelerators are in the same location, local high availability is possible. In a disaster, the disaster recovery is only possible through core DB2, that is, if you lose both Accelerators at the same time, the queries that are allowed to run in DB2 can run (but obviously it will not be as fast as the original routed query). Therefore, in this scenario, you should request a fast shipment of the replacement Accelerator immediately after the

disaster (if bringing your query response time to the same level as before the disaster is critical to your business operation).

The high availability disaster recovery configuration that is shown in Figure 11-2 on page 288 requires the following components:

- ▶ Two DB2 Analytics Accelerators, which also contain both hardware and software and include short range (SR) transceivers on the Accelerator servers
- ▶ Four 10 GbE SR OSA Express 4S cards (single port each). This means, two cards per System z server
- ▶ Four 10 GbE Switches
- ▶ Appropriate number of SR transceivers at the switches
- ▶ Network infrastructure between the two sites if the remote site option is selected and all the required fiber optic cables
- ▶ Long-distance connectivity

11.2.2 Option 2: Accelerators not in the same physical location

This option is depicted in Figure 11-3.

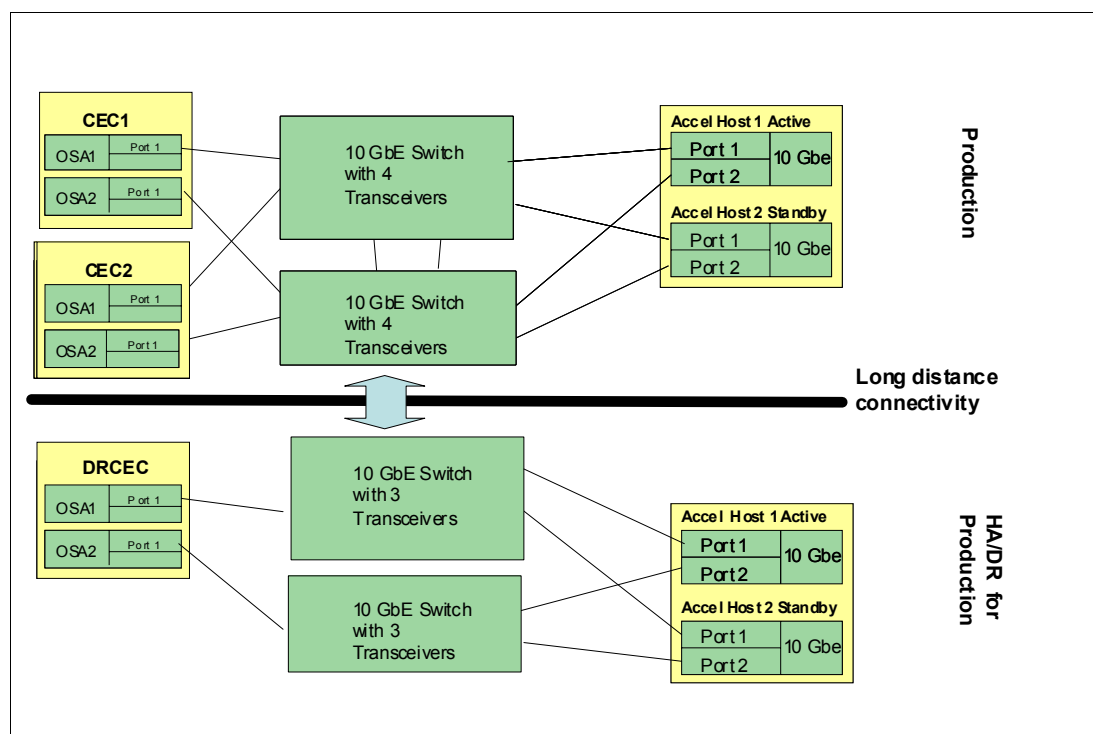


Figure 11-3 High availability/disaster recovery configuration with the Accelerators in two different locations

In this option, the Accelerators are not in the same physical location, so the standby Accelerator serves as a remote high availability and disaster recovery backup.

The high availability disaster recovery configuration that is shown in Figure 11-3 requires the following components:

- ▶ Two DB2 Analytics Accelerators, which also contain both hardware and software and include SR transceivers on the Accelerator servers

- ▶ Six 10 GbE SR OSA Express 4S cards (single port each), that is, two cards per System z server
- ▶ Four 10 GbE switches
- ▶ Appropriate number of SR transceivers at the switches
- ▶ Network infrastructure between the two sites if the remote site option is selected
- ▶ All required fiber optic cables

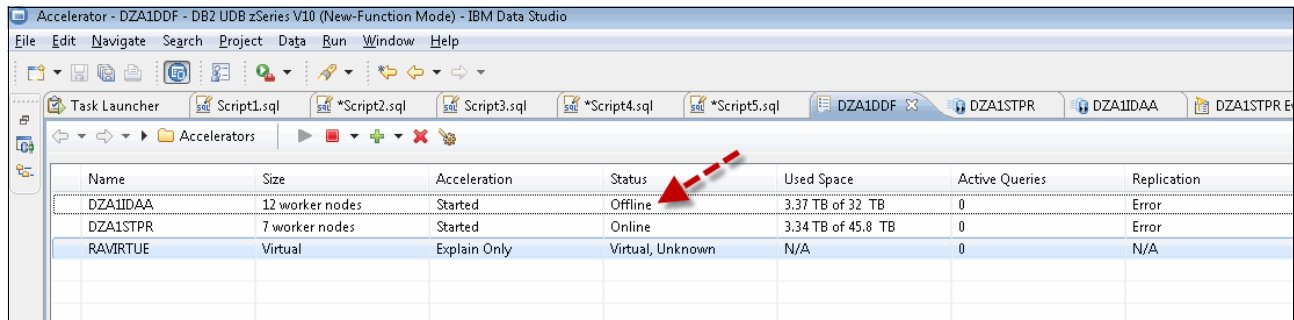
Additional considerations:

- ▶ The Accelerator network must be private
- ▶ Defining IP address for a class C network is sufficient

11.2.3 Observations from a simulated failover

In a failover, the only operation that has to be performed is to disable the acceleration for the failing site (primary Accelerator) and to enable the acceleration on the backup site (backup Accelerator). This simulation was performed by abruptly stopping the Accelerator without the knowledge of DB2 for z/OS server.

Figure 11-4 illustrates this scenario when the primary Accelerator named DZA1IDAA is offline (due to an outage), as seen under the *Status* column. As soon as such a failure is detected, the backup Accelerator should be started. After a successful start of the backup Accelerator, the status can be seen from the *Acceleration* column of the backup Accelerator named DZA1STPR, which has a value of “Started”.



Name	Size	Acceleration	Status	Used Space	Active Queries	Replication
DZA1IDAA	12 worker nodes	Started	Offline	3.37 TB of 32 TB	0	Error
DZA1STPR	7 worker nodes	Started	Online	3.34 TB of 45.8 TB	0	Error
RAVIRTUE	Virtual	Explain Only	Virtual, Unknown	N/A	0	N/A

Figure 11-4 Failover scenario with primary Accelerator offline and backup Accelerator started

Immediately after starting the backup Accelerator, the primary Accelerator must be stopped (if the status is not automatically changed to *stopped*); otherwise, the queries would still be routed to the primary Accelerator, and fail. During this brief interval during which the failed Accelerator is still in *started* condition, if you issue a **DISPLAY ACCEL(*) DETAIL** command for the failed Accelerator, you might see the output similar to the one shown in Figure 11-5 on page 291.

```

DSNX810I  -DZA1 DSNX8CMD DISPLAY ACCEL FOLLOWS -
DSNX830I  -DZA1 DSNX8CDA
ACCELERATOR                                MEMB  STATUS  REQUESTS  ACTV  QUED  MAXQ
-----
DZA1IDAA                                DZA1  STARTED              0    0    0    0
LOCATION=DZA1IDAA HEALTHY
DETAIL STATISTICS
  LEVEL   = AQT03012
  STATUS  = STOPPED
  FAILED QUERY REQUESTS                      =      14
  AVERAGE QUEUE WAIT                        =      0 MS
  MAXIMUM QUEUE WAIT                        =      0 MS
  TOTAL NUMBER OF PROCESSORS                 =      0
  AVERAGE CPU UTILIZATION ON COORDINATOR NODES =    .00%
  AVERAGE CPU UTILIZATION ON WORKER NODES    =    .00%
  NUMBER OF ACTIVE WORKER NODES              =      0
  TOTAL DISK STORAGE AVAILABLE                =      0 MB
  TOTAL DISK STORAGE IN USE                  =    .00%
  DISK STORAGE IN USE FOR DATABASE            = 3534847 MB

```

Figure 11-5 -DIS ACCEL DET output with Accelerator offline, but connection is healthy

During such an error or failure scenario, the **DISPLAY ACCEL DETAIL** output might look inconsistent and the following explanation of the key fields in Figure 11-5 might help:

- ▶ “STARTED” in the first line indicates that the **-STA ACCEL DB2** command has been issued and the Accelerator was started successfully
- ▶ “STATUS = STOPPED”
- ▶ “HEALTHY” is an internal state (other possible states: BUSY, FLATLINE, AUTHFAIL, DDDFAIL) that indicates that the connection between DB2 and the Accelerator is still alive and healthy

Example 11-1 is the message taken from the MSTR address space of the DB2 subsystem that is affected by the failed Accelerator during the simulation in a lab setting.

Since this was done by stopping only the Netezza backend via “nzstop” while keeping the Accelerator server process running, DB2 still has the Accelerator server process to talk to (hence, the internal state of HEALTHY). Because the Accelerator is still reporting the state of the Netezza backend to **DIS ACCEL**, the “STATUS = STOPPED” is observed. Even though this is not a valid disaster or failure scenario, it could happen. But, it will not cause a failover to happen automatically, that is, DB2 will not go from STARTED to STOPPED by itself.

Example 11-1 Sample DSNX881I message from the MSTR address space during failure

```

STC05580 DSNX881I  -DZA1 1 I 16695 (09-May-13, 16:46:49 EDT) 517
517          NPS system isas-h2 went from online to offliningNow at 09-May
517          16:46:49 EDT User initiated. Event: eventType: sysStateChange
517          eventTimestamp: 09-May-13, 16:46:49 EDT eventArgs:
517          previousState=online, currentState=offline

```

Note: If the Accelerator server process is interrupted (for instance, by logging into the Accelerator box and running “dwa-admin stop”), that would terminate the Accelerator process and simulate a cut network connection. But, if the Accelerator server process is interrupted (for instance, during a real disaster scenario), the DSNX881I messages would no longer appear in the syslog, as shown in Example 11-1 on page 291. Also, in such a failure scenario, if the backup Accelerator is kept “Active”, the queries are transparently routed to the surviving backup Accelerator (because the primary Accelerator state will automatically switch to “Stopped”).

11.3 CDC considerations for high availability

It is possible to apply the incremental updates from a DB2 source table to multiple Accelerator targets simultaneously. The details about how to set up two targets (apply processes) with the same source DB2 tables (capture process) are described in Chapter 7, “Incremental update” on page 161.

Section 11.3.1, “High availability setup for the incremental update capture engine” on page 292, provides a sample high-availability setup for the replication agent of IBM InfoSphere Change Data Capture (CDC) for z/OS, a component that is required for the incremental update function of IBM DB2 Analytics Accelerator for z/OS. For details, see the IBM techdoc: *High Availability Setup for the Incremental Update Capture Engine*, available at the following site:

<http://www.ibm.com/support/docview.wss?uid=swg27037912>

11.3.1 High availability setup for the incremental update capture engine

The setup that is described here uses a DB2 data sharing group with multiple members, each of which is running in a different logical partition (LPAR), and separate instances of the CDC replication agent for each LPAR. Only a single instance of the replication agent can be active at any time. The other instance will be in Hot Standby mode, which means that the instance has a connection to DB2, but remains in a waiting state and will not propagate data changes or accept agent log-on requests before the instance is granted exclusive access to the metadata tables.

When an instance enters the active mode, it initializes its own TCP/IP task, which in turn starts a BIND operation that associates a TCP port with a TCP/IP stack. For the z/OS Communication Server, this BIND operation is a request to pass control to the TCP/IP stack of the LPAR whose IP address is specified in the **BIND** command. The entire operation is carried out in a dynamic virtual IP address (DVIPA) environment.

If the CDC instance that is active becomes unavailable, the TCP/IP address of this instance is passed to one of the Hot Standby instances. As a result, one of the Hot Standby instances goes into active mode.

Figure 11-6 on page 293 illustrates this setup with a DB2 data sharing group that consists of two members in different LPARs. The data sharing group is called *DWCDBZA*; its members are called *SZA1* and *SZA2*. The member *SZA1* runs in an LPAR called *DWC1*; whereas, the other member, *SZA2*, runs in an LPAR called *DWC2*. Both LPARs belong to the same sysplex. Separate instances of the CDC replication agent are running in each LPAR. The TCP/IP stacks of both LPARs are named *TCPIP*. The connected Accelerator hardware, an IBM PureData System for Analytics N1001 with the name *dwa0112*, communicates with the

active instance over DVIPA address 10.101.8.249. A similar setup would work on an IBM PureData System for Analytics N2001 system, too.

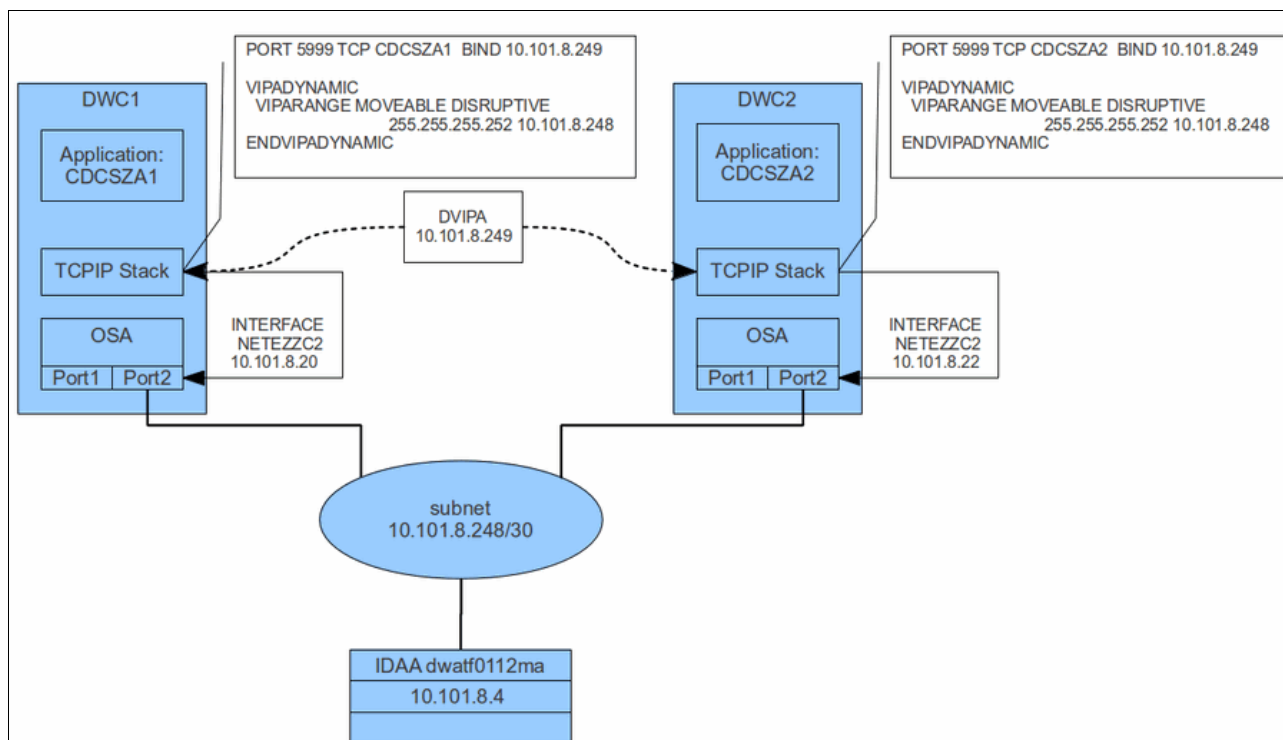


Figure 11-6 High availability setup for the incremental update capture engine

InfoSphere CDC for z/OS setup

Follow the instructions in Chapter 7, “Incremental update” on page 161 to install and run the InfoSphere CDC instance on each LPAR of your DB2 data sharing group. Ensure that the latest maintenance has been applied.

Ensure that the VSAM Metadata cluster (the name is defined by replacing the <MetaDatacluster> placeholder in the CHCDFMTD installation job) has the value (3 3) for the **SHAREOPTIONS** keyword. The VSAM Metadata cluster is referenced by the CHCMTDTA DD name in the product startup job control language (JCL) and must be shared across the active and all corresponding hot-standby instances. All instances using the same metadata tables (identified by the owner) within a DB2 subsystem or data sharing group must reference the same VSAM Metadata cluster.

You can use the ARMWRAP program to ensure that the address space is restarted in case it fails unexpectedly. For more details, see *IBM z/OS Automatic Restart Manager*, REDP-1073.

The IP configuration for the InfoSphere CDC for z/OS instances is defined in the CHCCMMxx member (xx is the suffix that you chose during the installation). In this example, the instances use TCP port 5999, which is defined by the **SERVICENAME** keyword, and a TCP/IP stack name **TCPIP**, which is defined by the **SUBSYSTEM** keyword:

```
TCP/IP SERVICENAME=5999,
      SUBSYSTEM=TCPIP
```

To verify this step, start InfoSphere CDC and review the CHCPRINT data set in the SPOOLED output. The output must contain the following information:

```
CHC9512I Using TCP/IP Subsystem address space TCPIP
```

```
CHC9523I TCP/IP Communication Support is active
CHC5099I CMO Task was initialized successfully
CHC6804I TCP Listener was initialized on service name 5999, port 5999
```

z/OS Communications Server setup

To reserve the port 5999 for the InfoSphere CDC instances and to associate the port number with the dynamic VIPA address 10.101.8.249, use the following statement in the PROFILE TCPIP data set for the TCP/IP stacks of both LPARs. Use different names for the CDC address space in each stack (CDCSZA1 is the name of first address space in this example. CDCSZA2 is the name of the second address space. You would have to enter CDCSZA2 when defining the second stack):

```
PORT
  5999 TCP CDCSZA1 BIND 10.101.8.249; CDC AGENT BIND TO DVIPA IP ADDRESS
```

The **VIPADYNAMIC** keyword sets the operation mode to *dynamic*. Because you need only one DVIPA address (10.101.8.249), you can use the smallest possible subnet (mask 255.255.255.252). This subnet gives you two possible client addresses: 10.101.8.249 and 10.101.8.250. Only the first one is used:

```
VIPADYNAMIC      ; VIPA DEFINITIONS
VIPARANGE MOVEABLE DISRUPTIVE 255.255.255.252 10.101.8.248
ENDVIPADYNAMIC
```

Verify the DVIPA configuration by entering the following command:

DISPLAY TCPIP,TCPIP,NETSTAT,VIPADCFG

The output of this command is:

```
DWC1      12342 14:38:32.08 STC01367 00000010 EZD0101I NETSTAT CS V1R12 TCPIP 234
                                                DYNAMIC VIPA INFORMATION:
                                                VIPA RANGE:
                                                IPADDR/PREFIXLEN: 10.101.8.248/30
                                                MOVEABLE: DISRUPT
                                                END OF THE REPORT
```

Verify the PORT configuration by entering the following command:

DISPLAY TCPIP,TCPIP,NETSTAT,PORTL

The output of this command is:

```
DWC1      12342 14:39:18.02 STC01367 00000010 EZD0101I NETSTAT CS V1R12 TCPIP 23
                                                PORT# PROT USER FLAGS RANGE SAF NAME
                                                5999 TCP CDCSZA1 DABU
                                                BINDSPECIFIC: 10.101.8.249
                                                ...
```

You can display the current DVIPA status of a system by using the following command:

DISPLAY TCPIP,TCPIP,NETSTAT,VIPADYN

This output of this command looks like this:

```
DWC1      12342 14:41:04.12 STC01367 00000010 EZD0101I NETSTAT CS V1R12 TCPIP 252
                                                DYNAMIC VIPA:
                                                IPADDR/PREFIXLEN: 10.101.8.249/30
```

```

STATUS: ACTIVE ORIGIN: VIPARANGE
BIND DISTSTAT:
ACTTIME: 12/07/2012 13:26:33
JOBNAME: CDCSZA1
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT

```

This example shows that CDC agent *CDCSZA1* is running and uses the DVIPA address 10.101.8.249.

