DB2 for OS/390 Version 5
Performance Topics

June 1997
## Contents

**Figures** .......................... vii

**Tables** .......................... ix

**Preface** ........................ xi

The Team That Wrote This Redbook .......... xi

Comments Welcome ........................ xii

**Chapter 1. Introduction** .................. 1

1.1 DB2 V5 Measurements and Performance .......... 2

1.2 DB2 V5 Performance Expectations ............... 3

**Chapter 2. Overview of DB2 for OS/390 Version 5 Enhancements** ................. 5

2.1 Performance, Availability, and Capacity ........ 5

2.1.1 Sysplex Query Parallelism .................. 5

2.1.2 Improved Data Availability with the REORG Utility ........ 5

2.1.3 Data Sharing Enhancements .................. 6

2.1.4 Caching Dynamic SQL Statements .............. 7

2.1.5 Reoptimizing the Access Path at Run Time .......... 8

2.1.6 CASE Expressions .......................... 8

2.1.7 Correlated Index Columns .................... 8

2.1.8 Greater Capacity with Large Partitioned Tables ........ 8

2.1.9 Fast and Usable Utilities ............... 9

2.1.10 Other Enhancements ....................... 9

2.2 Client/Server and Open Systems Processing .......... 11

2.2.1 Native Support for TCP/IP Networks ............ 11

2.2.2 Enhancements to Stored Procedures ............ 11

2.2.3 Package Authorization Cache .................. 12

2.2.4 DRDA Application Performance Enhancements .......... 12

2.2.5 Improved Application Portability ............ 13

2.2.6 Improved Security .......................... 13

2.3 User Productivity Enhancements ............... 14

2.3.1 Image Copy Enhancements .................... 14

2.3.2 RENAME TABLE statement .................... 14

2.3.3 DB2 Visual Explain .......................... 14

2.3.4 DB2 Installer .............................. 14

2.3.5 OS/390 Server Solution ...................... 15

**Chapter 3. SQL Performance** ................. 17

3.1 Caching Dynamic SQL Statements ............... 17

3.1.1 Why Cache Dynamic SQL Statements ............ 17

3.1.2 Conditions for Prepared Statement Caching and Retrieval .......... 19

3.1.3 Controlling the Cache ...................... 20

3.1.4 EDM Pool Considerations .................... 20

3.1.5 Performance Measurements for Cached Statements .......... 22

3.1.6 Summary for Dynamic Statement Caching .......... 23

3.2 Keeping Prepared Statements Past Commit .......... 23

3.2.1 Why Keep Prepared Statements Past Commit .......... 24

3.2.2 How to Keep Prepared Statements Past Commit .......... 24

3.2.3 Recommendations .......................... 27

3.3 Reoptimization at Run Time ................... 28
# 3. Reoptimization and Performance Measurements

3.1 Why Reoptimization Is Required ................................... 28
3.2 How to Reoptimize .................................................. 29
3.3 How to Use EXPLAIN with Reoptimization .......................... 30
3.4 How to Monitor Reoptimization ...................................... 32
3.5 Catalog Table Changes .............................................. 33
3.6 Reoptimization Performance Measurements .......................... 35
3.7 Considerations for Reoptimization at Run Time ...................... 39

# 4. Use of Key Column Correlation Statistics

4.1 Definition of Key Column Correlation ............................. 39
4.2 How Queries Can Benefit from DB2 V5 Statistics .................. 40
4.3 How Access Path Selection Is Influenced by Correlated Statistics 41
4.4 Query Performance Measurement with Key Column Correlation Statistics ........................................... 44
4.5 Conclusion for Use of Key Column Correlation Statistics ....... 44

# 5. Nonpartitioning Index Pieces

5.1 DB2 V5 Improvements ............................................... 47
5.2 Impact with REOPT(VARS) ........................................... 47

# 6. Enhanced Limited Partition Scan

6.1 DB2 V5 Improvements ............................................... 47
6.2 Impact with REOPT(VARS) ........................................... 47

# 7. Optimizer Enhancements

7.1 Noncolumn Expressions .............................................. 48
7.2 Scalar Functions .................................................... 49
7.3 Nonuniform Distribution Statistics ................................ 49
7.4 Enabling Parallel Access ............................................ 51

# Chapter 4. Query Parallelism

4.1 Parallelism Enhancements .......................................... 53
4.2 Sysplex Query Parallelism .......................................... 54
4.2.1 Introduction ..................................................... 54
4.2.2 Configuring for Sysplex Query Parallelism .................. 56
4.2.3 Data Sharing Coordinator and Assistant Configurations ........ 64
4.2.4 Performance Measurements .................................... 65
4.2.5 Conclusions and Recommendations ........................... 70

# Chapter 5. Utility Performance Enhancements

5.1 LOAD and REORG .................................................... 73
5.1.1 PREFORMAT ..................................................... 74
5.1.2 SORTKEYS ...................................................... 75
5.1.3 COPYDDN and RECOVERYDDN ................................. 81
5.2 RECOVER INDEX .................................................... 84
5.3 COPY .............................................................. 85
5.3.1 CHANGELIMIT Option .......................................... 86
5.3.2 Values For CHANGELIMIT Option ............................ 86
5.3.3 The COPY CHANGELIMIT Option and Output Report ......... 87
5.3.4 How to Automate the Image Copy Process .................... 88
5.3.5 Conclusions ..................................................... 90
5.4 RUNSTATS ........................................................... 90
5.4.1 Why RUNSTATS Should Be Run ................................. 90
5.4.2 Changes to Catalog Columns ................................... 91
5.4.3 Faster RUNSTATS with Sampling ............................... 91
5.4.4 Measurements for RUNSTATS Sampling ....................... 93
5.4.5 Key Column Correlation Statistics ............................ 100
5.5 Online REORG ....................................................... 106
5.5.1 SHRLEVEL ....................................................... 106
5.5.2 The Shadow Data Sets .......................................... 107
5.5.3 Phases of the Online REORG Utility ........................... 109
<table>
<thead>
<tr>
<th>Figures</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Accessing Nonpartitioned Indexes: With and Without Pieces</td>
<td>45</td>
</tr>
<tr>
<td>3. Query Parallelism: Enhancements from DB2 V3 to DB2 V5</td>
<td>55</td>
</tr>
<tr>
<td>4. Access Customer Partitioned Table with Sysplex Query Parallelism</td>
<td>56</td>
</tr>
<tr>
<td>5. Configuring Coordinator and Assistant</td>
<td>57</td>
</tr>
<tr>
<td>6. Sysplex Query Parallelism: Relationships of Buffer Pool Thresholds.</td>
<td>58</td>
</tr>
<tr>
<td>7. Example of Multiple Performance Period Goals</td>
<td>60</td>
</tr>
<tr>
<td>8. WLM Classification Rules for the Coordinator TSO Queries</td>
<td>61</td>
</tr>
<tr>
<td>9. WLM Classification Rules for the Assistant TSO Queries</td>
<td>62</td>
</tr>
<tr>
<td>10. A Query with Two Parallel Groups Running at The Same Time</td>
<td>63</td>
</tr>
<tr>
<td>11. Separate Query Processing Configuration</td>
<td>64</td>
</tr>
<tr>
<td>12. Dedicated Query Processing Configuration</td>
<td>65</td>
</tr>
<tr>
<td>13. Table Space Scan Queries Elapsed Time</td>
<td>66</td>
</tr>
<tr>
<td>14. Table Space Scan Queries CPU Time</td>
<td>67</td>
</tr>
<tr>
<td>15. Different Query Types: Comparison by Elapsed Time</td>
<td>67</td>
</tr>
<tr>
<td>16. Scalability Factors: Comparison by Elapsed Time</td>
<td>68</td>
</tr>
<tr>
<td>17. Sysplex Scalability: Comparison by Elapsed Time</td>
<td>69</td>
</tr>
<tr>
<td>18. Query Throughput Scalability: Comparison by Number of Queries per Hour</td>
<td>69</td>
</tr>
<tr>
<td>19. LOAD and REORG Processing without SORTKEYS</td>
<td>76</td>
</tr>
<tr>
<td>20. LOAD and REORG Processing with SORTKEYS</td>
<td>77</td>
</tr>
<tr>
<td>21. The Different Phases in REORG SHRLEVEL NONE</td>
<td>111</td>
</tr>
<tr>
<td>22. The Different Phases in REORG SHRLEVEL REFERENCE</td>
<td>111</td>
</tr>
<tr>
<td>23. The Different Phases in REORG SHRLEVEL CHANGE</td>
<td>112</td>
</tr>
<tr>
<td>24. The Different Phases in REORG PART SHRLEVEL CHANGE</td>
<td>113</td>
</tr>
<tr>
<td>25. Stored Procedure Query Result Sets</td>
<td>127</td>
</tr>
<tr>
<td>26. Stored Procedure Environment</td>
<td>132</td>
</tr>
<tr>
<td>27. The DSNTIPR Installation Panel</td>
<td>137</td>
</tr>
<tr>
<td>28. The DSNTIP5 Installation Panel</td>
<td>138</td>
</tr>
<tr>
<td>29. PREPARE without DEFER(PREPARE) Bind Option in DB2 V4</td>
<td>141</td>
</tr>
<tr>
<td>30. PREPARE without DEFER(PREPARE) Bind Option in DB2 V5</td>
<td>142</td>
</tr>
<tr>
<td>31. Relationships between DB2 V5 CLI Components</td>
<td>145</td>
</tr>
<tr>
<td>32. End to End Elapsed Times</td>
<td>151</td>
</tr>
<tr>
<td>33. Native Class 2 CPU Times</td>
<td>152</td>
</tr>
<tr>
<td>34. Client CPU Usage</td>
<td>153</td>
</tr>
<tr>
<td>35. Group Buffer Pool Rebuild: Scenario of a Rebuild of Structure</td>
<td>156</td>
</tr>
<tr>
<td>DB0G_GBP3 in CF2</td>
<td></td>
</tr>
<tr>
<td>37. Group Buffer Pool Automatic Recovery: Scenario with Different GBP Checkpoint Intervals</td>
<td></td>
</tr>
<tr>
<td>38. Selective Partition Locking in DB2 V5 Compared with Parent L-locking in DB2 V4</td>
<td></td>
</tr>
</tbody>
</table>
# Tables

1. Fetch Performance Enhancement by Dynamic Statement Caching  . . . . 22
2. Insert Performance Enhancement by Dynamic Statement Caching  . . . . 23
3. New WHEN_OPTIMIZE Column in PLAN_TABLE  . . . . . . . . . . . . . . . 32
4. Values of Column STATUS in SYSIBM.SYSTMT  . . . . . . . . . . . . . . 34
5. REOPT(VARS) Compared to NOREOPT(VARS) for Static SQL  . . . . . . . 36
6. REOPT(VARS) Compared to NOREOPT(VARS) for Dynamic SQL  . . . 37
7. REOPT(VARS) Compared to Use of Literals for Static SQL  . . . . . . . 37
8. REOPT(VARS) Compared to Use of Literals for Dynamic SQL  . . . . . . 38
9. REOPT(VARS) Comparison for Static SQL and Dynamic SQL  . . . . . . 38
10. Effect of Column Correlation on Matching Columns in V4 .........  43
11. Query Performance Enhancement by RUNSTATS KEYCARD ......... 44
12. COLVALUE and FREQUENCYF Columns in SYSIBM.SYSCOLDIST .... 51
13. LOAD Utility (Two Type 1 Indexes): V4 and V5 Comparisons .... 78
14. LOAD Utility (Two Type 2 Indexes): V4 and V5 Comparisons .... 79
15. REORG Utility (Two Type 1 Indexes): V4 and V5 Comparisons .... 79
16. REORG Utility (Two Type 2 Indexes): V4 and V5 Comparisons .... 80
17. REORG Utility (Two Type 1 Indexes): V5 Comparisons .... 80
18. REORG Utility (Two Type 2 Indexes): V5 Comparisons .... 80
19. Inline COPY in LOAD and REORG: Two Type 2 Indexes .... 83
20. RECOVER INDEX ALL (with SYSUT1): V4 and V5 Comparisons .... 84
21. RECOVER INDEX ALL (without SYSUT1): V4 and V5 Comparisons .... 85
22. RECOVER INDEX ALL (with SYSUT1 and without SYSUT1): V5 Comparisons .... 85
23. Tables with Column Changes During Migration from V4 to V5 .... 91
24. RUNSTATS Sampling on a Large Partitioned Table Space .... 94
25. RUNSTATS Sampling on Database A ... 95
26. RUNSTATS on Database A: 1% Sampling and no Statistics Comparison .... 95
27. Elapsed Times for Seven-Table Nested Loop Query, Sort for ORDER BY .... 96
28. Elapsed Times for Eight-Table Nested Loop Query, Sort for GROUP BY .... 97
29. Elapsed Times for Two-Table Merge Scan Query, Sort for DISTINCT .... 97
30. Query Performance: 1% Sampling and no Statistics Comparisons .... 98
31. RUNSTATS Sampling on Database B .... 99
32. Statistics in SYSIBM.SYSCOLDIST and SYSIBM.SYSCOLDISTSTATS ... 104
33. RUNSTATS INDEX with KEYCARD and FREQVAL ... 105
34. REORG Table Space: Scenario 1 ... 115
35. REORG Table Space: Scenario 2 ... 115
36. REORG Type 2 Partitioning Index without Concurrent Updates ... 116
37. REORG Type 2 Nonpartitioning Index without Concurrent Updates ... 116
38. REORG Table Space: Scenario 3 ... 116
39. REORG Table Space: Scenario 4 ... 118
40. REORG Table Space Partition with Concurrent OLTP ... 119
41. REORG Type 2 Partitioning Index with Concurrent OLTP ... 120
42. Performance of OLTP ... 121
43. New Columns in SYSIBM.SYSPROCEDURES ... 126
44. Comparison of Direct CLI and Multiple Result Sets with CLI ... 129
45. Stored Procedures with and without Commit on Return ... 130
46. Use of DB2-Established and WLM-Established Address Spaces ... 131
47. Comparison of DB2-Established and WLM-Established Address Spaces ... 135
48. SNA, AnyNet, and TCP/IP Performance Measurement Results ... 140
49. ASCII Client: End to End Elapsed Times ... 151
50. Native Class 2 CPU Times ... 152
<table>
<thead>
<tr>
<th>Number</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>51</td>
<td>Client CPU Usage</td>
<td>153</td>
</tr>
<tr>
<td>52</td>
<td>Castout Ownership for 300 MB measurement</td>
<td>159</td>
</tr>
<tr>
<td>53</td>
<td>Rebuilt Pages by Castout Owner</td>
<td>159</td>
</tr>
<tr>
<td>54</td>
<td>External Throughput Rate Under IRWW Workload, Manual Rebuild</td>
<td>160</td>
</tr>
<tr>
<td>55</td>
<td>External Throughput Rate Under IRWW Workload, Dynamic Rebuild</td>
<td>161</td>
</tr>
<tr>
<td>56</td>
<td>GBP Automatic Recovery: GBP Checkpoint Interval 8 Minutes</td>
<td>162</td>
</tr>
<tr>
<td>57</td>
<td>GBP Automatic Recovery: GBP Checkpoint Interval 2 Minutes</td>
<td>163</td>
</tr>
<tr>
<td>58</td>
<td>GBP Automatic Recovery: GBP Checkpoint Interval 1 Minute</td>
<td>163</td>
</tr>
<tr>
<td>59</td>
<td>Conditional Page P-Lock Measurements</td>
<td>166</td>
</tr>
<tr>
<td>60</td>
<td>Table CREWINFO</td>
<td>173</td>
</tr>
</tbody>
</table>
Preface

This redbook describes performance enhancements in Version 5 of IBM DATABASE 2 for OS/390 (also called DB2 V5). The book covers enhancements that help improve the performance of static and dynamic structured query language (SQL) applications, reduce the elapsed time of complex read-only queries, take DB2 closer to 24X7 data availability. It also covers enhancements that facilitate application portability in the client/server and open systems environments, and improved performance in the data sharing environment.

Recommendations are based on performance measurements done at the IBM Santa Teresa Laboratory.

This book will help managers and technical professionals understand why migrating to Version 5 is beneficial and enable them to take advantage of the enhancements to improve the performance of both their applications and subsystems.

The Team That Wrote This Redbook

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

Ravi Kumar is a Senior ITSO Specialist for DB2 for System 390 and MVS/ESA at the Application Development and Data Management ITSO Center, San Jose. He writes extensively and teaches IBM classes worldwide on all areas of DB2 for System 390 and MVS/ESA. Before joining the ITSO two years ago, Ravi worked in IBM Australia as a DB2 specialist.

Stephen Elford is DB2 Consultant in Product Support Services, IBM Sweden. He has 26 years of experience in the computer industry. His areas of expertise include MVS performance, systems management, and software development. He teaches IBM classes on all areas of OS/390 and DB2 for System 390. Previously he was the chief designer for EPDM, now known as Performance Reporter for OS/390.

Alan Hawkins is a Systems Support Specialist with IBM United Kingdom. He has 12 years experience with DB2, in database administration, systems programming and performance management roles. He holds a Ph.D. in Minerals Engineering from Birmingham University and joined IBM in 1982 as an analytical chemist. His areas of expertise include DB2 Performance Monitor (DB2 PM) and Query Management Facility (QMF).

Roar Syversen is a Consultant in IBM Global Services, Product Support Services, IBM Norway. He has 25 years of experience in the computer industry, 10 years of experience with DB2. His areas of expertise include MVS system programming, Customer Information Control System (CICS), and Resource Access Control Facility (RACF).

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

- Brian Baggett IBM Development, Santa Teresa
- Ted Blank IBM Development, Santa Teresa
- Viviane-Anavi Chaput IBM France
Comments Welcome

We want our redbooks to be as helpful as possible. Should you have any comments about this or other redbooks, please send us a note at the following address:

redbook@vnet.ibm.com

Your comments are important to us!
DB2 for OS/390 Version 5 Enterprise Data Server (DB2 V5) addresses the demands of client/server and data warehouse applications. By delivering a flexible client/server solution, DB2 V5 increases processing capacity and provides a scalable solution capable of managing terabytes (TB) of data. It delivers functions to improve application design decisions and ease DB2 system administration.

DB2 V5 also demonstrates IBM’s leadership in parallel database technology by further exploiting the System/390 Parallel Sysplex and DB2 data sharing. Sysplex query parallelism expands the power available to a single query, making it possible not only to share data but also to split a query into a number of smaller tasks and to run these tasks in parallel across multiple DB2 subsystems on multiple complexes in a data sharing group. The enhanced function combines with new support for very large tables to provide the performance needed for complex data analysis used by data mining applications. It increases the limit for table space size to 1 TB, and the number of partitions to 254.

DB2 for OS/390 takes a big step toward continuous availability with major improvements in the REORG utility. A new option allows read and write access to the data during most phases of the reorganization. A read-only access option allows the utility to process more quickly. Also, LOAD and REORG now support in-line COPY which allows the table space to be available as soon as the utility completes. Another new option for LOAD and REORG can bypass intermediate data sets which can allow the utility to complete its task in less elapsed time. Improvements to RECOVER INDEX unload phase and LOAD reload phase reduce CPU time. You can now use a SAMPLE option with RUNSTATS to reduce processing costs when collecting statistics on nonindexed columns.

Several enhancements improve availability and performance for DB2 data sharing. DB2 V5 takes advantage of the MVS function that permits a rebuild of the group buffer pools (GBP). You can use this function to rebuild a GBP to another coupling facility before applying maintenance or it may be invoked by DB2 to recover after a link failure has occurred. Other improvements include coexistence support, which permits phased migration of a DB2 data sharing group from V4 to V5 without stopping the entire Sysplex. Also, you can automatically recover data from a damaged GBP without manual operator intervention, retained locks are released sooner following a member failure, which makes the data available faster, and improved detection of changes in the read and write interest of shared data can lead to reduced overhead costs.

DB2 V5 has added a number of run-time performance improvements. Dynamic SQL that is called repeatedly can now be prepared and cached for use by subsequent application processes. A new bind option allows DB2 to reevaluate the access path after the values of host variables or parameter markers have been determined. Access path selection for correlated columns, such as city and state, can be improved by using RUNSTATS to collect statistics on correlated index columns. Also, you can now use indexes for some common expressions such as DATECOLUMN > CURRENT DATE – 30 DAYS. Previously, indexes could not be used in these expressions.

DB2, an industry leader in distributed technology, enhances the Distributed Relational Data Architecture (DRDA) significantly to improve performance and
communication costs for applications such as DataJoiner, faster block fetch operations, reduced network traffic due to improved dynamic SQL which permits a single message exchange to both send and receive a query, optimize for “n” rows on SELECT which allows the application to specify the number of rows returned, and improved Virtual Telecommunications Access Method (VTAM) performance to improve transaction throughput. Also, DB2 for OS/390 can store data in ASCII format, thereby eliminating the need, for example, for special sorting routines when data is returned from an ASCII client.

DB2 V5 extends the ability of DB2 for OS/390 to act as an enterprise server for client/server applications, by providing native Transmission Control Protocol/Internet Protocol (TCP/IP) support. IBM’s new Distributed Database Connection Services (DDCS) for Windows product uses this new capability to connect users of Windows 3.x and Windows 95 directly to DB2, without requiring an intermediate gateway machine.

DB2 V4 provides stored procedure support, allowing a DB2 client to call a procedure stored at the DB2 server. DB2 V5 extends this support and provides stored procedure query result sets. This allows DRDA clients to use tools like VisualAge, PowerBuilder, and VisualBasic to process one or more query answer sets returned by a DB2 stored procedure. Stored procedures take advantage of the Multiple Virtual Storage (MVS) workload manager (WLM) to prioritize individual procedure requests and run them in multiple address spaces. Performance can be enhanced by allowing procedures to run as subroutines as well as main routines. Security is enhanced by allowing clients to access the result sets without requiring the end users to have access to the underlying SQL tables and by using the end user’s RACF user ID to control access to non-SQL resources.

You can now choose between host-based or OS/2 workstation-based installation of DB2 for OS/390. Application design and development flexibility are extended by support for C++ and for object-oriented COBOL extensions, and by conformance to the SQL92 Entry Level Standard. For increased usability and simplification, DB2 Version 5 provides, as a server solution, DB2 Estimator and DB2 PM. For Internet connectivity, DB2 Version 5 provides Net.Data for MVS/ESA.

## 1.1 DB2 V5 Measurements and Performance

The common goal for all versions of DB2 is to provide performance that is equivalent to or better than the previous release of the product. The DB2 product development staff at IBM’s Santa Teresa Laboratory perform measurement tests to validate the changes and new functions available in DB2 V5.

This book outlines the performance related functions of DB2 V5, describes the measurement environments, and provides recommendations and conclusions based on the performance measurement tests.
1.2 DB2 V5 Performance Expectations

DB2 V5 is a well balanced version that offers new capabilities and possible performance improvements for the transaction, query, batch, utility, and client/server environments.

When considering the migration of a DB2 production environment from DB2 V4 to V5, it is useful to be aware of the investment you require to obtain the benefits of the new enhancements. You need to know about:

- The performance benefits available immediately after migration without implementing any new functions:
  Query (conditional P-locks in a data sharing environment, and stage 1 noncolumn expressions), operations (automatic recovery in data sharing), utility (CPU time in LOAD and RECOVER INDEX) will automatically benefit.

- The performance benefits can be obtained mostly from system programming and database administration effort, without modifying the applications:
  Benefits can be obtained with minimal effort for query performance and concurrency (dynamic statement caching, stage 1 noncolumn expressions, correlated index columns, package authorization caching, nonpartitioning index pieces, and selective partition locking), utility performance (online REORG, LOAD, REORG, RUNSTATS, and RECOVER INDEX), and operational performance (GBP rebuild, and log preformat).

- The performance impacts of new functions, in particular those available to end users, and the recommendations on how to design or tune the applications for optimum benefit:
  Sysplex query parallelism expands the power available to a single query, and, combined with support for large tables, provides the scalability for applications requiring access to very large amounts of data with reduced elapsed times.
  Application code and bind option changes (reoptimization of SQL, stored procedures, keeping prepared statements past commit), and query (CASE expression) provide performance benefits.

- The functions designed to improve the client/server environment and provide portability:
  These functions require detailed planning (native TCP/IP support, ASCII data support, WLM support for stored procedures, and call level interface (CLI)).
Chapter 2. Overview of DB2 for OS/390 Version 5 Enhancements

In this chapter, we discuss the DB2 for OS/390 Version 5 enhancements related to performance, client/server processing, availability, capacity, and user productivity.

2.1 Performance, Availability, and Capacity

Better performance for query and batch processing make DB2 for OS/390 the best choice for both online transaction processing and data warehousing. You can store a terabyte of data in your table spaces and keep data available with fast utilities.

DB2 V5 provides data availability improvements, enhances the data sharing environment with restart, recovery, and rebuild functionality, and provides greater capacity with large partitioned tables.

2.1.1 Sysplex Query Parallelism

Businesses have an increasing need to analyze large quantities of data, whether to validate a hypothesis or to discover new relationships between data. This information is often critical to business success, and it can be difficult to get the information in a timely manner. DB2 V4 lets you split and run a single query within a DB2 subsystem. With Sysplex query parallelism, DB2 V5 extends parallel processing to allow a single query to use all the power of a data sharing group.

Sysplex query parallelism is when members of a data sharing group process a single query. DB2 determines an optimal degree of parallelism based on estimated I/O and processing costs. Different DB2 members process different ranges of the data. Applications that are primarily read only and are processor intensive or I/O intensive can benefit from Sysplex query parallelism.

Because a data sharing group can grow, you can put more power behind your queries even as those queries become increasingly complex and run on larger and larger sets of data.

2.1.2 Improved Data Availability with the REORG Utility

DB2 takes a big step toward continuous availability with major improvements to the REORG utility. Previous releases of DB2 allow only limited access to data or indexes that are being reorganized. A new option of the REORG utility, SHRLEVEL CHANGE, allows read and write access during most phases of table space or index reorganization. You can also choose read-only access, which allows the utility to execute more quickly.

One of the major performance concerns with REORG is that the data is unavailable most of the time during the process. Online REORG removes the performance concerns from REORG by allowing the data to be available most of the time during the process.
2.1.3 Data Sharing Enhancements

Several enhancements improve availability, performance, and usability for DB2 data sharing.

2.1.3.1 Rebuild Support for Group Buffer Pools

DB2 V5 takes advantage of the MVS function that lets you rebuild the structure of GBPs:

- When doing planned maintenance of a coupling facility (CF) using DB2 V5, you can manually rebuild the GBPs to another CF before applying the maintenance. You no longer have to manually deallocate the GBPs before reallocating them to another CF.
- When a link failure occurs, DB2 dynamically rebuilds GBPs on another CF to avoid adding pages to the logical page list (LPL) and to avoid returning "resource unavailable" to applications. Continuous access to the data in the GBP is improved.

2.1.3.2 Automatic Recovery of Data in Damaged Group Buffer Pools

Now you can recover data automatically from a damaged GBP. In DB2 V4, an operator must issue START DATABASE commands to invoke recovery. The new method of recovery in DB2 V5 is faster, is less prone to human error, and makes data available more quickly.

2.1.3.3 Restart and Recovery Performance Improvements

When a member of a data sharing group fails, data that is being updated remains locked to maintain data integrity. DB2 V5 reduces the length of time retained locks are held, which makes data available sooner.

2.1.3.4 Coexistence Support

You can migrate a DB2 data sharing group from DB2 V4 to DB2 V5 without bringing down the entire data sharing group. When you migrate, members in DB2 V4 and DB2 V5 can coexist within the same data sharing group. By letting the releases coexist, DB2 provides nearly continuous availability across a release migration.

Mixed releases create an added complexity that you must manage. When you run mixed releases, you cannot use new functions, and operational problems may occur. Catalog migration does not provide concurrent access to the DB2 catalog, so plan to run mixed releases over a very short period of time, such as a weekend.

2.1.3.5 Reduced Data Sharing Costs

DB2 V5 introduces many enhancements that reduce the cost of data sharing.

*Conditional P-locks for Data Pages with Row Locking:* Data sharing in DB2 V4 introduced a new type of lock, called a physical lock (P-lock) which plays an important role in how DB2 tracks inter-DB2 interest. For example, in DB2 V4 if one DB2 member requests a P-lock that another DB2 member holds, the mode of a P-lock can be negotiated. In this process, the first DB2 member relinquishes the P-lock by forcing a write to the active log, a write to a GBP, and an unlock of the page P-lock. Now DB2 V5 uses a conditional P-lock for the insert, so the requesting member will move to a new page instead of negotiating the P-lock.
Specifying the Maximum Number of Rows in a Page: In DB2 V5, you can specify MAXROWS option with CREATE TABLESPACE to control the number of rows to be placed on a page. This maximum number persists through inserts or deletes and when data is loaded or reorganized. You can, for example, use this option to ensure that only one row is placed on a page. This helps in eliminating the data page P-lock requests when row locking is used in the data sharing environment.