Verify the current DVIPA status of a sysplex by entering the following command:

DISPLAY TCPIP,TCPIP,SYSPLEX,VIPADYN

This output of this command is:

```

DWC1 12342 14:43:52.32 STC01367 00000010 EZZ8260I SYSPLEX CS V1R12 105
                                           VIPA DYNAMIC DISPLAY FROM TCPIP AT
DWC1
                                           LINKNAME: VIPLOA6508F9
                                           IPADDR/PREFIXLEN: 10.101.8.249/30
                                           ORIGIN: VIPARANGE BIND
                                           TCPNAME MVSNAME STATUS RANK DIST
                                           -----
                                           TCPIP   DWC1   ACTIVE
                                           1 OF 1 RECORDS DISPLAYED

```

The example shows that the DVIPA address 10.101.8.249 is used by an application in LPAR DWC1.

Testing the failover

When you start the first instance of the InfoSphere CDC replication agent, which will become the active instance, verify that the DVIPA address is created correctly. This is indicated by the following message, which is displayed during the start procedure:

```
EZD1205I DYNAMIC VIPA 10.101.8.249 WAS CREATED USING BIND BY CDCSZA1 ON TCPIP
```

All subsequently started instances will enter the Hot Standby mode. This is indicated by the following message, which is shown at the start of each additional instance:

```

CHC9212I The meta data owned by <owner> is being used by
          another instance of InfoSphere CDC for z/OS, this
          instance is entering Hot Standby mode

```

If the active instance ends abnormally or the system on which it is running becomes unavailable, one of the Hot Standby instances takes over. This is indicated by the following message:

```

CHC9211I Termination of Active instance of InfoSphere CDC for z/OS using meta data
owned by <owner> detected, this instance is leaving Hot Standby mode and will
become Active

```

To initiate a failover manually, you can use the **HANDOVER** parameter of the **SHUTDOWN** command. The syntax of the command is as follows:

MODIFY <A/SName>, SHUTDOWN, [SCHED=<DateTime>|NORM|IMMED|ABORT], [HANDOVER]

Where <A/SName> is the name of the instance of the InfoSphere CDC replication agent.

If you shut down the active instance with the **HANDOVER** parameter, one of the Hot Standby instances reacts as though the active instance had been abnormally terminated, and becomes the new active instance. If you use this command for the active instance without the **HANDOVER** parameter, or issue a z/OS **STOP** command for this instance, the active instance and all Hot Standby instances are shut down. If you use this command for a Hot Standby instance or issue a z/OS **STOP** command for such an instance, only the specified Hot Standby instance is shut down.

The following example shows the output of a **SHUTDOWN** command invocation with the **IMMED** and **HANDOVER** parameters for the CDCSZA1 instance.

Command:

MODIFY CDCSZA1,SHUTDOWN,IMMED,HANDOVER

The message output is:

```
DWC1      12342 15:01:37.08 STC01367 00000010  EZD1298I DYNAMIC VIPA 10.101.8.249
DELETED FROM TCPIP
DWC1      12342 15:01:37.08 STC01367 00000010  EZD1207I DYNAMIC VIPA 10.101.8.249
WAS DELETED USING CLOSE API BY
                                CDCSZA1 ON TCPIP

DWC2      12342 15:01:37.14 STC01855 00000010  +CHC9211I (CDCSZA2) Termination of
Active instance of IBM InfoSphere
                                CDC for z/OS using meta data owned by
CD ...
DWC2      12342 15:01:37.14 STC01855 00000010  +CHC9211I (CDCSZA2) ... CUSER
detected, this instance is leaving Hot
                                Standby mode and will become Active
DWC2      12342 15:01:37.49 STC01459 00000010  EZD1205I DYNAMIC VIPA 10.101.8.249
WAS CREATED USING BIND BY CDCSZA2 ON
                                TCPIP
```

11.3.2 Accelerator coordinator node failover considerations

The InfoSphere CDC components are integrated into the Accelerator fail-over configuration. If an Accelerator SMP host fails, the secondary host would take over and restart all replication-related components on the apply side automatically. See Example 11-2 for the system messages issued.

Note: If the apply components are started on all the Accelerators that are involved in the high availability configuration, an SMP host failure will not have a negative impact.

Example 11-2 System messages for a failover from Accelerator host 1 to host 2

```
SDSF SYSLOG
Command ==>
Date Time      Message ID
*****
17OCT 00:05:59 DSNX809I  DSNX809I  -DA35 DSNX8TER ACCELERATOR PROCESSING STOP COMPLETE
17OCT 00:09:04 DSNX922I  DSNX922I  -DA12 DSN3AMT3 BEGINNING DISCONNECTION OF
17OCT 00:09:04 DSNX922I  DSNX922I  STORED PROCEDURE ADDRESS SPACES FROM DB2
17OCT 00:09:14 DSNX923I  DSNX923I  -DA12 DSN3AMT3 ALL STORED PROCEDURE ADDRESS
17OCT 00:09:14 DSNX923I  DSNX923I  SPACES ARE NOW DISCONNECTED FROM DB2
17OCT 00:09:28 DSNX809I  DSNX809I  -DA12 DSNX8TER ACCELERATOR PROCESSING STOP COMPLETE
```



```

170CT 00:31:50 DSNX801I DSNX801I -DA12 DSNX8INI ACCELERATOR PROCESSING STARTING
170CT 00:33:19 DSNX801I DSNX801I -DA35 DSNX8INI ACCELERATOR PROCESSING STARTING
170CT 00:33:22 DSNX891I DSNX891I -DA35 DSNX8CTG DSNACCEL OBJECT DSNACCEL DOES NOT EXIST
170CT 00:33:22 DSNX895I DSNX895I -DA35 DSNX8IN2 DSNACCEL IS UNAVAILABLE
170CT 00:33:22 DSNX800I DSNX800I -DA35 DSNX8IN2 ACCELERATOR FUNCTION IS NOT AVAILABLE
170CT 10:51:47 DSNX881I DSNX881I -DA12 1 I 1 (17-Oct-12, 10:41:59 EDT) NPS
170CT 10:51:47 DSNX881I system netezza1 went from online to offliningNow at 17-Oct-12,
170CT 10:51:47 DSNX881I 10:41:59 EDT User initiated. Event: eventType: sysStateChanged
170CT 10:51:47 DSNX881I eventTimestamp: 17-Oct-12, 10:41:59 EDT eventArgs:
170CT 10:51:47 DSNX881I previousState=online, currentState=offlin
170CT 10:51:47 DSNX881I DSNX881I -DA12 1 I 2 (17-Oct-12, 10:42:01 EDT) NPS
170CT 10:51:47 DSNX881I system netezza1 went from offliningNow to offlineNow at
17-Oct-12,
170CT 10:51:47 DSNX881I 10:42:01 EDT User initiated. Event: eventType: sysStateChanged
170CT 10:51:47 DSNX881I eventTimestamp: 17-Oct-12, 10:42:01 EDT eventArgs:
170CT 10:51:47 DSNX881I previousState=offliningNow, currentSt
170CT 10:51:47 DSNX881I DSNX881I -DA12 1 I 3 (17-Oct-12, 10:50:35 EDT) NPS
170CT 10:51:47 DSNX881I system netezza2 went from discovering to initializing at
17-Oct-12,
170CT 10:51:47 DSNX881I 10:50:35 EDT User initiated. Event: eventType: sysStateChanged
170CT 10:51:47 DSNX881I eventTimestamp: 17-Oct-12, 10:50:35 EDT eventArgs:
170CT 10:51:47 DSNX881I previousState=discovering, currentSt
170CT 10:51:47 DSNX881I DSNX881I -DA12 1 I 4 (17-Oct-12, 10:51:02 EDT) NPS
170CT 10:51:47 DSNX881I system netezza2 went from initializing to initialized at
17-Oct-12,
170CT 10:51:47 DSNX881I 10:51:02 EDT User initiated. Event: eventType: sysStateChanged
170CT 10:51:47 DSNX881I eventTimestamp: 17-Oct-12, 10:51:02 EDT eventArgs:
170CT 10:51:47 DSNX881I previousState=initializing, currentS
170CT 10:51:47 DSNX881I DSNX881I -DA12 1 I 5 (17-Oct-12, 10:51:04 EDT) NPS
170CT 10:51:47 DSNX881I system netezza2 went from initialized to preOnlining at
17-Oct-12,
170CT 10:51:47 DSNX881I 10:51:04 EDT User initiated. Event: eventType: sysStateChanged
170CT 10:51:47 DSNX881I eventTimestamp: 17-Oct-12, 10:51:04 EDT eventArgs:
170CT 10:51:47 DSNX881I previousState=initialized, currentSta
170CT 10:51:47 DSNX881I DSNX881I -DA12 1 I 6 (17-Oct-12, 10:51:08 EDT) NPS
170CT 10:51:47 DSNX881I system netezza2 went from preOnlining to resuming at 17-Oct-12,
170CT 10:51:47 DSNX881I 10:51:08 EDT User initiated. Event: eventType: sysStateChanged
170CT 10:51:47 DSNX881I eventTimestamp: 17-Oct-12, 10:51:08 EDT eventArgs:
170CT 10:51:47 DSNX881I previousState=preOnlining, currentState=
170CT 10:51:47 DSNX881I DSNX881I -DA12 1 I 7 (17-Oct-12, 10:51:10 EDT) NPS
170CT 10:51:47 DSNX881I system netezza2 went from resuming to online at 17-Oct-12,
10:51:10
170CT 10:51:47 DSNX881I EDT User initiated. Event: eventType: sysStateChanged
eventTimestamp:
170CT 10:51:47 DSNX881I 17-Oct-12, 10:51:10 EDT eventArgs: previousState=resuming,
170CT 10:51:47 DSNX881I currentState=online,
***** ***** ***** ***** ***** BOTTOM OF MESSAGES *****

```



A

Accelerated queries

This appendix provides details about the queries that are used in Chapter 4, “Accelerator resource management” on page 87, and Chapter 8, “Impact of new Netezza hardware and software components” on page 187.

The following topics are covered:

- ▶ Query CPU bound
- ▶ Selective query

The queries for 8.5.3, “Scenario #3: Mixed: I/O and CPU bound query” on page 202, are the same as the queries in 8.5.1, “Scenario #1: I/O bound query” on page 199.

A.1 Query CPU bound

The query used in section 4.3.2, “Test scenario A: Prioritizing one user (query) higher than the others” on page 100, where it is called query Q3A; and the query used in section 8.5.2, “Scenario #2: CPU bound query” on page 201 are listed in Example A-1.

Example A-1 CPU bound query

```
WITH
ORDER_METHOD_DIMENSION14
AS(
SELECT
ORDER_METHOD_DIMENSION.
ORDER_METHOD_KEY ORDER_METHOD_KEY,
MIN(ORDER_METHOD_DIMENSION.ORDER_METHOD_EN)ORDER_METHOD
FROM GOSLDW_ORDER_METHOD_DIMENSION ORDER_METHOD_DIMENSION
GROUP BY ORDER_METHOD_DIMENSION.ORDER_METHOD_KEY),
PRODUCT_LINE15
AS(
SELECT PRODUCT_LINE.PRODUCT_LINE_CODE PRODUCT_LINE_CODE,
MIN(PRODUCT_LINE.PRODUCT_LINE_EN)PRODUCT_LINE
FROM GOSLDW_PRODUCT_LINE PRODUCT_LINE
GROUP BY PRODUCT_LINE.PRODUCT_LINE_CODE),
Sales_territory_dimension12
AS(
SELECT
SALES_TERRITORY_DIMENSION.
COUNTRY_KEY COUNTRY_KEY,
SALES_TERRITORY_DIMENSION.
COUNTRY_CODE COUNTRY_CODE,
SALES_TERRITORY_DIMENSION.
SALES_TERRITORY_KEY SALES_TERRITORY_KEY,
SALES_TERRITORY_DIMENSION.
SALES_TERRITORY_CODE SALES_TERRITORY_CODE,
Sales_territory_dimension.
COUNTRY_EN COUNTRY_EN,
Sales_territory_dimension.
FLAG_IMAGE FLAG_IMAGE31,
SALES_TERRITORY_DIMENSION.
SALES_TERRITORY_EN SALES_TERRITORY_EN
FROM
GOSLDW_SALES_TERRITORY_DIMENSION SALES_TERRITORY_DIMENSION),
GENDER_LOOKUP13 AS(
SELECT
GENDER_LOOKUP.GENDER_CODE GENDER_CODE,
MIN(GENDER_LOOKUP.GENDER)GENDER
FROM GOSLDW_GENDER_LOOKUP GENDER_LOOKUP
WHERE GENDER_LOOKUP.LANGUAGE='EN'
GROUP BY GENDER_LOOKUP.GENDER_CODE),
Retailer__model_
AS(
SELECT
RETAILER_DIMENSION11.
RETAILER_SITE_KEY RETAILER_SITE_KEY,
SALES_TERRITORY_DIMENSION12.
```

```

SALES_TERRITORY_KEY SALES_TERRITORY_KEY,
SALES_TERRITORY_DIMENSION12.
SALES_TERRITORY_EN SALES_TERRITORY
FROM
GOSLDW_RETAILER_DIMENSION RETAILER_DIMENSION11,
SALES_TERRITORY_DIMENSION12,
GENDER_LOOKUP13
WHERE RETAILER_DIMENSION11.GENDER_CODE =
GENDER_LOOKUP13.GENDER_CODE
AND RETAILER_DIMENSION11.COUNTRY_KEY =
SALES_TERRITORY_DIMENSION12.COUNTRY_KEY)
SELECT
ORDER_METHOD_DIMENSION14.
ORDER_METHOD_KEY ORDER_METHODOKEY,
ORDER_METHOD_DIMENSION14.
ORDER_METHOD ORDER_METHOD1,
PRODUCT_LINE15.PRODUCT_LINE_CODE PRODUCT_LINEKEY,
PRODUCT_LINE15.PRODUCT_LINE PRODUCT_LINEO,
RETAILER__MODEL_.SALES_TERRITORY_KEY RETAILER_TERRITORYKEY,
RETAILER__MODEL_.SALES_TERRITORY SALES_TERRITORY,
CAST(TIME_DIMENSION17.CURRENT_YEAR AS CHAR(4))YEARKEY,
CAST(TIME_DIMENSION17.QUARTER_KEY AS CHAR(6))QUARTERKEY,
CAST(TIME_DIMENSION17.MONTH_KEY AS CHAR(6))MONTHKEY,
SUM(SALES_FACT18.GROSS_PROFIT)GROSS_PROFIT
FROM
ORDER_METHOD_DIMENSION14,
PRODUCT_LINE15,RETAILER__MODEL_,
GOSLDW_TIME_DIMENSION TIME_DIMENSION17,
GOSLDW_SALES_FACT SALES_FACT18,
GOSLDW_PRODUCT_TYPE PRODUCT_TYPE19,
GOSLDW_PRODUCT_DIMENSION PRODUCT_DIMENSION20
WHERE ORDER_METHOD_DIMENSION14.ORDER_METHOD_KEY =
SALES_FACT18.ORDER_METHOD_KEY
AND PRODUCT_DIMENSION20.PRODUCT_KEY =
SALES_FACT18.PRODUCT_KEY
AND PRODUCT_TYPE19.PRODUCT_TYPE_CODE =
PRODUCT_DIMENSION20.PRODUCT_TYPE_CODE
AND PRODUCT_LINE15.PRODUCT_LINE_CODE =
PRODUCT_TYPE19.PRODUCT_LINE_CODE
AND TIME_DIMENSION17.DAY_KEY =
SALES_FACT18.ORDER_DAY_KEY
AND RETAILER__MODEL_.RETAILER_SITE_KEY =
SALES_FACT18.RETAILER_SITE_KEY
GROUP BY
ORDER_METHOD_DIMENSION14.ORDER_METHOD_KEY,
ORDER_METHOD_DIMENSION14.ORDER_METHOD,
PRODUCT_LINE15.PRODUCT_LINE_CODE,
PRODUCT_LINE15.PRODUCT_LINE,
RETAILER__MODEL_.SALES_TERRITORY_KEY,
RETAILER__MODEL_.SALES_TERRITORY,
CAST(TIME_DIMENSION17.CURRENT_YEAR AS CHAR(4)),
CAST(TIME_DIMENSION17.QUARTER_KEY AS CHAR(6)),
CAST(TIME_DIMENSION17.MONTH_KEY AS CHAR(6));

```

A.2 Selective query

The query used in section 8.5.4, “Scenario #4: Selective query with equal predicates benefitting from zone maps” on page 204 is listed in Example A-2.

Example A-2 Selective query with equal predicates

```
WITH
PRODUCT_LINE13
AS
  (SELECT
    PRODUCT_LINE.PRODUCT_LINE_CODE PRODUCT_LINE_CODE,
    MIN(PRODUCT_LINE.PRODUCT_LINE_EN)PRODUCT_LINE
  FROM
    GOSLDW_PRODUCT_LINE PRODUCT_LINE
  GROUP BY
    PRODUCT_LINE.PRODUCT_LINE_CODE),
SALES_TERRITORY_DIMENSION11
AS
  (SELECT
    SALES_TERRITORY_DIMENSION.COUNTRY_KEY COUNTRY_KEY,
    SALES_TERRITORY_DIMENSION.COUNTRY_CODE COUNTRY_CODE,
    SALES_TERRITORY_DIMENSION.
      SALES_TERRITORY_KEY SALES_TERRITORY_KEY,
    SALES_TERRITORY_DIMENSION.
      SALES_TERRITORY_CODE SALES_TERRITORY_CODE,
    SALES_TERRITORY_DIMENSION.COUNTRY_EN COUNTRY_EN,
    SALES_TERRITORY_DIMENSION.FLAG_IMAGE FLAG_IMAGE31,
    SALES_TERRITORY_DIMENSION.
      SALES_TERRITORY_EN SALES_TERRITORY_EN
  FROM
    GOSLDW_SALES_TERRITORY_DIMENSION SALES_TERRITORY_DIMENSION),
GENDER_LOOKUP12
AS
  (SELECT
    GENDER_LOOKUP.GENDER_CODE GENDER_CODE,
    MIN(GENDER_LOOKUP.GENDER)GENDER
  FROM
    GOSLDW_GENDER_LOOKUP GENDER_LOOKUP
  WHERE
    GENDER_LOOKUP.LANGUAGE='EN'
  GROUP BY
    GENDER_LOOKUP.GENDER_CODE),
RETAILER__MODEL_
AS
  (SELECT
    RETAILER_DIMENSION10.RETAILER_SITE_KEY RETAILER_SITE_KEY,
    RETAILER_DIMENSION10.RETAILER_NAME RETAILER_NAME,
    RETAILER_DIMENSION10.CITY CITY,
    SALES_TERRITORY_DIMENSION11.COUNTRY_KEY COUNTRY_KEY,
    SALES_TERRITORY_DIMENSION11.
      SALES_TERRITORY_KEY SALES_TERRITORY_KEY,
    SALES_TERRITORY_DIMENSION11.COUNTRY_EN COUNTRY,
    SALES_TERRITORY_DIMENSION11.
      SALES_TERRITORY_EN SALES_TERRITORY,
```

```

        RETAILER_DIMENSION10.RETAILER_KEY RETAILER_KEY
FROM
    GOSLDW_RETAILER_DIMENSION RETAILER_DIMENSION10,
    SALES_TERRITORY_DIMENSION11,
    GENDER_LOOKUP12
WHERE
    RETAILER_DIMENSION10.GENDER_CODE=
        GENDER_LOOKUP12.GENDER_CODE
    AND
    RETAILER_DIMENSION10.COUNTRY_KEY=
        SALES_TERRITORY_DIMENSION11.COUNTRY_KEY)
SELECT
    PRODUCT_LINE13.PRODUCT_LINE_CODE PRODUCT_LINEKEY,
    PRODUCT_LINE13.PRODUCT_LINE PRODUCT_LINE0,
    RETAILER__MODEL_.SALES_TERRITORY_KEY RETAILER_TERRITORYKEY,
    RETAILER__MODEL_.SALES_TERRITORY SALES_TERRITORY,
    RETAILER__MODEL_.COUNTRY_KEY RETAILER_COUNTRYKEY,
    RETAILER__MODEL_.COUNTRY COUNTRY,
    RETAILER__MODEL_.RETAILER_KEY RETAILER_NAMEKEY,
    RETAILER__MODEL_.RETAILER_NAME RETAILER_NAME0,
    RETAILER__MODEL_.RETAILER_SITE_KEY RETAILER_SITE0KEY,
    RETAILER__MODEL_.CITY CITY,
    CAST(TIME_DIMENSION15.CURRENT_YEAR AS CHAR(4))YEARKEY,
    SUM(SALES_FACT16.SALE_TOTAL)REVENUE
FROM
    PRODUCT_LINE13,
    RETAILER__MODEL_,
    GOSLDW_TIME_DIMENSION TIME_DIMENSION15,
    GOSLDW_SALES_FACT SALES_FACT16,
    GOSLDW_PRODUCT_TYPE PRODUCT_TYPE17,
    GOSLDW_PRODUCT_DIMENSION PRODUCT_DIMENSION18
WHERE
    RETAILER__MODEL_.RETAILER_SITE_KEY IN (5000)
    AND
    PRODUCT_DIMENSION18.PRODUCT_KEY IN (30116)
    AND
    PRODUCT_LINE13.PRODUCT_LINE_CODE=
        PRODUCT_TYPE17.PRODUCT_LINE_CODE
    AND
    PRODUCT_TYPE17.PRODUCT_TYPE_CODE=
        PRODUCT_DIMENSION18.PRODUCT_TYPE_CODE
    AND
    PRODUCT_DIMENSION18.PRODUCT_KEY=
        SALES_FACT16.PRODUCT_KEY
    AND
    TIME_DIMENSION15.DAY_KEY=SALES_FACT16.ORDER_DAY_KEY
    AND
    RETAILER__MODEL_.RETAILER_SITE_KEY=
        SALES_FACT16.RETAILER_SITE_KEY
GROUP BY
    PRODUCT_LINE13.PRODUCT_LINE_CODE,
    PRODUCT_LINE13.PRODUCT_LINE,
    RETAILER__MODEL_.SALES_TERRITORY_KEY,
    RETAILER__MODEL_.SALES_TERRITORY,
    RETAILER__MODEL_.COUNTRY_KEY,

```

```

RETAILER__MODEL_.COUNTRY,
RETAILER__MODEL_.RETAILER_KEY,
RETAILER__MODEL_.RETAILER_NAME,
RETAILER__MODEL_.RETAILER_SITE_KEY,
RETAILER__MODEL_.CITY,
CAST(TIME_DIMENSION15.CURRENT_YEAR AS CHAR(4))
;

```

A.3 Other CPU bound queries

The CPU bound query Q1B, used for workload isolation tests that are described in section 4.5.1, “Test environment and data collection procedures” on page 110 is listed in Example A-3.

Example A-3 CPU bound query Q1B used for workload isolation

```

WITH
PRODUCT_LINE13
AS
  (SELECT
    PRODUCT_LINE.PRODUCT_LINE_CODE PRODUCT_LINE_CODE,
    MIN(PRODUCT_LINE.PRODUCT_LINE_EN)PRODUCT_LINE
  FROM
    GOSLDW_PRODUCT_LINE PRODUCT_LINE
  GROUP BY
    PRODUCT_LINE.PRODUCT_LINE_CODE),
SALES_TERRITORY_DIMENSION11
AS
  (SELECT
    SALES_TERRITORY_DIMENSION.COUNTRY_KEY COUNTRY_KEY,
    SALES_TERRITORY_DIMENSION.COUNTRY_CODE COUNTRY_CODE,
    SALES_TERRITORY_DIMENSION.
      SALES_TERRITORY_KEY SALES_TERRITORY_KEY,
    SALES_TERRITORY_DIMENSION.
      SALES_TERRITORY_CODE SALES_TERRITORY_CODE,
    SALES_TERRITORY_DIMENSION.COUNTRY_EN COUNTRY_EN,
    SALES_TERRITORY_DIMENSION.FLAG_IMAGE FLAG_IMAGE31,
    SALES_TERRITORY_DIMENSION.
      SALES_TERRITORY_EN SALES_TERRITORY_EN
  FROM
    GOSLDW_SALES_TERRITORY_DIMENSION SALES_TERRITORY_DIMENSION),
GENDER_LOOKUP12
AS
  (SELECT
    GENDER_LOOKUP.GENDER_CODE GENDER_CODE,
    MIN(GENDER_LOOKUP.GENDER)GENDER
  FROM
    GOSLDW_GENDER_LOOKUP GENDER_LOOKUP
  WHERE
    GENDER_LOOKUP.LANGUAGE='EN'
  GROUP BY
    GENDER_LOOKUP.GENDER_CODE),
RETAILER__MODEL_
AS

```



```

(SELECT
    RETAILER_DIMENSION10.RETAILER_SITE_KEY RETAILER_SITE_KEY,
    RETAILER_DIMENSION10.RETAILER_NAME RETAILER_NAME,
    RETAILER_DIMENSION10.CITY CITY,
    SALES_TERRITORY_DIMENSION11.COUNTRY_KEY COUNTRY_KEY,
    SALES_TERRITORY_DIMENSION11.
        SALES_TERRITORY_KEY SALES_TERRITORY_KEY,
    SALES_TERRITORY_DIMENSION11.COUNTRY_EN COUNTRY,
    SALES_TERRITORY_DIMENSION11.
        SALES_TERRITORY_EN SALES_TERRITORY,
    RETAILER_DIMENSION10.RETAILER_KEY RETAILER_KEY
FROM
    GOSLDW_RETAILER_DIMENSION RETAILER_DIMENSION10,
    SALES_TERRITORY_DIMENSION11,
    GENDER_LOOKUP12
WHERE
    RETAILER_DIMENSION10.GENDER_CODE=
        GENDER_LOOKUP12.GENDER_CODE
AND
    RETAILER_DIMENSION10.COUNTRY_KEY=
        SALES_TERRITORY_DIMENSION11.COUNTRY_KEY)
SELECT
    PRODUCT_LINE13.PRODUCT_LINE_CODE PRODUCT_LINEKEY,
    PRODUCT_LINE13.PRODUCT_LINE PRODUCT_LINE0,
    RETAILER__MODEL_.SALES_TERRITORY_KEY RETAILER_TERRITORYKEY,
    RETAILER__MODEL_.SALES_TERRITORY SALES_TERRITORY,
    RETAILER__MODEL_.COUNTRY_KEY RETAILER_COUNTRYKEY,
    RETAILER__MODEL_.COUNTRY COUNTRY,
    RETAILER__MODEL_.RETAILER_KEY RETAILER_NAMEKEY,
    RETAILER__MODEL_.RETAILER_NAME RETAILER_NAME0,
    RETAILER__MODEL_.RETAILER_SITE_KEY RETAILER_SITE0KEY,
    RETAILER__MODEL_.CITY CITY,
    CAST(TIME_DIMENSION15.CURRENT_YEAR AS CHAR(4))YEARKEY,
    SUM(SALES_FACT16.SALE_TOTAL)REVENUE
FROM
    PRODUCT_LINE13,
    RETAILER__MODEL_,
    GOSLDW_TIME_DIMENSION TIME_DIMENSION15,
    GOSLDW_SALES_FACT SALES_FACT16,
    GOSLDW_PRODUCT_TYPE PRODUCT_TYPE17,
    GOSLDW_PRODUCT_DIMENSION PRODUCT_DIMENSION18
WHERE
    PRODUCT_LINE13.PRODUCT_LINE_CODE=
        PRODUCT_TYPE17.PRODUCT_LINE_CODE
AND
    PRODUCT_TYPE17.PRODUCT_TYPE_CODE=
        PRODUCT_DIMENSION18.PRODUCT_TYPE_CODE
AND
    PRODUCT_DIMENSION18.PRODUCT_KEY=
        SALES_FACT16.PRODUCT_KEY
AND
    TIME_DIMENSION15.DAY_KEY=SALES_FACT16.ORDER_DAY_KEY
AND
    RETAILER__MODEL_.RETAILER_SITE_KEY=
        SALES_FACT16.RETAILER_SITE_KEY

```

```

GROUP BY
    PRODUCT_LINE13.PRODUCT_LINE_CODE,
    PRODUCT_LINE13.PRODUCT_LINE,
    RETAILER__MODEL_.SALES_TERRITORY_KEY,
    RETAILER__MODEL_.SALES_TERRITORY,
    RETAILER__MODEL_.COUNTRY_KEY,
    RETAILER__MODEL_.COUNTRY,
    RETAILER__MODEL_.RETAILER_KEY,
    RETAILER__MODEL_.RETAILER_NAME,
    RETAILER__MODEL_.RETAILER_SITE_KEY,
    RETAILER__MODEL_.CITY,
    CAST(TIME_DIMENSION15.CURRENT_YEAR AS CHAR(4))
;

```

The CPU bound query Q8B, which is used for workload isolation tests that are described in section 4.5.1, “Test environment and data collection procedures” on page 110 is listed in Example A-4.

Example A-4 CPU bound query Q8B used for workload isolation

```

SELECT COUNT_BIG(*)
FROM "PH3DW"."CUSTOMER_DIM" "CUSTDIM", "PH3DW"."TITLE_DIM" "TITDIM"
WHERE
--      "CUSTDIM"."C_CUSTOMER_ID" < 7000000 ;
      "TITDIM"."T_SEQ_NUM" < 10000 ;

```



B

Notes on maintenance

In this appendix, we provide a collection of notes that can help in identifying and documenting problems in DB2 Analytics Accelerator environments.

This appendix contains the following topics:

- ▶ Preventive maintenance
- ▶ What to do if you encounter an Accelerator problem
- ▶ Recent fixes

B.1 Preventive maintenance

Valid combinations of code levels for different components of the Accelerator are documented on the IBM support website. Some useful links are provided in this section.

B.1.1 Studio

Support for Data Studio Version 4.1 is documented at this site:

<http://www.ibm.com/support/docview.wss?uid=swg27036848>

You are encouraged to download the latest version of the Accelerator plug-ins as and when they become available because it is usually backward compatible.

What is new in Accelerator Studio 3.1.2 plug-ins is that it now includes the full Accelerator documentation. If you go to **Help** → **Contents** within Accelerator Studio, you can get all the manuals.

Also, the Task Launcher (**Help** → **Task Launcher**) now contains an updated “Accelerate” tab that points to updated Accelerator-related tutorials.

If you have an earlier version of the IBM Data Studio product, review the information that is provided in the following link:

http://pic.dhe.ibm.com/infocenter/dstudio/v4r1/topic/com.ibm.datatools.base.install.doc/topics/c_plan_consider_upgrade_product.html

You can upgrade the Studio product to the current release (for example, V4.1 from V3.2) by using the Update wizard in IBM Installation Manager.

B.1.2 DB2

Use the Consolidated Service Test (CST) as the base for service as described at this website:

<http://www.ibm.com/systems/z/os/zos/support/servicetest>

The current quarterly recommended service upgrade (RSU) is CST2Q13 (RSU1306), dated 1 July 2013, for DB2 9 and DB2 10. It contains all service through the end of March 2013 that is not already marked RSU, PE resolution, and HIPER/Security/Integrity/Pervasive program temporary fixes (PTFs), and their associated requisites and supersedes through the end of May 2013. The upgrade is described in the following document:

<http://www.ibm.com/systems/resources/RSU1306.pdf>

Most DB2 manuals were refreshed in June 2013. For the diagnostic guide, see APAR PM76924 for updated PDF files for V9 and V10.

The **RETAIN**® keyword for the Accelerator V3 related items is `idaav3r1/k`.

B.1.3 IBM DB2 Analytics Accelerator for z/OS

For IBM DB2 Analytics Accelerator for z/OS Version 3.1.0, apply the required or enabling APARs and PTFs and the recommended program update tape (PUT) level for your version of DB2 from the following link:

<http://www.ibm.com/support/docview.wss?uid=swg27035960>

System requirements for IBM Data Studio:

<http://www.ibm.com/support/docview.wss?uid=swg27016018>

<http://www.ibm.com/support/docview.wss?uid=swg27037912>

Network connections for IBM DB2 Analytics Accelerator:

<http://www.ibm.com/support/docview.wss?uid=swg27023654>

Network requirements for System z:

<http://www.ibm.com/support/docview.wss?uid=swg27024236>

Determining missing PTFs

You can use the **SMP/E REPORT MISSINGFIX** command in conjunction with **FIXCAT HOLDDATA** to determine if you have the recommended service for IBM DB2 Analytics Accelerator for z/OS, as explained here:

1. Acquire and RECEIVE the latest HOLDDATA file onto your z/OS system.
2. Go to the Enhanced HOLDDATA for z/OS and IBM OS/390® website and click **Download Enhanced OS/390 Enhanced HOLDDATA**:

<http://service.software.ibm.com/holdata/390holddata.html#download>

Download OS/390 Enhanced HOLDDATA

For complete information about the **SMP/E REPORT MISSINGFIX** command, see “SMP/E Commands” at the following site:

http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/GIMCOM50/CCONTENTS?SHEL F=gim2bk90&DN=SA22-7771-15&DT=20110523141208

Following is an example of the **REPORT MISSINGFIX** command:

```
SET    BOUNDARY (GLOBAL) .
REPORT MISSINGFIX ZONES(list of z/OS 1.13 target zones)
FIXCAT (IBM.DB2.AnalyticsAccelerator.V3R1) .
```

B.1.4 Change Data Capture

Change Data Capture is a component of the IBM InfoSphere Data Replication product bundled in with the Accelerator software.

For the IBM InfoSphere Data Replication version 10.1.3 Information Center, go to:

<http://pic.dhe.ibm.com/infocenter/iidr/v10r1m2/index.jsp>

For the High Availability Setup for the Incremental Update Capture Engine site, go to:

<http://www.ibm.com/support/docview.wss?uid=swg27037912>

Network requirements for System z:

<http://www.ibm.com/support/docview.wss?uid=swg27024236>

<http://www.ibm.com/support/docview.wss?uid=swg27028171>

B.1.5 OMEGAMON for DB2 Performance Expert

PM55637 introduced the instrumentation changes in batch reporting (Accounting, Statistics, and Record Trace).

Refer to APAR II14642 for recent issues and tips on OMPE V510 and V511 batch reporting.

APAR PM72622 introduced support of DB2 IFI enhancements for IBM DB2 Analytics Accelerator V3 in batch reporting, such as additional locking types.

More support for IBM DB2 Analytics Accelerator (after V510 PTFs UK75097/UK77225) is available in the following resources:

- ▶ Save data into Performance Database:
V511 NF PTF UK81551, APAR PM68928
- ▶ V3 support – plan and package level Accel.wait time in batch reports:
V511 NF PTFs UK90526/UK94985, APAR PM72766/PM87384
- ▶ Display accelerator data in Classic Real Time:
V511 NF PTF UK94132, APAR PM80739
- ▶ Spread sheet input data generator for statistics and accounting:
PM73732, V510 PTF UK90267, V511 PTF UK90268

Documentation can be accessed at the following links:

- ▶ IBM Tivoli OMEGAMON XE for DB2 Performance Expert information center:
http://pic.dhe.ibm.com/infocenter/tivihelp/v15r1/topic/com.ibm.omegamon.xe.pe_db2.doc_5.1.1/ko2welcome_pe.htm
- ▶ DB2 Tools Product Page (PDF manuals):
<http://www.ibm.com/support/docview.wss?uid=swg27020910#omegaxepe-lib>

B.2 What to do if you encounter an Accelerator problem

In case of unexpected behavior by the IBM DB2 Analytics Accelerator, you should always open a problem management record (PMR) with IBM support. To expedite the support process, it is advised to always collect a trace (with DEFAULT profile) regardless of the complexity of the problem or the type of problem, and make it available to IBM support. The *IBM DB2 Analytics Accelerator for z/OS Version 3.1.0 User's Guide*, SH12-6985, contains detailed information about how to collect a trace. The manual can be accessed from the following URL:

<http://publibfp.dhe.ibm.com/epubs/pdf/h1269851.pdf>

Note: The techdocs are updated for various scenarios and key links are made available on the IBM support home page:

http://www-947.ibm.com/support/entry/portal/Overview/Software/Information_Management/DB2_Analytics_Accelerator_for_z~OS

B.2.1 Accelerator trace

If you open an Accelerator-related PMR, it is almost always required to collect the Accelerator trace information. When a trace is collected, several options can influence the detail and volume of information included in the trace.