Selective Locking of Partitions in Partitioned Table Spaces: By recognizing when inter-DB2 read-write interest changes, DB2 V5 can reduce the amount of overhead associated with sharing data.

If you have a batch job on different members processing separate partitions, you can indicate that you want individual partitions locked only as they are accessed. You can specify the LOCKPART YES option of CREATE and ALTER TABLESPACE; each locked partition is then a separate lock parent. DB2 and integrated resource lock manager (IRLM) can detect when no inter-DB2 read-write interest exists on that partition and therefore does not propagate child locks unnecessarily. This can significantly reduce the cost of locking in these situations.

Dynamic Inter-DB2 Interest Tracking Improvement: In DB2 V4, when a member has read-write interest in a page set or partition and then the inter-DB2 interest on the page set or partition goes away, the page set or partition remains GBP-dependent until the last updating member switches to read-only. DB2 V5 removes the GBP-dependent status earlier by switching to a non-GBP-dependent as soon as the reader does a physical close.

Other Enhancements: The following are other enhancements that reduce data sharing costs:

- IRLM latch contention from P-lock unlock is reduced.
- Code is optimized for each subsequent insert and update to the same page in a GBP-dependent page set or partition.
- Inter-DB2 contention for an index tree P-lock when a Type 2 index split occurs is reduced.
- Hashing algorithm change reduces false contention.
- The default for the global committed log sequence number (GCLSN) is changed, resulting in more effective lock avoidance.
- Space map page update frequency and member-private space map pages help to reduce the space map page P-lock contention.

2.1.4 Caching Dynamic SQL Statements

For cases where the application issues the same dynamic SQL statement repeatedly, the ability to cache the previously prepared statement can significantly reduce the cost of running those statements. Different application processes can share the same prepared statements.

A new bind option lets you retain a prepared statement past a commit point, so that you no longer have to reissue a PREPARE statement. In addition, fewer messages are transmitted in a client/server environment when issuing dynamic SQL.
2.1.5 Reoptimizing the Access Path at Run Time

When host variables, parameter markers, or special registers are used in releases prior to DB2 V5, DB2 cannot always determine the best access path because the values for these variables are unknown. In DB2 V5, you can choose to let DB2 reevaluate the access path after these previously unknown values are determined. For dynamic queries, getting the latest information is often crucial and to obtain the information through dynamic SQL caching is a small additional resource burden in comparison to not having the reoptimization. Static queries make use of the reoptimization technique to resolve the situation when one access path is not sufficient.

2.1.6 CASE Expressions

DB2 V5 accepts CASE expressions wherever an expression is allowed in an SQL statement. The CASE expression allows you to write conditional logic where an expression is selected based on the evaluation of one or more conditions.

CASE expressions significantly reduce the elapsed time of queries because they reduce the number of data pages scanned. With CASE expressions, you can simplify the query so that it accesses the table only once.

2.1.7 Correlated Index Columns

Using the RUNSTATS utility, you can have DB2 collect statistics on a set of concatenated key columns of an index instead of limiting the collection to just the first key column or all key columns.

With these enhanced statistics, DB2 can choose a more efficient access path for those queries that have correlated columns (such as CITY and STATE) with three or more columns in the index. You can manually update column information for correlated columns that are not in the index.

2.1.8 Greater Capacity with Large Partitioned Tables

You can store important business records collected over years and decades in large partitioned tables. Large partitioned tables have the following advantages.

- More data in one table space
  A table space can hold 1 TB of data, instead of only 64 GB.

- More and larger partitions
  Large table spaces can have a maximum of 254 partitions of 4 GB each, instead of only 64 partitions of 1 GB each. Nonpartitioning indexes on large partitioned table spaces can have up to 128 data sets of up to 4 GB each.

- More flexible database design
  You no longer have to separate large amounts of data into different tables because of size constraints. This eliminates complex design work and simplifies queries.
2.1.9 Fast and Usable Utilities

In addition to the major improvements to the REORG utility, DB2 V5 delivers extensive improvements to several utilities that reduce processor or elapsed time and increase the availability of your data. Making copies is easy, and it is also easy to control when a copy should be made and what kind of copy to make.

2.1.9.1 LOAD and REORG with Inline COPY
A new option of the LOAD and REORG utilities, COPYDDN, lets you make an image copy without specifying a separate copy step. As a result, a separate scan of the data to create the copy is no longer needed. The table space is available sooner because the table space is not left in the copy pending state, regardless of the specified LOG option.

2.1.9.2 LOAD and REORG with SORTKEYS
A new option of the LOAD and REORG utilities, SORTKEYS, can be used to avoid unloading the index keys to an intermediate work data set. As a result, the elapsed times are reduced.

2.1.9.3 RECOVER INDEX and LOAD
Changes to the RECOVER INDEX unload phase and LOAD reload phase have reduced their CPU times.

2.1.9.4 RUNSTATS with Sampling
A new SAMPLE option of the RUNSTATS utility reduces the CPU cost of collecting statistics on nonindexed columns. With sampling, you define a percentage of the table rows, such as 10%, that is used to calculate nonindexed column statistics.

2.1.9.5 Preformatting of Data
In DB2 V5, the PREFORMAT option of LOAD and REORG allows you to preformat the data in your table spaces. The PRIQTY value in the CREATE TABLESPACE statement determines how many tracks or cylinders to preformat.

2.1.9.6 Data Transfer with Striping for BSAM Data
DB2 utilities can take advantage of Data Facility Storage Management Subsystem/MVS (DFSMS/MVS) support for extended sequential data sets to improve the I/O speed of DB2 utilities. With Basic Sequential Access Method (BSAM) striping, data is transferred between direct access storage device (DASD) and memory at a rate that is faster than an individual DASD can handle. BSAM striping is best for data sets that have large block sizes and high sequential processing activity.

2.1.10 Other Enhancements
DB2 V5 provides improvements in using indexes, caching authorization checks, reducing I/O contention, increasing number of rows in a page, using multiple service tasks, using deferred close at partition level, improving data definition language (DDL) concurrency, and preformatting active logs.
2.1.10.1 Use of Indexes
DB2 V5 improves performance by using indexes for some common expressions such as \texttt{DATECOLUMN > CURRENT DATE - 30 DAYS}. Previous releases of DB2 did not use indexes for these common expressions. Also in DB2 V5 all scalar functions can now use indexes.

2.1.10.2 Authorization Caching for Packages
DB2 caches information about package authorization to avoid repeated checks of the catalog at run time. This function improves the performance of applications that use wild cards in a package list or applications in a client/server environment.

2.1.10.3 Spreading Out Data Sets of Nonpartitioning Indexes
A new option lets you specify a maximum size for each data set (piece) of a nonpartitioning index. By specifying a size smaller than the default size, you can have more pieces and therefore greater control over where to place those pieces. Placing the pieces on separate I/O paths can reduce I/O contention for SQL operations that access nonpartitioning indexes.

2.1.10.4 Multiple Service Tasks
DB2 V5 introduces multiple service tasks for data set open and close, SYSLGRNX update, Access Method Services (AMS) define, and Data Space Manager (DSM) Create. In DB2 V4, we have only one task each for the above, resulting in waits and longer elapsed times. In DB2 V5, we have multiple tasks for each of the above. The tasks are generally associated with task control blocks (TCB). In the case of SYSLGRNX, service request blocks (SRBs) are used instead of TCBs, and therefore no limit on how many SRBs can be scheduled concurrently.

2.1.10.5 Deferred Close by Partition
You have a partitioned table space with 20 partitions, and only one of these partitions is in use whereas the remaining 19 partitions are no longer being referenced. In DB2 V4, defer close is possible only at table space level and so the 19 unreferenced partitions cannot be closed. In DB2 V5, defer close is possible at partition level, so that the 19 unreferenced partitions can be closed.

2.1.10.6 DDL Concurrency Improvements
DB2 V5 improves concurrency for DDL because of the ability to have multiple service tasks discussed in 2.1.10.4, “Multiple Service Tasks.”

2.1.10.7 Preformatting Active Logs
A new stand-alone batch utility preformats active log data sets that are newly defined. Preformatting new active log data sets improves performance when writing to logs.

2.1.10.8 Keeping Update Locks
For a SELECT statement which includes both \texttt{FOR UPDATE OF}, and either \texttt{WITH RS} or \texttt{WITH RR}, a new option \texttt{KEEP UPDATE LOCKS} enables DB2 to acquire an X-lock instead of the U or S-lock on all qualified pages/rows. All the X-locks acquired are retained until the next commit point.
2.2 Client/Server and Open Systems Processing

You can access more data with additional network and interface support. The enhanced functions of DB2 stored procedures allow you to improve the use of your desktop tools by using DRDA applications such as DataJoiner, retrieve data more quickly from DB2 by improvements to network processing. You can port applications to DB2 for OS/390 more easily by using the CLI. DB2 V5 improves security for your client/server environment.

2.2.1 Native Support for TCP/IP Networks

Many businesses choose TCP/IP as the communication protocol for local area networks and wide area networks. Previous releases of DB2 support TCP/IP networks only if AnyNet is installed on both the client and the server. DB2 V5 native support for TCP/IP requires no additional software to run network applications, which reduces complexity and software cost. Now, DRDA clients on many platforms, including OS/2, AIX, Sun Solaris, HP/UX, and Windows, can connect directly to the OS/390 platform without using a Systems Network Architecture (SNA) gateway machine. The native TCP/IP support relies on services available in OS/390 Release 3.

2.2.2 Enhancements to Stored Procedures

DB2 V5 enhancements to stored procedures are more query result sets, more concurrent transactions, more address spaces for stored procedures, flexible dispatching priorities, and two-phase commit processing between DB2 and non-SQL resources.

2.2.2.1 Multiple Query Result Sets

Many organizations use VisualAge, PowerBuilder, VisualBasic, or Lotus Approach to access data managed by DB2. These popular tools let you access data from many relational data sources. Many desktop tools provide powerful application development functions that exploit query result sets returned from stored procedures. DB2 stored procedures can now return one or more query result sets to a DRDA client, which improves network performance between clients and DB2, and lets you take full advantage of your desktop tools. You can also return nonrelational data, such as data from Information Management System (IMS) or virtual storage access method (VSAM) files, to a DRDA client.

2.2.2.2 Improved Environment

With DB2 V5 and enhancements to the MVS workload manager that are available in OS/390 Release 3, the DB2 stored-procedures environment supports a higher volume of transaction processing than in previous releases. Enhancements include the following:

- Multiple DB2 address spaces for stored procedures allow clients to run more concurrent transactions. The MVS workload manager optimizes performance based on system workload. The multiple address spaces also provide improved program isolation for stored procedures.
- The MVS enclave support is extended to include central processing unit (CPU) processing performed by stored procedures. Thus the MVS workload manager can assign individual MVS dispatching priorities to each DB2 stored procedure without requiring any additional tuning parameters.
- Stored procedures access non-SQL resources based on the caller’s RACF capabilities, which can improve security.
• The new Recoverable Resource Manager Services Attachment Facility (RRSAF) can use the OS/390 Transaction Manager and recoverable resource manager services (OS/390 RRS) to synchronize two-phase commit processing between DB2 and non-SQL resources that are accessed by a stored procedure.

2.2.3 Package Authorization Cache

In DB2 V5, all packages running on a DB2 system are eligible to store and retrieve authorization information from the cache. There is a new system parameter (ZPARM CACHEPAC in DSN6SPRM) to specify the cache size. The default size for the cache is 32K. With the package authorization cache turned on, you can expect a performance improvement for distributed applications.

2.2.4 DRDA Application Performance Enhancements

New functions improve the performance of high-volume transaction processing for client/server applications. DB2 V5 introduces changes to help reduce network traffic.

2.2.4.1 Support for High Volume Data Transfer

DB2 V5 adds support for asynchronous transfer mode (ATM) networks. ATM is a low-level communication protocol that routes either TCP/IP or SNA network traffic. ATM is an open solution with broad vendor support. The hardware is extremely fast, which makes it ideal for transmitting multimedia data for applications such as video on demand and digital library.

2.2.4.2 DRDA Support for OPTIMIZE for n ROWS on SELECT

Users of some client/server applications browse only a limited amount of data, perhaps only enough to fill one screen. To reduce the elapsed time for applications that read only a subset of the data, use the SELECT clause, OPTIMIZE for n ROWS, to control the amount of data returned to the application at one time.

2.2.4.3 Support for ASCII Tables

You can have DB2 store data in ASCII format. This support can improve the performance of queries in a client/server environment when you require ASCII comparisons and sorting.

2.2.4.4 Reduced Network Traffic

Network processing for DRDA requests is one of the biggest contributors to the overall processing cost of DRDA applications. Two changes reduce that cost:

• Faster dynamic SQL by a single message exchange on the network. The message both sends and receives a dynamic SQL query. In previous releases of DB2, the dynamic SQL SELECT statement requires two network message exchanges. In DB2 V5, a dynamic SQL SELECT statement requires only a single network message exchange.

• More efficient processing of VTAM send and receive operations by VTAM issuing a single request that performs a network send and receive operation. This change reduces the processing overhead at both the DB2 server and the DB2 requester. DB2 also transmits reply messages faster because data moves fewer times during network I/O operations.
2.2.5 Improved Application Portability

The call level interface improves application portability. Tools such as Visual Basic use standard interfaces such as Open Database Connectivity (ODBC) and X/Open Call Level Interface. These standard interfaces require only one version of an application to access many data sources. The application is not dependent on a specific precompiler.

Although client applications that use these standard interfaces can access DB2 for OS/390 data by using DDCS, DB2 for OS/390 now provides a CLI for programs that run on MVS. You can port applications from UNIX and PC workstations to MVS with little or no modification.

MVS applications can also issue ODBC or CLI calls from within a stored procedure. For example, you can write a stored procedure in C, using VisualAge, and then issue CLI calls to execute the SQL statements.

2.2.6 Improved Security

DB2 V5 improves security for your client/server environment. Distributed computing environment (DCE) security with simplified password management lets users more easily access data on a remote DB2 for OS/390 server. DB2 for OS/390 authenticates the DCE security token of a remote workstation user who connects to DB2 by way of DRDA. The user needs to maintain only one password, the DCE password, to access any DB2 that supports DCE security.

RACF PassTickets let you send user authentication information to the server without having to transmit passwords across the network.

In releases prior to DB2 V5, when DB2 detected a security error concerning user ID and password validity, the client received a single reason code that described the error. In DB2 V5, DB2 can send the DRDA requester more detailed security error codes for common security errors such as invalid user IDs, invalid passwords, and expired passwords.

In DB2 V5 DRDA lets users change expired passwords. The new security messages let DRDA requesters determine whether their RACF passwords have expired. When a DRDA requester is notified that a RACF password has expired, then the requester can prompt the end user for the old password and a new password and send those to the DB2 for OS/390 server.

The Security Server for OS/390 Release 4 provides a new function that allows you to control access to DB2 objects using RACF profiles. DB2 V5 uses the DB2 access control authorization exit point to access the OS/390 function. The benefits from this enhancement are these:

- Security rules are defined before a DB2 object is created.
- Security rules persist when a DB2 object is dropped.
- Access to DB2 objects is controlled with generic profiles.
- Flexible control of access to DB2 objects for single or multiple subsystems is provided using a single set of RACF profiles.
- User IDs are validated before permitting access to a DB2 object.
- DB2 cascading revoke is eliminated.
2.3 User Productivity Enhancements

The DB2 V5 productivity enhancements enable you to capitalize on enhanced DB2 family consistency and standardized SQL to simplify queries. You can use enhancements to the COPY utility to assist you in making decisions about the scope of the image copy. You can use the RENAME TABLE statement to change the characteristics of a table without copying the data multiple times.

DB2 V5 simplifies system administration and application programming tasks by using a graphical interface on an OS/2 workstation to install DB2 and to explain your queries. You can take advantage of functions that better integrate object oriented programming and improve access to data in your client/server environment. You can use the DB2 solution for the OS/390 server features to make full use of your DB2 subsystems.

2.3.1 Image Copy Enhancements

Usability enhancements to the COPY utility simplify the task of creating image copies. The new COPY option, CHANGELIMIT helps you make informed decisions about whether to make an image copy and, if so, whether it should be full or incremental. The CHANGELIMIT option lets you tell DB2 whether it should make a copy based on the percentage of changed pages in a table space, partition, or data set. DB2 also reports the number of changed pages, which lets you monitor the rate of change of a particular object.

2.3.2 RENAME TABLE statement

The RENAME TABLE SQL statement renames an existing table. This statement lets you change the characteristics of a table without physically copying the table multiple times. The authorizations for the old table are automatically transferred to the new table. Plans or packages dependent on the old table are not invalidated. Applications can still access the old table name until the plan or package is subject to a rebind.

2.3.3 DB2 Visual Explain

DB2 Visual Explain is an easy-to-use workstation tool that presents the DB2 Explain output in a graphical format. Relationships between database objects, such as tables and indexes, are instantly clear as are various operations, such as table space scans and sorts. DB2 Visual Explain also includes a browser for viewing DB2 subsystem parameters. In addition, context-sensitive help provides the user with direct access to performance suggestions in online DB2 books.

2.3.4 DB2 Installer

DB2 Installer enhances your productivity whether you are installing DB2 for the first time or are an experienced installer. From your workstation, you can install or migrate DB2 for OS/390 from a graphical command center. The graphical interface makes it easy to see the overall installation process. Also, you can keep a graphical record of how each subsystem is defined. The DB2 Installer helps you customize your DB2 subsystem as much or as little as you need. You can get a basic subsystem running quickly or customize every installation option. Also, you can run System Modification Program/Extended (SMP/E), installation, and sample jobs from the workstation. You receive job status dynamically, and you can edit job control language (JCL), perform some job cleanup, and examine job output.
2.3.5 OS/390 Server Solution

DB2 delivers a database server solution for the OS/390 server. Three optional features work directly with DB2 V5 for OS/390 applications to help you use the full potential of DB2:

- IBM DB2 Estimator for Windows estimates performance, resource use, efficiency, and cost of both existing and planned DB2 applications.

- DB2 PM produces reports and can perform real-time online monitoring to help you analyze, control, and tune performance of your DB2 subsystem and DB2 applications.

  DB2 PM V5 has an OS/2 version of the online monitor that substantially improves usability. You can monitor one or more DB2 subsystems from a single OS/2 workstation.

- Net.Data for MVS enables World Wide Web clients to access your DB2 data from the Internet. Net.Data works with the Internet Connection Secure Server and other popular servers to add live data from DB2 or other data sources to your Web pages.
Chapter 3. SQL Performance

This chapter provides details for the following enhancements related to SQL performance:

- Caching dynamic SQL statements
- Keeping prepared statements past commit
- Reoptimizing at run time
- Using key-column correlated statistics
- Nonpartitioning index pieces
- Enhanced limited partition scan
- Optimizer enhancements

3.1 Caching Dynamic SQL Statements

This section describes why you should consider caching dynamic SQL statements, and what type of applications can benefit. We describe the criteria that determine whether a dynamic statement is eligible for caching and under what circumstances DB2 can retrieve the prepared statement from the cache. We show how to enable the cache, and discuss the environmental description manager (EDM) pool implications for storage and performance. We include performance measurements that show the improved performance for queries that can benefit from the cache, and how dynamic SQL can perform almost as well as static SQL when using the cache.

3.1.1 Why Cache Dynamic SQL Statements

When you use dynamic SQL, DB2 must prepare each statement before use. The PREPARE statement creates an executable SQL statement from the character string that the application supplies. The executable form is called a prepared statement, while the character string is called a statement string. The preparation cost includes the optimization process, which depends on the complexity of the statement. Considerable catalog access can occur for the optimization process as well as for satisfying authorization checks.

An application has to pay the cost of a further PREPARE for every COMMIT issued, except in the case of SELECT statements declared with a cursor WITH HOLD. A cursor declared WITH HOLD has locking overheads because DB2 holds database descriptor (DBD) locks for all objects that the statement depends on.

In DB2 V5, you can cache prepared dynamic statements in the EDM pool. Both the statement string and the prepared statement are stored. All application processes use a single shared cache to store and retrieve prepared dynamic SQL statements. The decision on whether you use the cache to store prepared statements is at a system level.

If the cache is activated, an SQL statement is automatically stored in the cache after preparation. DB2 checks the cache whenever a prepare request is issued. If the same statement is already in the cache, the PREPARE and its cost are avoided. If the statement is not in the cache, then it is validated, optimized and placed in the cache for possible later use. An application must issue a PREPARE request after a COMMIT, but should find the already prepared statement in the cache.
Cached statements can be shared by different application processes. A prepared statement cached by one application process can be used by another application. Provided the SQL statement string matches and other conditions are met, the same prepared statement can be shared by different plans and packages.

### 3.1.1.1 Applications That Can Benefit

Dynamic statement caching is most suitable for applications with highly repetitive dynamic SQL statements. There is little benefit for applications that normally issue ad-hoc queries—for example, using SPUFI and DSNTEP2.

QMF makes very limited use of prepared statement caching. QMF simulates host variable use by inserting literal values into the SQL rather than by using parameter markers. This is sometimes referred to as parameterized SQL and is different from normal dynamic SQL applications. When these literal values change, DB2 does not identify the statements as being the same.

The following example illustrates the maximum benefit you can expect from dynamic statement caching. Consider a typical dynamic SQL application which retrieves a large number of rows of data by opening a cursor C1 for the SELECT statement at STMT1. The application prepares and executes statement STMT2 many times within a loop to update table EMP for each row it fetches from the PROJ table. Frequent commit points are set up within the loop:

```sql
DECLARE CURSOR C1
WITH HOLD FOR STMT1

PREPARE STMT1 FROM
  SELECT RESPEMP FROM PROJ
  WHERE PRENDATE<CURRENT DATE;

DESCRIBE STMT1
INTO DESCRIPTOR :SQLDA

OPEN C1

DO UNTIL SQLCODE=100
  FETCH FROM C1
  USING DESCRIPTOR :SQLDA

PREPARE STMT2 FROM
  UPDATE EMP SET BONUS=? WHERE EMPNO=??

EXECUTE STMT2
  USING DESCRIPTOR :SQLDA

COMMIT

END
```

Both statements STMT1 and STMT2 are cached in the EDM pool the first time they are prepared. STMT1 is probably never reused. STMT2 is executed many times within the loop but is prepared only once; on every subsequent execution, the prepared statement is found in the cache.
At commit time, cursor C1, declared WITH HOLD, stays open and the prepared statement STMT2 remains in the cache. Although the prepared statement is in cache and is not reprepared, it is essential that a PREPARE statement follows the COMMIT within the processing loop for cursor C1. You should need no changes to your existing application code to benefit from dynamic statement caching. You should continue to code the PREPARE statement within the processing loop for new application programs. If you wish to share prepared statements between applications, you need to ensure that the statements match exactly and that other criteria we show in 3.1.2.2, “Criteria for Prepared Statement Retrieval” are met.

3.1.2 Conditions for Prepared Statement Caching and Retrieval

Certain criteria must be met to enable prepared statements to be both cached and retrieved.

3.1.2.1 Criteria for Prepared Statement Caching

You can cache the following SQL statements:

- SELECT
- UPDATE
- INSERT
- DELETE.

Prepared statements from both locally and remotely connected applications are eligible for cache storage and retrieval. The following restrictions apply:

- Statements bound with the REOPT(VARS) option to enable reoptimization at run time are not eligible for caching. The principle of caching a statement is to optimize it once only, not again at run time.

- If you are using DB2 to DB2 private protocol, the server caches dynamic SQL statements only if any are present in the plan. Although static SQL statements are actually prepared dynamically at the server, they are not true dynamic SQL statements, and are thus not eligible for the cache.

3.1.2.2 Criteria for Prepared Statement Retrieval

DB2 searches the cache to compare the statement string the application supplies with the stored statement string. If the statement strings match according to the following rules, the application retrieves the prepared statement and the prepare step is avoided.

Consider the case where S1 and S2 are source statement strings, and P1 is the prepared statement for S1 in the cache. For DB2 to use P1 instead of preparing S2, the following must apply:

- S1 and S2 must be identical:
  - They must match character by character
  - There can be no intervening blanks in the text; however, any leading or trailing blanks are ignored.

If there is the slightest difference, DB2 cannot use P1 and instead prepares S2 and stores the prepared statement P2 in the cache.

- The same authorization ID must be used in preparing S2 and S1. The authorization ID that DB2 considers is:
• Current SQLID for DYNAMICRULES(RUN)
• Plan owner for DYNAMICRULES(BIND)
• Package owner for DDF server thread.

• The following bind options for S2 must match the bind time options in force for S1 at bind time.
  – CURRENTDATA
  – DYNAMICRULES
  – ISOLATION
  – SQLRULES
  – QUALIFIER.

• The PREPARE options for S2 must match the PREPARE options used for S1:
  – CURRENT DEGREE
  – CURRENT RULES.

If any one of the criteria is not met, DB2 must do a real prepare step and save the prepared statement in the cache.

It is clear that you need to implement closely controlled standards for SQL coding and bind options to take advantage of sharing prepared statements between different packages and plans.

We recommend the use of DYNAMICRULES(BIND) with the same plan and package owner to minimize authorization problems. The main benefits are for repetitive statement processing within an application, where all the retrieval criteria are automatically met.

3.1.3 Controlling the Cache
The system parameter CACHEDYN=YES/NO controls whether or not you have dynamic statement caching. You enable caching by specifying YES for CACHE DYNAMIC SQL on the Application Programming Defaults Panel 2 (DSNTIP4). The default is NO.

3.1.4 EDM Pool Considerations
In a data sharing environment, the cache is local to a single DB2 subsystem because each member of a data sharing group has its own EDM pool. There is no sharing of cached statements between members.

The EDM pool holds one skeleton copy of each prepared statement. The EDM pool does not treat dynamic statements like static statements, where copies of a statement are maintained in the EDM pool for each requester. Prepared statements are subject to a least recently used algorithm and can be discarded. The difference is that the dynamic statements are not stored in the DB2 directory on DASD; DB2 has to prepare them again if the prepared statements have been discarded.
3.1.4.1 How to Estimate EDM Pool Space Requirements

If you enable the cache option, you must increase the size of your EDM pool to accommodate the skeletons for your most frequently used dynamic SQL statements. If you do not do this, static statements are also discarded as the EDM pool nears its capacity, and the performance of static applications suffers.

You estimate the storage needed for the prepared statement cache using the following terminology:

- **Maxdynplans** is the maximum number of dynamic plans containing embedded dynamic SQL you expect to be running at any given time.
- **Dynstatsize** is the average statement size for a prepared statement in the cache. To calculate, add 1.6 KB for each single table statement and 0.2 KB for each additional table in the statement. For example, if you assume an average of three tables per per SQL statement, the average statement size is (1.6 + 0.2 +0.2)KB = 2 KB.
- **Dynstatexec** is the average number of dynamic SQL statements executed by plans containing dynamic SQL that are likely to be used repeatedly.
- **Adhocexec** is the maximum number of unique SQL statements likely to be used once only by plans that are executed by all users at a given time. You can estimate this by multiplying the maximum number of users running ad-hoc SQL programs and multiplying by 5.

Then calculate storage requirement as follows:

\[
\text{Storage in KB} = \text{dynstatsize} \times (\text{maxdynplans} \times \text{dynstatexec}) + \text{adhocexec}
\]

3.1.4.2 How to Monitor the EDM Pool

You can monitor the effectiveness of the prepared statement cache by using IFCIDs 0002, 0003, and 0148, added to the statistics trace in DB2 V5. The following fields are added to the EDM pool mapping macro DSNDQISE:

- **QISEDSC** shows the number of pages used for the cache.
- **QISEDSG** records the number of requests to search the cache.
- **QISEDSI** records the number of times a statement was inserted into the cache, which indicates it was not already in the cache.

To calculate how often statements are found ready prepared in the dynamic statement cache, use the formula:

\[
\text{hit ratio} = \frac{(\text{QISEDSG} - \text{QISEDSI})}{\text{QISEDSG}}
\]

DB2 PM V5 displays information from the new fields, and the prepared statement hit ratio, in the EDM pool activity block of the statistics report and in the online monitor EDM pool window. This helps you evaluate the effectiveness of the dynamic statement cache.

If your overall workload contains heavy use of dynamic SQL statements, many of which you execute repeatedly, then the hit ratio should be high, indicating that your selection of dynamic statement caching is a good option.