The following steps should be adhered to when initially collecting information for a PMR:

1. Never clear the trace unless instructed to do so by IBM support, even if the scenario is reproducible. Having all the data from the past in the trace allows IBM support to find references to earlier processes that might be relevant in diagnosing your problem.
2. Include a detailed description of the scenario that resulted in the observed unacceptable behavior, if possible. This also includes information about how the failing function was called, for example, via a batch job or via the IBM DB2 Analytics Accelerator Data Studio.
3. Include the error message and a screen capture, if available.
4. Collect an Accelerator trace:
 - a. Trace profiles should always be set to DEFAULT for Accelerator trace and OFF for Stored Procedure trace, as indicated in Chapter 2, “Using the Studio client to manage the environment” on page 11 (refer to Figure 2-34 on page 44, Figure 2-35 on page 45, and Figure 2-36 on page 46). There are other trace profiles available that give more detailed information. But, if the other trace profiles are turned on, there is a possibility that the trace files get voluminous and may impact the overall performance. Therefore, other traces should only be activated upon request from the IBM support personnel.
 - b. On the Save Trace panel, only the options that are shown in Figure B-1 on page 312 need to be selected (which are basically the default options).

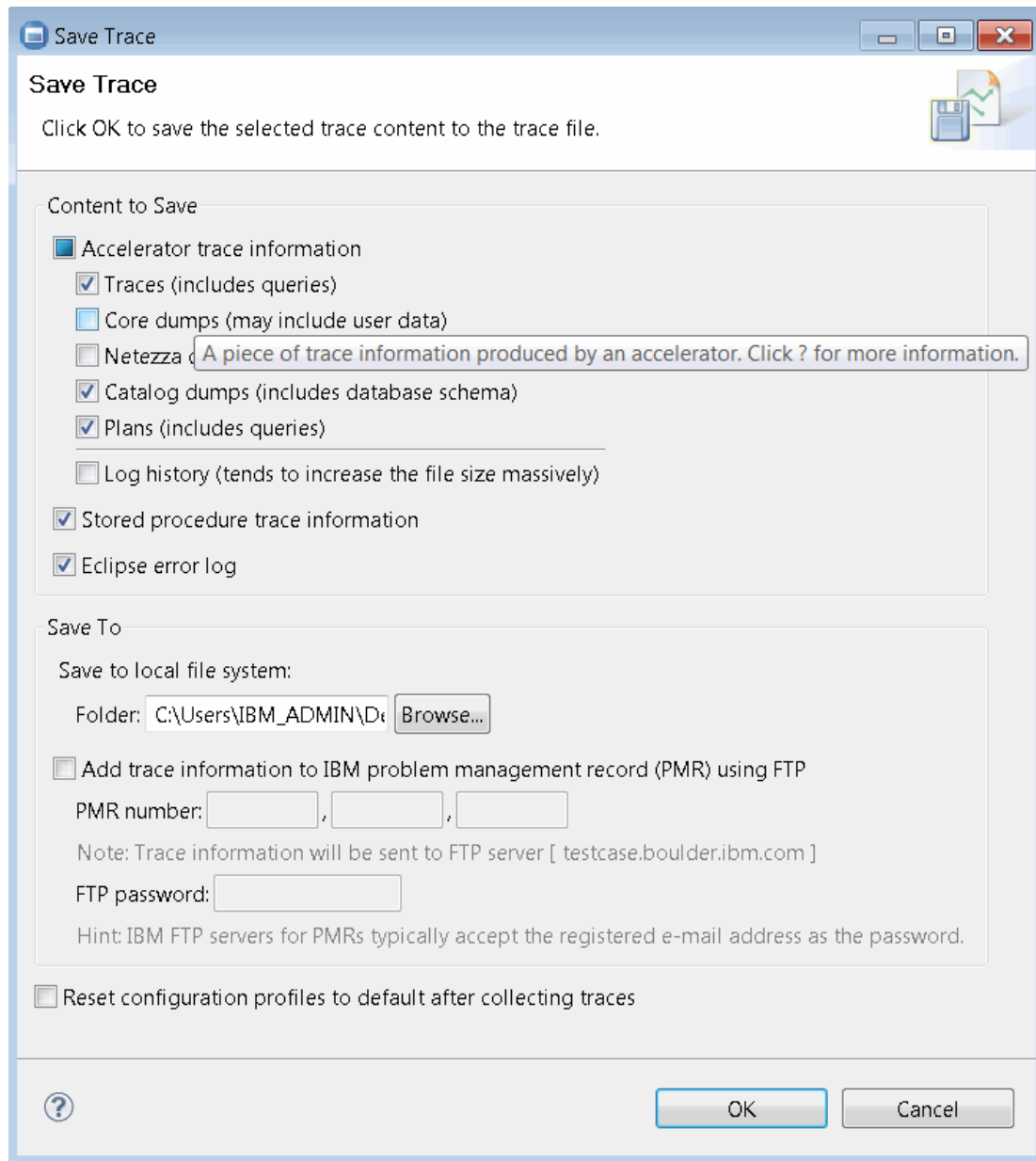


Figure B-1 Studio: Save trace options

B.2.2 FTPing the Accelerator trace to IBM support

You can also collect the required trace data via the Data Studio graphical user interface (GUI) and automatically attach it to the PMR, as shown in Figure B-1, by selecting the “Add trace information to IBM problem management record (PMR) using FTP” option. You should enter the PMR number that is assigned by IBM support and follow the password guidelines to upload the trace data by point-and-click if you are connected to the Internet. The other three options (which are not selected by default) as shown in Figure B-1 might contain sensitive user data and therefore, care should be taken to follow your shop’s security guidelines in dealing with such data.

B.3 Recent fixes

Accelerator support on DB2 10 for z/OS was added by APARs PM50434, PM50435, PM50436, PM50437, and PM51918. These APARs together provide the code in DB2 to support the use of Accelerator V3.

Following is a list of APARs of interest:

PM72274: Support for several new features of IBM DB2 Analytics Accelerator V3.

PM85936: Extensions to ADMIN_INFO_SQL stored procedure to include more functions and support for accelerated queries.

PM87554: Accelerator-related database access threads (DBATs) require transaction accounting records and should not be susceptible to Idle Thread Time-Out conditions.

PM88071: Support routing of implicit cast for DECFLOAT, bit operations, and TIMESTAMPDIFF.

PM90591: When running with IBM DB2 Analytics Accelerator, SQL CODE -901 can occur for the routing of a statement containing a DATE, TIME, and TIMESTAMP constant. Allow time stamp precision constants greater than 6 with zeroes.

PM90886 and PM95035 (both open): A new online changeable **ACCELMODEL** subsystem parameter determines whether to enable modeling of query workload for evaluating potential savings for both the accumulated elapsed time and CPU time if the plan is executed on an Accelerator. Only queries that are deemed eligible for execution on an Accelerator by DB2 are included in Accelerator-related fields of accounting trace, IFCID 3.

PM93789 and PM94515: Blocking and error messages for correlated subqueries are not supported on the Accelerator.



DSNZPARM parameters setting for data warehouse

This appendix describes groupings of DB2 for z/OS system parameter settings (DSNZPARM keywords) that could be used in the implementation of a new data warehouse using DB2 for z/OS (referred to as *DB2* in this book unless otherwise noted) or as a simple way of level-setting your current DSNZPARM keywords on an existing warehouse. We believe the first group of keywords could have a significant impact on both the successful implementation and performance of a new or existing data warehouse using DB2. The second group could still have an impact on a warehouse using DB2, but might not be as critical to the success of running a warehouse using DB2.

We group keywords logically together; that is, with other keywords used in related areas.

All system parameter settings described here are for DB2 10 for z/OS. Defaults, minimum, and maximum values of any one system parameter could be different from previous versions and could change in the future. Always verify the values that are about to be used for any DSNZPARM keyword by using the latest DB2 product documentation before making changes. As a “best practice”, it is also suggested that defaults should never be assumed and that all keywords should be coded even if the intent is to use that keyword's default. This could minimize the impact by changes made to a system parameter's default value in a future release, version, or maintenance level.

C.1 Suggested DSNZPARM parameters setting

The suggested keyword values described in this book should be used as a starting point only and should not be considered the “be-all and end-all” of DSNZPARM keywords. They were primarily assembled here for a new data warehouse using DB2. The effects of using the following DSNZPARM keyword settings should be monitored and adjustments made as necessary. The values used in this document in most cases will not be your final choices for any particular set of DSNZPARM keywords.

In the following descriptions, the term *updatable* means that the DSNZPARM keyword can be changed without restarting (recycling) the DB2 subsystem or DB2 member if it is a data sharing group. To activate the newly changed keyword, the DSNZPARM source member needs to be modified making the wanted adjustments to the DSNZPARM keyword settings, reassembled, and relinked. The DB2 **-SET SYSPARM** command is then used to activate the changed keywords. Refer to Chapter 7 of *DB2 10 for z/OS Command Reference*, SC19-2972, for a complete description of the **-SET SYSPARM** command.

On the following pages, the terms *installation keyword*, *opaque keyword*, and *hidden keyword* will be used to describe the different DSNZPARM keywords. An *installation keyword* is available through the installation panels and is documented in the Installation and Migration Guide listed below. The term *opaque keyword* is used to identify system parameters delivered through the maintenance stream. Some of the *opaque keywords* are now documented in the Installation Guide as of DB2 9. Finally, in the rare case that the term *hidden ZPARM* is used to describe a keyword, it is referring to a system parameter whose description has not been externalized and is primarily in place for the use of DB2 Level 2 for problem analysis and resolution. Normally, it is advised that you avoid modifying hidden ZPARMs.

The IBM DB2 for z/OS publication that was primarily used while putting this appendix together was the *DB2 10 Installation and Migration Guide*, GC19-2974-09.

Be careful when you pick up a presentation describing what DSNZPARM keywords should be used. First, they should almost always be taken as suggestions or as a starting point. ZPARMs are unique to your environment; that is their purpose. Not all DB2 subsystems behave the same. Second, determine when the paper, presentation, or book was written. If it was written for DB2 Version 8 for example, it is possible that some of the DSNZPARM keywords described have been replaced or even removed. Finally, you will come across everything from outdated parameters to misspelled ones. A lot of time can be spent trying to determine if something is useful only to find out that it no longer exists or never existed.

We discuss the following topics:

- ▶ Parallelism
- ▶ Query processing and diagnostics
- ▶ Disk storage allocation
- ▶ Storage allocation
- ▶ Star join processing
- ▶ Utility sort processing
- ▶ Query Accelerator
- ▶ LOB specific keywords
- ▶ XML specific keywords
- ▶ Miscellaneous system parameters
- ▶ Deprecated in DB2 10
- ▶ Pre DB2 9 DSNZPARM parameters removed, but still documented

C.1.1 Parallelism

This first group of keywords is used to manage the behavior of DB2's parallelism. DB2's query parallelism targets I/O intensive queries (such as table space scans, large index scans) and CP-intensive queries (such as joins, sorts, complex expressions). Its objective is to reduce the overall elapsed time of a query by taking advantage of available I/O bandwidth and processor power. Parallelism can be a critical DB2 component in reducing the elapsed time of queries running in DB2. On the other hand, a query that runs quick enough that it does not qualify to be moved to the Accelerator, probably could not, and should not be using parallelism.

The following five DSNZPARM keywords deal with enabling parallelism and managing the behavior of parallelism in DB2:

- ▶ CDSSRDEF
- ▶ PARAMDEG
- ▶ PARA_EFF
- ▶ SPRMPH
- ▶ PTASKROL

CDSSRDEF

Macro: DSN6SPRM
Allowed values: 1, ANY
Default: 1 (Disable the use of parallelism)
Installation panel: DSNTIP8
Updatable via **-SET SYSPARM = YES**
Recommended value: ANY

The special register "CURRENT DEGREE" is used to enable and disable parallelism in a dynamic SQL environment, the SQL type that is most likely to be used in a data warehousing environment. If the CURRENT DEGREE register is set to "1" or "1 ", parallelism is disabled or not available as an access path choice. If CURRENT DEGREE is set to "ANY", parallelism is enabled, which allows the optimizer to choose parallelism as a potential access path. The system-wide default value for CURRENT DEGREE is set on the installation panel DSNTIP8 or by the DSNZPARM keyword, **CDSSRDEF**, on the **DSN6SPRM** macro. Regardless of the value set as the DB2 subsystem default, the value of the CURRENT DEGREE special register can be modified (overridden) via the SET CURRENT DEGREE SQL statement.

When a dynamic SQL statement is executed without first setting the CURRENT DEGREE special register, DB2 will use the value specified on this DSNZPARM keyword.

The default for CDSSRDEF is 1. Because this is a warehouse system and we want to minimize the run time (elapsed time) of most SQL statements, you might want to consider setting CDSSRDEF to "ANY" if you want ALL dynamic SQL to consider using parallelism. However, if this is your first venture into parallelism or warehouse queries, consider running EXPLAIN against your queries to determine how much parallelism could be used. Use this information to adjust your PARAMDEG (discussed next) to an appropriate value. Also, if new to using the CURRENT DEGREE special register set to ANY, do not initially set PARAMDEG to 0 (zero). Use the information from EXPLAIN and the formula of (2 x # CPs and zIIPs). Specifying "ANY" is again system wide so all dynamic SQL will have the opportunity to use parallelism. These suggestions are designed to give some added processor usage control.

The SQL statement SET CURRENT DEGREE sets the current degree special register. It does not have any affect on the actual degree of parallelism or the number of processors used for parallelism. Parallelism is decided by DB2 based on PARAMDEG, the number of CPs and zIIPs, and other factors.

For static SQL, a BIND or REBIND of a package specifying the **DEGREE** keyword can be used to manage whether or not the SQL in that package will be considered for parallelism.

PARAMDEG

Macro = DSN6SPRM
Allowed values = 0 - 254
Default value = 0
Installation panel ID = DSNTIP8
Updatable via **-SET SYSPARM = YES**
Recommended value: 2 x # CPs and zIIPs

The next parallelism control to consider is the number of CPs that DB2 will be allowed to use. MAX DEGREE on the installation panel DSNTIP8 or the DSNZPARM keyword, **PARAMDEG**, on the **DSN6SPRM** macro can be used to set the maximum number of CPs that DB2 can use for CPU query parallelism. The default for this value is 0 (zero), which allows DB2 to choose the degree of parallelism. When the default is used, DB2 favors a degree of parallelism that is 2 times the number of available processors, including IBM System z Integrated Information Processor (zIIP specialty engine).

For a new data warehouse, where query performance metrics have not yet been established, it is suggested that the default value 0 not be used. Specifying 0 or taking the default could allow the over-allocation of processor resources, potentially causing performance issues. Use the information from running EXPLAINS for the warehouse queries along with the formula of (2 x # CPs and zIIPs) to assist in determining a starting value for PARAMDEG.

The actual degree of parallelism used by DB2 is highly influenced by the number of processors that are available and the number of partitions in the table space being accessed. The higher that either or both of these values are, the higher the degree of parallelism that might be used.

Parallel processing is one of the functions within DB2 that is eligible to run on a zIIP. When an internal threshold is reached for the amount of processing performed by a parallel query, the parallel child tasks will be zIIP eligible. Up to 80% of all parallel child task work is eligible to run on a zIIP after the initial threshold is reached.

In most cases however, the time should be taken to determine the best value for MAX DEGREE (PARAMDEG). A guideline for choosing a starting point for MAX DEGREE is to choose a value somewhere midpoint between the maximum number of CPs available and the maximum number of partitions that will be processed. If you believe that the queries will tend to be more CPU intensive, make this value closer to the number of CPs that you have available. On the other hand, if you predict that the queries will be more I/O intensive, make this number closer to the number of partitions. If you are concerned only about processor usage, consider using the formula: # Processors \leq X \leq 2 x # Processors, where "X" represents your degree of parallelism.

The degree of parallelism used should be closely monitored and adjusted accordingly.

PARA_EFF

Macro = DSN6SPRM
Allowed values = 0 - 100
Default value = 50
Installation panel ID = DSNTIP8
Updatable via **-SET SYSPARM = YES**
Recommended value: See Description

APAR PM16020 added the **PARA_EFF** keyword to the DSNZPARM **DSN6SPRM** macro. This DSNZPARM keyword controls the parallelism efficiency assumption of DB2.

PARA_EFF is specified as a percentage 0 - 100. The two extremes are easy to explain and even easier to understand. If you specify **PARA_EFF** = 0, DB2 behaves exactly as pre DB2 9 did when picking an access path. If you use **PARA_EFF** = 100, DB2 behaves as though the APAR PM16020 had never been applied using DB2 9 cost reduction. Setting **PARA_EFF** to 0 or 100 are special cases. It is the values 1 - 99 inclusively that make this keyword interesting.

Any value of 1 - 99 still gives you the DB2 behavior of attempting to include parallelism as part of the access path decision, a behavior change that arrived in DB2 9. However, you get to control how aggressive that DB2 gets when making the decision to use parallelism.

DB2 needs to make some assumptions about how much savings would be gained by using parallelism against the actual cost of parallelism. The closer the value specified on **PARA_EFF** is to 99, the more likely DB2 is to choose an access path that includes a higher degree of parallelism to maximize the elapsed time reduction. It is possible however, that the higher degree of parallelism could come at a higher total cost if compared to a plan with a lower degree of parallelism or with a sequential plan. The total cost of using parallelism could outweigh the elapsed time savings gained with parallelism.

The closer **PARA_EFF** is set to 1, the lower the possible estimated elapsed time savings could be if parallelism is included. The high value for the **PARAMDEG** system parameter can accentuate all this. A lower value for **PARAMDEG** can coexist with a higher **PARA_EFF** percentage. However, a higher **PARAMDEG** percentage will likely warrant a lower value for **PARA_EFF**.

The goal is to pick an access path that includes parallelism were parallelism's elapsed time savings outweigh the additional CPU cost of using parallelism. The **PARA_EFF** parameter controls how aggressively DB2 estimates the elapsed time savings before including parallelism as part of the access path.

If detailed examples of how this keyword affects parallelism are of interest to you, see the examples of how varying the value of **PARA_EFF** affects the optimization outcome in the "Problem Conclusion" section of cover for APAR PM16020.

SPRMPTH

Macro = DSN6SPRC
Allowed values =
Default value = 120 msec
Installation panel ID = None
Updatable via **-SET SYSPARM = NO**
Recommended value: 120

DSNZPARM **SPRMPTH** on the **DSN6SPRC** macro is the hidden ZPARM. Normally, discussions about hidden ZPARM keywords are avoided. However, **SPRMPTH** can be an extremely helpful keyword that is well documented in numerous DB2 presentations.

SPRMPTH sets the threshold that a query must reach before being considered for parallelism. Because of the initial cost to set up parallelism, often parallelism is not wanted for short-running queries. This threshold can help prevent that from happening. When using the default as an example, any queries that DB2 thinks will run in less the 120 msec will have parallelism disabled. There are cases where the default value is not high enough to eliminate everything that you do not want, to use parallelism. It is in these situations that you might consider increasing this value. Although there are situations where you might want to make this threshold higher than the default, there is no reason to ever decrease it.

The internal cost used in the compare against SPRMPH is stored in field QW0022CO in IFCID 0022. This field/value can be used if you are trying to determine a more appropriate value, rather than the default, to set SPRMPH.

APAR PQ45820, PQ25135

PTASKROL

Macro = DSN6SYSP
Allowed values = YES/NO
Default value = YES
Installation panel ID = none
Updatable via **-SET SYSPARM = YES**
Recommended value: YES

The DSNZPARM **PTASKROL** keyword is an opaque keyword that determines if DB2 is to roll up all of the accounting trace records from a parallel query task into a single record. If you set **PTASKROL** to YES, all of the parallel child task records are rolled up into the single originating parallel parent task record. When YES is specified, less SMF data is collected and processing costs are reduced. However, some detail is lost. The general recommendation is to take the default (YES) for performance reasons. If you are attempting to diagnose a balancing issue, NO should be considered to obtain the more detailed individual records. If NO is specified, each parallel child task produces its own accounting trace record, increasing the total number of SMF records collected for that task.

C.1.2 Query processing and diagnostics

The following parameters relate to query processing and diagnostics:

- ▶ CACHEDYN
- ▶ EDMSTMTC
- ▶ DECDIV3
- ▶ DECARTH
- ▶ MXQBCE

CACHEDYN

Macro = DSN6SPRM
Allowed values = YES/NO
Default value = YES
Installation panel ID = DSNTIP8
Updatable via **SET SYSPARM = YES**
Recommended value: YES

DB2 has a storage area that is used to cache dynamic SQL statements that DB2 refers to as the *dynamic statement cache*. Its purpose is to reduce the necessity for full PREPAREs, thus resulting in a potential reduction in CPU by caching (saving) the dynamic SQL statement text and the executable statement dynamic statement cache.

The dynamic statement cache is defined by setting the **CACHEDYN** keyword on the **DSN6SPRM** macro, or CACHE DYNAMIC SQL on the installation panel DSNTIP8 to YES, the default.

Note: The dynamic statement cache can also be used to capture dynamic SQL statements for tuning.

Assuming your warehouse environment is predominantly dynamic SQL, it is recommended that the dynamic statement cache be left on (**CACHEDYN YES**) to reduce the cost of using dynamic SQL.

Note: If you specify **CACHEDYN** as YES (or just take the default), you must also set the **AUTH** system parameter on the **DSN6SPRM** macro to YES (its default). **AUTH** enables DB2's authorization checking. This is the "USE PROTECTION" field on the DSNTIPP installation panel.

EDMSTMTC

Macro = DSN6SPRM
Allowed values = 5000 to 1048576
Default value = 113386
Installation panel ID = DSNTIP8
Updatable via **-SET SYSPARM = YES**
Recommended value: See description

To capture anything in the dynamic statement cache, you are going to have to first have a dynamic statement cache. The size of the statement cache storage area is defined by the **EDMSTMTC** keyword on the **DSN6SPRM** macro or EDM STATEMENT CACHE on the DSNTIPC installation panel. If you take the defaults, and it is recommended that you always code the system parameter value, be aware that the default value for EDMSTMTC changed from 102396 K for Version 8 and 56693 K for DB2 9, to 113386 K in DB2 10. The larger the dynamic statement cache, the more dynamic SQL statements it can hold. However, monitor how the cache is used. Over-allocating the statement cache is just wasting storage, and not making the cache large enough for all of your dynamic SQL workload can result in the reuse of cache storage. The dynamic statement cache is a least recently used (LRU) wrap around storage area. When full, it wraps and reuses space already occupied by dynamic SQL that it thinks is no longer being used.

DECDIV3

Macro = DSN6SPRM
Allowed values = 5000 to 1048576
Default value = 113386
Installation panel ID = DSNTIP4
Updatable via **-SET SYSPARM = NO**
Recommended value: None, application dependent

The DECDIV3 controls whether the last three digits to the right of the decimal point should be retained after any decimal division. YES retains the last three digits; NO does not. NO is the default and follows normal SQL rules for decimal division.

This value can be set using the **DECDIV3** system parameter on the **DSN6SPRM** macro or via the MINIMUM DIVIDE SCALE field on installation panel DSNTIP4.

DECARTH

Macro = DSNHDECP
Allowed values = DEC15, DEC31, 15, 31, or DPP.S
Default value = DEC15
Installation panel ID = DSNTIP4
Updatable via **-SET SYSPARM = NO**
Recommended value: None, application dependent

DECARTH specifies the rules that DB2 will follow when both operands in a decimal operation have a precision of 15 or less. DEC15 or 15 does not allow precision greater than 15 digits. DEC31 or 31 allows for precision up to 31 digits. Whatever value this parameter is set to becomes the default for the CURRENT PRECISION special register.

This is *not* a DSNZPARM keyword. It belongs to DECP. It is mentioned here because it has affected dynamic SQL in our testing so that we had to use the CURRENT PRECISION special register to resolve the issue.

MXQBCE

Macro = DSN6SPRM
Allowed values = 1 to 32767
Default value = 1023
Installation panel ID = none
Updatable via **-SET SYSPARM = NO**
Recommended value: 1023 (default)

MXQBCE limits how many different join sequences the DB2 optimizer will consider before deciding on an access path. The lower the number specified for MXQBCE, the fewer the number of join sequences considered. By reducing the number of join sequences considered, the time that DB2 spends on bind processing could also be reduced. Of course, spending less time could also mean that a less appropriate access path could be chosen.

The value used as the default is 1023, which is determined using the formula $(2^{**} N - 1)$ where $N = 10$.

Setting this threshold too low can result in DB2 completing too small of a percentage of prepares, which might not be enough to distinguish which is the best plan. So, with a value much less than 1023, you run the risk of clipping a lower cost and possibly better access path. In other words, if you decide to change this keyword, do *not* make it any lower than the default value of 1024. Keep in mind that as you raise MXQBCE, you raise the cost of preparing the statement, and in most cases, you still will not get a better access path.

MXQBCE is included here because it is one of those values that you constantly see mentioned in discussions on how to set up DB2 for a warehouse on System z. If it is understood what it is and how it is used, it might minimize the number of people who try to change it. Also, be aware of MAX_OPT_STOR, the maximum amount of storage that can be consumed by the DB2 optimizer; and MAX_OPT_CPU, the maximum amount of CPU time that can be used by the DB2 optimizer. For all practical purposes, these values should be left at their defaults unless DB2 Level 2 instructs you to increase them. Again, these are mentioned here only because they occasionally show up in presentations and articles as suggestion ZPARMs that should be modified.

Note: The DSNZPARM **TABLES_JOINED_THRESHOLD** keyword was removed from DB2 back in DB2 9 and should not be considered anymore for these optimizer calculations. Any references to TABLES_JOINED_THRESHOLD should be ignored.

C.1.3 Disk storage allocation

The following parameters relate to disk allocation:

- ▶ MGEXTSZ
- ▶ DSVCI

MGEXTSZ

Macro = DSN6SYSP
Allowed values = YES, NO
Default value = YES
Installation panel ID = DSNTIP7
Updatable via **-SET SYSPARM = YES**
Recommended value: YES (default)

When MGEXTSZ on the **DSN6SYSP** macro or **OPTIMIZE EXTENT SIZGIN** on the DSNTIP7 installation panel is set to YES, the default, DB2 will manage secondary extent processing to minimize the chances of the table space running out of extents before using all the space. YES enables a sliding secondary quantity for DB2 managed (SMS managed) page sets.

There is no reason to *not* specify YES. MGEXTSZ has been successfully used by customers and in our labs since it first appeared in DB2 Version 8. Specifying YES also reduces the administrative cost of managing data sets while preventing possible -904 errors from running out of extents before exceeding the table space maximum size.

This parameter is included here to ensure that it is set to YES or that the default is taken. YES will improve performance with no negative effects.

DSVCI

Macro = DSN6SYSP
Allowed values = YES, NO
Default value = YES
Installation panel ID = DSNTIP7
Updatable via **-SET SYSPARM = YES**
Recommended value: YES (default)

When **DSVCI** on the **DSN6SYSP** macro, or when **VARY DS CONTROL INTERVAL** on the DSNTIP7 installation panel is set to YES, the DB2 managed data sets will use the same VSAM control interval as the buffer pool that is associated with that table space of index space.

If **DSVCI** is set to NO, DB2 managed page sets are created with a 4 K fix interval regardless of what size buffer pool they use.

If **DSVCI** is changed from NO to YES, any pre-existing page sets will continue to use a 4 K page interval until they are redefined (for example, through a REORG or LOAD REPLACE).

This parameter is included here to ensure that it is set to YES or that the default is taken. YES will improve performance with no negative effects.

C.1.4 Storage allocation

The following parameters relate to virtual storage allocation:

- ▶ MXDTCACH
- ▶ SRTPOOL
- ▶ MAXRBLK
- ▶ MAXTEMPS_RID
- ▶ CONTSTOR
- ▶ MINSTOR
- ▶ DSMAX

For any storage recommendation, remember that the amount of usable virtual storage should never exceed the amount of real storage available. All efforts should be made to ensure that the system never exceeds a zero (0) paging rate.

The amount of storage being used by DB2 is available through the IFCID 0225 trace record, which is part of the Class 1 Statistics trace. Storage usage can also be verified using the z/OS **DISPLAY VIRTSTOR** command, and paging rates can be checked by using IBM Resource Measurement Facility™ (RMF).

MXDTCACH

Macro = DSN6SPRM
Allowed values = 0 to 512 (in megabytes)
Default value = 20
Installation panel ID = DSNTIP8
Updatable via **-SET SYSPARM = YES**
Recommended value: 128 (see warning below)

The **MXDTCACH** keyword on the **DSN6SPRM** macro, or the **MAX DATA CACHING** field on the installation panel DSNTIP8 is used to specify the maximum amount of memory in megabytes that can be allocated per thread for data caching.

As of DB2 9, in-memory data caching was extended to *joins* other than *star joins*. Base tables, temporary tables, table expressions, and materialized views that lack appropriate indexes can all benefit from sparse index in memory data caching. MXDTCACH manages the size of that cache. Increasing **MXDTCACH** from its 20 MB default could minimize the amount of random activity that might spill over to the sort work buffer pool.

WARNING: The storage specified on MXDTCACH is allocated *per* query. Although all of the memory allocated for MXDTCACH comes from above the 2 GB bar, verify that sufficient memory exists to support the possible number of anticipated concurrent running queries. Ensure that virtual to real storage usage for DB2 is being monitored using the DB2 IFCID 0225 and the z/OS **VIRTSTOR** command, and that the z/OS system paging rate stays at 0 (zero) using a tool like RMF. In addition, IFCID 27 tracks the memory utilization for data caching.

Consider starting **MXDTCACH** off at 128 MB for a data warehousing DB2 subsystem. If you find your queries displaying poor performance that have the column "PRIMARY_ACCESTYPE = T" (the base table or result file will be materialized into a work file, and that work file will be accessed using a sparse index) in the EXPLAIN's PLAN_TABLE, consider increasing the size of MXDTCACH to an even higher value.

SRTPOOL

Macro = DSN6SPRM
Allowed values = 240 to 128000
Default value = 10000
Installation panel ID = DSNTIPC
Updatable via **-SET SYSPARM = YES**
Recommended value: Large (based on installation)

The **SRTPOOL** subsystem parameter on the **DSN6SPRM** macro, or the SORT POOL SIZE field on installation panel DSNTIPC, defines the size of the sort pool used by an SQL statement in kilobytes. When it comes to the sort pool, larger is almost always better. However, keep in mind that this value is *per* SQL statement. If the maximum value 128000 is used and there are 10 SQL statements requiring a sort running simultaneously, each of those 10 SQL tasks has

the potential of using 128000 KB of sort pool work area. A little common sense comes in to play here when picking a value for SRTPOOL. You are trading SQL performance for the possibility of a storage shortage. Again, it is important to remember that the sort pool value specified with SRTPOOL is *per* concurrent SQL user.

If you do not have unlimited storage available, a potential starting point for the SRTPOOL size can be calculated using the formula found in the *DB2 10 Installation and Migration Guide*, GC19-2974, or *DB2 10 for z/OS Managing Performance*, SC19-2978:

$32000 \times (16 + \text{sort key length} + \text{sort data length})$

In the above formula, sort key length is the sum of the lengths of all columns used on the ORDER BY clause. The second value, sort data length, is the sum of the lengths of all columns used in the entire SQL statement. That includes both the columns on the SELECT clause and the columns on the ORDER BY clause, as long as the columns are not repeated in both places.

As an alternative, you can also use IFCID 0096, which contains value information about each sort process that takes place in DB2. A 0096 record is recorded at the end of every sort (an IFCID 0095 is recorded at sort start). The information necessary can be found in the fields QW0096KL (key length), and QW0096DL (data length).

You can expect a fair amount of SQL sort activity when running data warehouse queries when indexes are not available or the Accelerator is not being used.

MAXRBLK

Macro = DSN6SPRM
Default value = 400000 (up from 8000 in previous releases)
Allowed values = 0, 128 to 10000000
Updatable via **-SET SYSPARM = YES**
Installation panel ID = DSNTIPC
Recommended value: See Description

The RID POOL SIZE field on the DSNTIPC installation panel or MAXRBLK on the DSN6SPRM set the size of the storage pool (in kilobytes) used for record identifier (RID) processing. Setting this value to 0 (zero) disables RID pool processing, meaning no list prefetch, hybrid joins, and multiple index access (and whatever else that might need to use the RID pool).

This keyword can be left at the default value and monitored. If RID pool use is significant (a high use of list prefetch, hybrid joins, and multiple index access), this pool size should be increased. A larger RID pool will improve the performance of SQL using RID processing. An example of when it might be necessary to increase the RID pool size is in support of XML index access (particularly DOCID with logical AND or OR).

If the size is increased, increase in 32 K increments and monitor pool usage, DB2 storage usage, and system paging to ensure that increasing the pool size does not cause some other performance issue. There is a formula in the *DB2 10 for z/OS Installation and Migration Guide*, SC19-2974 that can be used to determine a starting point for RID pool size.

WARNING: Be careful if MAXTEMPS_RID, explained below, is being used. Do not reduce the RID pool size in an attempt to save space by allowing RID processing to fail over to using the work files. Access path selection is still based on the RID pool size. If the optimizer thinks the pool is too small to process the anticipated number of RIDs, an access path using RID will not be chosen. Also, when considering the amount of space used by the RID pool, this space comes from above the 2 GB bar.

MAXTEMPS_RID

Macro = DSN6SPRM
Default value = NOLIMIT
Allowed values = NONE, NOLIMIT, 1 to 329166
Updatable via -SET SYSPARM = YES
Installation panel ID = DSNTIP9
Recommended value: NOLIMIT

MAXTEMPS_RID on the **DSN6SPRM** macro, or MAX TEMP RID field on installation panel, DSNTIP9, determines the maximum amount of storage in the work file database that a single RID list can use. Storage is used in the work file database when the sorted RID list is too large for the RID pool. Be careful with this one. The default NOLIMIT is being recommended. However, if queries consistently fail over to the work file database, the value of MAXRBLK above should be increased. MAXTEMP_RID is not a crutch for poor DB2 subsystem tuning.

CONTSTOR

Macro = DSN6SPRM
Allowed values = YES, NO
Default value = YES
Installation panel ID = DSNTIPE
Updatable via -SET SYSPARM = YES
Recommended value: NO

The **CONTSTOR** keyword on the **DSN6SPRM** macro or the CONTRACT THREAD STG on the installation panel, DSNTIPE, determines whether DB2 should contract (reduce) the working storage area for threads. **CONTSTOR** defaulted to NO in releases prior to DB2 10. In DB2 10, the default was changed to YES. Unless you are experiencing some kind of virtual storage issue, or anticipate having a storage issue, **CONTSTOR** should be set to NO. A performance price is paid when DB2 manages thread storage (CONTSTOR YES). It is a trade-off between performance and minimizing storage usage.

For best performance, specify NO for **CONTSTOR**. To resolve storage constraints in DBM1 address space, specify YES for **CONTSTOR**. The associated CPU cost of turning this parameter on is on average about 2%. When switched on (YES), the cost is incurred because DB2 for z/OS periodically contacts the agent local non-system pool for each DB2 thread. Storage that is acquired by a thread normally remains allocated to that thread until thread deallocation for any threads bound with RELEASE(DEALLOCATE). Threads bound with RELEASE(COMMIT) will mark these blocks as free and they can be reclaimed at commit time. If YES is specified for **CONTSTOR**, DB2 will examine each thread at every commit to see if the thread has exceeded an internal threshold for a set number of commits or if the total size of the thread's storage pool has exceeded a second internal threshold. If either of these two conditions test true, storage blocks that are no longer in use are freed and returned to the operating system.

Specifying CONTSTOR=YES for subsystems that have many long-running persistent threads and constrained on storage in the DBM1 address space, can reduce the total storage used in the DBM1 address space.

MINSTOR

Macro = DSN6SPRM
Allowed values = YES, NO
Default value = NO (Default was set to YES for DB2 9)
Installation panel ID = DSNTIPE
Updatable via -SET SYSPARM = YES
Recommended value: NO

The **MINSTOR** subsystem parameter in a way is similar to **CONTSTOR** because it also helps manage thread storage. **MINSTOR** controls whether DB2 uses storage management algorithms to minimize the amount of working storage that is consumed by individual threads. When set to **N0** (the DB2 10 default), it uses a first fit algorithm, which can result in free space fragmentation. If **MINSTOR** is set to **YES**, a best fit algorithm is used instead. DB2 will follow all the chains across all segments searching for a “best fit”, resulting in denser storage. The CPU overhead is about 1%. However, the real danger of using **MINSTOR YES** may be that it can mask storage leaks, making them more difficult to debug while the storage leaks are causing slower performance.

Because the default was changed to **YES** in DB2 9 and then back to **N0** in DB2 10, it is mentioned here to make sure that it is set back to **N0**. It should be set to **YES** if the subsystem is experiencing storage issues that must be resolved. It could also be set to **YES** in those cases where the subsystem is fully tuned with epitomized storage and it has been determined that there are no leaks.

DSMAX

Macro = DSN6SPRM
 Allowed values = 1 to 100000
 Default value = 20000
 Installation panel ID = DSNTIPC
 Updatable via **-SET SYSPARM = YES**
 Recommended value: See Description

DSMAX determines the maximum number of data sets that can be open in DB2 at one time. At first installation, the initial value for this system parameter is calculated, although in many cases it still needs to be resized based on the actual database configuration implemented. When determining the number of possible open page sets, consider partitioned table spaces and indexes, LOBs, XML, and indexes with small **PIECESIZE**. DB2 calculated **DSMAX** value at installation does not account for the additional data sets that will exist when using the previous listed page sets.

Because a data warehouse environment could consist of a very large number of page sets because of partitioning and indexing, **DSMAX** is included here for two reasons.

First, open page sets use storage below the line, so having a **DSMAX** value that approaches 100,000 may not be realistic. Lowering the value of **DSMAX** will reduce the below the line storage requirement.