If the hit ratio is low you should consider that:

- There is a cost every time DB2 searches the cache and does not find the prepared statement in the cache.
• There is a very slight increase in the cost of preparing a statement that is not already in the cache. There is no benefit if the prepared statement is not executed again before being removed from the cache.

• The EDM pool is shared by both static and dynamic statements, so you may adversely affect the performance of static SQL applications.

Even if your hit ratio is low, you may still consider that the benefit from a few applications that run frequently, and execute the same dynamic SQL statements repeatedly, can justify the cache option.

You are unlikely to have a high hit ratio if your DB2 subsystem has a significant ad-hoc query usage.

We advise you to monitor the EDM pool hit ratios for cursor table, package table and DBD requests before and after you enable the dynamic statement cache. This enables you to assess the possible impact of caching dynamic SQL statements on the EDM pool usage for static SQL statements, and to make sure that your estimated EDM pool increase is sufficient.

### 3.1.5 Performance Measurements for Cached Statements

The two main objectives of this test are to:

• Test how effective retrieval of a prepared statement is from the cache compared with having to prepare the same statement each time it is executed.

• Compare the execution times for repetition of cached dynamic SQL statements with static SQL.

We use two examples to illustrate the effect of dynamic statement caching, one for a fetch of two rows with a cursor, the second for the insert of one row. Both the fetch of two rows and the insert of one row are repeated to test the effect of having the statements in cache for the second execution.

Table 1 shows a comparison of the CPU times for the fetch of two rows. The times for the first and second fetch of two rows are shown when the statement is run as dynamic and as static SQL.

<table>
<thead>
<tr>
<th></th>
<th>CPU Time for First Fetch of Two Rows</th>
<th>CPU Time for Second Fetch of Two Rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static SQL</td>
<td>Dynamic SQL</td>
<td>Delta %</td>
</tr>
<tr>
<td>0.0402</td>
<td>0.0619</td>
<td>+54</td>
</tr>
</tbody>
</table>

**Note:** All times are in seconds.

The first fetch of two rows shows the expected overhead for dynamic SQL compared to static SQL, mainly from the prepare step. The second fetch of two rows shows the reduction in CPU time when the prepared statement is retrieved from the cache. There is a very small overhead of around 0.0004 second for the second fetch of two rows for dynamic SQL compared to static SQL, partly due to searching the cache for the statement.
Table 2 on page 23 shows a comparison of the CPU time for an insert statement repeated twice. The times for the first and second insert are shown when the statement is run as dynamic and as static SQL.

<table>
<thead>
<tr>
<th>Static SQL</th>
<th>Dynamic SQL</th>
<th>Delta %</th>
<th>Static SQL</th>
<th>Dynamic SQL</th>
<th>Delta %</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0377</td>
<td>0.0604</td>
<td>+60</td>
<td>0.0330</td>
<td>0.0334</td>
<td>+1.2</td>
</tr>
</tbody>
</table>

Note: All times are in seconds.

The first insert shows the expected overhead for dynamic SQL compared to static SQL, mainly from the PREPARE step. The second insert shows the reduction in CPU time when the prepared statement is retrieved from the cache. There is only a very small overhead of 0.0004 second for the second insert for dynamic SQL compared with static SQL. The cost of dynamic SQL over static SQL is restricted to the first PREPARE step.

The conclusion from the study is that the overhead of a repetitively executed dynamic SQL statement is usually restricted to only the first execution of the statement. The more times an application executes a statement repetitively, the lower the overhead for dynamic SQL.

### 3.1.6 Summary for Dynamic Statement Caching

You should consider caching of dynamic SQL statements if a significant proportion of your workload is dynamic SQL which is frequently executed.

We particularly recommend caching if your important applications have many instances where the same dynamic SQL statement is executed repetitively within a loop. In this case you get performance almost equivalent to that of static SQL. The only overhead is in the first execution of the statement within the loop.

Because caching is a systemwide choice, you must increase the size of the EDM pool for the cache. Monitor the use of the EDM pool to make sure that there is no impact on static SQL statements, and that sufficient use is made of the cache to justify the extra storage.

You can also reuse statements between applications, but only if you have strict SQL and bind option control.

Again, we do not recommend caching if your dynamic SQL workload consists mainly of ad-hoc statements.

### 3.2 Keeping Prepared Statements Past Commit

In this section, we describe what is meant by keeping prepared statements past commit and the application programming considerations. We discuss how to keep prepared statements past a commit point, and the additional implications when prepared statement caching is active. We describe the storage requirements in the DBM1 address space.
3.2.1 Why Keep Prepared Statements Past Commit

In DB2 V4, you can only keep SELECT statements past a commit point, and only if you specify the WITH HOLD option in your DECLARE CURSOR statement. The use of WITH HOLD means you have an overhead because DBD locks for all dependent objects are also held past the commit point. All other statements must be prepared again after the commit point.

In DB2 V5, the use of prepared statement caching enables you to save dynamic SQL statements in a systemwide, or global, cache and to reuse them without the overhead of repreparation. With the prepared statement cache option alone, you must still have the PREPARE statement even though it may not have to be executed.

In addition you can now save dynamic SQL statements past a commit point by specifying KEEPDYNAMIC(YES). This enables an application to completely eliminate the PREPARE statement for a statement already prepared before the commit.

3.2.2 How to Keep Prepared Statements Past Commit

In this discussion you need to differentiate between the character string form of an SQL statement, the statement string, and the executable form of an SQL statement after it is prepared, the prepared statement.

To keep prepared statements past commit, use the new bind option KEEPDYNAMIC. Valid values are:

- KEEPDYNAMIC(NO)
  This default means DB2 discards prepared statements at commit points, unless they are associated with a cursor having the WITH HOLD option. The statement must be prepared again before it can be executed.

- KEEPDYNAMIC(YES)
  DB2 preserves prepared statements past a commit point. DB2 handles SQL statements differently, depending on whether or not you have the prepared statement cache option enabled on your DB2 system.

3.2.2.1 KEEPDYNAMIC(YES) without Prepared Statement Cache

DB2 keeps only the SQL statement string past a commit point. The prepared statement is discarded. If the application issues another OPEN, EXECUTE or DESCRIBE for the statement, then DB2 uses the saved statement string to prepare the statement. This is often called an implicit prepare. The prepared statement is kept by DB2 until:

- The application issues a PREPARE for the same statement.
- The application process ends.
- A rollback occurs.
- A commit occurs.

This can be illustrated by:
3.2.2.2 KEEPDYNAMIC(YES) with Prepared Statement Cache

If you specify KEEPDYNAMIC(YES) and have prepared statement caching active, DB2 retains a copy of both the statement string and the prepared statement. These copies are considered to be locally cached; the statement is also globally cached in the EDM pool.

DB2 retains both local statement string and prepared statement past a commit. The local copy of the prepared statement is marked “not in use.” When the application issues a subsequent EXECUTE, OPEN or DESCRIBE statement, DB2 uses the local copy of the prepared statement after validity checking. A DROP, ALTER, or REVOKE statement executed on a dependent object renders the local statement invalid.

The local instance of the prepared statement is kept, in the DBM1 address space until:

- The application issues a PREPARE for the same statement
  This causes the local prepared copy to be discarded, and the statement is prepared again.
- The application process ends.
- A rollback occurs.
- The kept local copy is removed on a least recently used basis.
- A DROP, ALTER or REVOKE statement is executed on an object the statement depends on.

This can be illustrated by:

```sql
PREPARE STMT1 FROM ..... Statement prepared and put in memory
EXECUTE STMT1
COMMIT

EXECUTE STMT1 ..... Application does not issue a PREPARE
COMMIT (DB2 issues an implicit prepare)

EXECUTE STMT1 ..... Again, no PREPARE needed
COMMIT
```

3.2.2.3 Application Coding Considerations

Consider the following code and note that the PREPARE statement for STMT2, which is executed many times in a loop, is removed from the loop:

```sql
PREPARE STMT1 FROM ..... Statement prepared and put in memory
EXECUTE STMT1
COMMIT

EXECUTE STMT1 ..... Application does not issue a PREPARE
COMMIT (DB2 uses the prepared statement in memory)

EXECUTE STMT1 ..... Again, no PREPARE needed
COMMIT
```
DECLARE CURSOR C1
WITH HOLD FOR STMT1

PREPARE STMT1 FROM
(SELECT RESPEMP FROM PROJ
  WHERE PRENDATE<CURRENT DATE)

DESCRIBE STMT1
INTO DESCRIPTOR :SQLDA

PREPARE STMT2 FROM
(SELECT EMP SET BONUS=?
  WHERE EMPNO=?)

OPEN C1

DO UNTIL SQLCODE=100
  FETCH FROM C1
    USING DESCRIPTOR :SQLDA
  EXECUTE STMT2
    USING DESCRIPTOR :SQLDA
  COMMIT
END

STMT2 is executed many times within a loop for each row returned from cursor C1, declared WITH HOLD. When a COMMIT statement is executed, DB2 retains position on cursor C1. DB2 retains the local copy of prepared statement STMT2 unless a PREPARE statement causes it to be discarded. The PREPARE statement for STMT2 is only needed once and must be outside the execution loop.

Existing applications must be recoded to eliminate unnecessary PREPARE requests following the commit point. If the application issues a PREPARE request for the same statement string, then the locally kept copy of the prepared statement is discarded and DB2 prepares the statement.

Your application must be able to handle SQLCODEs for implicit commits. DB2 keeps the application’s local copy of a prepared SQL statement past commit, marked “not in use.” If DB2 removes a “not in use” statement from the local cache during execution of an application, perhaps by executing a DROP, ALTER, or REVOKE statement on a dependent object, DB2 tries to retrieve the statement from the global cache.

If this attempt fails, DB2 must implicitly prepare the statement again. This probably occurs when the application next issues an OPEN, EXECUTE or DESCRIBE for the statement. The application must handle the possible SQLCODE from any error that may occur during the implicit PREPARE. An error also occurs if the application runs on a DB2 system at DB2 V4 or without dynamic cache.
3.2.2.4 Controlling Storage
You need to consider the effect of each application on DBM1 storage, and limit
the number of prepared statements to be kept locally past a commit point by any
applications bound with KEEP_DYNAMIC(YES).

DB2 uses the system parameter MAXKEEPD to provide control. You specify the
chosen value in the MAX KEPT DYN STMTS field of installation panel DSNTIPE.
The default of 5000 statements means storage of 9 MB is required, assuming an
average of 1.8 KB for each prepared SQL statement. You can specify zero in
this field to effectively disable KEEP_DYNAMIC(YES) globally. Statements are
discarded on a least recently used basis, and active statements are not
considered for discard.

*Instrumentation*: New SQL statement statistics are introduced to help monitor
how effective you are at keeping prepared statements past a commit point,
particularly when dynamic statement caching is enabled:

- QXSTFND—Number of successful cache searches
- QXSTNFND—Number of unsuccessful cache searches
- QXSTIPRP—Number of implicit PREPAREs
- QXSTNPRP—Number of PREPAREs avoided following a commit point
  when KEEP_DYNAMIC(YES) and cache are active
- QXSTDEXP—Number of times inactive prepared statements were discarded
  from DBM1 because the limit was exceeded
- QXSTDINV—Number of times a cached statement was purged from the cache
  because a DROP, ALTER, or REVOKE statement was executed.

3.2.2.5 Data Sharing
If the prepared statement cache is not active on all members, then an
application bound with KEEP_DYNAMIC(YES) must be implicitly prepared if the
statement is assigned to a member without the cache. For this reason, do not
introduce KEEP_DYNAMIC(YES) on any member until all members are at DB2 V5
and have the cache active.

3.2.3 Recommendations
You need to recode existing applications to benefit from keeping prepared
statements past a commit point. You also need to handle SQLCODES for implicit
PREPARE done when a OPEN, EXECUTE, or DESCRIBE is executed.

Maximum benefit is obtained if the cache option for prepared statements is
active. We expect you to have dynamic statement caching enabled if you wish to
exploit keeping prepared statements past a commit point. Remember to allow
for the extra storage in your DBM1 address space. Also, limit the use of
KEEP_DYNAMIC(YES) to only those applications with specific performance
requirements.
3.3 Reoptimization at Run Time

You can invoke a facility in DB2 V5 to reconsider the access path at run time, when the values for host variables, parameter markers and special registers are known. This process, called reoptimization, should be considered before trying to force an access path.

We discuss changes that must be made to enable existing dynamic SQL applications to use reoptimization and how to minimize performance overheads when you write new applications. We discuss also the use of EXPLAIN facility and how you can monitor reoptimization of SQL queries at run time. The changes to catalog tables are described. The performance improvement for queries with reoptimization is shown.

3.3.1 Why Reoptimization Is Required

All SQL queries, both static and dynamic go through a bind optimization process before they can be run. During the bind process, DB2 determines the best access path based on column predicates and on the statistics available in the catalog.

At bind time, DB2 knows only the actual values of any literals in predicates; it does not know the actual values of the following variables:

- Input host variables
- Parameter markers
- Values in special registers, for example CURRENT TIMESTAMP.

For static SQL, bind and execution occur at different times, unless an incremental bind is required when VALIDATE(RUN) is used. For dynamic SQL with parameter markers, access path selection occurs when the statement is prepared. The host variables associated with the parameter markers are not passed to DB2 until the OPEN or EXECUTE statement is executed.

With variables, the optimizer must use default filter factors to estimate the number of rows to eliminate for each predicate. The optimizer cannot make full use of available catalog statistics: for example, frequency statistics in SYSIBM.SYSCOLDIST are ignored. In most cases the optimizer still makes a good decision but in some cases it does not. The contents of a variable may be critical to access path selection and subsequent performance, for example it may determine which partition range of a partitioned table to scan.

In DB2 V4, if you have a performance problem with the optimization of variables, you often have to force a change of access path. When you do force a change of access paths, for example by rewriting a query, the access paths may then change when you install maintenance or migrate to a new release of DB2.

Another method of addressing a performance problem is to update the catalog. However, this can introduce unpredictable problems elsewhere. You must also remember to redo the update after each time RUNSTATS is run. It is important to record all catalog changes and be prepared to undo them. If you must update the catalog, then always ensure there is a well qualified WHERE clause in every update statement, and consider running the update in SPUFI with AUTOCOMMIT(NO) to check how many rows are potentially updated.
Query optimization improves with every release of DB2, so see if any existing query problems are resolved in DB2 V5 before you consider reoptimization.

### 3.3.2 How to Reoptimize

DB2 V5 introduces new options for package and plan BIND and REBIND commands:

- **NOREOPT(VARS)**
  - This is the default. No redetermination of access path occurs at run time.
- **REOPT(VARS)**
  - Redetermination of access path occurs at run time.

With REOPT(VARS) DB2 considers all statements with input variables for reoptimization. Any statements without input variables are not considered.

DB2 V5 introduces the DEFER(PREPARE) bind option for packages. Use of this option causes the preparation of the statements in the package to be deferred for application-directed access using remote packages. The default is NODEFER(PREPARE). In DB2 V4, this bind option is available only for plans and applies to DB2 to DB2 private-protocol access only.

REOPT(VARS) implies DEFER(PREPARE). If you specify REOPT(VARS), you must defer preparation of dynamic SQL statements until the OPEN or EXECUTE statement is executed. So you cannot specify NODEFER(PREPARE).

DB2 V5 introduces the KEEPDYNAMIC option for plans and packages to enable you to keep dynamic SQL statements after a commit point. The default is KEEPDYNAMIC(NO). If you specify REOPT(VARS), you cannot keep dynamic SQL statements after commit points so KEEPDYNAMIC(YES) cannot be used.

#### 3.3.2.1 Reoptimization of Static SQL Statements

If you specify REOPT(VARS) the processing order for static SQL is:

1. At bind time DB2 carries out the normal process:
   - Check authorizations.
   - Identify run time validation requirements.
   - Select access paths.
   - Update the PLAN_TABLE if EXPLAIN(YES) is specified.
   - Generate the plan or package.

2. At run time, DB2 selects the access path based on the known value of the host variable or special register:
   - If no errors at bind time, the SQL statement is reparsed.
   - For statements bound with VALIDATE(RUN) DB2 resolves outstanding validations and reoptimizes variables during a single incremental bind.
   - Reoptimization occurs:
     - Every time a cursor statement is opened.
     - Every time a noncursor statement is executed.

**Consideration for Applications:** For a cursor statement, reoptimization occurs only once, not for each execution of the FETCH statement in a loop. Reoptimization occurs every time you execute a noncursor statement in a loop.
You have to close the cursor by issuing a CLOSE CURSOR, COMMIT, or ROLLBACK statement, and reopen the cursor for reoptimization to reoccur.

Design your new applications in such a manner that you can separate queries that benefit from reoptimization from those that do not need it. This enables you to limit reoptimization of static statements and prevent unnecessary minibinds at run time.

### 3.3.2.2 Reoptimization of Dynamic SQL Statements

If you specify REOPT(VARS), the processing for dynamic SQL is done as follows:

DB2 defers the prepare processing and returns SQLCODE 0, unless you issue a DESCRIBE statement.

If you issue a DESCRIBE statement before an OPEN or EXECUTE statement, DB2 prepares the statement without input variables, and prepares the statement again with input variables when the OPEN or EXECUTE statement is executed.

When the OPEN or EXECUTE statement is executed, DB2 prepares the SQL statement, optimizes the access path using the actual values that the application supplies, and returns the SQLCODE for the deferred PREPARE statement.

**Note:** The OPEN statement refers to dynamic SQL SELECT statements.  
The EXECUTE statement refers to dynamic SQL non-SELECT statements.

**Consideration for Applications:** You must check existing applications to ensure that they can handle SQLCODEs for prepares which are issued by OPEN or EXECUTE statement when you select REOPT(VARS).

We recommend that you consider REOPT(VARS) for application programs that contain dynamic SQL statements with input variables. All statements go through a bind process; it is better to have the actual values for input variables available to get the best access path selected.

You should also ensure that you do not prepare the same statement twice, that is both before and after the input variables are known. You should:

- Issue a DESCRIBE statement after the OPEN or EXECUTE statement.
- Avoid the use of the PREPARE INTO statement before you open a cursor. 
  PREPARE INTO statement causes DESCRIBE statement to be executed implicitly, causing the preparation of the SQL statement both before and after you open a cursor.
- Handle SQLCODEs because the SQL statement is prepared when OPEN or EXECUTE statement is executed.
  Issuing a DESCRIBE statement always causes the SQL statement to be prepared twice if you use DB2 to DB2 private protocol. Therefore, consider using application-directed distribution for new applications.

### 3.3.3 How to Use EXPLAIN with Reoptimization

There are considerable differences in how you use the EXPLAIN facility with REOPT(VARS) for static and dynamic SQL statements.
3.3.3.1 EXPLAIN for Static SQL Statements
There are differences in the use of the EXPLAIN facility for static SQL statements with REOPT(VARS) that depend on the choice of the VALIDATE option. Here we assume the use of EXPLAIN(YES).

**VALIDATE(BIND):** This is the normal bind option for plans and packages. The actual values of input variables are not known at bind time and the PLAN_TABLE entries reflect the access paths without reoptimization, using default filter factors, as in DB2 V4.

To see the new access path determined at run time, it is necessary to use the trace record for instrumentation facility component ID (IFCID) 0022.

**VALIDATE(RUN):** If the bind process for a statement is successful, DB2 inserts PLAN_TABLE entries at bind time, without reoptimization.

If the bind process for a statement is unsuccessful, then there is no entry in PLAN_TABLE at bind time. DB2 inserts the reoptimized access path into PLAN_TABLE at run time.

3.3.3.2 EXPLAIN for Dynamic SQL Statements
If you specify EXPLAIN(YES) the entries in PLAN_TABLE show the access path optimized with the actual values of input variables at run time. If you DESCRIBE the statement before OPEN or EXECUTE you also have additional entries in PLAN_TABLE showing the access path using default filter factors. Because the additional rows represent an avoidable extra preparation of the statement, do not use this method merely to compare access paths with and without reoptimization.

EXPLAIN(YES) is useful to obtain reoptimized access path information for a statement. Rebind the plan or package with EXPLAIN(NO) after a short initial test period, or you produce PLAN_TABLE entries at every execution.

3.3.3.3 EXPLAIN with SPUFI, QMF, DB2 PM, and Visual Explain
You may use SPUFI, QMF, DB2 PM, or Visual Explain to get the access path information for your SQL statements interactively, one at a time, as you write them. QMF has the added benefit of producing a formatted report while DB2 PM and Visual Explain interpret the PLAN_TABLE entries for you.

In DB2 V4, if you have host variables in a predicate for an original query, and you wish to EXPLAIN the query interactively, you should replace each host variable with a parameter marker represented by a question mark (?). If you use a literal then DB2 uses a filter factor calculated from the literal value instead of the default filter factor.

In DB2 V5, if you bind the original query with REOPT(VARS), it is more useful to provide a typical literal value in the SQL statement because DB2 knows the real value of the host variable at run time. You can test the query with a range of literals to see if the access path changes according to the value of the literal.

Be careful that the length and precision of any test literal does not exceed the corresponding column definition or you make the predicate stage 2 and therefore nonindexable.
3.3.3.4 PLAN_TABLE Changes
DB2 V5 introduces a new PLAN_TABLE column WHEN_OPTIMIZE to indicate whether the access path is determined at bind time, or at run time using known values of input variables. This information is also recorded by IFCID 0022.

Table 3 describes values of the new column.

<table>
<thead>
<tr>
<th>Column Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>blank</td>
<td>A blank value indicates the access path was determined at bind time using a default filter factor for SQL statements which contain input host variables, parameter markers, or special registers. This is possible only if you have not used REOPT(VARS).</td>
</tr>
<tr>
<td>B</td>
<td>The access path was determined at bind time using a default filter factor for SQL statements that contain input host variables, parameter markers, or special registers. However, the statement will be reoptimized at execution time using values of input variables for host variables, parameter markers, or special registers. This indicates you have used REOPT(VARS) for static SQL, or for dynamic SQL with DESCRIBE or PREPARE INTO before the OPEN.</td>
</tr>
<tr>
<td>R</td>
<td>The access path was determined at execution time by using values of input variables for host variables, parameter markers, or special registers. This indicates you have used REOPT(VARS) for static SQL requiring an incremental bind at run time or, for dynamic SQL</td>
</tr>
</tbody>
</table>

The new column will have an initial value of blank.

3.3.4 How to Monitor Reoptimization
You may wish to monitor the execution of plans and packages to determine the reoptimized access path at run time for statements with input variables. This information is not available in PLAN_TABLE for static SQL statements bound with the recommended option of VALIDATE(BIND).

IFCID 0022 enables you to record the reoptimized access path information at runtime. It records access path information in the same format as PLAN_TABLE, except that the record always contains the actual access path with input variables at run time.

We recommend monitoring with IFCID 0022 to examine reoptimization of dynamic SQL statements rather than regular use of EXPLAIN(YES). Performance class 3 activates IFCID 0022, but you incur less overhead if you use performance class 30, 31, or 32 and specify only the IFCIDs you require. DB2 PM record trace can be used to format the records.

IFCID 0247 can be used to see the actual values for input variables. This is for serviceability use only in DB2 V4 but is for general use in DB2 V5.

You should always limit the number of trace records. Therefore, you should qualify the trace. For example, to start a trace to activate IFCIDs 0022 and 0247 for plan P1, and record the output in system management facility (SMF) you specify the command:
You can also use online DB2 PM to capture the trace records to your own user data set.

When you format the report, for example with a DB2 PM record trace, you can identify the statement and relate it both to the PLAN_TABLE entries and to SYSIBM.SYSSTMT or SYSIBM.SYSPACKSTMT. You also see the actual input variables for the query.

You have an additional use for IFCID 0247 in the debugging of application programs. A common use for the DB2 PM record trace is to examine the last SQLCODE an application program issues before the application program ends abnormally (abends). You now also have the ability to record input variables to give even more valuable diagnostics.

### 3.3.5 Catalog Table Changes

DB2 V5 has changes to four catalog tables to support reoptimization:

- **SYSIBM.SYSPLAN**
  
  REOPTVAR column added, CHAR(1) not null.
  
  Indicates if the access path is redetermined at execution time using input variables. Values can be:
  
  - N—NOREOPT(VARS) bind option, determine access path at bind time.
  
  - Y—REOPT(VARS) bind option, determine access path at run time for SQL statements with input variables.

- **SYSIBM.SYSPACKAGE**
  
  REOPTVAR column added with same purpose and allowable values as REOPTVAR in SYSIBM.SYSPLAN.
  
  DEFERPREPARE column added, CHAR(1) not null.
  
  Indicates whether prepare processing is deferred until the OPEN statement is executed. Values can be:
  
  - N—NODEFER(PREPARE) bind option, prepare processing is not deferred until OPEN statement is executed.
  
  - Y—DEFER(PREPARE) bind option, prepare processing is deferred until OPEN statement is executed.
  
  - blank—Bind option is not specified for the package; it is inherited from the plan.

- **SYSIBM.SYSSTMT**
  
  STATUS column added, CHAR(1) not null.
  
  Indicates the status of the bind process. Descriptions of the possible values are shown in Table 4 on page 34.
The values and purpose of the STATUS column in SYSIBM.SYSSTMT are the same as in SYSIBM.SYSSTMT.

### Table 4. Values of Column STATUS in SYSIBM.SYSSTMT

<table>
<thead>
<tr>
<th>Column Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Distributed—Statement uses DB2 private protocol (system directed access). The statement is parsed and executed at the server using defaults for input variables during access path selection.</td>
</tr>
<tr>
<td>B</td>
<td>Distributed—Statement uses DB2 private protocol (system directed access). The statement is parsed and executed at the server using values for input variables during access path selection.</td>
</tr>
<tr>
<td>C</td>
<td>Compiled—Statement was bound successfully using defaults for input variables during access path selection.</td>
</tr>
<tr>
<td>D</td>
<td>Interpretive—Statement is interpreted at execution time. This is mainly for DDL.</td>
</tr>
<tr>
<td>E</td>
<td>Explain—Statement is an SQL EXPLAIN statement. The EXPLAIN is done at bind time using defaults for input variables during access path selection.</td>
</tr>
<tr>
<td>F</td>
<td>Parsed—Statement did not bind successfully and VALIDATE(RUN) was used. This statement is rebound at execution time using values for input variables during access path selection.</td>
</tr>
<tr>
<td>G</td>
<td>Compiled—Statement bound successfully but REOPT is specified. This statement is rebound at execution time using values for host variables during access path selection.</td>
</tr>
<tr>
<td>H</td>
<td>Parsed—Statement did not bind successfully and VALIDATE(RUN) was used. The statement is rebound at execution time using defaults for input variables during access path selection.</td>
</tr>
<tr>
<td>I</td>
<td>Indefinite—Statement is dynamic. This statement will be bound at execution time using defaults for input variables during access path selection.</td>
</tr>
<tr>
<td>J</td>
<td>Indefinite—Statement is dynamic. This statement will be bound at execution time using values for input variables during access path selection.</td>
</tr>
<tr>
<td>K</td>
<td>Control—CALL statement.</td>
</tr>
<tr>
<td>L</td>
<td>Bad—The statement has some allowable error. The bind continues but the statement cannot be executed.</td>
</tr>
<tr>
<td>blank</td>
<td>The statement is not executable or was bound in a DB2 release prior to DB2 V5.</td>
</tr>
</tbody>
</table>

3.3.5.1 Useful Catalog Queries
You can determine which of your SQL statements will be optimized at run time using actual input variable values by running the following SQL queries:

- **For plans:**
  ```sql
  SELECT PLNAME, STMTNO, SEQNO, TEXT
  FROM SYSIBM.SYSSTMT
  WHERE STATUS IN ('B', 'F', 'G', 'J')
  ORDER BY PLNAME, STMTNO, SEQNO
  ```

- **For packages:**
  ```sql
  SELECT COLLID, NAME, VERSION, STMTNO, SEQNO, STMT
  FROM SYSIBM.SYSPACKSTMT
  WHERE STATUS IN ('B', 'F', 'G', 'J')
  ORDER BY COLLID, NAME, VERSION, STMTNO, SEQNO
  ```
To determine which plans and packages will be incrementally bound because they originally have the VALIDATE(RUN) bind option and the bind is unsuccessful, run the following SQL queries:

- For plans:
  ```sql
  SELECT DISTINCT NAME
  FROM SYSIBM.SYSSTMT
  WHERE STATUS IN ('F', 'H')
  ```

- For packages:
  ```sql
  SELECT COLLID, NAME, VERSION
  FROM SYSIBM.SYSPACKSTMT
  WHERE STATUS IN ('F', 'H')
  ```

No index is defined on column STATUS in either SYSIBM.SYSSTMT or SYSIBM.SYSPACKSTMT. You may find it highly useful to create an index to improve the performance of these catalog queries in your own environment.