Next, too low of a value for **DSMAX** could cause excessive page set open/close page set processing, which can have a negative effect on the overall performance of DB2. The deferred close process, the closing, and de-allocating of open page sets by DB2, is affected by **DSMAX** because the thresholds used for deferred close are percentages. If 99% of **DSMAX** is reached, DB2 will perform an asynchronous physical VSAM close of 3% of the open data sets. DB2 will first close any **CLOSE=YES** page sets on LRU chain. If the 3% is not reached, DB2 will then close any **CLOSE=NO** page sets on LRU chain. Excessive data set open/close activity will result in high DBM1 task control block (TCB) time.

C.1.5 Star join processing

The following parameters relate to star join processing:

- ▶ **STARJOIN**
- ▶ **SJTABLES**
- ▶ **EN_PJSJ**

STARJOIN

Macro = DSN6SPRM
Allowed values = DISABLE, ENABLE, 1 to 32768
Default value = DISABLE
Installation panel ID = DSNTIP8
Updatable via -SET SYSPARM = YES
Recommended value: DISABLE (unless star schemas are in use)

The **STARJOIN** keyword on the **DSN6SPRM** macro or STAR JOIN QUERIES field on the DSNTIP8 installation panel enables or disables star join processing for the DB2 subsystem. It is suggested that you start out with the default value DISABLE until you know that you are going to use star schemas. If star schemas are in use and star join processing is going to be used, this keyword must be set to ENABLE.

If you specify a numeric value to enable star join processing, a “1” indicates that the fact table will be the largest table in a star join query. A value of 2 - 32768 means that DB2 should use the ratio of the star join table and the largest dimension table.

SJTABLES

Macro = DSN6SPRM
Allowed values = 0 to 32767
Default value = 10
Installation panel ID = None
Updatable via -SET SYSPARM = YES
Recommended value: 5 (if DW only environment)

The DSNZPARM keyword **SJTABLES** has meaning *only* when star join processing is enabled (see STARJOIN above). It is used to set the number of tables that must be contained in a query block before star join processing can be considered:

0	Signifies to use the default value of 10 or more tables.
1, 2, 3	Star join will always be considered.
4 - 225	Star joins are considered only when the query block has that number of tables.
226 - 32767	Star join will never be considered.

If this DB2 subsystem is in support of a data warehouse-only environment, enable star join processing with a starting value of SJTABLES = 5 and STARJOIN = ENABLE (or a value of 1 - 32768). If this DB2 subsystem could be used for a mixed workload (DW and OLTP for example), use a higher value for SJTABLES like the default of 10 as the starting point. SJTABLES is used to lower the number of tables necessary for DB2 to determine if the query should participate in star join processing.

STARJOIN and **SJTABLES** are both DB2 system-wide system parameters. To gain more granularity, manage them by using the profile table, DSN_PROFILE_TABLE.

SJTABLES has NO meaning when STARJOIN = DISABLE.

EN_PJSJ

Macro = DSN6SPRM
Allowed values = OFF, ON
Default value = OFF
Installation panel ID = None
Updatable via -SET SYSPARM = YES
Recommended value: ON (if STARJOIN enabled)

The **EN_PJSJ** keyword on the **DSN6SPRM** macro has meaning only if star join processing is enabled (**STARJOIN + ENABLE**). When **EN_PJSJ** is set **ON**, pairwise join, or dynamic indexing **ANDing**, processing is enabled. Pairwise join can improve join processing performance for qualified queries.

C.1.6 Utility sort processing

DB2 also passes the value of **SORTNUM** to **DFSORT** to assist **DFSORT** in determining how many sort work data sets to allocate. It is strongly recommended that **SORTNUM** not be used and that you let **DFSORT** determine the number of sort work data sets that it needs. In addition to **SORTNUM**, there are also two DB2 **DSNZPARM** keys that should be considered when making the decision to include or exclude **SORTNUM**; **IGNSORTN**=**N0** and **UTSORTAL**=**YES**, both on the **DSN6SPRM** macro.

The following parameters relate to sort processing:

- ▶ **IGNSORTN**
- ▶ **UTSORTAL**

IGNSORTN

Macro = **DSN6SPRM**
Allowed values = **YES, NO**
Default value = **NO**
Installation panel ID = **DSNTIP61**
Updatable via **-SET SYSPARM = YES**
Recommended value: **NO**

IGNSORTN instructs DB2 to ignore the **SORTNUM** keyword, if specified. It is recommended that this keyword be left at its default, **N0**. If for some reason a need arises where **SORTNUM** must be specified to solve a specific sort problem, having this set to **YES** will prevent that. Leaving this keyword at **N0** always gives you the option of specifying **SORTNUM**, if necessary. However, as stated, it is highly recommended that you *not* use the **SORTNUM** keyword.

This value can be set using the system parameter **IGNSORTN** on the **DSN6SPRM** macro or via the **IGNORE SORTNUM STAT** field on installation panel, **DSNTIP61**.

UTSORTAL

Macro = **DSN6SPRM**
Allowed values = **YES, NO**
Default value = **YES**
Installation panel ID = **DSNTIP61**
Updatable via **-SET SYSPARM = YES**
Recommended value: **YES**

When **UTSORTAL** is set to **YES**, again the default, DB2 can use real-time statistics, now always available as of DB2 10, to determine the sort work data set sizes. Specifying **N0** forces you to either specify the sort work data sets in the **JCL** or pre-allocate the data sets. **UTSORTAL** should always be set to its default, **YES**. It is highly recommended that this keyword be allowed to take its default value, **YES**.

This value can be set using the system parameter **UTSORTAL** on the **DSN6SPRM** macro or via the **UT SORT DATA SET ALLOCATION** field on installation panel, **DSNTIP61**.

C.1.7 Query Accelerator

The following four DSNZPARM keywords are only meaningful if the DB2 Analytics Accelerator for z/OS is installed. Before changing any of these keywords, it should first be discussed with IBM:

- ▶ ACCEL
- ▶ QUERY_ACCELERATION
- ▶ QUERY_ACCEL_OPTIONS
- ▶ GET_ACCEL_ARCHIVE

ACCEL

Macro = DSN6SPRM
Default value = NO
Allowed values = YES, NO, COMMAND
Updatable via **-SET SYSPARM = NO**
Installation panel ID = None
Recommended value: If Accelerator installed: AUTO; NOT installed: NO

The DSNZPARM keyword **ACCEL** controls whether the DB2 subsystem can access the DB2 Analytics Accelerator, and how.

If **ACCEL** is set to NO, the use of the Accelerator is not allowed. This is the default. If this option is selected, Accelerator processing cannot be started.

When **ACCEL** is set to AUTO, the Accelerator is automatically enabled and started when the DB2 subsystem is started.

If **ACCEL** is set to COMMAND, Accelerator processing is allowed and the Accelerator can be started using the DB2 **-START ACCEL (accelerator_name)...**

QUERY_ACCELERATION

Macro = DSN6SPRM
Default value = NONE
Allowed values (v2) = NONE, ENABLE, and ENABLE_WITH_FALLBACK
Allowed values (v3) = NONE, ENABLE, ENABLE_WITH_FALLBACK, ELIGIBLE, and ALL
Installation panel ID = None
Updatable via **-SET SYSPARM = YES**
Recommended value: ENABLE

The **QUERY_ACCELERATION** keyword on the **DSN6SPRM** macro defines the default value for the CURRENT QUERY ACCELERATION special register. The QUERY_ACCELERATION default is used when the SET CURRENT QUERY ACCELERATION SQL statement has not been issued. Valid values for the SET statement and for the DSNZPARM keyword are:

NONE No query acceleration takes place. (default)

ENABLE A query is accelerated only if DB2 determines that there is an advantage to running the query on the Accelerator. Costing and heuristics contribute to the decision of DB2 to send to the Accelerator.

If there is an Accelerator failure while running the query, or the Accelerator returns an error, DB2 returns a negative SQL code to the application.

ENABLE_WITH_FAILBACK

This option is similar to ENABLE with a slight change to DB2's behavior for the cases when there is an Accelerator failure or when the Accelerator returns an error.

If the Accelerator returns an error during the PREPARE or first OPEN for the query, DB2 executes the query without the Accelerator. If the Accelerator returns an error during a FETCH or a subsequent OPEN, DB2 returns the error to the user, and does not execute the query.

ELIGIBLE

Indicates that any given query will be accelerated if it is eligible for acceleration without heuristics and costing. Queries that are not eligible will be executed in DB2. Costing and heuristics are not considered.

If there is an Accelerator failure while running the query, or the Accelerator returns an error, DB2 returns a negative SQL code to the application.

ALL

Indicates that any given query will be accelerated if it is eligible for acceleration without heuristics and costing. Queries that are not eligible will not be executed in DB2, but will raise an SQL error instead. There is no failback with this option.

Note: ELIGIBLE and ALL are available for the DB2 Analytics Accelerator for z/OS V3 and above only.

QUERY_ACCEL_OPTIONS

Macro = DSN6SPRM
Default value = NONE
Allowed values = NONE, 1, 2, 3, 98, and 99
Updatable via **-SET SYSPARM = YES**
Installation panel ID = None
Recommended value: NONE

The **QUERY_ACCEL_OPTIONS** keyword on the **DSN6SPRM** macro specifies additional types of SQL queries to be included in query routing. Valid options are:

NONE

Query routing is restricted to the standard SQL statements. (default)

1

The queries that include data encoded by multi-byte character set EBCDIC encoding scheme are not blocked from executing on the Accelerator, although the Accelerator encodes the same data in the UTF-8 Unicode encoding scheme. EBCDIC and Unicode implement different collating sequences. The collating sequence for Unicode is numeric, uppercase characters, and then lower case characters (1, 2, 3, A, B, C, a, b, c). In EBCDIC, the collating sequence is lower case, upper case, and then numeric (a, b, c, A, B, C, 1, 2, 3). There are also differences in collating for the national characters. This affects both data ordering and the results from range predicates. Therefore, if the tables include character columns where more than one of these groups can be found in the column values, and the SQL statements include range predicates or ordering on these columns, a query executed in DB2 might return a different result set than the same query executed in the Accelerator.

- 2 For the queries that include an INSERT FROM SELECT statement, the SELECT part is not blocked from executing on the Accelerator, although the data operated on by SELECT might not be current in the Accelerator.
- 3 The queries that include DB2 byte-based string functions on data encoded by multi-byte character sets encoding schemes (like Unicode) are not blocked from executing on the Accelerator, although the Accelerator supports only character-based string functions. If the data on which the string function is specified contains only single-byte characters, executing the function on the Accelerator will return the same result as executing the function on DB2 regardless of what encoding scheme is used for the data. However, if the data contains multi-byte characters, the results will not be the same.

GET_ACCEL_ARCHIVE

Macro = DSN6SPRM
 Default value = NO
 Allowed values = NO, YES
 Updatable via -SET SYSPARM = YES
 Installation panel ID = None
 Recommended value: NO

GET_ACCEL_ARCHIVE specifies the default setting of the CURRENT GET_ACCEL_ARCHIVE special register.

If the keyword is set to NO, when a table is archived in the Accelerator and a reference is made to that table, the query does not use the data in the archived table.

However, if the keyword is set to YES and the data is archived in the Accelerator, when a query references that data the query can use the archived data.

C.1.8 LOB specific keywords

DB2 has two DSNZPARM keywords that are used to size the storage used for large object (LOB) values: One keyword for storage per user (**LOBVALA**), and one keyword for the storage use by the entire DB2 subsystem (**LOBVALS**):

- ▶ LOBVALA
- ▶ LOBVALS

LOBVALA

Macro = DSN6SYSP
 Default value = 10240
 Allowed values = 1 to 2097152 (value is in kilobytes)
 Updatable via -SET SYSPARM = YES
 Installation panel ID = DSNTIPD
 Recommended value: 51200

LOBVALA on the **DSN6SYSP** macro is the maximum amount of memory in kilobytes that a user (or agent) can use for storing LOB values.

A suggested starting point for LOBVALA could be 50 MB. That value should be monitored and adjusted as query workload is developed.

LOBVALS

Macro = DSN6SYSP
Default value = 2048
Allowed values = 1 to 51200 (value is in megabytes)
Updatable via **-SET SYSPARM = YES**
Installation panel ID = DSNTIPD
Recommended value: 10240

LOBVALS on the **DSN6SYSP** macro is the amount of storage that a DB2 subsystem can use for storing LOB values. Note that LOBVALS is in megabytes (MB), not kilobytes.

A suggested starting point for LOBVALS could be 10 GB. That value should be monitored and adjusted as query workload is developed.

C.1.9 XML specific keywords

DB2 has two DSNZPARM keywords that are used to size the storage used for Extensible Markup Language (XML) values: One keyword for storage per agents (**XMLVALA**), and one keyword for the DB2 subsystem (**XMLVALS**):

- ▶ XMLVALA
- ▶ XMLVALS

XMLVALA

Macro = DSN6SYSP
Default value = 204800
Allowed values = 1 to 2097152 (value is in kilobytes)
Updatable via **-SET SYSPARM = YES**
Installation panel ID = DSNTIPD
Recommended value: See description

XMLVALA on the **DSN6SYSP** macro is the maximum amount of memory in kilobytes that a user (or agent) can use for storing XML values.

For constructing an XML document, consider setting XMLVALA to at least twice the maximum length of the document being generated. If querying XML data, the starting point for XMLVALA should be at least four times the maximum document size, for either stored or generated documents. These are suggested starting points that should be monitored and adjusted as needed.

XMLVALS

Macro = DSN6SYSP
Default value = 10240
Allowed values = 1 to 51200 (value is in megabytes)
Updatable via **-SET SYSPARM = YES**
Installation panel ID = DSNTIPD
Recommended value: See description

XMLVALS on the **DSN6SYSP** macro is the amount of storage that a DB2 subsystem can use for storing XML values. Note that XMLVALS is in megabytes (MB), not kilobytes.

Consider setting XMLVALS to the maximum size allowable by the system memory that is available. A possible starting point for XMLVALS is to take the maximum number of XML threads multiplied by either 2 GB or the value specified by XMLVALA:

(max # of XML threads x 2GB or XMLVALA)

There are metrics available in the DB2 Statistics and DB2 Accounting records that report the maximum XML storage used by the DB2 subsystem or the XML agent (user).

C.1.10 Miscellaneous system parameters

This section contains “other” DSNZPARM keywords that might be of interest, but do not fit into any of the other previous categories.

LRDRTHLD

Macro = DSN6SPRM
Default value = 10
Allowed values = 0 to 1439 (minutes)
Updatable via **-SET SYSPARM = YES**
Installation panel ID = DSNTIPE
Recommended value: 0

LRDRTHLD on the **DSN6SPRM** macro or LONG-RUNNING READER field on the installation panel, DSNTIPE, specifies in minutes hold that a read claim can be held. If this time is exceeded, a warning message is issued and a trace record created to report the long running query.

LRDRTHLD is mentioned here for two reasons:

- ▶ The default was changed from 0 (zero), which disables the long running query reporting to 10 minutes, meaning if a read claim is held for more than 10 minutes, a warning message and trace record is created.
- ▶ In a data warehouse environment, long running queries are usually expected. Setting this to 0 (zero) and disabling the warnings prevents being overwhelmed by warning messages.

Metrics: SMFACCT, SMFSTAT, STATIME, and SYNCVAL

These four DSNZPARM keywords, all on the DSN6SYSP macro, have something to do with gathering metrics about the DB2 subsystem or the applications and SQL running in a DB2 subsystem. They should also all be “turned on”. If a performance issue arises, having the information made available by activating these system parameters will be invaluable. Not having them active might mean that a problem will have to continue until adequate metrics are gathered for use in problem determination. All four keywords are initially defined on the DSNTIPN installation panel.

Only the keywords and their suggested values are listed here.

SMFACCT=(1,2,3,7,8)

SMFACCT specifies whether DB2 is to automatically start the listed classes when DB2 is started and send accounting data to SMF. The classes specified on this keyword are usually used in the diagnostics of an application or SQL performance issue.

Although this keyword cannot be modified via the **-SET SYSPARM** command, the started classes can be started and stopped with the **-START TRACE** and **-STOP TRACE** commands.

SMFSTAT=(1,3,4,5,6)

SMFSTAT specifies whether DB2 is to automatically start the listed classes when DB2 is started and send statistical data to SMF. The classes specified on this keyword are usually used in the diagnostics of a system performance issue.

Although this keyword cannot be modified via the **-SET SYSPARM** command, the started classes can be started and stopped with the **-START TRACE** and **-STOP TRACE** commands.

STATIME=1

STATIME specifies the interval in minutes for gathering statistics. Instrumentation facility component identifiers (IFCIDs) are written at the end of the interval. This keyword can be modified using the **-SET SYSPARM** command; however, when set to **1** there is no need to change it.

STATIME has changed in definition as of DB2 10. Not only is the default now 1 minute and also the suggested value, but this value *only* applies to IFCIDs 0105, 0106, 0199, and 0365.

SYNCVAL=0

SYNCVAL specifies whether stats recording should be coordinated with some part of the hour. This is almost essential in a data sharing environment to match trace records across members. This keyword can be modified using the **-SET SYSPARM** command; however, when set to **1** there is no need to change it.

SYNCVAL, like **STATIME**, has changed in definition as of DB2 10. This keyword *only* applies to IFCIDs 0105, 0106, 0199, and 0365.

Accounting Rollup: ACCUMACC and ACCUMUID

ACCUMACC and **ACCUMUID** manage whether or not DB2 will roll up accounting records. Both of these keywords are on the **DSN6SYSP** macro. They are both initially set on the **DSNTIPN** installation panel, and they can both be modified using the **-SET SYSPARM** command.

ACCUMACC=NO

ACCUMACC determines if accounting records should be accumulated based on the value of **ACCUMUID**. Setting **ACCUMACC** to a value between 2 - 65335 determines the number of accounting records in an interval between writes. Setting this keyword to **NO** causes accounting data to be written every time that a distributed data facility (DDF) thread goes in active or for each **RRSAF** thread signon. The tradeoff is granularity, one record per DDF thread occurrence, versus a reduction in the number of accounting records being written.

ACCUMUID=0

ACCUMUID determines the aggregation fields used for rollup. The suggested value for this keyword in this paper is 0 (zero) for rollup on: end user ID, application or transaction name, and workstation name. However, any criteria that you decide on is OK. The 18 values (0 - 17) are described in detail in the *DB2 10 Installation and Migration Guide*, GC19-2974.

Managing temp storage in work files

As of DB2 9, temp work files (declared *global temporary tables*) and sort work files (RDS sort) were combined using only one set of work files. There are long explanations in other places that describe what that means and the affects this can have on DB2. For now, and here, only the two system parameters will be described, and only to help prevent potential future problems.

The two **DSNZPARM** keywords that are affecting how this all works are **MAXTEMPS** and **WFDBSEP**.

MAXTEMPS = 0

The **MAXTEMPS** system parameter on the **DSN6SPRM** macro (MAX TEMP STG/AGENT field on the **DSNTIP9** installation panel) allows the definition of the maximum amount of temporary storage that can be used in the work files database, regardless of how it will be used. The number that is specified is assumed to be in megabytes (MB) unless the number is followed by a G (for gigabytes), then the number is in GB. A value of 0 (zero) means that no limit is enforced. 0 (zero) is the default.

WFDBSEP = NO

System parameter **WFDBSEP** on the **DSN6SPRM** macro (or SEPARATE WORK FILES field on the DSNTIP9 install panel) defines to DB2 whether sort and DGTT must be separated:

- | | |
|------------|--|
| YES | DB2 will always enforce the preferred separation described above. However, if the preferred table space type is not available, the task gets a negative return code and fails. |
| NO | Use the preferred separation. However, if the preferred table space type is not available, DB2 will use whatever work file table space that is available. NO is the default. It is also the suggested choice when <i>not</i> using declared global temporary tables. NO will cause the least number of problems should a DGTT get created or if work file table spaces get defined with secondary extents. |

The important difference between the two options is that NO will allow failover to a preferred table space type; whereas, YES will cause a failure should that incorrect table space choice be encountered.

C.1.11 Deprecated in DB2 10

There are five DSNZPARM keywords, **OPHYBCST**, **OJPERFEH**, **OPTIOWGT**, **OPTIXIO**, and **PTCDIO**, which are often mentioned in discussions of improved query performance. These system parameters are all deprecated in DB2 10 and planned for removal in a future release of DB2. The DB2 behavior enabled by taking their current default value is the behavior that will remain in DB2 going forward. It is recommended that these five DSNZPARM keywords be left at their default. It should be verified that a value other than what is described here is not hard coded in an older DB2 subsystem.

OJPERFEH on the **DSN6SPRM** macro by default (YES) enables a couple of performance enhancements in outer joins; overriding the default by specifying NO disabled these enhancements. This keyword in almost *all* cases should be set to YES. This opaque parameter was first introduced using a hidden ZPARM back in DB2 Version 5, and later updated to an opaque ZPARM. APARs PQ29780 and PQ48485 have additional details.

OPTIOWGT on macro **DSN6SPRM** enables (the default was changed to ENABLE by APAR PK75643) support for an improved cost estimation for balancing the improvements in I/O response and disk caching with CPU speeds. This support was added in DB2 9 via APAR PK61277. See the two APARs, PK61277 and PK75643, for additional details. This keyword should be left at its default ENABLE.

OPTIXIO on macro **DSN6SPRM** is an opaque parameter that provides a more stable I/O costing formula with significantly less sensitivity to buffer pool and object size when the current default (ON) is used. This function was delivered back in DB2 Version 8 via APAR PK12803, and the default was changed to ON with APAR PK26613.

PTCDIO is another opaque parameter on macro **DSN6SPRM**. It is a switch to turn off an index costing change made by APAR PQ86763 way back in DB2 Version 7. The actual ZPARM parameter was also added in DB2 Version 7 via APAR PQ97866 with a default of OFF and the recommendation to *not* turn it on without IBM support's direction. Leave this keyword set to its default value, OFF.

C.1.12 Pre DB2 9 DSNZPARM parameters removed, but still documented

There are a couple of DSNZPARM keywords that you might still come across if you happen to get a hold of an older presentation, book, white paper, and so on. They are listed below along with the APAR that introduced them to DB2:

OPTIXOPREF (PK68986, PK51734, PK77426)
OPTOPSE
OPTCCOS4 (PK26760)
OPTX0IRC (PK30857)
OPHYBCST* (PK90334)
PARAPAR1 (PQ87352)
SJMISSKY (PK06964)
SJMXPOOL * (V8 Install Guide)
TABLES_JOINED_THRESHOLD (PK33532)
MAX_OPT_ELAP

These system parameters were removed from DB2 with the introduction of DB2 9 or DB2 10. Whatever functionality that they enabled has become part of the DB2 9 or DB2 10 product.



D

Tools for Accelerator

This appendix briefly describes what DB2 tools are available for the IBM DB2 Analytics Accelerator.

The following sections are covered:

- ▶ Tools for Accelerator
- ▶ OMEGAMON XE for DB2 Performance Expert on z/OS
- ▶ DB2 Query Monitor for z/OS
- ▶ InfoSphere Optim Query Workload Tuner for DB2 for z/OS
- ▶ InfoSphere Optim Configuration Manager

D.1 Tools

The IBM DB2 Analytics Accelerator can leverage powerful DB2 tool capabilities to further maximize Accelerator investment. With additional tools you should be able to:

- ▶ Proactively monitor and manage your accelerated queries
- ▶ Validate performance ROI of accelerated queries
- ▶ Streamline query candidate selection
- ▶ Filter out unaccelerated queries to maximize workload tuning efforts
- ▶ Administer and manage your Accelerator

Figure D-1 shows a high-level overview of three IBM DB2 for z/OS performance tools that are available for IBM DB2 Analytics Accelerator.

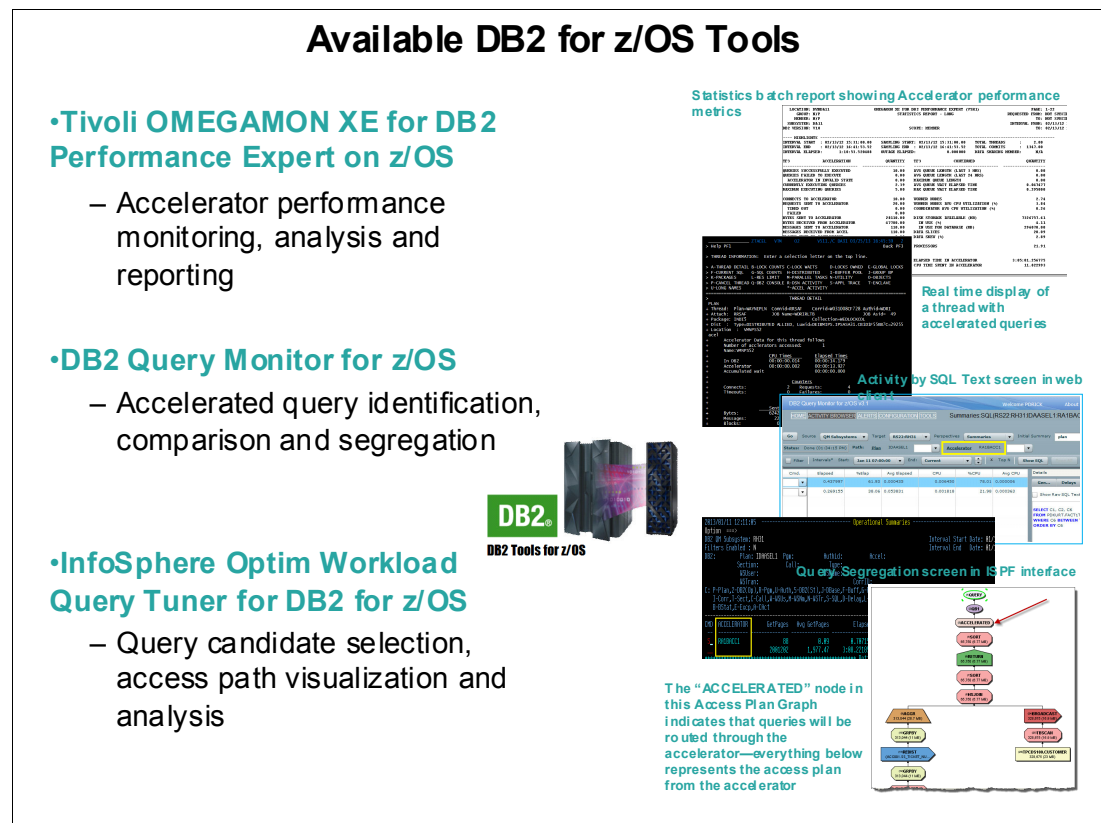


Figure D-1 Available DB2 for z/OS tools for IBM Analytics Accelerator

In the next sections, we briefly describe the three performance tools and a configuration tool:

- ▶ OMEGAMON XE for DB2 Performance Expert on z/OS
- ▶ DB2 Query Monitor for z/OS
- ▶ InfoSphere Optim Query Workload Tuner for DB2 for z/OS
- ▶ InfoSphere Optim Configuration Manager

D.2 OMEGAMON XE for DB2 Performance Expert on z/OS

Just as with any operational system in your IT organization, to get the most out of IBM DB2 Analytics Accelerator, you will want to monitor and report on its performance on an ongoing basis. The industry-leading systems monitor for DB2 for z/OS is the IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS. Beyond real-time and short-term monitoring capabilities, OMEGAMON XE for DB2 unmatched reporting capabilities provides multitudes of real time and batch reports to show how your Accelerator is performing. This immediate feedback dramatically increases your understanding of the value that the DB2 Analytics Accelerator provides to your organization, and validates your return on investment. This knowledge has other tangible benefits such as performance trend identification, ongoing appliance utilization, and charge back accounting.

D.3 DB2 Query Monitor for z/OS

DB2 Query Monitor for z/OS can be a powerful tool to help monitor, compare, and filter through the accelerated queries on your DB2 Analytics Accelerator. Use Query Monitor to identify which queries in your applications are being accelerated to show that your applications are utilizing the DB2 Analytics Accelerator.

Use Query Monitor to measure the performance of the same query with and without acceleration to provide valuable return-on-investment information clearly showing the value of the Accelerator implementation in your organization.

Finally, utilize Query Monitor to sort through DB2 queries and quickly segregate those that are not accelerated. This will greatly assist you in targeting the appropriate queries and workloads for your tuning efforts.

D.4 InfoSphere Optim Query Workload Tuner for DB2 for z/OS

InfoSphere Optim Query Workload Tuner for DB2 for z/OS assists in the selection, tuning, and access plan analysis of accelerated query workloads. As you might know, not every DB2 query is a good candidate for acceleration.

By using Optim Query Workload Tuner, you can quickly identify potential candidates without manually analyzing each query. Among Optim Query Workload Tuner's strengths is the ability to tune complete workloads by taking the entire workload into account. With a mix of accelerated and unaccelerated queries in your DB2 for z/OS environment, Optim Query Workload Tuner's advisors will sort through the workload and continue to offer the same expert advice.

You can use the same Visual Explain feature to view access plans, irrespective of where the query is executed. You continue to use the same tool with the same workflow. You also have the option of installing the IBM DB2 Analytics Accelerator Studio on top of Optim Query Workload Tuner. With this combination, you have an even more powerful tool, which is one tool to perform your tuning and the administration of the IBM DB2 Analytics Accelerator.

D.5 InfoSphere Optim Configuration Manager

To take advantage of some features of DB2, it is advantageous to set special registers within the application rather than to set it at a DSNZPARM level. This allows you to control the granularity of the feature or to test the feature before making it available to all applications.

By using Optim Configuration Manager, you can set special registers without needing to make any application change. For example, you can confirm if an individual application can benefit from using the Accelerator appliance, using Optim Configuration Manager to set the special register QUERY_ACCELERATION dynamically without amending the programs. This change does not affect any other application that is executing, and uses the DB2 tables and not the appliance.



Additional material

This book refers to additional material that can be downloaded from the Internet as described in the following sections.

Locating the Web material

The Web material associated with this book is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser at:

<ftp://www.redbooks.ibm.com/redbooks/SG248151>

Alternatively, you can go to the IBM Redbooks website at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG24-8151.

Using the Web material

The additional Web material that accompanies this book includes the file, sg248151.zip, which includes the following files that are used in Chapter 7, “Incremental update” on page 161:

<i>File name</i>	<i>Description</i>
REXXProgram.zip	Sample REXX program that calls several DB2 Analytics Accelerator stored procedures.
REXXJCL.zip	Sample JCL to run REXXProgram as a batch job.
LATMONRX.zip	Sample REXX program to obtain replication information and insert into a DB2 table. This program can be used to gather historical information on the performance of the replication process.
LATMONDDL.zip	Sample DDL for the DB2 table used by the REXX program, LATMONRX.

LATMONJ.zip

Sample JCL to run LATMONRX as a batch job. This job can be incorporated into a job scheduler to automatically run at an interval of one minute or greater.

System requirements for downloading the Web material

The Web material requires the following system configuration:

Hard disk space:	2 MB minimum
Operating System:	Windows
Processor:	Intel 386 or higher
Memory:	16 MB

Downloading and extracting the Web material

Create a subdirectory (folder) on your workstation, and extract the contents of the web material .zip file into this folder.

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

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- ▶ *Optimizing DB2 Queries with IBM DB2 Analytics Accelerator for z/OS*, SG24-8005
- ▶ *Co-locating Transactional and Data Warehouse Workloads on System z*, SG24-7726
- ▶ *IBM z/OS Automatic Restart Manager*, REDP-1073
- ▶ *Smarter Business Dynamic Information with IBM InfoSphere Data Replication CDC*, SG24-7941
- ▶ *Implementing IBM InfoSphere Change Data Capture for DB2 z/OS V6.5*, REDP-4726

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- ▶ *IBM DB2 Analytics Accelerator for z/OS Version 3.1.0 Installation Guide*, SH12-6983
- ▶ *IBM DB2 Analytics Accelerator for z/OS Version 3.1.0 Stored Procedures Reference*, SH12-6984
- ▶ *IBM DB2 Analytics Accelerator for z/OS Version 3.1.0 User's Guide*, SH12-6985
- ▶ *IBM DB2 Analytics Accelerator for z/OS Version 3.1.0 Getting Started*, SH12-6986
- ▶ *IBM DB2 10 for z/OS Command Reference*, SC19-2972
- ▶ *IBM DB2 10 for z/OS Installation and Migration Guide*, SC19-2974
- ▶ *IBM DB2 10 for z/OS Managing Performance*, SC19-2978
- ▶ *IBM DB2 10 for z/OS SQL Reference*, SC19-2983
- ▶ *IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS Report Reference Version 5.1.0*, SH12-6921
- ▶ *IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS Version 5.1.1 Configuration and Customization*, GH12-6970

Online resources

These websites are also relevant as further information sources:

- ▶ IBM DB2 Analytics Accelerator support home
http://www-947.ibm.com/support/entry/portal/overview//software/information_management/db2_analytics_accelerator_for_z~os
- ▶ IBM PureData System
<http://www.ibm.com/software/data/puredata/analytics>
- ▶ *Paper: Enabling query acceleration with IBM DB2 Analytics Accelerator for ODBC and JDBC applications without modifying the applications*
<http://www.ibm.com/support/docview.wss?uid=swg27038078>
- ▶ Data Studio V3.2 installation
<http://www.ibm.com/support/docview.wss?uid=swg24033663>
- ▶ Near Real-Time Analytics with IBM DB2 Analytics Accelerator
<http://www.edbt.org/Proceedings/2013-Genova/papers/edbt/a52-martin.pdf>
- ▶ IBM DB2 Analytics Accelerator product documentation
<http://www.ibm.com/support/docview.wss?uid=swg27036909>
- ▶ IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS Support home page:
http://www.ibm.com/support/entry/portal/Overview/Software/Tivoli/Tivoli_OMEGAMON_XE_for_DB2_on_z~OS
- ▶ Netezza Developer Network
<https://www.ibm.com/developerworks/mydeveloperworks/groups/service/html/communityview?communityUuid=35ac05e2-4e00-42fe-b252-111e5f3ad8fa>
- ▶ IBM zEnterprise Analytics System 9700
<http://www.ibm.com/software/data/infosphere/zenterprise-analytics-system/9700>
- ▶ IBM zEnterprise Analytics System 9710
<http://www.ibm.com/software/data/infosphere/zenterprise-analytics-system/9710>
- ▶ Rapid SAP NetWeaver BW ad-hoc Reporting Supported by IBM DB2 Analytics Accelerator for z/OS
<http://scn.sap.com/docs/DOC-19436>
- ▶ IBM z/OS Management Facility
<http://www.ibm.com/systems/z/os/zos/zosmf>
- ▶ IBM Whitepaper: *Possible network configurations*
<http://www.ibm.com/support/docview.wss?uid=swg27028171>
- ▶ Structure of DSNX881I Messages
<http://www.ibm.com/support/docview.wss?uid=swg27037905>
- ▶ IBM InfoSphere Data Replication version 10.2 Information Center
http://pic.dhe.ibm.com/infocenter/iidr/v10r2m0/index.jsp?topic=%2Fcom.ibm.cdcdoc.homepage.doc%2Fic-homepage_iidr.html
- ▶ IBM technote: Synchronizing data in IBM DB2 Analytics Accelerator for z/OS
<http://www.ibm.com/support/docview.wss?uid=swg27038501>

- ▶ IBM white paper: Near Real-Time Analytics with IBM DB2 Analytics Accelerator
<http://www.edbt.org/Proceedings/2013-Genova/papers/edbt/a52-martin.pdf>
- ▶ IBM technote: How IBM DB2 Analytics Accelerator for z/OS handles correlated subqueries
<http://www.ibm.com/support/docview.wss?uid=swg27037928>
- ▶ IBM technote: Preventing outages of the incremental update process
<http://www.ibm.com/support/docview.wss?uid=swg27039147>
- ▶ IBM techdoc: *Structure of DSNX881I Messages*
<http://www.ibm.com/support/docview.wss?uid=swg27037905>
- ▶ IBM techdoc: *High Availability Setup for the Incremental Update Capture Engine*
<http://www.ibm.com/support/docview.wss?uid=swg27037912>

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

A

- AC 99
- ACCEL 123
- ACCEL GET_TABLES_DETAILS 159
- ACCEL GET_TABLES_INFO 159
- ACCEL_ADD_TABLES stored procedure 146
- ACCEL_ARCHIVE_TABLES stored procedure 156
- ACCEL_LOAD_TABLES 116
- ACCEL_LOAD_TABLES stored procedure 59
- ACCEL_SET_TABLES_ACCELERATION 41, 80
- Accelerator wizard 32
- accelerator_name 229, 330
- ACCELMODEL 313
- access 6, 12, 55, 107, 122, 144–145, 161, 246, 270, 292, 317
- access path 36, 136, 317
- access plan diagram 136
- ADDTABLES 80
- administration xx, 11, 159, 164, 190, 269
- Administration Explorer 30
- aggregate functions 131
- aggregation 200, 335
- ALTER BUFFERPOOL 211
- ALTER TABLE 63
- analytics 143
- APAR 65, 240, 308, 318
- APARs 66, 112, 308, 336
- APIs 264
- application 3, 8, 16, 56, 96, 122, 145, 148, 169, 208, 225, 268, 285, 321
- application server 283
- AQT_MAX_UNLOAD_IN_PARALLEL 40, 60
- AQT10000I 85, 232
- AQTSCALL 80, 229
- AQTSCALL program 80
- AQTSJI03 80, 229
- architectural overview 267
- architecture 2, 8, 14, 163, 189, 267
- argument 130–131
- attribute 49, 140, 204
- authentication token 50
- availability
 - data warehouse 10

B

- base table 324
- batch 6, 55, 105, 156, 161, 208, 229, 274, 309, 343
 - update 55–56, 161
- Batch reporting 310
- BIGINT 130
- BLOB 76
- buffer pool 239, 323
 - activity 95, 324
- buffer pools 95