### 3.3.6 Reoptimization Performance Measurements

The test environment was a 1 GB database comprising seven partitioned tables. Eight queries were examined using static and dynamic SQL with and without REOPT(VARS). Host variables were used as the input variables for static SQL, while parameter markers were used as the input variables for dynamic SQL. In addition, the queries were run using dynamic SQL but with literal values rather than with parameter markers to set a base level for elapsed times.

The following traces were activated:

- Statistics Class (1,3,4)
- Accounting Class (1,2,3)
- Performance Class (31) IFCID (0022,0221,0222,0231)

DB2 PM was used to provide accounting information, to report on access path reoptimization at run time, and to analyze the degree and mode of parallelism.

The following comparisons are made:

- Compare reoptimized access path selection and nonreoptimized access path selection for static SQL with host variables.
- Compare reoptimized access path selection and nonreoptimized access path selection for dynamic SQL with parameter markers.
- Compare reoptimized access path selection and nonreoptimized access path selection for static SQL with literals.
- Compare reoptimized access path selection and nonreoptimized access path selection for dynamic SQL with literals.

All seven queries for both static and dynamic SQL show an identical access path change on reoptimization.

The queries were of the following type:

- Query 1, single table select with column function
- Query 2, three-table join with column function, GROUP BY, and ORDER BY
- Query 3, six-table join with column function, GROUP BY, and ORDER BY
• Query 4, six-table join with GROUP BY, and ORDER BY
• Query 5, eight-table join with GROUP BY, and ORDER BY
• Query 6, six-table join with column function, GROUP BY, and ORDER BY
• Query 7, four-table join with column function, GROUP BY, and ORDER BY.

3.3.6.1 Comparison of REOPT(VARS) with NOREOPT(VARS)
The results are examined separately for static and dynamic SQL.

**Static SQL:** Table 5 shows measurements relating to query performance using REOPT(VARS) and NOREOPT(VARS) for static SQL.

<table>
<thead>
<tr>
<th>QUERY</th>
<th>Elapsed Time with NOREOPT(VARS)</th>
<th>Elapsed Time with REOPT(VARS)</th>
<th>Elapsed Time Delta, REOPT(VARS) minus NOREOPT(VARS)</th>
<th>Elapsed Time Delta %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query 1</td>
<td>308</td>
<td>225</td>
<td>-83</td>
<td>-27</td>
</tr>
<tr>
<td>Query 2</td>
<td>381</td>
<td>38</td>
<td>-343</td>
<td>-90</td>
</tr>
<tr>
<td>Query 3</td>
<td>2951</td>
<td>111</td>
<td>-2840</td>
<td>-96</td>
</tr>
<tr>
<td>Query 4</td>
<td>617</td>
<td>548</td>
<td>-69</td>
<td>-11</td>
</tr>
<tr>
<td>Query 5</td>
<td>56</td>
<td>59</td>
<td>+3</td>
<td>+5</td>
</tr>
<tr>
<td>Query 6</td>
<td>709</td>
<td>147</td>
<td>-562</td>
<td>-79</td>
</tr>
<tr>
<td>Query 7</td>
<td>858</td>
<td>90</td>
<td>-768</td>
<td>-90</td>
</tr>
</tbody>
</table>

**Note:** All times are in seconds.

The access path changes with reoptimization at run time for all the queries. The changes occur because DB2 now knows the actual values for input variables and no longer uses default filter factors.

Savings in elapsed time occur for seven of the eight queries as the result of improved optimization. One query has the same elapsed time despite a change in access path.

The actual access path changes that occur are summarized below:

• Query 1 changes from a table space scan with a parallel degree of 7 to an index scan with a parallel degree of 10.
• Query 2 includes a composite table sort.
• Query 3 changes the join method and sequence of tables.
• Query 4 changes the join method and sequence of tables.
• Query 5 changes the join method and the degree of parallelism.
• Query 6 adds a composite table sort and changes the degree of parallelism.
• Query 7 uses a different index and reduced degree of parallelism.

**Dynamic SQL** Table 6 on page 37 shows measurements relating to query performance using REOPT(VARS) and NOREOPT(VARS) for dynamic SQL.
The same access path changes occur for reoptimization of the queries with dynamic SQL and parameter markers as for static SQL and host variables. The elapsed time savings are significant for six of the seven queries.

### 3.3.6.2 Comparison of REOPT(VARS) with Use of Literals

The results are examined separately for static and dynamic SQL.

#### Static SQL:
Table 7 shows measurements relating to query performance using literals and parameter markers with REOPT(VARS) for static SQL.

<table>
<thead>
<tr>
<th>QUERY</th>
<th>Elapsed Time with Literals</th>
<th>Elapsed Time with REOPT(VARS)</th>
<th>Elapsed Time Delta, REOPT(VARS) minus Literals</th>
<th>Elapsed Time Delta %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query 1</td>
<td>224</td>
<td>225</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>Query 2</td>
<td>38</td>
<td>38</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Query 3</td>
<td>114</td>
<td>111</td>
<td>−3</td>
<td>−3</td>
</tr>
<tr>
<td>Query 4</td>
<td>546</td>
<td>548</td>
<td>+2</td>
<td>+1</td>
</tr>
<tr>
<td>Query 5</td>
<td>56</td>
<td>59</td>
<td>+3</td>
<td>+5</td>
</tr>
<tr>
<td>Query 6</td>
<td>145</td>
<td>147</td>
<td>+2</td>
<td>+1</td>
</tr>
<tr>
<td>Query 7</td>
<td>91</td>
<td>90</td>
<td>−1</td>
<td>−1</td>
</tr>
</tbody>
</table>

**Note:** All times are in seconds.

There is very little difference in the performance of the queries that go through the reoptimization process when literals are used. Exactly the same access paths are observed whether the statement is prepared with literals or is reoptimized after the input host variables are replaced with the actual values at run time.

#### Dynamic SQL:
Table 8 on page 38 shows measurements relating to query performance using literals and parameter markers with REOPT(VARS) for dynamic SQL.
There is very little difference in the performance of the queries when they go through the reoptimization process when literals are used. Exactly the same access paths are observed whether the statement is prepared with literals or is reoptimized after the parameter markers in the input variables are replaced with the actual values at run time.

Although there are actually some small increases in the times for the test with literals, there should in fact be no significant difference under the same conditions. Work file preformatting during the test run with literals almost certainly accounts for this increased time.

### 3.3.6.3 Comparison of REOPT(VARS) for Static and Dynamic SQL

Table 9 shows measurements relating to query performance using REOPT(VARS) for dynamic and static SQL.

<table>
<thead>
<tr>
<th>QUERY</th>
<th>Elapsed Time Static SQL</th>
<th>Elapsed Time Dynamic SQL</th>
<th>Elapsed Time Delta, Dynamic SQL minus Static SQL</th>
<th>Elapsed Time Delta %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query 1</td>
<td>224</td>
<td>221</td>
<td>−3</td>
<td>−1.3</td>
</tr>
<tr>
<td>Query 2</td>
<td>36</td>
<td>36</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Query 3</td>
<td>104</td>
<td>106</td>
<td>+2</td>
<td>+1.9</td>
</tr>
<tr>
<td>Query 4</td>
<td>547</td>
<td>545</td>
<td>−2</td>
<td>−0.4</td>
</tr>
<tr>
<td>Query 5</td>
<td>56</td>
<td>56</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Query 6</td>
<td>145</td>
<td>146</td>
<td>+1</td>
<td>+0.7</td>
</tr>
<tr>
<td>Query 7</td>
<td>96</td>
<td>95</td>
<td>−1</td>
<td>−1.0</td>
</tr>
</tbody>
</table>

**Note:** All times are in seconds.

There is no significant difference in the elapsed times for equivalent static and dynamic SQL statements that are reoptimized. The access paths are the same for each query.
3.3.6.4 Conclusion of Reoptimization Performance Measurements

We conclude that:

- The elapsed times for reoptimized queries are identical to those selected when there are literals in the queries instead of input variables.
- Both static and dynamic SQL show equivalent access paths and elapsed times when reoptimized.
- Reoptimization shows significant elapsed time improvements for some SQL statements when default filter factors are avoided.
- Not all queries benefit from reoptimization. Many show a change in access path; those that do usually have a reduced elapsed time.

3.3.7 Considerations for Reoptimization at Run Time

You can get significant benefits in elapsed time for SQL queries reoptimized at run time when the actual values of input variables are known.

To implement reoptimization for dynamic SQL, make sure that your existing applications can handle SQLCODEs normally returned when the PREPARE statement is executed. Always consider reoptimization for new dynamic SQL applications that have statements with input variables, and code new dynamic SQL applications to avoid preparing statements twice, with and without values for input variables.

For static SQL, be very selective in the use of REOPT(VARS). Restrict the use of REOPT(VARS) to statements whose access path is so inappropriate that they have a totally unacceptable response time. Do not incur the reoptimization overhead for online transactions that already perform within your requirements. Separate the code in new applications so that you bind with REOPT(VARS) only those packages containing statements that require reoptimization at run time.

There are likely to be greater potential savings in batch jobs where the value of a host variable or special register can limit the range of partitions scanned for a partitioned table.

3.4 Use of Key Column Correlation Statistics

Here we describe what is meant by key column correlation, and what the impact of key correlation is in terms of access path selection and query performance. We describe how query access paths can be improved by using the statistics collected on correlated key columns using the new RUNSTATS options in DB2 V5. The performance improvement when correlated statistics are available is shown for queries with predicates on correlated key columns.

3.4.1 Definition of Key Column Correlation

Two columns of data are correlated when the value of one column is related to the value of the other column. For example consider a query on a table with a three column index on CITY, STATE and ZIPCODE:

```
SELECT * FROM PLACE_TABLE
WHERE STATE = 'CA'
    AND CITY = 'SAN FRANCISCO'
```

In DB2 V4, the optimizer assumes that the column rows are independent and that the following combinations all have the same chance of occurring:
• CITY = 'SAN FRANCISCO' AND STATE = 'CA'
• CITY = 'LOS ANGELES' AND STATE = 'CA'
• CITY = 'LOS ANGELES' AND STATE = 'NY'
• CITY = 'SAN FRANCISCO' AND STATE = 'NY'.

However, the value of STATE is related to the value of CITY, so that all combinations cannot occur, and the columns are said to be correlated.

In the example, the optimizer underestimates the number of qualifying rows for the query because it assumes it can apply filter factors for both columns independently. The optimizer assumes the index is the best one to use. It is unaware that STATE does virtually no additional filtering once the value of CITY has been applied and that far more rows qualify than the optimizer assumes. Although the optimizer detects a matching index scan on two columns CITY and STATE the filtering is virtually equivalent to a matching index scan on only the first column CITY.

3.4.2 How Queries Can Benefit from DB2 V5 Statistics

In DB2 V5, RUNSTATS can collect cardinality statistics on a set of concatenated columns of an index instead of just the first key column (FIRSTKEYCARDF) and all key columns (FULLKEYCARDF).

RUNSTATS can also collect frequent value statistics on a concatenation of columns in SYSIBM.SYSCOLDIST and SYSIBM.SYSCOLDISTSTATS instead of statistics only on the first key column. By holding the frequency of the concatenated key column values 'SAN FRANCISCO,CA' the optimizer now knows the percentage of rows that match the query and can establish an exact filter factor for access by the index.

3.4.2.1 Column Correlation on Matching Predicates

If all columns of an index are matched, then filter factors are established from FULLKEYCARD in SYSIBM.SYSINDEXES and SYSIBM.SYSINDEXSTATS. SYSIBM.SYSINDEXSTATS. Concatenated key column statistics provide accurate filter factor estimates for local equal predicates on less than fully matching columns of an index. An example of the type of query that benefits is:

SELECT * FROM TABLE
WHERE C1 = 'A'
     AND C2 = 'B'
     AND C3 = 'C'

where

• An index exists on C1, C2, C3, C4.
• The first three index columns have matching predicates (MATCHCOLS 3).
• Some or all of the index columns are correlated.

You can collect the statistics to improve this type of query using RUNSTATS.
3.4.2.2 Column Correlation and Index Screening

Column correlation may also occur on nonmatching index columns used for index screening. In this case the predicates do not filter as much data as their filter factors suggest. An example of this type of query is:

```
SELECT * FROM TABLE
WHERE C2 = 'B'
  AND C3 = 'C'
```

where

- An index exists on C1, C2, C3, C4.
- There are no matching index key columns (MATCHCOLS 0).
- Columns C2 and C3 are correlated.

You must update the catalog manually to provide the statistics to improve this type of query. RUNSTATS collects correlation statistics only on a concatenation of index columns that includes the first key column.

3.4.2.3 Column Correlation on Table Joins

Joins can benefit from a better join method and join sequence because the optimizer uses filter factors to determine the correct method with which to join tables. If the optimizer incorrectly assumes that too few rows are collected by predicates on correlated columns then either the wrong join sequence may be chosen for a nested loop join, or a better join type may be available.

3.4.3 How Access Path Selection Is Influenced by Correlated Statistics

We show here a more detailed example of a query with matching equal predicates and how collecting statistics on correlated index columns influences the access path chosen. The access path is first shown for DB2 V4 and then for DB2 V5 after collecting statistics on the correlated index columns.

Consider the following holding data about the current and previous storage sites for parts. Table and index definitions are:
Table Definition
-------------------

CREATE TABLE PART_HISTORY ,
  (PART_TYPE CHAR(2) , -- TYPE OF PART
   PART_SUFFIX CHAR(10) , -- PART IDENTIFIER
   W_NOW INTEGER , -- WHERE THE PART IS NOW
   W_FROM INTEGER , -- WHERE THE PART CAME FROM
   DEVIATIONS INTEGER ,
   COMMENTS CHAR(254),
   DESCRIPTION CHAR(254),
   DATE1 DATE ,
   DATE2 DATE ,
   DATE3 DATE );

Index Definitions
-------------------

CREATE UNIQUE INDEX IX1
  (PART_TYPE , (W_NOW ,
   PART_SUFFIX , W_FROM ,
   W_NOW , DATE1 ,
   W_FROM));

CREATE UNIQUE INDEX IX2
  (W_NOW ,
   W_FROM ,
   DATE1);

The following catalog statistics are available in DB2 V4:

<table>
<thead>
<tr>
<th>Table Statistics</th>
<th>Index Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARD 100000</td>
<td>FIRSTKEYCARD 1000</td>
</tr>
<tr>
<td>NPAGES 10000</td>
<td>FULLKEYCARD 100000</td>
</tr>
<tr>
<td></td>
<td>CLUSTERRATIO 99</td>
</tr>
<tr>
<td></td>
<td>NLEAF 3000</td>
</tr>
<tr>
<td></td>
<td>NLEVELS 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Column Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLUMN</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>PART_TYPE</td>
</tr>
<tr>
<td>W_NOW</td>
</tr>
<tr>
<td>W_FROM</td>
</tr>
</tbody>
</table>

Access Path for DB2 V4: The following query using available statistics:

```
SELECT * FROM PART_HISTORY
WHERE PART_TYPE = 'BB'
AND W_NOW = 3
AND W_FROM = 3
```
In the example in Table 10, use of index IX2 appears to be more efficient and the optimizer selects it. The optimizer assumes that W_NOW and W_FROM are independent columns. However, 50% of parts are in the same location they came from and have the same value of W_NOW and W_FROM. The optimizer thus does not make a proper decision.

**Access Path for DB2 V5:** We now look at the same query in DB2 V5. There is no change in access path until RUNSTATS is run to collect statistics on the correlated key columns of index IX2. From the results, your knowledge of the data model will tell you whether correlation exists.

The following statistics are recorded in the DB2 catalog after running RUNSTATS with the KEYCARD option, and are now available in SYSIBM.SYSCOLDIST:

<table>
<thead>
<tr>
<th>Concatenated Key Column Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>C</td>
</tr>
</tbody>
</table>

**Note:** Cardinality information on the concatenation of all index columns, “1,2,3, and 4” for index IX1 and “3,4, and 8” for Index IX2, is held in the FULLKEYCARDF column of SYSIBM.SYSCINDEXES.

With this additional information, the optimizer now calculates a revised filter factor of 0.01 for index IX2 based on the concatenated cardinality for Columns 3
and 4 of 100. The filter factor is unchanged for index IX1. The optimizer now knows that 20 index leaf pages are to be read for index IX2 compared with only 3 for index IX1, so the optimizer selects the correct index, IX1.

### 3.4.4 Query Performance Measurement with Key Column Correlation Statistics

To illustrate the benefits of key column correlation statistics on query performance, we use two examples of complex nested loop joins, where some of the matching index key columns for equal predicates are correlated.

One query is a seven-table nested loop join and the other query is an eight-table nested loop join. Each query was run with DB2 V4 statistics available and then repeated after running RUNSTATS with the KEYCARD option to collect the cardinality statistics for the concatenated columns of all indexes with more than two columns. Indexes with only two columns already have full cardinality information in SYSIBM.SYSINDEXES and SYSIBM.SYSINDEXSTATS columns FIRSTKEYCARD and FULLKEYCARD.

Table 11 shows a comparison of the query CPU consumption with statistics provided without and with the KEYCARD option.

<table>
<thead>
<tr>
<th>Table 11. Query Performance Enhancement by RUNSTATS KEYCARD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seven-Table Nested Loop Join</td>
</tr>
<tr>
<td>--------------------------------</td>
</tr>
<tr>
<td>CPU Time Without KEYCARD</td>
</tr>
<tr>
<td>18</td>
</tr>
</tbody>
</table>

Note: All times are in seconds.

The seven-table join shows a 50% saving in CPU time and the eight-table join shows a 17% saving in CPU time. This saving is a combination of better index access and better sequence selection for the table joins.

### 3.4.5 Conclusion for Use of Key Column Correlation Statistics

Key column correlation statistics are primarily aimed at queries with less than fully matching equal index predicates on correlated index key columns and can provide substantial CPU and elapsed time savings.

If you know that certain key columns in your indexes are highly correlated, consider collecting concatenated key column statistics. You may prefer first to determine whether you actually have a query performance problem, and if the problem is likely to arise from correlation.

Queries with nonmatching index predicates can also benefit, but the benefit is offset by the need to manually update the catalog. Manual update requires extreme caution and may cause unpredictable access paths for other queries.
3.5 Nonpartitioning Index Pieces

DB2 V5 introduces a new option so you can specify a maximum size for each data set (piece) of a nonpartitioning index. By specifying a smaller size, you can have more pieces and therefore greater control over where to place those pieces. Placing the pieces on separate I/O paths can reduce I/O contention for SQL operations that access nonpartitioning indexes during read or update processing.

The use of pieces spreads out the index pages of a nonpartitioning index in a table space by a collection of physical data sets. A new keyword PIECESIZE on the CREATE and ALTER INDEX specifies PIECESIZE. The value must be a power of two and you can indicate the amount in kilo-, mega-, and gigabytes. The default is 2 GB for regular table spaces and 4 GB for LARGE table spaces; it can be as small as 256 KB.

For example,

```
CREATE TYPE 2 INDEX ON DBEES.TABLE1(COL1,COL2) PIECESIZE 128M;
```

Figure 1 shows that in DB2 V4, multiple inserts from several partition index partitions or data partitions result in I/O contention on the nonpartitioning index. In DB2 V5, pieces provide multiple I/O access paths to the the nonpartitioning index and inserts are done in parallel. Figure 2 on page 46 shows the elapsed time improvement from using the nonpartitioning index pieces.
The job inserts 75000 rows. The nonpartitioning index pieces with a single data set need 24.3 minutes elapsed time. Having 10 pieces allows inserts to go in 10 parallel streams and reduces the elapsed time to 7.2 minutes. The improvement is more significant when multiple tasks (many inserts, updates, or deletes) are accessing the nonpartitioning index, since all the tasks must read and update the nonpartitioning index.

### 3.6 Enhanced Limited Partition Scan

DB2 can limit the scan of data in a partitioned table space to one or more partitions. This method is called limited partition scan, and means that DB2, from the values provided in the predicate, determines the lower and the higher partition that can possibly contain the rows that are requested, and limits the scan to that range of partitions.

To take advantage of this method, the query must provide a predicate on the first key column of the partitioning index. Only the first key column is significant for limiting the range of the partition scan.

In DB2 V4, if there are multiple predicates, DB2 eliminates partitions only if the predicates are connected by the AND operator.

Limited partition scan has also been used in DB2 V4 for each table accessed in a join, as long as the access is sequential.

If the query exploits query parallelism, DB2 V4 uses limited partition scan only for single table queries.
3.6.1 DB2 V5 Improvements

Only predicates that contain the first key column of the partitioning index are significant when determining which partitions to eliminate from the scan, as in DB2 V4.

Partition scan is enhanced compared with DB2 V4, as DB2 V5 eliminates partitions when multiple predicates are connected by OR.

**Note:** If multiple predicates are connected by the OR operator, and one of the predicates does not refer to the first key column of the partitioning index, DB2 does not use limited partition scan.

Limited partition scan can also be used on each table accessed in a join processed in parallel. DB2 V5 determines which individual partitions can possibly contain records to be retrieved, and limits the scan to these partitions only.

Let us assume a table, TABLE1, with a partitioning index on column C1 and five partitions. Partition 1 contains values of C1 up to 1500, partition 2 contains values of C1 from 1501 to 3000, partition 3 contains values of C1 from 3001 to 4500, partition 4 contains values of C1 from 4501 to 6000, and the last partition contains values of C1 above 6000.

Consider the following query:

```sql
SELECT ....
FROM TABLE1
WHERE
  C1 BETWEEN $1000$ AND $2000$
OR
  C1 BETWEEN $7000$ AND $8000$
```

Here, DB2 disqualifies partitions 3 and 4, and limits the scan to partitions 1, 2, and 5.

Whether this query executes in parallel or not depends on many factors, although the setting of the DEGREE keyword in the bind process is the most important. DEGREE=ANY makes the queries in a package eligible for parallelism. The optimizing process then decides if the query benefits from being executed in parallel groups.

3.6.2 Impact with REOPT(VARS)

Limited partition scan is not supported when host variables or parameter markers are used on the first key column of the partitioning index. This is because there is no way DB2 can disqualify one or more partitions as the actual values in the predicate are not known at bind time. If you have queries that might benefit from limited partition scan, but have host variables or parameter markers in the predicate instead of literals, consider binding the package or plan with REOPT(VARS). This makes the query eligible for limited partition scan, as the actual values in the predicate are known at time of the open processing, when the reoptimization takes place. See 3.3, “Reoptimization at Run Time” on page 28, where REOPT(VARS) are discussed in more detail.
3.7 Optimizer Enhancements

Stage 2 predicates are not indexable. Stage 1 predicates are indexable, although not all stage 1 predicates are indexable. Some predicates which were stage 2 in DB2 V4 are stage 1 in DB2 V5 and some of these are indexable. The optimizer can use the additional nonuniform distribution statistics that you can collect with RUNSTATS to arrive at more accurate filter factors resulting in better access paths. In this section we discuss the following optimizer enhancements:

3.7.1 Noncolumn Expressions

In DB2 V4, predicates with noncolumn expressions (expressions that do not contain a column reference), such as

```
SELECT ....
FROM
  TABLE2
WHERE
  COL1 = :HOSTV + 3
```

are stage 2 predicates.

In DB2 V5, simple predicates such as

```
COL Operator Noncolumn Expression
```

are stage 1 and indexable predicates, except when the operator is:

- `<>` (not equal to)
- `<=` (not equal to)
  - the predicate is stage 1, but not indexable
- `NOT BETWEEN`
  - the predicate is stage 1, but not indexable
- Some special cases:
  - `+ 0`
  - `−0`
  - `"0`
  - `/0`
  - `||"`
  - `CONCAT"

- the predicate is stage 1 but not indexable.

The reason these special cases are stage 1 but not indexable is that many applications use noncolumn expressions such as these to force the optimizer to select a different access path, for example another index, than it would have otherwise used.

To avoid changes in these applications, predicates with these special nonoperational noncolumn expressions continue to be not indexable.

Quantified predicates and predicates using EXISTS, IN, LIKE, and NULL are not changed in DB2 V5.
3.7.1.1 Impact with REOPT(VARS)
Predicates using noncolumn expressions that have been made indexable or stage 1 through this enhancement allow reoptimization to take place when REOPT(VARS) is specified.

This means that noncolumn expressions are considered variables in the reoptimization process, along with host variables, parameter markers, and special registers.

Thus, the following query will be reoptimized when REOPT(VARS) is specified:

```
SELECT ....
FROM TABLE1
WHERE COL1 = :HOSTV1 + COL2
AND COL3 >= CURRENT DATE - 14 DAYS
```

**Note:** In this example, it is not only the presence of a host variable (HOSTV1) that causes the reoptimization, but also the noncolumn expression (CURRENT DATE - 14 DAYS).

See 3.3, "Reoptimization at Run Time" on page 28, where REOPT(VARS) are discussed in more detail.

3.7.2 Scalar Functions
In DB2 V5, any scalar functions using noncolumn expressions are indexable.

In the following query, for instance, the predicate is stage 1 and indexable in DB2 V5:

```
SELECT ....
FROM TABLE1
WHERE COL1 = DECIMAL(:HOSTV,7,2)
```

**Note:** However, there are some exceptions. Please refer to DB2 for OS/390 Administration Guide for a complete list of exceptions.

3.7.3 Nonuniform Distribution Statistics
You can collect nonuniform distribution statistics for the first column of an index by executing the RUNSTATS INDEX utility. RUNSTATS inserts one row with the column value (COLVALUE) and frequency of that value (FREQUENCYF) into table SYSCOLDIST for each of the 10 most frequent values.

In DB2 V5, by specifying FREQVAL with NUMCOLS and COUNT, RUNSTATS also collects nonuniform distribution statistics for concatenated index columns; NUMCOLS specifying the number of concatenated columns to collect for, COUNT specifying the number of most frequent values to collect.

When nonuniform distribution statistics are collected, either for the first column of an index, or for concatenated index columns, whenever a predicate references the first or the concatenated columns, DB2 matches the values from the predicate against the values in column COLVALUE in the SYSCOLDIST table. If
a match is found the frequency of that value in the table, recorded in column FREQUENCYF, is used as filter factor for that predicate.

If there is no match—that is, the value from the predicate is not within the number of most frequent values that have been collected—DB2 assumes that the remainder of values in the table are uniformly distributed, and calculates the filter factor to use accordingly.

5.4, "RUNSTATS" on page 90 discusses how and when to collect key correlated column statistics, and 3.4, "Use of Key Column Correlation Statistics" on page 39 discusses the use of key correlated-column statistics in the query environment.

The following example illustrates another new exploitation of nonuniform distribution statistics in DB2 V5 when there is a LIKE predicate with a wild card.

The SQL SELECT is on a table from which nonuniform distribution statistics has been collected on the first column of the index (RUNSTATS INDEX without FREQVAL).

```sql
SELECT ... FROM TABLE1 WHERE COL3 LIKE '%%IF%%'
```

DB2 V4 calculates the filter factor based on information found in the statistics and the length of, and the number of possible alternatives (blank, A - Z, 0 - 9), per position in the predicate.

This gives a very low filter factor, which tells DB2 that probably very few rows have to be read from the table and returned. In this and in many other cases, assuming uniform distribution of values on the index column throughout the table, may lead DB2 to select access paths that are not optimal in regard to the number of rows returned.

If nonuniform distribution statistics are collected for this column, then, if possible DB2 V5 uses numbers calculated by RUNSTATS in the process of establishing a filter factor for the predicate.

Let us assume that nonuniform distribution statistics have been collected on indexed column COL3, and that the SYSCOLDIST table contains the values in Table 12 on page 51:
Column COLVALUE in the third row in Table 12 contains the string “%RIF%” provided in the predicate. In this case, where the string in the predicate is found among the 10 most frequent values of column COL3, DB2 V5 uses the value in column FREQUENCYF in table SYSCOLDIST as the filter factor for this predicate.

The filter factor from column FREQUENCYF in the SYSCOLDIST table, 0.0676, indicates that this statement returns more rows than was indicated by the low filter factor calculated and used in DB2 V4.

As the number of rows returned is one of the factors influencing the access path (access method and index use), DB2 V5 is able to select better access paths than earlier releases did.

There may be additional rows in that table that contains the string “RIF” in column COL3, but the filter factor provided in the nonuniform distribution statistics in the SYSCOLDIST table still gives DB2 V5 a better indication on how many rows are likely to be retrieved than when assuming uniform distribution throughout the table.

Even if the string is not found among the values collected in the SYSCOLDIST table, collecting nonuniform distribution statistics enables DB2 V5 to select better access paths. It assumes uniform distribution for the “no match values” among the remaining rows—that is the rows not holding any of the x most frequent values.

Collecting nonuniform distribution statistics, and often for more than the 10 most frequent values, improves both query CPU time and elapsed time. Especially for tables with static data and frequent access, nonuniform distribution statistics can be of benefit. (See 3.4, “Use of Key Column Correlation Statistics” on page 39 for guidelines and a recommendation.)

3.7.4 Enabling Parallel Access

When the plan or package is bound with DEGREE(ANY) or the CURRENT DEGREE special register is set to ANY, then in releases prior to DB2 V5, the optimizer comes up first with the best sequential plan at bind time, without considering parallelism. It then determines whether parallelism can be enabled, and the sequential plan is refined further into a parallel plan to support this.
For example, for a particular query, the optimizer determines that the use of a type-N hybrid join is the best method for sequential access. However, since parallelism is not supported for this type of join, no refinement is done to the sequential plan.

In DB2 V5, the optimizer considers whether parallelism is possible with the best sequential plan. If parallelism is not possible, then the next best sequential plan that allows for parallelism is chosen. Thus, in the example just discussed, the optimizer does not choose type-N hybrid join.
Chapter 4. Query Parallelism

DB2 V5 provides additional types of queries that can use parallelism and introduces Sysplex query parallelism, which extends parallel processing to allow a single query to use all the processing power of a data sharing group.

Query parallelism is a way of reducing the elapsed time for long running queries which can either be I/O intensive or CP intensive. An I/O-intensive query scans large volumes of data and requires considerable I/O processing and minimal CPU processing. A CP-intensive query usually performs sorts and may have query functions such as joins on multiple tables, column and scalar functions, and grouping and ordering of data.

Query I/O parallelism was introduced in DB2 V3. See Chapter 2 of DB2 V3 Performance Topics (GG24-4284) for more information about query I/O parallelism.

Query CP parallelism in DB2 V4 extended the benefits of parallel processing to CP intensive queries. Query CP parallelism is designed to make full use of the total processing power in a central processing complex (CPC) and take advantage of the increased I/O bandwidth from partitioned table spaces. See Chapter 2 of DB2 V4 Non-Data-Sharing Performance Topics (SG24-4562) for more information about query CP parallelism.

The additional improvements to parallelism that are introduced in DB2 V5 are mentioned in 4.1, “Parallelism Enhancements.” We present Sysplex query parallelism in 4.2, “Sysplex Query Parallelism” on page 54.

4.1 Parallelism Enhancements

Several improvements to the way DB2 processes queries have implications for the choice of access paths. DB2 V5 exploitation of parallelism enhances the possibility of using parallelism for queries. Any query that is a candidate for CP parallelism is also a potential candidate for Sysplex query parallelism, if your system is set up to run Sysplex query parallelism (as in 4.2.2, “Configuring for Sysplex Query Parallelism” on page 56). The following is a list of enhancements in DB2 V5:

- INSERT with a subselect.
  In DB2 V3 Query I/O parallelism, and in DB2 V4 Query I/O or CP parallelism is not possible for subselect within inserts. In DB2 V5, for example, the subselect on the following INSERT statement can be run in parallel:

  ```sql
  INSERT INTO T1
  SELECT * FROM T2 WHERE COL1 > 'TPCD';
  ```
  The SELECT can be run in any parallel mode in DB2 V5.

- Nonpartitioned index scan.
  In DB2 V4, the scan of a nonpartitioned index is almost always done sequentially. In DB2 V5, the nonpartitioned index scan can be done in parallel and the use of PIECESIZE helps to further improve performance. The impact of using PIECESIZE results in improvements with nonpartitioning indexes for parallelism. See 3.5, “Nonpartitioning Index Pieces” on page 45 for further information.
GROUP BY and ORDER BY on the same key in join.

In DB2 V4, DB2 does only one sort if both GROUP BY and ORDER BY are specified on the same key in the SQL statement joining two or more tables. However, the sort is sequential. In DB2 V5, the sort is done in parallel.

Discontiguous partition scan for both single-table query and join.

Consider a partitioned table space with eight partitions. If the data to be retrieved is in Partitions 2, 4, and 6, then DB2 V4 scans the data in Partitions 2, 3, 4, 5, and 6. If you have parallelism enabled, the scan of these partitions is done in parallel. However, this is possible only for a single-table query.

In DB2 V5, only Partitions 2, 4, and 6 are scanned in parallel; Partitions 3 and 5 are not accessed. This is also possible for a query joining two or more tables and not just for a single-table query, as in DB2 V4. See 3.6, “Enhanced Limited Partition Scan” on page 46 for more information.

Central processor (CP) parallelism is enabled for queries specifying cursor with hold when isolation level is uncommitted read (UR) or cursor stability (CS).

The single qualified partition and a table space defined with only one partition are now enabled for CP parallelism. Support for this enhancement is also available for DB2 V4 with authorized program analysis report (APAR) PN88570.

4.2 Sysplex Query Parallelism

DB2 data sharing in a Parallel Sysplex configuration is a requirement for Sysplex query parallelism. The MVS level should be at MVS 5.2 or later releases. It is important that you are familiar with the environment that can influence the performance of Sysplex query parallelism.

We describe how to configure for Sysplex query parallelism in 4.2.2, “Configuring for Sysplex Query Parallelism” on page 56. The list of items is comprehensive and we give concise information about the aspects of each area.

We examine some possible Sysplex query parallelism configurations in 4.2.3, “Data Sharing Coordinator and Assistant Configurations” on page 64.

We evaluate the expectations of Sysplex query parallelism and present measurement results from a laboratory environment in 4.2.4, “Performance Measurements” on page 65.

Finally, in 4.2.5, “Conclusions and Recommendations” on page 70, we give additional recommendations and guidelines for implementing Sysplex query parallelism.

4.2.1 Introduction

In Sysplex query parallelism, members of a data sharing group process a single query and each member is on a separate CPC. DB2 V4 lets you split and run a query within a DB2 subsystem. With Sysplex query parallelism, DB2 V5 extends parallel processing to allow a query to use all the power of a data sharing group.

DB2 determines an optimal degree of parallelism based on estimated I/O and processing costs. Different DB2 members process different ranges of the data. Sysplex query parallelism now makes the Parallel Sysplex attractive as a
complex query environment as well. DB2’s solution allows update activity to occur alongside long-running complex queries.

DB2 has been evolving to enhance parallel processing for your applications. Figure 3 shows the evolution of parallelism from DB2 V3 to DB2 V5.

Figure 3 shows that DB2’s ability to process read-only queries in parallel commenced with the introduction of I/O parallelism in DB2 V3. Then DB2 V4 allowed a query to be processed by multiple tasks. Both these types of query parallelism were limited to the processing capacity available within a single CPC. Therefore, a query could not use any more processing power than was available for a single DB2 subsystem. DB2 V5 introduces Sysplex query parallelism, which allows DB2 to use the Parallel Sysplex data sharing technology to process a query across DB2 members in a data sharing group.

Applications that are primarily read only and are processor or I/O intensive can benefit from Sysplex query parallelism. Because a data sharing group can grow, you can put more power behind your queries even as those queries become increasingly complex and run on larger and larger sets of data. The ability to reduce elapsed time for long-running queries assumes that processing power is available.

Figure 4 on page 56 shows how several DB2 members in a data sharing group can use the full power of the CPCs to process the customer table in parallel. The members DB1G, DB2G, and DB3G all access different ranges of the Customer Table and the results are returned to the application issuing the SELECT statement.
If a query qualifies for parallel processing, DB2 V5 determines the optimal degree of parallelism at bind or prepare time. DB2 can decide to distribute parts of the query for processing to other members in the data sharing group or to process the query within the originating subsystem only.

4.2.2 Configuring for Sysplex Query Parallelism

To use Sysplex query parallelism, you must:

- Configure the coordinator and assistant members in the data sharing group.
- Place work files on a shared disk.
- Specify buffer pool thresholds.
- Allow for the effect of a number of online CPs.
- Increase storage requirements for IRLM.
- Set correct MVS WLM goals.
- Use partitioning for optimal parallel performance.
- Set limits using the resource limit specification table (RLST).

The following subsections assist you in understanding how the permutations of these values influence the performance of your Sysplex query parallelism environment.

4.2.2.1 Configuring for Coordinator and Assistant

To use Sysplex query parallelism, each member in a data sharing group participates as either a coordinator or an assistant. The coordinator is the member where the query originates. The assistants are the other members where parts of the query are sent. Figure 5 on page 57 shows how, for each member on the installation panel DSNTIPK, you can specify the role of the member in the data sharing group.
**Figure 5. Configuring Coordinator and Assistant**

1. A member can be both coordinator and assistant.

The coordinator and assistant are ZPARM values. When these parameters are active, the members can participate in Sysplex query parallelism.

Different configurations of coordinator and assistant setups are explained in 4.2.3, “Data Sharing Coordinator and Assistant Configurations” on page 64.

### 4.2.2.2 Work Files on Shared Disk

Sysplex query parallelism requires that all data sharing members that participate in processing parallel queries must have access to each other’s work files. When an assistant member is processing work on behalf of a coordinator member’s query, it uses its own work file. To coordinate the work, the work files must be accessible to all members.

For data sharing, we recommend that you keep your work files in a separate buffer pool that is not backed by a GBP. This recommendation is because work files can never be GBP-dependent. Use the same buffer pool number for work file table spaces on all data sharing group members. DB2 makes the assumption that all work file table spaces are in the same buffer pool and makes run-time decisions based upon this assumption.

### 4.2.2.3 Setting Buffer Pool Allocation Thresholds

Buffer pool resources are checked at run time and not at bind time. DB2 V5 provides a threshold parameter (VPXPSEQT) for changing the assisting parallel sequential threshold for the virtual buffer pool. This threshold determines the portion of the virtual buffer pool that is used for processing queries that originate from other members of the data sharing group.

Figure 6 on page 58 shows the relationships of the buffer pool thresholds.
The ALTER BUFFERPOOL command allows you to dynamically control the extent to which a DB2 member can participate as an assistant in Sysplex query parallelism at run time. If the VPXPSEQT=0, that member can not be used as a parallel assistant for a query using that buffer pool. To disable the member from participating in Sysplex query parallelism as an assistant, set either VPXPSEQT=0 or VPPSEQT=0 for buffer pools.

If the virtual buffer pool that is used by the assistant has a shortage of resources, it can cause the query to run at a degree of parallelism that is less than the optimal degree determined at bind time or it can cause a poor distribution of parallel tasks across the data sharing group.

Also, define relevant table spaces and indexes with GBPCACHE CHANGED, as it is unlikely that DB2’s processing of a large query repeatedly reads the same pages. Therefore there is no need to cache those pages in a GBP.

4.2.2.4 Allow for the Effect of the Number of Online CPs

The information about the number of CPs available to the data sharing group is saved in the shared communications area (SCA) in the coupling facility at DB2 startup. Any changes to the number of CPs online after startup are not reflected in the SCA. You need to stop and start the member to update the information in the SCA.

DB2 determines a parallel degree and a parallel mode at bind time. One of the factors taken into account by the optimizer is the number of CPs online.
As it is feasible to have a variation in the configuration, we recommend that you plan to run your binds when your data sharing group is configured the way it will be at run time. Any deviation from the number of CPs online at run time does not provide the optimal access path selection for DB2.

4.2.2.5 Increase Storage Requirements for IRLM
Sysplex query parallelism uses IRLM to notify messages when the coordinator communicates with the assistants and vice versa. The main use of these messages is to pass data between a specific assistant and the coordinator. Ensure that the MAXCSA parameter has sufficient storage specified in the IRLM startup procedure for these messages.

An estimation of the amount of extra storage required by an assisting IRLM is the number of parallel tasks that can run concurrently, divided by the number of members sharing the work. Multiply the result by 32 KB to get a storage estimate.

For example, if you have a data sharing group in which three members participate in processing Sysplex queries, a maximum of eight queries can be executing concurrently, and the highest number of tasks is approximately 30. The calculation is:

\[
\frac{8 \times 30}{3} = 80
\]

\[80 \times 32 \text{ KB} = 2.6 \text{ MB extra storage on each assistant IRLM}\]

Additional storage for the coordinator is approximately 200 KB for the messages that are sent.

4.2.2.6 Setting MVS Workload Management Goals
Define performance goals that meet your Sysplex query parallelism workload requirements. If you do not classify the work for the assistant DB2s, the following can occur:

- In compatibility mode, the part of the query that runs on the assistant is assigned the same dispatching priority as the DB2 database services address space.
- In goal mode, the part of the query that runs on the assistant is discretionary work.

A service class with a goal and an importance is called a service class period. You assign a performance goal to each service class period, such as a response time goal, and you indicate an importance level, showing how important it is to your business that the performance goal be achieved.

There are three kinds of goals: response-time, execution velocity, and discretionary.

- Response time goals indicate how quickly you want your work to be processed. Since response time goals are not appropriate for all kinds of work, such as long-running queries, execution velocity goals can be used instead.
- Execution velocity goals define how fast work should run when ready, without being delayed for processor, storage, or I/O access. Execution velocity goals are the most appropriate goal for started tasks, for long-running queries, or for batch work.
- Discretionary goals are for low-priority work for which you do not have any particular performance goal, for long-running queries, or for batch work. Workload management processes the work using resources not required to meet the goals of other service classes.

Performance periods are available for work that has variable resource requirements and for which your goals change as the work uses more resources. You specify a goal, an importance level, and a duration for a performance period. (Duration is the amount of service units that period should consume before going on the next goal.) As you go from one performance period to the next, you can change the type of goal. Goals should become less stringent going from one period to the next.

Figure 7 shows the decreasing stringency of goal and decreasing importance from period 1 to period 2.

<table>
<thead>
<tr>
<th>Service Class</th>
<th>DSVEL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Period 1</strong></td>
<td></td>
</tr>
<tr>
<td>Velocity</td>
<td>40</td>
</tr>
<tr>
<td>Importance</td>
<td>3</td>
</tr>
<tr>
<td>Duration</td>
<td>25000 SU</td>
</tr>
<tr>
<td><strong>Period 2</strong></td>
<td></td>
</tr>
<tr>
<td>Velocity</td>
<td>15</td>
</tr>
<tr>
<td>Importance</td>
<td>5</td>
</tr>
</tbody>
</table>

*Figure 7. Example of Multiple Performance Period Goals*

When WLM associates work with the service class DSVEL, the work consumes 25000 services in Period 1. If the work continues to use resources, WLM now associates the work with Period 2; velocity of 15 and an importance level of 5 apply for that workload. Use this period switching method to ensure that very long queries do not receive too much resources, or when your applications use queries with varying resource demands and you cannot identify the variations with a classification rule.

Response time and response time percentile goals are best used for work that has at least three transaction completions over 20 minutes of time. Consider these goals if your Sysplex query parallelism workload fits this profile and, in addition, your workload is more important than other workloads in the Parallel Sysplex. If your Sysplex query parallelism workload has too few completions to use response time goals, then consider using a velocity goal. The discretionary goal is a suitable goal for the Sysplex query parallelism workload when all other work is more important.

Classification rules are the rules workload management uses to associate a performance goal or a resource group (or both) with work by associating incoming work with a service class. Optionally, classification rules can also associate incoming work with a report class, similar to report performance groups. Classification rules work like the IEAICSxx PARMLIB member does in compatibility mode, although they are more flexible. The classification rules for a subsystem are specified in terms of transaction qualifiers such as job name, user ID, or transaction class.
Using the WLM interactive system productivity facility (ISPF) dialog you can define the goals for your Sysplex query parallelism workload. Coordinators and assistants can use different classification rules.

Figure 8 and Figure 9 on page 62 show examples of how to define your WLM classification definitions for your Sysplex query parallelism workload. One definition is for the coordinator as the query is originating from time sharing option (TSO) and one definition is for the assistants as the distributed parts of the query come from within the data sharing group.

**Figure 8. WLM Classification Rules for the Coordinator TSO Queries**

Figure 8 shows the classification for the the coordinator queries that originate from TSO. WLM associates work from user IDs (UI) that start with GRP9 to the service class DSVEL and work from the GRPAnnnn user IDs with the service class DSRT1. If the user IDs GRPAnnnn have accounting information (AI) of D10* then this work is associated with service class DSRTX. Any other work that arrives from user IDs that do not meet the classification rules is assigned the default service class DSDEFLT.
### WLM Classification Rules for the Assistant TSO Queries

Figure 9 shows the classification for the assistants.

1. The DB2 in the subsystem type is for all the DB2s in the Parallel Sysplex where goal mode is active.
2. The subsystem instance (SI) level 1 classification rule associates query work originating from other members of the data sharing group. If your system is not running at OS/390 Release 3, you no longer use different service class names for the assistant work. In Figure 9, the service classes XQnnnnnn are different from the services classes DSnnnnnn defined for the coordinator in Figure 8 on page 61.
3. Report classes can be the same.

For more information about using workload management see *OS/390 MVS Planning: Workload Management* GC28-1761.

#### 4.2.2.7 Partitioning for Optimal Parallel Performance

To exploit Sysplex query parallelism, the number of partitions should be increased to a number close to the total number of the CPs that you plan to allocate for processing queries across the data sharing group. For CP-intensive queries, specify a number of partitions that is equal to the number of CPs in the system that you plan to allocate for the Sysplex query parallelism workload. Do not include CPs that are dedicated to other more important workloads.

For I/O-intensive queries, an approach is to calculate the ratio of elapsed time to CPU time. Multiply this ratio by the number of Sysplex query parallelism CPs that you plan to allocate for this workload. The rounded-up number determines how many partitions you can use, provided that the I/O configuration is balanced to match the number of possible parallel I/O tasks.

Also, try to balance the data in each part, as the degree of parallelism is influenced by skews in the partitions. In most cases, DB2 divides the table space into logical pieces based on the size of the largest physical partition.
Suppose a table consists of 20000 pages and 10 physical partitions and the largest of these contains 10000 pages, then DB2 is most likely to create only two logical pieces and the degree of parallelism is 2. If the same table has evenly sized partitions of 1000 pages each and the query is I/O intensive, then it results in a more optimal degree of parallelism of 20 and reduced elapsed time.

4.2.2.8 Setting Limits Using the Resource Limit Specification Table

DB2’s resource limit specification table helps to control dynamic SQL queries that use Sysplex query parallelism. Resource limits are local in scope to ensure that a single dynamic query does not overrun any DB2 subsystem. Data sharing group members can share the same RLST, or each member can have its own RLST.

If more than one parallel group processes the work for a query on a member, then each parallel group can use the service unit limit set for that member. However, the coordinator works differently in this situation because the total service units for all parallel groups are governed by the limit on the coordinating member. Figure 10 shows how the RLST definitions work.

You can disable query parallelism for dynamic SQL statements by specifying the following value in the RLFFUNC column in the RLF specification table:

- 3 = Query I/O parallelism is disabled
- 4 = Query CP parallelism is disabled
- 5 = Sysplex query parallelism is disabled
4.2.3 Data Sharing Coordinator and Assistant Configurations

All the components in 4.2.2, "Configuring for Sysplex Query Parallelism" on page 56 can influence how your Sysplex query parallelism environment runs. Figure 11 and Figure 12 on page 65 show how different settings can give two different profiles to Sysplex query parallelism configurations.

Figure 11. Separate Query Processing Configuration. In this configuration, the transaction managers (IMSA, IMSB, CICSY, and CICSZ), TSO, and BATCH members do not assist in query parallelism.

In Figure 11 1. The coordinator subsystem parameter is set to YES so the transaction managers can accept queries but as the virtual buffer pool settings for 2 VPPSEQT and VPXPSEQT are set to zero, the DB2 members DB1G, DB2G, DB3G, and DB4G do not process the parallel queries. In addition, 3 the assistant subsystem parameter is set to NO and therefore DB1G, DB2G, DB3G, and DB4G do not accept parallel queries from any other member in the data sharing group (DB0G).

The designated query processors CPC6 and CPC8 process the parallel queries for the transaction managers as DB6G and DB8G have 4 the ASSISTANT subsystem parameter set to YES and all the virtual buffer pool is available for parallelism.

CPC5 and CPC7 which run the TSO and batch workloads have the same settings for the coordinator and assistant parameters as the transaction managers. But by specifying 5 VPPSEQT = 70, this parameter setting allows DB5G and DB7G to run queries for their respective TSO and batch systems, but they do not assist other members.
In Figure 12, all members can coordinate and assist parallel queries as the subsystem parameters are set to YES. The virtual buffer pool sequential thresholds are high as this data sharing group is dedicated to query processing.

4.2.4 Performance Measurements

Sysplex query parallelism performance improvements come from further elapsed time reduction because query work can be spread across more resources such as CPs, number of channels, and buffer pools. The scalability of Sysplex query parallelism is linear in most cases. As the amount of parallel processing grows, then the performance improvement grows correspondingly. The cost of managing the Sysplex query parallelism incurs minimal system overhead.

4.2.4.1 Measurement Environment

The measurement environment was as follows:

- A data sharing group with two members running on a Parallel Sysplex with 9672-RX2 and a coupling facility 9674-C02.
- All members coordinator and assistant parameters set to YES.
- 30 partition table spaces with 30 GB of data.
- I/O configuration of eight RAMAC2 controllers with four partitions on seven controllers and two partitions on the eighth controller.
4.2.4.2 Measurement Results

The measurement results presented in this section were run in a controlled laboratory environment using specific workloads. There is no guarantee that you will have the same or similar results in your environment. Performance results depend upon the workload characteristics and the software and hardware configurations.

Figure 13, Figure 14 on page 67, and Figure 15 on page 67 show the results from comparisons of Sequential, I/O Parallelism, CP Parallelism, and Sysplex query parallelism for queries.

Figure 13 shows the elapsed times for an I/O-intensive query and a CP-intensive query with Sequential, I/O Parallelism, CP Parallelism, and Sysplex query parallelism access. 1 The sequential access needs 160.6 minutes to complete and the I/O parallelism access takes 22.5 minutes. The I/O parallelism access is a 7.1 times improvement in comparison to sequential. The CP parallelism access needs 5.6 minutes, which is a 28.7 improvement in comparison to sequential.

2 The Sysplex query parallelism results show a minor improvement for the I/O-intensive query over CP parallelism but a significant improvement for a CP-intensive query as would be expected with more CPs available. However, if the I/O bandwidth were widened, then Sysplex query parallelism would have shown significant reduction in elapsed time as well. The measurement environment does not change the I/O configuration, but when increasing the processing resources you should also increase the I/O resources to obtain the benefit of Sysplex query parallelism.

4 With I/O parallelism, elapsed time is greater than or equal to CPU time for the Sequential access as only one task is spawning multiple I/O streams.
Figure 14. Table Space Scan Queries CPU Time. This figure shows CPU time comparison between I/O-intensive and CP-intensive queries.

Figure 14 shows the CPU times for the table space scan queries. The CPU overhead is quite small for all types of query parallelism. The evaluation of these results with the elapsed time reductions in Figure 13 on page 66 shows that the benefits of using Sysplex query parallelism are affordable in CPU costs.

Figure 15. Different Query Types: Comparison by Elapsed Time

Figure 15 shows that the reduction in elapsed times for a sample of query types that are normally found in a decision-support workload. These results provide you with an indication of how much the improvement gains in elapsed time can be derived from either I/O-intensive or CP-intensive types of queries.

1 The Select Distinct query sequential access needs 172.3 minutes to complete and the Sysplex query parallelism access takes 17.4 minutes which is an improvement by 9.9 times. However it is only a minor improvement in
comparison to the CP parallelism result. 2 The Select Count, in-list Sysplex query parallelism access shows an improvement by 27 times the elapsed time for sequential. The Sysplex query parallelism access is a two-times improvement in comparison to the CP parallelism results.

Figure 16, Figure 17 on page 69, and Figure 18 on page 69 show the scalability of Sysplex query parallelism by elapsed time and throughput.

Figure 16 shows how the scalability of your queries influences performance.
1 These results show the balanced query scales for Sysplex query parallelism as additional CPs provide a reduction in elapsed-time for Sysplex query parallelism in comparison to the CP parallelism access. 2 Queries that produce a large answer set show only a minor elapsed-time improvement for Sysplex query parallelism in comparison to CP parallelism because of the sequential nature of returning the rows.

Some queries do not scale well. For example, a query which selects a limited range of data, presumably using an index, may require access to only a small number of partitions and therefore may not utilize all the processing power within a data sharing member. When such a query is split to run on multiple members, the work has been offloaded to make the coordinator less CPU busy, but the elapsed time may not be significantly reduced. Another example would be serial processing by an application program. A query that produces a large answer set waits for the application, as DB2 can deliver data faster than the application can process it.
Figure 17. Sysplex Scalability: Comparison by Elapsed Time

Figure 17 shows an example of a very CP-intensive query which can continue to exploit the processing power of a DB2 data sharing group as more and more CPCs are added. Each CPC has one DB2 member. This query achieved greater than 90% scalability in all cases.

Figure 18. Query Throughput Scalability: Comparison by Number of Queries per Hour

Figure 18 shows the results of a mixture of short- and long-running queries, which were submitted as distributed requests from a separate DB2 system. The WLM performance goals were set up to use different service periods to give
priority to some short and important queries while preventing the very long
queries from monopolizing the system resources.

The measurement of 10 CPs shows 44 queries completed during the hour; with
20 CPs in a Sysplex, 81 queries were completed. The proportion of short- and
long-running queries doubled, which shows excellent scalability and consistent
response times for queries in a Sysplex managed by WLM, and using Sysplex
query parallelism

4.2.5 Conclusions and Recommendations

Sysplex query parallelism in DB2 V5 provides significant improvements in
elapsed time with minimal increase in CPU time for your application. When your
application requires more CPU resources for your queries, the use of Sysplex
query parallelism assists in reducing the elapsed times. The results from 4.2.4.2,
“Measurement Results” on page 66 show that Sysplex query parallelism is an
important addition to the query parallelism options.

To assure that your workloads benefit from Sysplex query parallelism, you must
measure and control the resources used by parallel operations in your data
sharing group. All the factors mentioned in 4.2.2, “Configuring for Sysplex Query
Parallelism” on page 56 can combine, so that the environment for Sysplex query
parallelism can vary depending upon parameter settings.

The throughput of your queries is controlled specifically either by MVS Workload
Manager in goal mode, or by SYS1.PARMLIB members IEAIPSxx, IEAOPTxx, and
IEAICSxx in compatibility mode. An incorrect setting in the MVS parameters can
easily cause a wrong distribution of resources within the data sharing group.
Because Sysplex query parallelism queries use CPU resources, configure your
MVS environment to match your goals for the Sysplex query parallelism
workload.

We recommend you partition for optimal performance by determining if a query
is I/O or CP intensive. For CP-intensive queries, specify a number of partitions
that is equal to the number of CPs in the system that you plan to allocate for the
Sysplex query parallelism workload. For I/O-intensive queries, use the ratio
calculation in 4.2.2.7, “Partitioning for Optimal Parallel Performance” on page 62.

Monitor the usage of your buffer pool thresholds. In DB2 V5, the DISPLAY
BUFFERPOOL command displays information about all the thresholds. You can
determine the amount of buffer pool space available for parallel processing and
how much is available to assist with processing queries from other members in
the data sharing group.

Identify queries that are candidates for Sysplex query parallelism and monitor
the resource utilization of the coordinator and assistants. Any query that is a
candidate for query CP parallelism is a potential candidate for Sysplex query
parallelism. See 4.1, “Parallelism Enhancements” on page 53 for more
information about candidate queries.

When DB2 executes a Sysplex query, the elapsed time is affected by dynamic
factors such as CPU utilization, I/O contention, DB2 buffer pool availability, and
cross-system coupling facility (XCF) capacity. As with CP parallelism, to
determine if performance tuning is needed to further improve the elapsed time
for a query, look at CPU utilization and elapsed time for each parallel task. Use
the information from the Class 8 performance trace records IFCIDs 0221 and 0231 to examine the tasks, specifically the ones with the longest times.

Because of the cost of transferring locks to the coordinating DB2 member, Sysplex query parallelism is not used if a query uses isolation repeatable read (RR) or read stability (RS). It is unlikely that long-running queries would use an isolation of RR or RS, but to provide this capability with Sysplex query parallelism, bind with CS or UR and issue a LOCK TABLE tname in SHARE MODE statement before running the query.
Chapter 5. Utility Performance Enhancements

In this chapter, we describe the DB2 V5 performance-related enhancements to the following DB2 utilities:

- LOAD and REORG
  A new option, SORTKEYS, reduces elapsed time for LOAD and REORG jobs. A second new option, PREFORMAT, enables preformatting of unused pages after the loading of data in LOAD and REORG jobs. Another new option, NOSYSREC, reduces elapsed time for REORG jobs. The new inline copy feature produces an image copy data set parallel to the LOAD and REORG operation and thus make the table space available sooner.