- BUFSIZE 170
- BUFTHRESHOLD 170
- business
 - challenges 6
 - metadata 283
 - process 4, 8
 - processes 2, 8, 162, 280
 - questions 268
 - reports 1, 282
 - scenario 3, 279
 - users 97

C

- CANCEL THREAD 241
- catalog table 50, 65, 166, 222
- CCSID 129
- Change Data Capture 162
- CHCCFGxx 171
- CHCDBMxx 170–171
- CHECK DATA 270
- CICS 3, 214
- class 90, 208, 212, 268
- classification rules 96
- CLI 98
- CLOB 76
- Cognos 7, 9, 198, 272
- Cognos BI 10, 279
- column value 137
- columns 5, 27, 64, 125, 128, 174, 190, 215, 255, 325
- Command Line Processor 261
- complex queries xix, 110, 144, 201, 268
- components 15, 66, 113, 162, 187, 267, 285, 308
- compression 190, 269
- concurrency
 - level 224
- condition 133, 227, 290
- CONNECT 214, 216
- connection profile 26
- COPY 65, 154, 270
- COUNT 76, 135
- CPU time 111, 210, 215, 322
- CREATOR 76
- CS 117, 294
- current data 6, 282
- CURRENT GET_ACCEL_ARCHIVE 148
- CURRENT QUERY ACCELERATION special register 12, 123, 134, 330
- CURRENT QUERY ACCELERATION 123
- CURRENT TIMESTAMP 130

D

- dashboards 1, 162
- data xx, 1, 8, 11, 55, 90, 126, 128, 143, 161, 187, 209, 268, 285, 310, 315

- access 264, 270
- characteristics 3, 56, 158
- consolidation 7, 273
- current 1, 9, 125, 332
- filter 191
- frequency 6, 56
- integrity 5, 149, 282
- operational 3, 9, 145, 162, 273
- processing 2, 9, 58, 145, 162, 190, 273, 320
- quality 66, 273
- requirements 2, 9, 56, 143, 189, 273
- transfer 66, 154, 274
- types 5, 67, 130, 149, 191, 270
- DATA CAPTURE CHANGES 170
- data mart 6
- data server 12, 90, 268
- data set 60, 95–96, 148, 171, 229, 280, 293, 327
- data sharing 8, 27, 170, 219, 227, 316
 - group 35, 170, 292
- data source 6, 256
 - DB2 256
- Data type 138–139
- data warehouse 2, 9, 79, 114, 179–180, 187, 189, 267, 274, 315
 - database 6, 274
 - environment 5, 327
 - queries 318
- Data Warehousing xx
- database management
 - system 117, 189
- DATE 37, 125, 132, 313
- DB2 xix, 6, 9, 11, 55, 87, 121, 161, 187, 207–208, 268, 285, 307, 315, 343
 - functionality 124, 128–129, 164, 337
 - instance 48, 224, 292
 - member 80, 219, 227, 316
 - optimizer 127, 129, 157–158, 322
 - tools xix, 215, 269
- DB2 10 7, 31, 95, 126, 144, 198, 226, 268, 315–316
 - change 126
 - data 7, 329
 - SQL 321
 - use 7, 329
- DB2 9 xx, 207, 316, 319
 - behavior 319
 - DB2 10 321
 - system 321
 - use 319
- DB2 accounting 96, 212, 254
- DB2 Analytics Accelerator
 - DB2 commands 226
- DB2 commands 207
- DB2 data
 - sharing 148, 287
- DB2 database
 - processing 56
- DB2 optimizer 129, 157–158, 322
- DB2 statistics 208
- DB2 subsystem 25, 50, 88, 123, 164, 166, 208, 210, 215, 291, 293, 316–317

- DB2 system 210, 328
- DB2 Version 5 336
- DB2-supplied stored procedures 114
- DBAT 215, 217
- DBMS 145
- DD DISP 96, 253
- DD DSN 253
- DDF 88, 215, 275, 335
- DDF command 261
- DDF thread 335
- DDL 26, 166, 233, 343
- decimal 321
- default value 148, 157, 315
- DEGREE 132, 216, 317
- DELETE 63, 148, 164, 216, 254, 274
- delete 62, 172
- DFSMS 66, 187, 270
- diagnostic information 44
- dimension 95, 177, 183, 198, 204, 328
- DIS ACCEL DETAIL 222
- DISPLAY ACCEL 211, 222, 290
- DISPLAY ACCEL DETAIL 228, 291
- DISPLAY THREAD 99
- Distributed 215, 217
- DRDA 122, 209, 212
- DSN_QUERYINFO_TABLE 12, 138
- DSN6SPRM 138, 317
- DSNT408I 225
- DSNT408I SQLCODE 117
- DSNT415I SQLERRP 117, 225
- DSNT416I SQLERRD 117, 225
- DSNT418I SQLSTATE 117, 225
- DSNTESC 26
- DSNTIP8A 124
- DSNUTILU 48, 59, 114
- DSNX830I 222, 227
- DSNX88 287
- DSNX880I 223
- DSNX881I 220
- DSNZPARM 123, 148, 243, 315
 - ACCEL 330
 - ACCUMACC 335
 - ACCUMUID 335
 - CACHEDYN 320
 - CDSSRDEF 317
 - CONTSTOR 326
 - DECARTH 322
 - DECDIV3 321
 - DSMAX 327
 - DSVCI 323
 - EDMSTMTC 321
 - EN_PJSJ 329
 - GET_ACCEL_ARCHIVE 332
 - IGNSORTN 329
 - LOBVALA 332
 - LOBVALS 333
 - LRDRTHLD 334
 - MAXRBLK 325
 - MAXTEMPS 335
 - MAXTEMPS_RID 326

- MGEXTSZ 323
- MINSTOR 327
- MXDTCACH 324
- MXQBCE 322
- PARA_EFF 318
- PARAMDEG 318
- PTASKROL 320
- QUERY_ACCEL_OPTIONS 331
- QUERY_ACCELERATION 123, 330
- SJTABLES 328
- SMFACCT 334
- SMFSTAT 334
- SPRMPH 319
- SRTPOOL 324
- STARJOIN 328
- STATIME 335
- SYNCVAL 335
- UTSORTAL 329
- WFDBSEP 336
- XMLVALA 333
- XMLVALS 333
- dynamic SQL 122, 149, 317
 - caching 320
 - query 122
 - statement 317

E

- efficiency 189, 269, 318
- element 229
- enabling tables 41
- environment xix, 3, 10–11, 60, 92, 149, 162, 190, 207, 270, 286, 316
- error message 171, 219, 311
- ETL 2, 55, 162, 274
- event 220, 282
- EXEC SQL 270
- EXPLAIN 12, 138, 227, 317
- Explain 12, 150
- EXPLAIN output 138
- expression 125, 129, 158

F

- fact 7, 64, 95, 101, 149, 177, 195, 328
- FAILED QUERY REQUESTS 222
- failover 269, 285, 326
- failure 2, 123, 126, 225, 330
- FETCH 124, 126, 216, 331
- fetch 140
- FINAL TABLE 129
- flexibility 9, 84, 270
- forceFullReload 83
- framework 282
- function 6, 12, 117, 125, 144, 162, 219, 275, 292, 311, 332

G

- granularity 56, 95, 178, 328

H

- handle 7, 56, 115, 149, 190
- hardware 6, 26, 73, 94, 141, 149, 173, 187, 219, 267, 285
- high availability 285
- High Performance Storage Saver 143
- history 7, 32, 50, 91, 144
- HPSS 143

I

- I/O 9, 100, 164, 196, 217, 269, 317
- IBM Cognos Business Intelligence 279
- IBM DB2 Analytics Accelerator 7, 11, 56, 88, 121, 143, 162, 187, 219, 268, 286, 308
 - setup 292
- IBM DB2 Analytics Accelerator Studio 12, 139, 144, 149, 269
- IBM InfoSphere Data Replication 162
- IBM Netezza 1000 187, 269
- IBM System
 - z Integrated Information Processor 318
- IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS 208
- idaav3r1/k 308
- II14642 309
- IMS xx, 3, 241
- incremental update 55
- index xxi, 109, 149, 166, 275, 317
- Information Server 272
- infrastructure 1, 8, 66, 94, 258, 268
- input parameter 49, 234
- INSERT 63, 125, 148, 164, 216, 274, 332
- insert 67, 172, 232, 343
- INSERT from SELECT 141
- installation xix, 12, 56, 112, 126, 165, 240, 270, 293, 316
- IP address 30, 90, 167, 215, 222, 279, 292
- IPNAMES 222
- IRLM 118, 217
- IS 98, 239, 297

J

- Java 17, 80, 270

K

- KB 170, 197, 233, 325
- keyword 156, 293, 315

L

- latency 55, 113, 162, 234, 280
- LENGTH 131–132, 210
- levels 7, 25, 66, 97, 149, 187, 308
- Linux 7, 9, 15, 162, 272
- Linux on System z 279
- list 37, 48, 59, 92, 123, 168, 201, 220, 226, 309, 325
- list prefetch 325
- LOAD 63, 148, 155, 187, 249, 270, 323
- loading tables 50, 79
- LOADTABLES 80

LOB 239, 270, 332
 LOBs 327
 LOCATIONS 33, 222
 LOCK 216–217
 locking 37, 81, 310
 locks 39
 LOG 132, 217
 log 30, 56, 90, 162, 219, 271, 292
 logging 179
 LOGPOLLINTERVAL 170, 172
 lookup 268
 LPAR 219–220, 272, 274, 292
 z/OS 278, 292

M

M 98, 240
 maintenance 7, 32, 63, 96, 149, 164, 190, 208, 270, 293, 308, 315
 maximum number 68, 139, 210, 318
 MAXSUBSCRSTAGESIZE 171
 memory 163, 191, 272, 324
 MERGE 216
 message 17, 82, 91, 164, 215, 219, 227, 287, 311, 334
 XML 84, 236
 metadata 9–10, 59–60, 166, 169, 277, 292
 mixed workload 6, 9, 105, 268, 328
 model 74, 177, 189, 269
 MODIFY 211, 270, 295
 Monitor III 100
 MQTs 277
 MSTR address space 220, 291

N

name 16, 76, 92, 130, 151, 154, 167, 215, 279, 292, 335, 343
 NFM 245
 node 106, 262
 nodes 40, 101, 173, 199
 Not Accounted time 217
 NULL 82, 98, 135, 251
 NUMTCB 72, 114
 NUMTCB=1 114

O

OA41706 66, 79
 Object 33, 240
 ODBC 256
 ODBC and JDBC query acceleration 127
 OLTP 2, 9, 127, 143, 269, 328
 performance 3, 280
 OMEGAMON XE 95, 208, 214, 310
 OMPE classic interface 240
 ONUTILITYACTION=IGNORE 156
 operational data 5, 158–159, 162
 optimization 6, 9, 78, 157–158, 196, 270, 319
 optimizer 157, 317
 options 22, 56, 91, 124, 156, 167, 187, 226, 229, 272, 310, 331

ORDER 130, 216, 325
 ORDER BY 325
 organizing keys 27

P

pairing code 26, 91
 parallelism 37, 67, 200, 317
 PART 117, 151
 PARTITION 63
 partition 39, 56, 145–146, 169, 205, 220, 292
 Partitioning 37
 partitioning 76, 327
 partitions 12, 55, 143–144, 162, 199, 287, 318
 Performance 46, 95, 173, 187, 208, 275, 309, 325
 performance xix, 1, 8, 25, 56, 88, 122, 143, 170, 187, 207–208, 268, 311, 315, 343
 performance goals 114
 performance improvement 189, 269
 PM55637 309
 PM68928 310
 PM71744 65
 PM72264 124
 PM72265 124
 PM72274 158, 313
 PM72622 310
 PM72766 310
 PM73732 310
 PM80739 240
 PM85936 313
 PM87384 310
 PM87554 313
 PM88071 132, 313
 PM90151 66, 72
 PM90591 313
 PM90886 313
 PM93789 313
 PM94515 313
 PM95035 313
 PMR 310
 PORT 222, 294
 port number 32, 264, 294
 POSITION 131, 232
 predicate 126, 190
 prefix 219
 processors 73, 177, 191, 269, 285, 317
 production 3, 79, 106, 224, 275, 297
 PTF 113, 166, 310

Q

QMF 7, 234
 query 6, 26, 56, 121, 127, 143–144, 173, 187, 207–208, 212, 268, 299, 317
 performance 90, 145, 173, 191, 320
 response time 58, 102, 142, 184
 table 37, 58, 90, 126, 129, 144, 151, 173, 200
 Query Accelerator 95, 330
 query parallelism 200, 317
 query performance 66, 197, 270, 318
 QUERYNO 138

R

RACF 50, 271
range-partitioned table 63
Real 7, 310
real storage 95, 324
real time 6, 226, 274, 329
real-time statistics 64–65
REASON=00D351FF 225
Redbooks website 343, 345
 Contact us xxii
redundancy 285
REGION 253
remote location 222
remote server 225
REMOVETABLES 80
REORG 63, 90, 270, 323
replication 12, 84, 237, 292, 343
REPORT 65, 99, 209, 213, 294, 309
report classes 112
reports 2, 96, 177, 213, 220, 280, 310
repository 22, 283
requirements 16, 56, 114, 144, 244, 267, 309, 344
resource groups 90
response time 105, 141, 186, 197, 224
return 32, 80, 92, 125, 128, 159, 224, 280, 331
RID 239, 325
RID POOL SIZE 325
RMF 95, 271, 324
ROLLBACK 171, 217
row length 177
RTS 64
RUNSTATS 65, 90, 270

S

same way 36, 276
SAQTSAMP 60, 166, 229
S-Blade 193
scalability
 data 5
scalar functions 131
schema 6, 25, 80, 251
SECURITY 239
SEQ 214, 217
Server 54, 162, 245, 269, 292
Service class 115
service classes 90
SET 99, 126, 159, 174, 216, 309, 316
SETTABLES 80
SHRLEVEL 117
side 7, 23, 50, 59, 132, 164, 279, 286
SJTABLES 239, 328
SKIP LOCKED DATA 117
SMF 95, 320
source data 5, 74, 178
SPT01 245
SQL xix, 12, 63, 88, 117, 122, 128, 145, 149, 208–209, 270, 317
SQL queries 124, 149, 196, 208, 248, 331
SQL Reference 126

SQL statement 25, 36, 76, 117, 130, 133, 148–149, 199, 225, 317
SQL statement text 320
SQLCODE 72, 117, 126, 223
SQLERROR 225
SQLSTATE 117, 223
SSID 219
standards 187, 268
STARJOIN 328
START ACCEL 27, 123, 211, 219, 330
state 9, 34, 62, 148, 150, 168, 210, 212, 219, 270, 287
statement 25, 50, 76, 117, 126, 148–149, 174, 199, 225, 294, 313, 317
static SQL 151, 318
statistics 63, 95, 129, 150, 208, 329
STATUS 53, 222, 227, 291
STOP ACCEL 27, 211, 227
stored procedures 12, 74, 112, 159, 163, 229, 270
subscription 164
SUBSTR 125, 132
synchronization 32, 56, 91, 163
SYSACCEL.SYSACCELERATEDTABLES 50, 78, 130, 155, 158
SYSADM 139
SYSCOLUMNS 76
SYSIBM.SYSTABLEPART 78
SYSIBM.SYSTABLES 78, 166
SYSIBM.SYSTABLESPACE 78
SYSIBM.USERNAMES 33
SYSIN 96, 239
sysplex 292
SYSPRINT 82, 96, 253
SYSPROC.ACCEL_ADD_TABLES 37
SYSPROC.ACCEL_CONTROL_ACCELERATOR stored procedure 234
SYSPROC.ACCEL_LOAD_TABLES 48, 80
SYSPROC.ACCEL_UPDATE_SOFTWARE 166
SYSTABLEPART 78
System z xix, 3, 9, 66, 87, 89, 145, 163–164, 208, 220, 267, 285, 309, 322
 component 283
 environment xix, 3, 9, 92, 267
 hardware 7, 267
 network 66, 164, 279
 partition 79, 272
 platform 4, 9, 84
 server 7, 268
 task 170

T

table expression 129
table space 39, 64–65, 148, 150–151, 156, 233, 277, 317–318
 data 323
tables 25, 50, 55, 101, 126, 143, 145, 161, 249, 282, 324
TCP/IP 169, 210, 214, 278, 292
temporal 7
TEXT 219
TIME 37, 96, 99, 130, 132, 210, 214, 313
time period 183

TIMESTAMP 37, 130, 132, 219, 313
trace profiles 44, 311
trace record 208, 320
traces 46, 95, 208, 311
transformation 2, 9, 276
TRUNCATE 63, 132
TYPE 138, 216

U

UDF 6, 130, 217
UK75097 310
UK77225 310
UNLOAD 117, 154, 270
UPDATE 63, 148, 216–217, 274
user ID 167, 279, 335
user-defined function 117, 130
UTF-8 81, 124, 131, 230, 331

V

VALUE 132, 239
VARCHAR 76, 126, 130, 233
variable 40, 60, 126, 166, 230
VERSION 210, 214
Version xix, 16, 66, 88, 140, 165, 210, 286, 308, 316
versions 7, 15, 215, 315
Virtual Accelerator 36

W

WITH 78, 126, 223, 300
WLM xx, 37, 72, 87, 214, 217, 271
WLM application environment 117
WLM environment 114
WLM_SET_CLIENT_INFO 48, 98
work file 324
workload 6, 9, 37, 73, 177, 213, 217, 224, 268, 286, 321
 performance 186, 275
Workload Manager 37, 72, 92

X

XML 50, 59, 156, 229, 325
XML data 333
XML documents 80
xmlns 82, 230

Z

z/OS xix, 3, 11, 66, 88, 198, 207, 210, 268, 285, 308, 315
 system 6, 143, 219, 270, 309, 315
zIIP 100, 275, 318



Hybrid Analytics Solution using IBM DB2 Analytics Accelerator for z/OS V3.1

(0.5" spine)

0.475" <-> 0.873"

250 <-> 459 pages



Hybrid Analytics Solution using IBM DB2 Analytics Accelerator for z/OS V3.1



**Leverage your
investment in
IBM System z for
business analytics**

**Transparently
accelerate DB2 for
z/OS complex queries**

**Implement high
performance and
available analytics**

The IBM DB2 Analytics Accelerator Version 3.1 for z/OS (simply called *Accelerator* in this book) is a union of the IBM System z quality of service and IBM Netezza technology to accelerate complex queries in a DB2 for z/OS highly secure and available environment. Superior performance and scalability with rapid appliance deployment provide an ideal solution for complex analysis.

In this IBM Redbooks publication, we provide technical decision-makers with a broad understanding of the benefits of Version 3.1 of the Accelerator's major new functions. We describe their installation and the advantages to existing analytical processes as measured in our test environment. We also describe the IBM zEnterprise Analytics System 9700, a hybrid System z solution offering that is surrounded by a complete set of optional packs to enable customers to custom tailor the system to their unique needs.

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

SG24-8151-00

ISBN 0738438790