- RECOVER INDEX
  Reduces CPU time because of reductions in path length in its unload phase.

- RUNSTATS
  The new sampling option of the RUNSTATS utility improves performance by reducing the processing required to collect nonindexed column statistics.

- COPY
  The CHANGELIMIT option of the COPY utility provides an easy way to determine whether an image copy needs to be taken, and whether the copy should be full or incremental.

- Online REORG
  The SHRLEVEL option of the REORG utility lets you choose the level of access you allow to your data during the reorganization. The levels you can choose are NONE, REFERENCE, and CHANGE where CHANGE permits read/write access to your data during almost the entire reorganization.

5.1 LOAD and REORG

DB2 V5 delivers extensive improvements to the LOAD and REORG utilities that reduce processor costs and elapsed time and increase the availability of your data.

- A change in the LOAD utility improves performance in the reload phase.

- A new keyword, PREFORMAT, enables preformatting of the remaining, unused pages in the current VSAM data sets (table space, index space, or partitions of those) following the reloading of data.

- A new keyword, SORTKEYS, significantly reduces I/O to intermediate work data sets and thereby reduces elapsed time.

- Two new keywords, COPYDDN and RECOVERYDDN, produce inline image copy data sets and make the table space available sooner by eliminating the need for a separate copy step.
5.1.1 PREFORMAT

Specify PREFORMAT when you want to preformat the unused pages in a new table space, or reset a table space (or partition of a table space) and all index spaces associated with the table space after the data has been loaded and the indexes built.

The PREFORMAT option can be specified as a main option of the LOAD utility or as a suboption of the PART option in the “into table spec” declaration. In the REORG utility, the PREFORMAT option is specified as an option within the “reorg options” declarations.

Examples of the LOAD and REORG statements with the PREFORMAT option:

- LOAD
  - LOAD DATA PREFORMAT INTO TABLE tname
  - LOAD DATA INTO TABLE tname PART 1 PREFORMAT

- REORG
  - REORG TABLESPACE tsname PREFORMAT
  - REORG INDEX ixname PREFORMAT
  - REORG TABLESPACE tsname PART 1 PREFORMAT
  - REORG INDEX ixname PART 1 PREFORMAT

After the data has been loaded and the indexes built, the PREFORMAT option on the LOAD and REORG utility command statement directs the utilities to format all the unused pages, starting with the high-used relative byte address (RBA) plus 1 (first unused page) up to the high-allocated RBA within the allocated space to store data. After preformatting, the high-used RBA and the high-allocated RBA are equal. Once the preformatted space is used up and DB2 V5 has to extend the table space, normal data set extending and formatting occurs.

During insert processing, DB2 preformats new pages before they can be used. This preformatting is performed in blocks as pages are needed.

Especially for insert-intensive applications, the delays caused by the preformatting process can affect the performance. To avoid the performance impact and also to achieve consistent execution times, the preformatting technique can be useful. If the final size of the table space can be predicted, and the access to the data has a high insert-to-read ratio, especially during the fill-up period, this technique should be considered.

5.1.1.1 User-Managed Data Sets Compared with DB2-Managed Data Sets

Since a user-managed data set is not deleted and reallocated during utility processing, the size remains unchanged, or is increased if data is added. The PREFORMAT option preformats all unused allocated pages in the actual table space, so this also includes secondary extensions that may have been previously allocated.

A DB2-managed data set is deleted and reallocated during REORG and LOAD REPLACE utilities. This means it is resized to the original allocation size and retains that size even if the amount of data being loaded does not fill the primary
allocation. Preformatting a DB2-managed table space always causes the full primary allocation amount of a data set to be preformatted.

For both user-managed and DB2-managed data sets if the data set goes into secondary extents during the utility processing, the high-allocated RBA becomes the end of the secondary extent and that becomes the high value for preformatting.

### 5.1.2 SORTKEYS

Specify SORTKEYS when you do not want to write extracted keys to the SYSUT1 or SORTOUT data sets, but to pass them in memory to the sort process.

The new syntax of the LOAD and REORG statements is

- **LOAD**
  - LOAD DATA SORTKEYS estimate INTO TABLE tname
- **REORG**
  - REORG TABLESPACE tsname SORTKEYS

The estimate on the LOAD statement is an estimate of the number of keys for this job to sort. This estimate is passed to sort for efficient allocation of memory and work data sets for the sort. If the estimate is omitted or specified as zero, LOAD writes extracted keys to SYSUT1 during the reload phase as if SORTKEYS were not specified. It then invokes the sort phase and after the keys have been sorted, they are passed to the build phase in memory. In this case, the performance improvement is approximately half as much as it would have been if an estimate had been provided, because the I/O to the SYSUT1 data set in the sort phase cannot be avoided.

Specifying SORTKEYS means that you do not want to use SYSUT1 and SORTOUT as intermediate work data sets for sorting extracted keys. However, you still must allocate both data sets when executing a LOAD utility because they are used in processing errors detected during the reload, validate, or enforce phase, and in passing foreign keys from the sort phase to the enforce phase.

Appendix A, "Estimates When SORTKEYS Is Used" on page 169, explains how to estimate the number of keys to sort, and the sizes for work data sets for the LOAD utility.

No estimate can be provided for REORG because REORG calculates the correct numbers during the unload phase. Also, if SORTKEYS is used, neither the SYSUT1 nor the SORTOUT data set is used for REORG.

### 5.1.2.1 Processing with and without SORTKEYS

Let us take a look at the phases observed, for instance through utility report messages, for LOAD and REORG utilities and the actual processing that occurs with and without SORTKEYS specified.

Figure 19 on page 76 shows the phases and the actual processing in versions previous to DB2 V5, and in DB2 V5 without SORTKEYS specified.

For example, issue the following control statement to achieve the processing shown in Figure 19 on page 76:
LOAD DATA INDDN EMPIDS
INTO TABLE DSN8510.EMP

Figure 19. LOAD and REORG Processing without SORTKEYS

1. During the reload phase, the data is loaded into the table space, and keys are extracted and written to the SYSUT1 data set.

2. The reload phase completes, and the sort phase starts.

3. The sort phase consist of five steps:
   a. LOAD or REORG links to sort and sort initiates.
   b. The records containing the keys are read from the SYSUT1 data set and passed to sort.
   c. Sort finishes sorting the records.
   d. Sort passes the records back to LOAD or REORG, and they are written to the SORTOUT data set.
   e. Sort terminates, and LOAD or REORG finish sort phase processing.

4. The build phase reads the sorted key records from the SORTOUT data set and uses them to build indexes. LOAD also uses the sorted key records to enforce referential constraints. For LOAD, foreign keys are sorted to the end of the SORTOUT data set and are read in a later phase.

Figure 20 on page 77 shows the phases and the actual processing in DB2 V5 LOAD and REORG with SORTKEYS specified (LOAD with the estimated number of keys supplied).

For example issue the following control statement to achieve the processing shown in Figure 20 on page 77:

LOAD DATA REPLACE INDDN INPUT
SORTKEYS 66000
INTO TABLE DSN8510.DEPT
Figure 20. LOAD and REORG Processing with SORTKEYS

1. At the beginning of the reload phase, a subtask is started to handle the work of sorting the keys. The sort subtask performs its initialization and waits for the main task (reload) to pass the keys to sort.

2. The reload phase loads the data into the table space, extracts the keys, and passes them in memory for sorting.

3. At the end of the reload phase, the last key is passed to sort, and record sorting completes.

4. Sort passes the sorted keys in memory to the build phase, which builds the indexes. Only the keys used to build indexes are passed to the build phase. For LOAD, foreign keys are sorted to the end of the SORTOUT data set and are read in a later phase.

5.1.2.2 Advantages

Index keys are passed to sort in memory rather than written to an intermediate data set. Similarly, the sorted keys are piped into the main build task when they are sorted. Avoiding the sort phase I/O to intermediate data sets improves the performance of the LOAD and REORG utilities.

The elapsed time from the start of the reload phase to the end of the build phase is reduced. The elapsed time of the observed sort phase is reduced more than the elapsed time of the reload and build phases. The sort initialization and termination and the record-in and record-out portion of the old sort phase are now in the reload and build phases. The elapsed time of the reload and build phases might increase over that when SORTKEYS is not used, but the overall elapsed time will be shorter.
5.1.2.3 Considerations
When SORTKEYS is specified, if the job abends in either the reload, sort, or build phase, then with no intermediate work data sets, it can be restarted only from the beginning of the reload phase.

If you do not specify SORTKEYS, LOAD examines each extracted key as it is written to the SYSUT1 data set to determine whether it is already in sorted order. If the keys are in sorted order, LOAD skips the sort phase. If SORTKEYS is specified, the sort phase is always invoked, which is not necessary with already sorted keys. Thus, any LOAD job that currently does not require a sort phase, whether because the index keys are already in sorted order or there are no indexes, is not a candidate for using SORTKEYS.

Similarly, REORG on tables with clustering index may not be the best candidate for using SORTKEYS as REORG skips sort if it is not necessary.

Note also that SORTKEYS is not available for SYSDBASE, SYDBAUTH, SYSPLAN, and SYSVIEWS in DSNDB06 and for DSNDB01.DBD01.

5.1.2.4 Measurements
The LOAD and REORG utilities were run on a table with 10 partitions, each partition having approximately 500 000 rows with 26 columns and a row length of 118 bytes. The table has two Type 1 indexes: one partitioning index, and one unique nonpartitioning index.

Note: The measurements were done using the 10-way RX3 CMOS processor (Model 9672 Version 0C), MVS/ESA V5.2 and RAMAC DASD.

LOAD: Table 13 shows the performance measurement results of LOAD in DB2 V4, in DB2 V5, and in DB2 V5 with SORTKEYS specified, all with Type 1 indexes.

<table>
<thead>
<tr>
<th></th>
<th>DB2 V4</th>
<th>DB2 V5</th>
<th>DB2 V5 with SORTKEYS</th>
<th>Delta %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU Time</td>
<td>Elapsed Time</td>
<td>CPU Time</td>
<td>Elapsed Time</td>
</tr>
<tr>
<td></td>
<td>1166</td>
<td>1295</td>
<td>899</td>
<td>1041</td>
</tr>
<tr>
<td></td>
<td>916</td>
<td>754</td>
<td>916</td>
<td>754</td>
</tr>
</tbody>
</table>

Note: All times are in seconds.

In this case, with LOAD of an entire table with two Type 1 indexes, DB2 V5 shows improvements over DB2 V4 of 23% in CPU time and 20% in elapsed time. These improvements are mainly due to a reduction of 36% in CPU time in the reload phase, and a reduction of 32% and 10% in elapsed time in the reload and build phases.

DB2 V5 with SORTKEYS specified shows a 21% reduction in CPU time and a 42% reduction in elapsed time over DB2 V4.

Table 14 on page 79 shows the performance measurement results of LOAD in DB2 V4, in DB2 V5, and in DB2 V5 with SORTKEYS specified, all with Type 2 indexes.
In this case with LOAD of an entire table with two Type 2 indexes, DB2 V5 overall reductions over DB2 V4 are 20% in CPU time and 24% in elapsed time. These improvements are mainly due to a reduction of 31% in CPU time in the reload phase, and a reduction of 27% and 40% in elapsed time in the reload and build phases.

DB2 V5 with SORTKEYS specified shows a 20% improvement in CPU time and a 46% improvement in elapsed time over DB2 V4.

**Specifying SORTKEYS in the LOAD Utility:**

The measurement results in Table 13 on page 78 and Table 14 show that specifying SORTKEYS when running the LOAD utility gives a significant reduction in elapsed time. It does so by avoiding intermediate data sets and by processing the sort in parallel with the reload and build phases.

Compared to DB2 V5 LOAD without specifying SORTKEYS, the reduction in elapsed time when SORTKEYS is specified, is about 29%. For example in Table 14, the difference in elapsed times is (1067 - 757) 310 seconds, which is 29% of 1067 seconds. This is achieved at virtually the same CPU consumption.

**REORG:** Table 15 shows the performance measurement results of REORG in DB2 V4 and in DB2 V5, type 1 indexes.

For REORG of an entire table with two Type 1 indexes, DB2 V5 shows overall reductions in CPU time of 2% and in elapsed time of 1.1% over DB2 V4.

Table 16 on page 80 shows the performance measurement results of REORG in DB2 V4 and in DB2 V5, Type 2 indexes.
For REORG of an entire table with two Type 2 indexes, DB2 V5 shows overall reductions in CPU time of 3.6% and in elapsed time of 11% over DB2 V4. These improvements are mainly due to a reduction of 13% in CPU time and 44% in elapsed time in the build phase.

Table 17 shows the performance measurement results of REORG in DB2 V5, in DB2 V5 with SORTKEYS specified, and in DB2 V5 with SORTKEYS and NOSYSREC specified, all with Type 1 indexes.

For REORG of an entire table with two Type 1 indexes, DB2 V5 with SORTKEYS and NOSYSREC specified shows an overall reduction of 34% in elapsed time over DB2 V5 without SORTKEYS and NOSYSREC specified. This reduction is achieved at virtually the same CPU consumption.

DB2 V5 with SORTKEYS alone shows an overall reduction of 18% in elapsed time over DB2 V5 without SORTKEYS and NOSYSREC specified. However, the CPU consumption goes up by 4%.

Table 18 shows the performance measurement results of REORG in DB2 V5, in DB2 V5 with SORTKEYS specified, and in DB2 V5 with SORTKEYS and NOSYSREC specified, all with Type 2 indexes.

### Table 16. REORG Utility (Two Type 2 Indexes): V4 and V5 Comparisons

<table>
<thead>
<tr>
<th></th>
<th>DB2 V4</th>
<th>DB2 V5</th>
<th>Delta %</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Time</td>
<td>872</td>
<td>829</td>
<td>−4.9</td>
</tr>
<tr>
<td>Elapsed Time</td>
<td>1437</td>
<td>1262</td>
<td>−12.2</td>
</tr>
</tbody>
</table>

**Note:** All times are in seconds.

### Table 17. REORG Utility (Two Type 1 Indexes): V5 Comparisons

<table>
<thead>
<tr>
<th></th>
<th>DB2 V5</th>
<th>DB2 V5 with SORTKEYS</th>
<th>DB2 V5 with SORTKEYS and NOSYSREC</th>
<th>Delta %</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Time</td>
<td>818</td>
<td>852</td>
<td>--</td>
<td>+4.0</td>
</tr>
<tr>
<td>Elapsed Time</td>
<td>1254</td>
<td>1022</td>
<td>--</td>
<td>−18</td>
</tr>
</tbody>
</table>

**Note:** All times are in seconds.

### Table 18. REORG Utility (Two Type 2 Indexes): V5 Comparisons

<table>
<thead>
<tr>
<th></th>
<th>DB2 V5</th>
<th>DB2 V5 with SORTKEYS</th>
<th>DB2 V5 with SORTKEYS and NOSYSREC</th>
<th>Delta %</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Time</td>
<td>829</td>
<td>812</td>
<td>--</td>
<td>−2.0</td>
</tr>
<tr>
<td>Elapsed Time</td>
<td>1262</td>
<td>988</td>
<td>--</td>
<td>−22</td>
</tr>
</tbody>
</table>

**Note:** All times are in seconds.
For REORG of an entire table with two Type 2 indexes, DB2 V5 with SORTKEYS and NOSYSREC specified shows an overall reduction of 4% in CPU time and 36% in elapsed time over DB2 V5 without SORTKEYS and NOSYSREC specified.

DB2 V5 with SORTKEYS alone shows a 2% reduction in CPU time and a 22% reduction in elapsed time over DB2 V5 without SORTKEYS and NOSYSREC specified.

**Specifying SORTKEYS and NOSYSREC in the REORG Utility:** You have to be careful about using NOSYSREC for regular REORG. The only time we recommend you should use this is when an image copy is made immediately prior to REORG.

Specifying SORTKEYS in the REORG utility will result in small reductions in CPU time in all of the phases and about 2% to 4% for the utility as a whole, compared with CPU time for REORG without SORTKEYS.

The important improvement is in elapsed time. Specifying SORTKEYS avoids the I/O to intermediate data sets, and the sort process is executed as a subtask run in parallel with the reload and build phases. This results in a small increase in elapsed time in both the reload and build phases, but since the sort process is no longer a separate phase executed in sequence with the other phases, no elapsed time is recorded for the sort. This alone accounts for about 25% of the total elapsed time of the REORG utility. This, besides an elapsed time reduction in the build phase, results in a 22% and 23% reduction in elapsed time when Type 1 and Type 2 indexes respectively are used, compared with elapsed time for REORG without SORTKEYS.

Specifying NOSYSREC in the REORG utility means that I/O to an intermediate unload data set is avoided, as data is passed in memory from the unload to the reload phase in parallel with the reload and build phases.

When both SORTKEYS and NOSYSREC are specified in REORG, all the improvements from specifying SORTKEYS alone (discussed above) still apply. In addition, CPU time is reduced by about 15% in the unload phase and elapsed time drops by more than 40%. For the utility as a whole, CPU time is reduced by more than 4%, and elapsed time by more than 35% over the REORG utility without either SORTKEYS or NOSYSREC specified.

### 5.1.3 COPYDDN and RECOVERYDDN

Specifying COPYDDN indicates that you want to create a full image copy data set (SHRLEVEL REF) during the execution of a LOAD REPLACE or REORG utility. That is, you want to take an inline copy.

**Note:** This discussion also applies to making image copies for recovery sites using RECOVERYDDN.

The new syntax of the LOAD REPLACE and REORG statements is

- **LOAD REPLACE**
  - LOAD DATA REPLACE COPYDDN ddname INTO TABLE tname
  - LOAD DATA INTO TABLE tname PART n REPLACE COPYDDN ddname
- **REORG**
  - REORG TABLESPACE tname COPYDDN ddname
For handling the output data set from an inline copy, DSN1COPY and DSN1PRNT have a new option, INLCOPY, which specifies that the input is an inline image copy data set.

When performing a LOAD or REORG operation, you can choose whether to log changes or not by specifying the keywords LOG YES or LOG NO. If you specify LOG NO the utility job completes in less time, but the object is not immediately available for use because it is left in a copy-pending state.

Note: Even with LOG NO specified, some logging can occur during the build phase if there is a Type 2 nonpartitioning index and concurrent access on a different partition of the same table space.

If you specify LOG YES, the utility job elapsed time increases, but the object is not left in a copy-pending state.

To reduce the length of time during which an object is not available, either because it is being processed by the utility or left in a copy-pending state, the new inline copy function (COPYDDN) is provided. When you specify COPYDDN, a full image copy data set is produced parallel to the execution of the LOAD and REORG utilities, and the object of the utility is not left in a copy-pending state regardless of which LOG option you specified.

Note: The inline copy is not valid unless the entire table space or partition is being replaced. If you specify COPYDDN in a LOAD utility without the REPLACE keyword, the LOAD utility terminates.

5.1.3.1 Considerations with Inline Copy
The inline image copy data set is not exactly the same as the image copy data set produced by the ordinary COPY utility:

• The data set produced by the inline copy is logically equivalent to a full image copy with SHRLEVEL REF, but the data within the data set differs in some respects:
  − Data pages may be out of sequence and some may be repeated. If pages are repeated, the last one is always the correct copy.
  − Space map pages are out of sequence and may be repeated.
  − If the compression dictionary is being rebuilt with LOAD, the set of dictionary pages occurs twice in the data set, the second set being the correct one.

Because of the small number of duplicate pages, the additional space required to hold the data is negligible.

• The SYSCOPY record produced by an inline copy in LOAD and REORG contains ICTYPE=F and SHRLEVEL=R.

• DSN1COPY and DSN1PRNT process the inline copy as full image copies from the COPY utility, with the following differences:
  − The INLCOPY key word must be specified.
  − DSN1COPY with or without OBIDXLAT and RESET can load a table space from an inline copy data set.
  − When printing from an inline copy data set, DSN1COPY and DSN1PRNT prints all instances of a page or page range, the last instance being the correct one.
Image copies taken during LOAD REPLACE are not recommended for use with RECOVER TOCOPY. Uncorrected unique index violations and referential constraint violations may exist in the image copy.

If a RECOVER TOCOPY is done, indexes are placed in recover pending status, and dependent table spaces, if any, are placed in check pending status. RECOVER INDEX and CHECK DATA must be run.

5.1.3.2 Measurements
Table 19 shows the performance measurement results of LOAD and REORG with inline COPY on a DB2 table with two Type 2 indexes, one partitioning index and one unique nonpartitioning index.

<table>
<thead>
<tr>
<th>Utility</th>
<th>Alone</th>
<th>+ COPY</th>
<th>With Inline Copy</th>
<th>Delta Alone (%)</th>
<th>Delta +COPY (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU Time</td>
<td>ET</td>
<td>CPU Time</td>
<td>ET</td>
<td>CPU Time</td>
</tr>
<tr>
<td>LOAD</td>
<td>926</td>
<td>1067</td>
<td>955</td>
<td>1302</td>
<td>954</td>
</tr>
<tr>
<td>Fast LOAD</td>
<td>918</td>
<td>757</td>
<td>947</td>
<td>992</td>
<td>956</td>
</tr>
<tr>
<td>REORG</td>
<td>829</td>
<td>1262</td>
<td>858</td>
<td>1497</td>
<td>856</td>
</tr>
<tr>
<td>Fast REORG</td>
<td>812</td>
<td>988</td>
<td>841</td>
<td>1223</td>
<td>860</td>
</tr>
</tbody>
</table>

Note: ET is elapsed time. All times are in seconds.

Full image copy on the same table space took 29 seconds CPU time and 235 seconds elapsed time.

When SORTKEYS is specified with LOAD, the utility is referenced as Fast LOAD.

When SORTKEYS is specified with REORG, the utility is referenced as Fast REORG.

In the + COPY column, the full-image copy numbers are added to the utility’s numbers to show the entire duration when two jobs are run.

The tables shows CPU and Elapsed Time in seconds for:
- The utility itself (Alone)
- The utility and COPY run as two jobs (+ COPY)
- The utility with inline COPY (With Inline Copy)

Deltas are given in percent, comparing utility with inline copy figures to:
- The utility alone (Delta Alone)
- The utility and COPY run as two separate jobs (Delta +COPY)

5.1.3.3 Conclusions on Inline Copy
These measurements show that the inline copy function of the LOAD and REORG utilities reduces the length of time a DB2 object is unavailable by 16.6% and 16% for LOAD and REORG, respectively, compared with the added elapsed times for LOAD and COPY (in the + COPY column). This reduction is achieved at virtually the same CPU consumption as the utility run on its own.

The rows for Fast LOAD and Fast REORG show that when SORTKEYS is specified for the utility run, the time the DB2 object is unavailable appears to be reduced even more. The actual reduction seen is the same as for the utilities without SORTKEYS, but as the percentage is calculated from a smaller number, the reduction appears to be greater.
Another way to calculate the improvement gained by using the inline copy feature is to observe the reduction in inline copy elapsed time compared with the elapsed times of the utility and COPY when run as separate jobs. The delta here is what is saved on the COPY utility alone.

The numbers for Fast REORG in Table 19 on page 83 show that the difference between the elapsed time in columns + COPY (1223 seconds) and With Inline Copy (1006 seconds) is 217 seconds, which is the elapsed time reduction when using inline copy compared with executing the REORG and COPY utilities as two jobs. Therefore the actual elapsed time for the copy part of the REORG with inline copy is (235 - 217): 18 seconds. Compared to the elapsed time for a separate COPY job, inline copy reduces the elapsed time for the copy process by 92%.

Compared to the added elapsed times for REORG and COPY (in the + COPY column) inline copy reduces the total elapsed time by 17.7%.

5.2 RECOVER INDEX

DB2 V4 introduced an enhancement to REORG SORTDATA UNLOAD CONTINUE to extract rows from a page and keys from a row with reduced CPU time in the unload phase of the REORG.

DB2 V5 RECOVER INDEX uses the same technique to reduce CPU time in its unload phase.

The RECOVER INDEX measurements were done on a table with 10 partitions, each partition having approximately 500 000 rows with 26 columns and a row length of 118 bytes. The table has two Type 2 indexes, one partitioning index and one unique nonpartitioning index.

Note: The measurements were done using the 10-way RX3 CMOS processor (Model 9672 Version 0C), MVS/ESA V5.2 and RAMAC DASD.

Table 20 shows the performance measurement results of RECOVER INDEX to recover all the indexes compared to RECOVER INDEX in DB2 V4.

<table>
<thead>
<tr>
<th></th>
<th>DB2 V4</th>
<th>DB2 V5</th>
<th>Delta %</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Time</td>
<td>700</td>
<td>593</td>
<td>−15</td>
</tr>
<tr>
<td>Elapsed Time</td>
<td>934</td>
<td>796</td>
<td>−15</td>
</tr>
</tbody>
</table>

Note: All times are in seconds.

In the case of RECOVER INDEX ALL (two Type 2 indexes), the overall reduction compared with DB2 V4 is 15% in CPU time mainly due to a 50% reduction in CPU time in the unload phase, and 15% reduction in elapsed time mainly due to a 38% reduction in elapsed time in the build phase.

DB2 V4 RECOVER INDEX introduced the option of not specifying nor allocating SYSUT1, thus eliminating I/O to intermediate data set. This option can also be used in RECOVER INDEX in DB2 V5.
Table 21 on page 85 shows the performance measurement results of RECOVER INDEX without SYSUT1 to recover all the indexes compared to RECOVER INDEX in DB2 V4 without SYSUT1.

<table>
<thead>
<tr>
<th></th>
<th>CPU Time</th>
<th>Elapsed Time</th>
<th>CPU Time</th>
<th>Elapsed Time</th>
<th>Delta CPU</th>
<th>Delta Elapsed</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2 V4</td>
<td>649</td>
<td>811</td>
<td>527</td>
<td>579</td>
<td>−19</td>
<td>−29</td>
</tr>
</tbody>
</table>

Note: All times are in seconds.

In the case of RECOVER INDEX ALL (two Type 2 indexes) without specifying or allocating SYSUT1 data set, the overall improvement compared with DB2 V4 is a 19% reduction in CPU time mainly due to a 26% reduction in the unload phase. The reason for the 26% reduction compared with 50% reduction when SYSUT1 is specified is that time normally attributed to the sort phase now is counted partly in the unload phase and partly in the build phase.

In this case, the elapsed time is reduced by 29% in DB2 V5 compared with DB2 V4, mainly because the number of I/Os is reduced.

Table 22 shows the performance measurement results of DB2 V5 RECOVER INDEX with SYSUT1 to recover all the indexes, compared to RECOVER INDEX in DB2 V5 without SYSUT1.

<table>
<thead>
<tr>
<th></th>
<th>CPU Time</th>
<th>Elapsed Time</th>
<th>CPU Time</th>
<th>Elapsed Time</th>
<th>Delta CPU</th>
<th>Delta Elapsed</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2 V5 with SYSUT1</td>
<td>593</td>
<td>796</td>
<td>527</td>
<td>579</td>
<td>−11</td>
<td>−27</td>
</tr>
</tbody>
</table>

Note: All times are in seconds.

In the case of RECOVER INDEX ALL (two Type 2 indexes) in DB2 V5 without specifying or allocating SYSUT1 data set, the overall improvement compared with DB2 V5 with SYSUT1 specified is an 11% reduction in CPU time and a 27% reduction in elapsed time.

Without SYSUT1 specified, the resources used are distributed differently among the phases involved. Compared to RECOVER INDEX with SYSUT1, both the unload and build phases have increased CPU and elapsed times because time usually spent in the sort phase is now accounted for here. This increase is, however, more than offset by the savings in the sort phase.

5.3 COPY

The change in the COPY utility introduced in DB2 V5 is a user productivity enhancement.

By using the new CHANGELIMIT option on the COPY statement, you can display image copy information about a table space, a partition, a data set, or a list of these objects. You can also let DB2 decide whether to take an image copy, and if so, what kind of image copy, based on this information.
5.3.1 CHANGELIMIT Option

When you use the new CHANGELIMIT option, the COPY utility reports image copy information for a table space. The image copy information is collected and reported during a new phase of the COPY utility, the report phase, and includes the following values:

- The total number of pages. This value is the number of pages in the table space and may include preformatted pages containing all zeroes. This number is also the number of pages copied if a full image copy is created.
- The number of empty pages if the table space is segmented.
- The number of changed pages. This number is the number of pages copied if an incremental image copy is created.
- The percentage of changed pages.
- The type of image copy recommended.

When table space is specified, image copy information is displayed (1) for each partition if the table space is partitioned, (2) for each data set if the table space is not partitioned, and (3) for the entire table space. The report can be used for an inspection of the state of changes in an object or a list of objects.

You can specify the REPORTONLY option only with the CHANGELIMIT option.

If you specify only CHANGELIMIT option, DB2 automatically makes the appropriate image copy based on the recommendation given in the image copy information.

If you specify both CHANGELIMIT and REPORTONLY options, DB2 does not make an image copy and produces only a report for the image copy information. Although an image copy is not created, the data set is allocated if you have the appropriate DD statement in the JCL. To prevent the data set being allocated for the image copy, you may want to specify DD DUMMY for the image copy data set in the JCL.

5.3.2 Values For CHANGELIMIT Option

The type of image copy recommended by the report phase of the COPY utility is based on values you provide for the CHANGELIMIT option. These values are ranges of changed pages expressed in percent of total pages in the table space. The following discussion assumes that both CHANGELIMIT and REPORTONLY options are specified.

You can provide one or two values:

- One value (percent_value1)
  - An incremental image copy is recommended if the percentage of changed pages in the table space is greater than 0 and less than percent_value1.
  - A full image copy is recommended if the percentage of changed pages is greater than or equal to percent_value1.
  - No image copy is recommended if no pages have been changed.
- Two values (percent_value1,percent_value2)
An incremental image copy is recommended if the percentage of changed pages is greater than \textit{percent\_value1} and less than \textit{percent\_value2}.

A full image copy is recommended if the percentage of changed pages is greater than or equal to \textit{percent\_value2}.

No image copy is recommended if the percentage of changed pages is less than, or equal to \textit{percent\_value1}.

If no values are provided on the \textit{CHANGELIMIT} statement, the default values used are \(1,10\).

\textbf{Note:} If \textit{CONCURRENT} is specified, only full image copies are allowed. The image copy information is displayed using both values, but copies are created only if the full image copy \textit{CHANGELIMIT} value is met.

### 5.3.3 The COPY CHANGELIMIT Option and Output Report

Suppose you have a table space DSN8S51C in database DSN8D51P. You want to check the table space to determine whether any space map pages are broken, or to identify any other problems that might prevent making an image copy (for instance, the object may be in recovery-pending state). You also want to determine the number of changes in that table space since the last image copy was created.

On table spaces like this, make an incremental image copy when the number of changed pages is between 10\% and 40\% of the total number of pages in the table space. When more than 40\% are changed, make a full image copy.

A COPY statement to achieve this would look like this example:

\begin{verbatim}
COPY TABLESPACE DSN8D51P.DSN8S51C CHANGELIMIT(10,40) REPORTONLY
\end{verbatim}

The new report phase in the COPY utility produces a report as shown in this example:

\begin{verbatim}
DSNU000I DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = RPRTO
DSNU050I DSNUGUTC - COPY TABLESPACE DSN8D51P.DSN8S51C
CHANGELIMIT(10,40) REPORTONLY
DSNU440I DSNUBCHG - IMAGE COPY INFORMATION

\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
\textbf{4KB} & \textbf{EMPTY} & \textbf{CHANGED} & \textbf{PERCENT OF} \\
\hline
\textbf{PAGE} & \textbf{PAGES} & \textbf{PAGES} & \textbf{PAGES} & \textbf{CHANGED PAGES} & \textbf{ICTYPE} \\
\hline
DSN8D51P & DSN8S51C & ALL & 64 & 5 & 25 & 39.0 & I \\
\hline
DSNU43I & DSNUBCHG - INCREMENTAL CHANGE LIMIT MET \\
DSNU010I & DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=2 \\
\hline
\end{tabular}
\end{verbatim}

This report shows that there is no broken page in this table space, and that the total percent of changed pages is greater than the \textit{CHANGELIMIT} low value 10 and less than the \textit{CHANGELIMIT} high value 40, so that an incremental image copy is recommended.

\textbf{Note:} If you use the \textit{CHANGELIMIT} and \textit{REPORTONLY} option, you must specify \texttt{DD DUMMY} on the \texttt{SYSCOPY DD} statement to prevent this data set from being allocated and opened.
Notice the return code from the utility execution. It can be used to automate the image copy process as shown in 5.3.4, “How to Automate the Image Copy Process” on page 88.

The return codes when CHANGELIMIT is specified are as follows:

1  If successful and none of the ranges specified on CHANGELIMIT is met, then no image copy is recommended.

2  If successful and the percent of changed pages is within the range specified for an incremental image copy, then an incremental image copy is recommended.

3  If successful and the percent of changed pages is within the range specified for a full image copy, then a full image copy is recommended.

8  During the report phase, COPY checks each space map page for any inconsistencies. If a broken space map page exists, the return code is set to 8.

The report from the COPY utility when a broken space map page is found looks like this:

<table>
<thead>
<tr>
<th>DBNAME</th>
<th>TSNAME</th>
<th>DSNUM</th>
<th>4KB CHANGED</th>
<th>PERCENT OF</th>
<th>ICTYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSN8D51P</td>
<td>PARTTSP</td>
<td>1</td>
<td>INFORMATION CANNOT BE DETERMINED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSN8D51P</td>
<td>PARTTSP</td>
<td>2</td>
<td>32 23</td>
<td>71.88</td>
<td></td>
</tr>
<tr>
<td>DSN8D51P</td>
<td>PARTTSP</td>
<td>ALL</td>
<td>INFORMATION CANNOT BE DETERMINED</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DSNU441I DSNUBASP - BROKEN PAGE FOUND
  DBNAME = DSN8D51P
  TSNAME = PARTTSP
  DSNUM = 1
  PAGE = 00000001

DSNU010I DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=8

5.3.4 How to Automate the Image Copy Process

Currently, to determine whether an image copy needs to be taken, you can either count the change bits in the space map pages of the data set (by using the DSN1PRNT utility), or execute an image copy and examine the utility report to see if any pages were changed. Both operations can be costly, time consuming, and disruptive, especially for large nonpartitioned table spaces.

Another method used is to execute an image copy with FULL NO specified and hope changes have been made to the table space. If not, an empty data set is created, which can represent a problem when generation data groups (GDGs) are used and valid image copies are rolled out.

The CHANGELIMIT option of the COPY utility provides an easy, secure, and cost-effective way to determine whether an image copy needs to be, or can be taken, whether the image copy should be full or incremental, and by providing status information through the return code, the means to automate the process of taking image copies.
The following example shows a job that automatically takes a full image copy of table space PARTTSP2 in database DSN8D51A (if it is found OK and the number of changed pages is equal to or greater than 20%). It does not attempt an image copy if the percent of changed pages is below 20.

```
//COPYREPT EXEC PGM=DSNUTILB,PARM=DSN,COPYR
//SYSCOPY DD DUMMY
//SYSIN DD *
COPY TABLESPACE DSN8D51A.PARTTSP2 CHANGELIMIT 20 REPORTONLY
/*
 // IF (COPYREPT.RC=3) THEN MAKE A FULL IMAGE COPY
 */
//COPYFULL EXEC PGM=DSNUTILB,PARM=DSN,COPYF
//SYSCOPY DD DSN=HLQ.ANYNAME,SPACE=(CYL,(15,1)),DISP=(,CATLG)
//SYSIN DD *
COPY TABLESPACE DSN8D51A.PARTTSP2 FULL YES
/*
 // ENDIF
 */
```

The percentage of changed pages meets the full image copy change limit, consequently a full image copy is taken.

The output written to SYSPRINT is shown in this example:

```
DSNU000I  DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = COPYR
DSNU050I  DSNUGUTC - COPY TABLESPACE DSN8D51A.PARTTSP2
       CHANGELIMIT 20 REPORTONLY
DSNU440I  DSNUBCHG - IMAGE COPY INFORMATION

        4KB  CHANGED   PERCENT OF
  DBNAME  TSNAME  DSNUM  PAGES  PAGES  CHANGED  PAGES  ICTYPE
  ------------------------------- ------- ------- ------- ------- -------
DSN8D51A  PARTTSP2  1    192    57  29.69
DSN8D51A  PARTTSP2  2    192    30  15.63
DSN8D51A  PARTTSP2  ALL  384    87  22.73  F

DSNU443I  DSNUBCHG - FULL CHANGE LIMIT MET
DSNU010I  DSNUBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=3

DSNU000I  DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = COPYF
DSNU050I  DSNUGUTC - COPY TABLESPACE DSN8D51A.PARTTSP2 FULL YES
DSNU400I  DSNUBBID - IMAGE COPY COMPLETE -
        NUMBER OF PAGES=384
        AVERAGE PERCENT FREE SPACE PER PAGE = 23.99
        PERCENT OF CHANGED PAGES = 22.73
        ELAPSED TIME=00:00:09
DSNU010I  DSNUBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

**Note:** DSNUM in this report means partition number if the table space is a partitioned table space, and data set number if the table space is not a partitioned table space.
5.3.5 Conclusions

The additional functionality introduced in the COPY utility in DB2 V5 enables you to manage the process of taking image copies of your data in a controlled and secure way. Based upon your policy for copying, you can construct jobs that use the return code from the report produced by the COPY utility to make the decision on what to do in the subsequent steps. The utility decides whether to take an image copy or not, and what kind of image copy should be taken, based on the number of changed pages in the table space.

This enhancement is accomplished with as good performance as in the COPY utility in DB2 V4.

5.4 RUNSTATS

DB2 V5 introduces a sampling option to improve performance when collecting nonindexed column statistics. When you decide to collect the statistics, you can define the percentage of table rows to use for the calculation. DB2 V5 also enables you to collect statistics on a concatenation of key columns to improve the performance of queries that reference correlated columns in the predicates.

Three new keywords control the use of the new RUNSTATS options:

- SAMPLE indicates the percentage of rows to sample when collecting nonindexed column statistics.
- KEYCARD collects cardinalities on a concatenation of index columns, beginning with the first index column.
- FREQVAL controls the collection of frequent value statistics by specifying:
  - NUMCOLS the number of key columns to concatenate when collecting frequent value statistics for an index.
  - COUNT the number of frequent values to be collected.

5.4.1 Why RUNSTATS Should Be Run

The RUNSTATS utility collects summary information about the characteristics of data in table spaces and indexes, updating the DB2 catalog with this information. There are two formats for RUNSTATS: RUNSTATS TABLESPACE and RUNSTATS INDEX. RUNSTATS TABLESPACE collects statistics on table spaces, and optionally on indexes. RUNSTATS INDEX collects statistics only on indexes.

The DB2 optimizer uses the information collected by the ACCESSPATH option to select the access path to the data. You can use the information collected by the SPACE option to evaluate the design and check the organization of tables and indexes. Here we discuss only items relating to the ACCESSPATH option.

The optimizer uses filter factors to estimate the number of rows that are qualified by a predicate. A filter factor is a combination of the literal value in the predicate, the operator in the predicate, and the statistics on the column in the predicate. The optimizer uses default filter factors to select access paths to data if RUNSTATS has never been run. The optimizer may select a poor access path unless the actual filter factors for the data match the defaults. For example, the optimizer may incorrectly assume that too few rows of a table are filtered by a range predicate and make that table the outer table of a nested loop join.

The standard recommendations for running RUNSTATS are these:
When a table is loaded.
When a table or index is reorganized.
When a new index is physically created.
After extensive update, delete, or insert activity.
After RECOVER TABLESPACE or RECOVER INDEX.

This action ensures that major and cumulative changes to data statistics are recorded in the DB2 catalog. In addition, the optimizer will not have to use default filter factors for new indexes when selecting access paths.

The more statistics that are available, the better chance the optimizer has of selecting the best access path. However, you have to balance the CPU time needed to collect the statistics against the benefit of a more efficient access path.

### 5.4.2 Changes to Catalog Columns

The integer columns in some catalog tables that record cardinality are changed to floating point in DB2 V5. This allows RUNSTATS to collect statistics on large tables introduced in DB2 V5. The migration process copies the old columns to the new columns. RUNSTATS updates only floating point columns for large tables, and both sets of columns for other tables. The optimizer will use only the floating column statistics for access path determination. Columns we refer to are shown in Table 23 below.

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Original Column Name</th>
<th>Migrated Column Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSOLDIST</td>
<td>FREQUENCY</td>
<td>FREQUENCYF</td>
</tr>
<tr>
<td>SYSOLDISTSTATS</td>
<td>FREQUENCY</td>
<td>FREQUENCYF</td>
</tr>
<tr>
<td>SYSOLDISTSTATS</td>
<td>FREQUENCY</td>
<td>FREQUENCYF</td>
</tr>
<tr>
<td>SYSINDEXES</td>
<td>COLCARD</td>
<td>COLCARDF</td>
</tr>
<tr>
<td>SYSINDEXES</td>
<td>FIRSTKEYCARD</td>
<td>FIRSTKEYCARDF</td>
</tr>
<tr>
<td></td>
<td>FULLKEYCARD</td>
<td>FULLKEYCARDF</td>
</tr>
</tbody>
</table>

### 5.4.3 Faster RUNSTATS with Sampling

In this section we describe how RUNSTATS sampling can be used to obtain statistics on nonindexed column data with reduced CPU consumption. We discuss the effect on access path determination with varying sampling percentages and demonstrate the clear benefit even of low sampling percentages compared with never running RUNSTATS.

#### 5.4.3.1 Statistics Collected by Sampling

RUNSTATS sampling affects the collection of statistics on columns that are not the first key column of an index, and which are recorded in columns HIGH2KEY, LOW2KEY, and COLCARD in SYSIBM.SYSCOLUMNS and in columns HIGH2KEY and LOW2KEY in SYSIBM.SYSCOLSTATS. The optimizer uses these statistics to determine filter factors when additional frequent value statistics are not available. In this case the optimizer has determined that there is no usable entry in SYSIBM.SYSCOLDIST because:

- The column is not the first column of an index.
• The column is the first column of an index, but the data is uniformly distributed by the column.
• The column is the first column of an index, but RUNSTATS has never been run on the index.

Frequent value statistics may be absent from SYSIBM.SYSCOLDIST for other reasons. For example, incorrect manual update of the catalog.

Column statistics are often not collected because of the high CPU consumption for RUNSTATS. To calculate HIGH2KEY, LOW2KEY, and COLCARD statistics in SYSIBM.SYSCOLUMNS for columns that are not the first column of an index, RUNSTATS uses a complex and CPU-intensive hashing and statistical algorithm.

In DB2 V4, you might typically collect column statistics as follows:
• Almost always on the first column of indexes, using the INDEX or INDEX ALL options of RUNSTATS INDEX
• Often on the remaining index columns, using the TABLE and COLUMN options of RUNSTATS TABLESPACE
• Occasionally on those nonindexed columns often specified in range or join predicates, particularly if bad access paths were found, using the TABLE and COLUMN options of RUNSTATS TABLESPACE
• Seldom on all columns by specifying TABLE ALL or TABLE, and taking the default of COLUMN ALL.

DB2 V5 enables you to sample the column values and trade accuracy with CPU requirements. Use the optional SAMPLE keyword to determine the percentage of table rows to sample. You can specify integer values between 1 and 100; the default of 25 means that 25% of the table rows are used to calculate statistics on nonindexed columns.

The COLCARDF value is scaled as if all column values were evaluated. Values of HIGH2KEY and LOW2KEY are not scaled. Be aware that the value for COLCARDF is also a statistical estimate, even if the SAMPLE keyword is omitted.

The following RUNSTATS utility control statement updates catalog statistics for all columns in the STLETALT table in table space DPABDBAH.SSLALT without sampling (this works the same way as in DB2 V4):

```
RUNSTATS TABLESPACE (DBAPDBAH.SSLALT)
  TABLE (STLETALT)
```

The following RUNSTATS utility control statement updates catalog statistics for all columns in the STLETALT table in table space DPABDBAH.SSLALT at the default of 25% sampling:

```
RUNSTATS TABLESPACE (DBAPDBAH.SSLALT)
  TABLE (STLETALT) SAMPLE
```

The following RUNSTATS utility control statement updates catalog statistics for all columns in the STLETALT table in table space DPABDBAH.SSLALT at 10% sampling:

```
RUNSTATS TABLESPACE (DBAPDBAH.SSLALT)
  TABLE (STLETALT) SAMPLE 10
```
5.4.3.2 Factors Affecting CPU Consumption
The following factors affect RUNSTATS CPU time:

- The number of rows in the table
- The sampling percentage
- The number of columns sampled
- Column lengths
- Column uniqueness.

More CPU is used for tables with a larger number of rows, and when a higher percentage of rows is sampled. Similarly, more CPU is used for tables with more columns, particularly if the columns are longer than average. Columns that are more unique require less CPU because less interchange of HIGH2KEY and LOW2KEY values occurs during the sampling process.

5.4.3.3 Factors Affecting Access Path Accuracy
The following factors affect RUNSTATS accuracy:

- The number of rows in a table
- The sampling percentage
- Column uniqueness.

The accuracy of the statistics collected will be greater the more rows there are in a table. For the same number of rows in a table, the statistics collected will be more accurate, the higher the percentage of rows that are sampled. Generally, the statistics collected will be more accurate the more unique the column value is. Sampling has no effect on accuracy in the special case when the column always contains the same value.

5.4.4 Measurements for RUNSTATS Sampling
We now examine the effect of sampling on the following:

- A large partitioned table space.
- Database "Database A"
  This database contains tables with millions of rows, with long record length and VARCHAR columns.
- Database "Database B"
  This database contains tables which, compared with database "Database A," have fewer rows, shorter record length, no VARCHAR columns, and column data that is generally more unique.

We report on the CPU reduction with reduced sampling, and on the corresponding access paths for queries at each sampling level. The queries against database "Database A" tend to be complex, with multitable joins and sorts. The queries against database "Database B" are much simpler and have fewer joins.
5.4.4.1 Effect of Sampling a Large Partitioned Table Space

Table 24 shows the effect on CPU consumption when RUNSTATS is run on a partitioned table space containing a 5 million row table in 10 partitions on a two-way H5 processor.

<table>
<thead>
<tr>
<th>SAMPLE X%</th>
<th>RUNSTATS CPU Time SAMPLE X%</th>
<th>RUNSTATS CPU Time Delta Compared with SAMPLE 100%</th>
<th>RUNSTATS CPU Time Delta (%) Compared with SAMPLE 100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>315</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>90</td>
<td>307</td>
<td>-8</td>
<td>-3</td>
</tr>
<tr>
<td>80</td>
<td>279</td>
<td>-36</td>
<td>-11</td>
</tr>
<tr>
<td>70</td>
<td>262</td>
<td>-53</td>
<td>-17</td>
</tr>
<tr>
<td>60</td>
<td>245</td>
<td>-70</td>
<td>-22</td>
</tr>
<tr>
<td>50</td>
<td>229</td>
<td>-86</td>
<td>-27</td>
</tr>
<tr>
<td>40</td>
<td>212</td>
<td>-103</td>
<td>-33</td>
</tr>
<tr>
<td>30</td>
<td>195</td>
<td>-120</td>
<td>-38</td>
</tr>
<tr>
<td>20</td>
<td>179</td>
<td>-136</td>
<td>-43</td>
</tr>
<tr>
<td>10</td>
<td>161</td>
<td>-154</td>
<td>-49</td>
</tr>
<tr>
<td>5</td>
<td>154</td>
<td>-161</td>
<td>-51</td>
</tr>
<tr>
<td>1</td>
<td>145</td>
<td>-170</td>
<td>-54</td>
</tr>
</tbody>
</table>

Note: All times are in seconds.

The results in Table 24 enable us to derive an interpolation formula where the percentage CPU saving at a given SAMPLE percentage X compared to a SAMPLE percentage of 100% is denoted by:

\[ \% \text{ CPU Saving} = (100 - X) \times 0.55 \]

For example, the percentage CPU saving at the default SAMPLE percentage value of 25% is 41%.

5.4.4.2 Effect of Sampling Database A

Table 25 on page 95 shows the effect on CPU consumption when RUNSTATS is run on a set of 15 tables. The workload is on a six-way 600J processor, and the CPU times are the total times for the 15 tables.
The results in Table 25 enable us to derive an interpolation formula where the percentage CPU saving at a given SAMPLE percentage X compared with a SAMPLE percentage of 100% is denoted by:

\[
\% \text{ CPU Saving} = (100 - X) \times 0.6
\]

For example, the percentage of CPU time saving at the default SAMPLE percentage value of 25% is 45%.

Table 25 shows the effect on CPU consumption when RUNSTATS is run on the same set of 15 tables in Database A, and on the same processor, but without specifying TABLE or SAMPLE, so that nonindexed column statistics were not collected. The results are compared to CPU consumption at 1% sampling.

<table>
<thead>
<tr>
<th>SAMPLE X %</th>
<th>RUNSTATS CPU Time</th>
<th>RUNSTATS CPU Time Delta</th>
<th>% CPU Saving Compared with SAMPLE 100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1445</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>75</td>
<td>1229</td>
<td>-216</td>
<td>-15</td>
</tr>
<tr>
<td>50</td>
<td>1013</td>
<td>-432</td>
<td>-30</td>
</tr>
<tr>
<td>40</td>
<td>925</td>
<td>-520</td>
<td>-36</td>
</tr>
<tr>
<td>20</td>
<td>750</td>
<td>-695</td>
<td>-48</td>
</tr>
<tr>
<td>10</td>
<td>661</td>
<td>-784</td>
<td>-54</td>
</tr>
<tr>
<td>5</td>
<td>615</td>
<td>-830</td>
<td>-57</td>
</tr>
<tr>
<td>1</td>
<td>580</td>
<td>-865</td>
<td>-60</td>
</tr>
</tbody>
</table>

Note: All times are in seconds.

The results in Table 25 enable us to derive an interpolation formula where the percentage CPU saving at a given SAMPLE percentage X compared with a SAMPLE percentage of 100% is denoted by:

\[
\% \text{ CPU Saving} = (100 - X) \times 0.6
\]

Table 26 shows the effect on CPU consumption when RUNSTATS is run on the same set of 15 tables in Database A, and on the same processor, but without specifying TABLE or SAMPLE, so that nonindexed column statistics were not collected. The results are compared to CPU consumption at 1% sampling.

<table>
<thead>
<tr>
<th>RUNSTATS CPU Time</th>
<th>RUNSTATS CPU Time Delta</th>
<th>% CPU Saving Compared with SAMPLE 1%</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAMPLE 1%</td>
<td>without TABLE Statistics</td>
<td>Compared with SAMPLE 1%</td>
</tr>
<tr>
<td>580</td>
<td>222</td>
<td>-358</td>
</tr>
</tbody>
</table>

Note: All times are in seconds.

The results in Table 26 show that in this case there is a 62% saving when RUNSTATS is run without specifying either TABLE or SAMPLE compared with the result of only 1% sampling. However, nonindexed column statistics were not collected, so the optimizer uses default filter factors.

5.4.4.3 Effect of Running Queries on Database A

The performance of 20 queries on tables in Database A is measured. These queries are complex in nature. The following results are noted:

- One query has a bad access path at 20% sampling.
- Two queries have a bad access path at 10% sampling.
- Five queries have a bad access path at 5% sampling.
Table 27 on page 96, Table 28 on page 97, and Table 29 on page 97 tabulate details of the effect on elapsed time of running three of the queries with statistics collected using different values of SAMPLE. We include the effect when RUNSTATS column information is not available.

Table 27 shows elapsed times for a seven-table nested loop join, with sort for ORDER BY.

### Table 27. Elapsed Times for Seven-Table Nested Loop Query, Sort for ORDER BY

<table>
<thead>
<tr>
<th>SAMPLE X %</th>
<th>Query Elapsed Time</th>
<th>Query Elapsed Time Delta Compared with SAMPLE 100%</th>
<th>Query Elapsed Time Delta (%) Compared with SAMPLE 100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>73</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>74</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>10</td>
<td>86</td>
<td>+13</td>
<td>+18</td>
</tr>
<tr>
<td>5</td>
<td>132</td>
<td>+59</td>
<td>+81</td>
</tr>
<tr>
<td>1</td>
<td>238</td>
<td>+165</td>
<td>+632</td>
</tr>
<tr>
<td>* A</td>
<td>705</td>
<td>+632</td>
<td>+866</td>
</tr>
</tbody>
</table>

**Note:** All times are in seconds.

The case where Sample X% = "*" denotes where RUNSTATS TABLE was not run on the data.

Results from this complex query show that a good access path is obtained even at 20% sampling. The access path is reasonable at 10% sampling but is bad at 5% and below, indicating inadequate statistical accuracy for the query concerned.

The optimizer must use less accurate filter-factor estimates to decide which table to select as the outer table in each of the seven joins. The worst case is where no column statistics are available. If default filter factors do not match the actual data, the optimizer probably selects the wrong sequence to join the tables.

Table 28 on page 97 shows the elapsed times for an eight-table nested loop join with sort for GROUP BY.
The case where Sample X% = "*" denotes where RUNSTATS TABLE was not run on the data.

Results from this complex query show that a good access path is obtained even at 10% sampling. The access path selected based on statistics collected with 5% sampling and below is bad. The worst case is where column statistics are not available. The access path is marginally worse with defaults than with statistics collected with 1% sampling.

Table 29 shows elapsed times for a two-table merge scan join with sort for DISTINCT.

<table>
<thead>
<tr>
<th>SAMPLE X %</th>
<th>Query Elapsed Time SAMPLE X %</th>
<th>Query Elapsed Time Delta Compared with SAMPLE 100%</th>
<th>Query Elapsed Time Delta (%) Compared with SAMPLE 100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>73</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>73</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>75</td>
<td>+2</td>
<td>+3</td>
</tr>
<tr>
<td>5</td>
<td>102</td>
<td>+29</td>
<td>+40</td>
</tr>
<tr>
<td>1</td>
<td>818</td>
<td>+745</td>
<td>+1020</td>
</tr>
<tr>
<td>* A</td>
<td>834</td>
<td>+761</td>
<td>+1042</td>
</tr>
</tbody>
</table>

Note: All times are in seconds.

The case where Sample X% = "*" denotes where RUNSTATS TABLE was not run on the data.

Results from the merge scan query show some access path deterioration at 20% sampling. The worst case is where column statistics are not available. The access path is significantly worse with defaults than with statistics collected from 1% sampling.
5.4.4.4 Comparison of No RUNSTATS and 1% Sampling

Table 30 shows the measurement data for query performance with default filter factors, compared with performance when RUNSTATS has been run with 1% sampling.

The results show a significant elapsed-time saving for sampling at 1% compared with performance when RUNSTATS had never been run so that no column statistics were available. Access paths selected using 1% sampling are better than access paths selected using default filter factors. The main improvements are found in

- Join table sequence
- Join method
- Avoidance of inefficient indexes.

### Table 30. Query Performance: 1% Sampling and no Statistics Comparisons

<table>
<thead>
<tr>
<th>QUERYNO</th>
<th>Query Elapsed Time without RUNSTATS</th>
<th>Query Elapsed Time with RUNSTATS, SAMPLE 1%</th>
<th>Query Elapsed Time Delta</th>
<th>Query Elapsed Time Delta %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query 1</td>
<td>995</td>
<td>307</td>
<td>-688</td>
<td>-69</td>
</tr>
<tr>
<td>Query 2</td>
<td>705</td>
<td>238</td>
<td>-467</td>
<td>-66</td>
</tr>
<tr>
<td>Query 3</td>
<td>245</td>
<td>73</td>
<td>-172</td>
<td>-70</td>
</tr>
<tr>
<td>Query 4</td>
<td>2197</td>
<td>425</td>
<td>-1772</td>
<td>-81</td>
</tr>
<tr>
<td>Query 5</td>
<td>703</td>
<td>42</td>
<td>-661</td>
<td>-94</td>
</tr>
<tr>
<td>Query 6</td>
<td>880</td>
<td>270</td>
<td>-610</td>
<td>-69</td>
</tr>
<tr>
<td>Query 7</td>
<td>266</td>
<td>23</td>
<td>-243</td>
<td>-91</td>
</tr>
<tr>
<td>Query 8</td>
<td>210</td>
<td>16</td>
<td>-194</td>
<td>-92</td>
</tr>
</tbody>
</table>

**Note:** All times are in seconds.

The results show a significant elapsed-time saving for sampling at 1% compared with performance when RUNSTATS had never been run so that no column statistics were available. Access paths selected using 1% sampling are better than access paths selected using default filter factors. The main improvements are found in

- Join table sequence
- Join method
- Avoidance of inefficient indexes.

5.4.4.5 Effect of Sampling Database B

Table 31 on page 99 shows the effect on CPU consumption when RUNSTATS is run on a set of 19 tables in Database B. The tables generally have shorter columns and rows than in Database A. The workload is on a two-way H5 processor, and the CPU times are the total times for the 19 tables.
The results in Table 31 enable us to derive an interpolation formula where the percentage of CPU saving at a given SAMPLE percentage X compared with a SAMPLE percentage of 100% is denoted by:

\[ \% \text{ CPU Saving} = (100 - X) \times 0.44 \]

The savings in CPU are proportionally less than for Database A, primarily because of the shorter column lengths and rows, which require less hashing.

5.4.4.6 Effect of Running Queries on Database B

A set of 60 queries are run against Database B at various sampling rates. The access paths change very little at most sampling rates. At 1% sampling, only one query shows a bad access path. These queries are very much simpler than those run on Database A and have far fewer joins. Also, queries with a matching index scan that eliminates many rows on the first index column are little affected by sampling, particularly if there are very few additional predicates.

5.4.4.7 Conclusion for RUNSTATS Sampling

Sampling provides the greatest RUNSTATS CPU savings for a database where there are tables with millions of rows and many VARCHAR columns or when the column data is far from unique. In these cases, you should expect to save around 45% CPU time at the default sampling percentage of 25.

RUNSTATS sampling at 1% is likely to produce significant saving in query elapsed times compared with performance when no RUNSTATS-generated statistics exist for nonindexed columns. This is particularly true for a database where multitable joins with many predicates are often processed.

Sampling is unlikely to improve either good matching index scans with few predicates or single table space scans; in these cases, nonindexed column statistics are of little benefit.

We recommend that you consider RUNSTATS TABLE with sampling for databases where nonindexed column statistics have never been collected. You can expect considerable query CPU savings at 1% sampling, better access paths
at 10% sampling, and in many cases very good or optimum access paths at 20% sampling.

If you currently collect nonindexed column statistics on a regular basis, we recommend you consider the introduction of sampling. You should first EXPLAIN any queries with critical performance requirements and then introduce sampling at the default of 25%. If there are no access path changes, consider further reductions in sampling percentage to maximize your potential CPU saving. If access path changes do result, you should run the queries concerned and check if the performance is acceptable.

If you do not want to introduce sampling, you need not make changes to your existing DB2 V4 RUNSTATS utility control statements. Sampling occurs only if the SAMPLE keyword is specified.

5.4.5 Key Column Correlation Statistics

This section describes the RUNSTATS utility control statements you use to collect statistics on a concatenation of key columns, and indicates where in the catalog DB2 holds the statistics. We show examples of RUNSTATS utility control statements for collection of statistics on concatenated key columns. The extra CPU cost to collect the statistics is discussed.

5.4.5.1 How RUNSTATS Collects Correlated Statistics

If you accept the DB2 V5 defaults, you will not collect additional statistics for correlated key columns. For example, consider the following RUNSTATS utility control statement:

```
RUNSTATS INDEX (DSN510.XEMPL1)
```

As in DB2 V4, specifying RUNSTATS INDEX collects frequent value statistics in tables SYSIBM.SYSCOLDIST, and in SYSIBM.SYSCOLDISTSTATS (if the index is partitioned) for only the first key column of index XEMPL1.

In addition, cardinality values are recorded in columns FIRSTKEYCARDF and FULLKEYCARDF in SYSIBM.SYSINDEXES and also in SYSIBM.SYSINDEXSTATS for a partitioned index.

In DB2 V5, you can collect additional statistics on correlated key columns using two new key words, KEYCARD and FREQVAL:

- **KEYCARD**
  Indicates the cardinalities of each column (concatenated with all previous columns) to collect. For example, the following RUNSTATS utility control statement on a four-column index XADDR4 collects the following cardinalities:

```
RUNSTATS INDEX (DBAPDBA.XADDR4)
  KEYCARD
```

  - First key (FIRSTKEYCARDF), stored in SYSIBM.SYSINDEXES and SYSIBM.SYSINDEXSTATS
  - Concatenation of first and second key columns, stored in SYSIBM.SYSCOLDIST and SYSIBM.SYSCOLDISTSTATS
  - Concatenation of first, second, and third key columns, stored in SYSIBM.SYSCOLDIST and SYSIBM.SYSCOLDISTSTATS
Concatenation of first, second, third, and fourth columns (FULLKEYCARDF), stored in SYSIBM.SYSINDEXES and SYSIBM.SYSINDEXESTATS.

**Note:** There is no value in specifying KEYCARD for a two-column index, RUNSTATS INDEX collects this information in FIRSTKEYCARDF and FULLKEYCARDF by default.

- FREQVAL controls the collection of frequent value statistics. If this keyword is specified, it must be followed by two additional keywords, NUMCOLS and COUNT:
  - NUMCOLS indicates the number of key columns to concatenate when collecting frequent value statistics. Specifying 3 means that frequent values are collected on the first three index key columns. The default is 1.
  - COUNT indicates the number of frequent values to collect statistics on. Specifying 15 means that statistics are collected for the 15 most frequent values; the default is 10.

For example, to collect statistics for 15 frequent values on the concatenation of the first three columns of a five-column index XADDR5 specify:

```
RUNSTATS INDEX (DBAPDBA.XADDR5)
    FREQVAL NUMCOLS 3 COUNT 15
```

**Default Considerations:** If you specify KEYCARD, RUNSTATS also collects by default statistics on 10 frequent values for the first index key column, but not if you also specify FREQVAL in the same RUNSTATS utility control statement.

If you specify FREQVAL, then RUNSTATS collects frequent value statistics on only the concatenation of key columns specified by NUMCOLS. Additional frequent value statistics for the first index column will not be collected by default. If you are unaware of this, you may accidentally stop collection of statistics on the first index column. These values will remain in the catalog and can then become out of date.

To prevent this, you need to provide a FREQVAL values for each NUMCOLS value you wish to specify. For example, to continue to collect statistics for 10 frequent values on the first index column, collect 20 frequent values on the concatenation of the first two columns, and 50 frequent values on the concatenation of the first three columns specify:

```
RUNSTATS INDEX (DBAPDBA.XADDR)
    FREQVAL NUMCOLS 1 COUNT 10
    FREQVAL NUMCOLS 2 COUNT 20
    FREQVAL NUMCOLS 3 COUNT 50
```

Running RUNSTATS for a given value of NUMCOLS deletes all the previous statistics from the catalog for the particular concatenation of columns specified by NUMCOLS.

You can also specify KEYCARD and FREQVAL in the same RUNSTATS utility control statement, and collect both cardinality and frequent value statistics. The following statement collects cardinality statistics on index XADDR5 and 15 frequent values on the concatenation of the first three index columns.
**RUNSTATS INDEX (DBAPDBA.XADDR5)**

**KEYCARD**

**FREQVAL NUMCOLS 3 COUNT 15**

*When to Use KEYCARD and FREQVAL:* In this discussion we refer to SYSIBM.SYSINDEXES and SYSIBM.SYSCOLDIST catalog tables. There are corresponding tables, SYsinDexStatSts and SYScoldistStatSts, to consider for partitioned table spaces.

Consider using KEYCARD and FREQVAL to collect index statistics when you are almost certain that you have correlation in your multicolumn index keys and you also have queries with less than fully matching equal predicates.

Queries with fully matching equal predicates use FULLKEYCARDF in SYSIBM.SYSINDEXES to establish filter factors. Queries with only one matching equal predicate use FIRSTKEYCARDF in SYSIBM.SYSINDEXES and FREQUENCYF values in SYSIBM.SYSCOLDIST to establish filter factors. These statistics are collected without using KEYCARD or FREQVAL keywords.

When you collect cardinality values using KEYCARD you can compare the resulting CARDF values in SYSIBM.SYSCOLDIST with the values of FIRSTKEYCARDF and FULLKEYCARDF in SYSIBM.SYSINDEXES to help estimate which column groups are correlated and the degree to which they are correlated.

This will influence your decision on your use of FREQVAL, which you should consider if your data is not uniformly distributed.

We recommend FREQVAL to collect frequent value statistics if there are literals in your partially matching equal predicates. The same recommendation applies if there are input host variables and the query is reoptimized at run time, which we recommend for dynamic SQL in DB2 V5.

Choose the values of NUMCOLS to collect frequent value statistics on those index key concatenations you know to be correlated. Do not for example specify NUMCOLS values that collect statistics on all concatenations of the index key columns if only the first two columns are correlated.

Your choice of COUNT for a given concatenation of columns will be influenced by the cardinality of the column group, the distribution of your data, and the type of queries you run:

- Start with the default COUNT of 10 and adjust upward, as required. Aim to collect frequent value statistics for most of the nonuniformly distributed values.
- You must balance the CPU cost of collecting the statistics, and the space needed for the extra catalog rows, against how effectively DB2 can estimate the filter factors for the particular values of the column group that are specified in the majority of your queries.

DB2 can establish an exact filter factor if your query predicates contain one of the columns for which you have collected frequent value statistics. Otherwise, DB2 calculates a filter factor based on the assumption that all distinct columns of the column group for which you have not collected frequent value statistics are evenly distributed across the remaining percentage of your data. We do not recommend using a high value for
COUNT in an attempt to collect statistics for 100% of your data; this is not necessary.

There is a benefit in collecting correlated statistics in cases where the first column has low cardinality or even a cardinality of one. This often occurs in database designs where the first column is company or country. In this case, the first two indexed columns will always be correlated but DB2 V5 can record more useful frequent value statistics in SYSIBM.SYSCOLDIST. This is a good example of where you need to know your data to identify possible, probable, or definite column correlation.

5.4.5.2 Where DB2 Stores Concatenated Key Column Statistics
In DB2 V5, there are considerable changes to the SYSIBM.SYSCOLDIST and SYSIBM.SYSCOLDISTSTATS catalog tables that store the new key column correlation statistics. Table 32 on page 104 shows a description of the new and changed columns that are updated when you use the new RUNSTATS options.
You can update all the columns manually, but you must use extreme caution to avoid unpredictable access paths. Always keep a record of updates so that you can undo them if necessary. The preferred method is always to run `RUNSTATS`.

### Table 32. Statistics in SYSIBM.SYSCOLDIST and SYSIBM.SYSCOLDISTSTATS

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>VARCHAR(18)</td>
<td>Change of use in DB2 V5. Name of the column. If NUMCOLUMNS is greater than 1,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>name now identifies the first column name of the set of concatenated columns</td>
</tr>
<tr>
<td></td>
<td></td>
<td>associated with the statistics.</td>
</tr>
<tr>
<td>TYPE</td>
<td>CHAR(1) NOT NULL</td>
<td>New in DB2 V5. The type of statistics collected:</td>
</tr>
<tr>
<td></td>
<td>DEFAULT &quot;F&quot;</td>
<td>&quot;C&quot; Cardinality, collected by the KEYCARD option.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;F&quot; Frequent value, collected by FREQVAL option or by default.</td>
</tr>
<tr>
<td>CARDF</td>
<td>FLOAT NOT NULL</td>
<td>New in DB2 V5. The number of distinct values for the column group. Valid</td>
</tr>
<tr>
<td></td>
<td>WITH DEFAULT -1</td>
<td>for statistics collected by KEYCARD.</td>
</tr>
<tr>
<td>COLGROUPCOLNO</td>
<td>VARCHAR(254) NOT</td>
<td>New in DB2 V5. Identifies the set of columns associated with the statistics.</td>
</tr>
<tr>
<td></td>
<td>NULL WITH DEFAULT</td>
<td>The field contains zero for statistics associated with a single column.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Otherwise the field is an array of SMALLINT columns with a dimension equal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>to the value in NUMCOLUMNS.</td>
</tr>
<tr>
<td>NUMCOLUMNS</td>
<td>SMALLINT NOT NULL</td>
<td>New in DB2 V5. The number of columns associated with the statistics.</td>
</tr>
<tr>
<td></td>
<td>WITH DEFAULT -1</td>
<td></td>
</tr>
<tr>
<td>FREQUENCYF</td>
<td>SMALLINT NOT NULL</td>
<td>New in DB2 V5. A number which, when multiplied by 100, gives the percentage</td>
</tr>
<tr>
<td></td>
<td>WITH DEFAULT -1</td>
<td>of rows that contain the value in COLVALUE. For example, a value of 0.153</td>
</tr>
<tr>
<td></td>
<td></td>
<td>indicates 15.3%.</td>
</tr>
<tr>
<td>COLVALUE</td>
<td>VARCHAR(254) NOT</td>
<td>New in DB2 V5. Contains the data of a frequently occurring value; this will</td>
</tr>
<tr>
<td></td>
<td>NULL WITH DEFAULT</td>
<td>be a concatenated string of column values for a TYPE F row with NUMCOLUMNS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>greater than 1. The data may not be printable for noncharacter data types.</td>
</tr>
</tbody>
</table>
You will observe a value of -1 in the FREQUENCYF column for a row with TYPE “C” statistics and a value of -1 in the CARDF column for a row with TYPE “F” statistics.

See Appendix B, “Example of Key Column Correlation Statistics” on page 173, for an example of SQL queries that extract cardinality and frequent value information from SYSIBM.SYSCOLDIST, and cardinality values from SYSIBM.SYSINDEXES.

5.4.5.3 Cost of RUNSTATS for Concatenated Column Statistics

When choosing the FREQUENCY and NUMCOLS options, be aware that many additional rows can be added to SYSIBM.SYSCOLDIST and SYSIBM.SYSCOLDISTSTATS catalog tables. To avoid this, limit use to areas where you have specific query performance problems. If you decide to collect frequent value statistics you should check that sufficient space is available in the SYSSTATS catalog table space which holds tables SYSIBM.SYSCOLDIST and SYSIBM.SYSCOLDISTSTATS.

Table 33 shows the effect on CPU consumption when key column correlation statistics are collected on 12 indexes using both KEYCARD and FREQVAL.

<p>| RUNSTATS CPU Time without KEYCARD and FREQVAL | KEYCARD | FREQVAL |</p>
<table>
<thead>
<tr>
<th>RUNSTATS CPU Time</th>
<th>CPU Time Delta % Compared with RUNSTATS without KEYCARD and FREQVAL</th>
<th>RUNSTATS CPU Time</th>
<th>CPU Time Delta % Compared with RUNSTATS without KEYCARD and FREQVAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>56</td>
<td>74</td>
<td>+32</td>
<td>116</td>
</tr>
</tbody>
</table>

Note: All times are in seconds.

The results show you that there is an increase in CPU time for the collection of cardinality and frequent value statistics on a concatenation of key columns.

5.4.5.4 Limitations of RUNSTATS for Concatenated Columns

RUNSTATS cannot collect statistics on key columns for index screening—that is, on a concatenation of index columns not including the first key column. Index screening predicates improve index access on nonmatching key columns by reducing the number of rows that qualify while searching an index.

RUNSTATS cannot collect statistics on a concatenation of nonindexed columns.

You can update SYSIBM.SYSCOLDIST manually to provide the necessary information if you have a particular problematic query that might benefit. We do not normally recommend this. Use extreme caution or you can get unpredictable access paths. Always keep a record of updates so that you can undo them if necessary.

Chapter 5. Utility Performance Enhancements 105
5.4.5.5 Considerations for RUNSTATS on Concatenated Key Columns

The statistics can improve the access paths and reduce CPU consumption for queries with less than fully matching equal index predicates on correlated index key columns. However, the statistics are collected at a cost of RUNSTATS CPU consumption and catalog growth.

Consider collecting statistics for an index whenever all of the following conditions apply:

- The index has three or more key columns.
- You have queries with equal predicates that match more than one but not all of the index key columns.
- You believe the matching index key columns are correlated.

You need to understand your data to know which key columns may be correlated.

You need to check the size of your SYSSTATS catalog table space for growth when you start to collect the new statistics, and allocate more space if needed.

5.5 Online REORG

DB2 V5 REORG offers great improvement in the availability of data to applications during a REORG.

5.5.1 SHRLEVEL

A new key word, SHRLEVEL, is added to the REORG utility control statement to allow you to choose standard, read-only online, and read-write online reorganization:

These three choices exist for reorganizing a table space (with all the indexes reorganized automatically at the same time), reorganizing an entire index, and reorganizing a partition of a partitioning index:

- SHRLEVEL NONE (standard reorganization)
  This is the default and represents the behavior of the REORG utility in DB2 V4 and prior releases.
- SHRLEVEL REFERENCE (read-only online reorganization)
  This alternative gives applications read access to the object being reorganized almost throughout the entire process of reorganization.

The process involves these activities:

1. DB2 unloads the data from the old (original) copy that is being reorganized, sorts the data by clustering key, and reloads the data into a new (shadow) copy. Concurrently, applications have read access to the original copy.

2. DB2 switches the application accessing to the shadow copy by renaming the data sets for the table space and the indexes. During the actual switching, the applications have no access to the data.

When reorganizing just a partition, DB2 renames the data sets for the partition of the table space, and for this partition of the partitioning index,
and then corrects the row identifiers (RIDs) for the corresponding logical partition in any nonpartitioning indexes.

3. DB2 gives applications read and write access to the shadow copy.

- SHRLEVEL CHANGE (read-write online reorganization)

This alternative gives applications read and write access to the object being reorganized almost throughout the entire process of reorganization.

The process involves these activities:

1. DB2 records the log’s current Log Record Sequence Number (LRSN) and then unloads the data from the old (original) copy of the object being reorganized, sorts the data by clustering key, and reloads the data into a shadow copy. Concurrently, applications have read and write access to the original copy, and DB2 uses its normal facilities to record the changes in the log.

2. DB2 processes the log (reads the log records and applies them to the shadow copy), to bring the shadow copy up to date. This step executes iteratively, with each iteration processing a sequence of log records. Applications have read and write access to the original copy during every iteration except the last. During the last iteration, they have only read access.

3. DB2 switches the application access to the shadow copy by renaming the data sets for the table and the indexes. During the actual switch, applications have no access to the data. When reorganizing just a partition, DB2 renames the data sets for the partition of the table space, and for this partition of the partitioning index, and then corrects the RIDs for the corresponding logical partition in any nonpartitioning indexes.

4. DB2 gives the applications read and write access to the renamed shadow copy.

5.5.2 The Shadow Data Sets

REORG with SHRLEVEL REFERENCE and SHRLEVEL CHANGE use shadow copies of the data to be reorganized during the reorganizing process. The handling of these new shadow data sets differs, depending on whether the objects to be reorganized are DB2 managed or not.

5.5.2.1 DB2-Managed Objects

If the objects to be reorganized are DB2 managed, the REORG utility automatically creates the shadow data sets needed for the reorganization. The shadow data sets are created in the same storage group (STOGROUP) as the objects to be reorganized. Sufficient room must be available on the volumes in the particular STOGROUP to support the additional space allocation of at least the size of the objects to be reorganized. The size and other characteristics of these shadow data sets may be influenced by using the DB2 ALTER command prior to executing the reorganization. Any new characteristics are picked up by the shadow data sets, which eventually become the original data sets. During the switch phase, REORG renames the shadow data sets to the original names. The original (un-REORGed) data sets are deleted during the UTILTERM phase.
5.5.2.2 User-Managed Objects
Performing an online REORG on user-managed objects requires more manual intervention. Shadow data sets must be created before executing the reorganization. They should be defined as LINEAR VSAM data sets with SHAREOPTIONS(3,3). One shadow data set is needed for each object to be reorganized:

- An entire table space
  - A shadow copy for the table space
  - A shadow copy for each index
- One or more partitions of a partitioned table space
  - A shadow copy for each partition in the table space being reorganized
  - A shadow copy for each partition of a partitioning index being reorganized
  - A shadow copy for each logical partition of any nonpartitioning index being reorganized.

There is no automatic deletion of user-managed original (un-REORGed) data sets (referred to as temporary data sets) upon completion of a reorganization.

Allocation size (PRIQTY and SECQTY) is managed through IDCAMS, while free space (PCTFREE and FREEPAGE) is controlled using the DB2 ALTER command.

5.5.2.3 Data-Set Naming Convention
The use of shadow data sets has resulted in an extension to the DB2 data-set naming convention. Temporary data sets are the original (un-REORGed) data sets after the renaming process.

- DB2 standard naming
  - cat.DSNDBx.database.page set.l0001.Annn
- DB2 shadow data-set naming for table space, index, or partitions of those
  - cat.DSNDBx.database.page set.S0001.Annn
- DB2 temporary data-set naming for table space, index, or partitions of those
  - cat.DSNDBx.database.page set.T0001.Annn
- DB2 shadow data-set naming for logical partition of a nonpartitioning index
  - cat.DSNDBx.database.page set.S0mmm.Annn, where mmm is the partition number of the partition being
- DB2 temporary data-set naming for logical partition of a nonpartitioning index
  - cat.DSNDBx.database.page set.T0mmm.Annn, where mmm is the partition number of the partition being reorganized.

Very often, DB2 data sets are managed by Storage Management Subsystem (SMS). The use of SHRLEVEL REFERENCE or SHRLEVEL CHANGE in the REORG utility might cause some changes to be made in the ACS routines. For example, l0001 is used to identify a data set as a DB2 data set.
5.5.3 Phases of the Online REORG Utility

To support online reorganization, the REORG utility has three new phases, as well as changes in some of the existing phases:

- **UTILINIT**
  Initialization and setup. Unchanged.

- **UNLOAD**
  Unloads data to be reorganized into a sequential data set (SYSREC). Data is sorted in clustering sequence if a clustering index exists and either SORTDATA or SHRLEVEL CHANGE is specified.
  You can use the NOSYSREC option (together with SORTDATA) to omit this unload data set, which gives a performance advantage by avoiding I/O to intermediate data sets (for SHRLEVEL CHANGE, NOSYSREC is the default).

- **RELOAD**
  Reloads the data from the unload phase and extracts keys to build indexes.
  Specifying SORTKEYS allows the extracted keys to be passed to sort in memory and sorted in parallel with the reload phase. This reduces elapsed time by avoiding I/O, besides the reduction gained by sorting in parallel. For SHRLEVEL CHANGE, SORTKEYS is the default.
  You can use the COPYDDN (and optionally the RECOVERYDDN) option to take an inline image copy (and optionally a remote copy) during the reload phase. This is a choice with SHRLEVEL NONE and mandatory in SHRLEVEL REFERENCE and SHRLEVEL CHANGE.

- **SORT**
  Sorts the extracted index keys.
  If the SORTKEYS option is specified, SORT passes the sorted keys in memory to the build phase rather than writing them to SORTOUT for the build phase to read. This reduces elapsed time.

- **BUILD**
  Builds indexes.
  If REORG TABLESPACE PART SHRLEVEL NONE is specified, this phase also corrects the logical partition of any nonpartitioning index.

- **LOG**
  A new phase for DB2 V5.
  Executes only if SHRLEVEL CHANGE is specified.
  During this phase, log records are applied to the shadow copy. This step executes iteratively, each iteration processing a sequence of log records. During the last iteration, applications have only read access to the original copy.
  Changed pages are appended to the inline full-image copy taken during the reload phase by two incremental copies taken during the last iteration—one just before writers are drained, and one just after.

- **SWITCH**
  A new phase for DB2 V5.
  Executes only if SHRLEVEL REFERENCE or SHRLEVEL CHANGE is specified.
During this phase, DB2 switches access to the shadow copies of the table and index spaces. This switching is done by IDCAMS rename, and there is no access to the data during this phase. The more physical data sets IDCAMS has to process, the longer the data remains unavailable to the applications.

If the object being reorganized is a partition of a partitioned table space, the actual partition of the table space and of the partitioning index are renamed. Any logical partition of a nonpartitioning index is not processed here.

- **BUILD2**
  
  A new phase for DB2 V5.
  
  Executes only if SHRLEVEL REFERENCE or SHRLEVEL CHANGE is specified and the object that is being reorganized is a partition of a partitioned table space with one or more nonpartitioning indexes.

  During this phase, DB2 corrects logical partitions in nonpartitioning indexes rather than replacing them. This method is used to allow concurrent reorganizations of several partitions. It uses the shadow created for the logical part of the nonpartitioning index to correct the RIDs in the original copy of the nonpartitioning index. When completed, the shadow of the logical partition is deleted.

  During this operation, no access, other than read only is allowed to the logical partition being corrected in the nonpartitioning index. Read and update are allowed in other logical partitions (those not being reorganized); however, insert and delete are permitted only for Type 2 indexes.

  This is the most time-consuming operation with limited application access to the data. As a result, trying to optimize the working conditions for this process seems worthwhile. One new option, PIECESIZE, introduced in DB2 V5 in the CREATE INDEX statement, allows you to divide nonpartitioning indexes into smaller pieces and spread these pieces evenly on the DASD available. This avoids I/O contention and thereby reduces the elapsed time for the corrections, which is the time when the reorganized object is unavailable to applications. For more information about PIECESIZE, see 3.5, "Nonpartitioning Index Pieces" on page 45.

- **UTILTERM**
  
  This is the cleanup phase. Here, the original DB2-managed data sets (un-REORGed) are deleted.

### 5.5.4 The Phases and the Locks in Effect

Figure 21 on page 111, Figure 22 on page 111, Figure 23 on page 112, and Figure 24 on page 113 illustrate the phases processed, and the level of concurrent application access to the data being reorganized in each of the three types of the REORG utility.
Figure 21. The Different Phases in REORG SHRLEVEL NONE

SHRLEVEL NONE is the default and the effect is the same as running the REORG utility in DB2 V4.

The unload phase allows read access to the data (indicated by DW = Drain Writers), while the other phases disallow all access (indicated by DA = Drain All).

**Note:** SYSREC, SYSUT1 and SORTOUT are all shown here but some, or all of these may be avoided by specifying NOSYSREC and/or SORTKEYS.

Figure 22. The Different Phases in REORG SHRLEVEL REFERENCE

SHRLEVEL REFERENCE allows read-only online REORG.
With SHRLEVEL REFERENCE, the read access to the object being reorganized is extended to almost the entire duration of the REORG utility (indicated by DW = Drain Writers). Only the switch phase (new in DB2 V5) disallows all access (indicated by DA = Drain All) to the data being reorganized.

**Note:** The indexes of the data being reorganized are treated slightly differently here than in SHRLEVEL NONE. Since they play no direct role in the online reorganization, they could potentially be in recovery-pending state without affecting the reorganization. During the switch phase, new copies take their place.

![Figure 23. The Different Phases in REORG SHRLEVEL CHANGE](image)

SHRLEVEL CHANGE allows read-write online REORG.

With SHRLEVEL CHANGE, the applications have read and write access to the data being reorganized through almost the entire process. By claiming the readers (indicated by CR = Claim Reader) the phases allow concurrent read and write access to the data. It is only through the final iteration in the log phase that the writers are drained (indicated by DW = Drain Writers) allowing read access only. During the last phase, the switch phase, no access to the data being reorganized is permitted (indicated by DA = Drain All).

**Note:** During the entire execution of the REORG utility, locks on the data prevent physical mass deletes. If a mass delete is requested, only space map pages are updated.

As described in 5.5.3, “Phases of the Online REORG Utility” on page 109, online REORG with SHRLEVEL CHANGE has three new phases. The last new phase, the Build 2 phase, has not yet been shown. This is the phase that corrects parts of nonpartitioning indexes when reorganizing a part of a partitioned table that has nonpartitioning indexes.
When reorganizing a partitioned table space, the switch phase handles (using rename, dedrain, and delete) the table space partition (indicated by TS Part N) and the part of the partitioning index (indicated by IX Part N). The Build2 phase then handles the logical partition of the nonpartitioning index by updating the RIDs. During this phase, all is drained (indicated by DA = Drain All) on this logical partition (indicated by NPI LPART N) of the nonpartitioning index. Read and update are allowed in other partitions that are not being reorganized, but no deletes or inserts are permitted.

5.5.5 The Mapping Table

During the unload and reload phases, the REORG utility uses a mapping table to map the RIDs used in the original copy of the data set to the RIDs used in the shadow copy. Updates to this mapping table are not logged; hence no restart is possible during these phases.

This mapping table is used later in the log phase when logged updates to the original copy are applied to the shadow.

This is a five-step process for each iteration:

1. Read log records.
2. Sort log by old RID.
3. Translate old to new RID.
4. Sort log by new RID.
5. Apply log records to shadow.

The mapping table is a DB2 table that must be created explicitly in a segmented table space:
CREATE TABLESPACE tsname SEGSIZE 4;

CREATE TABLE tbname
    (TYPE CHAR(1) NOT NULL,
     SOURCE_RID CHAR(5) NOT NULL,
     TARGET_XRID CHAR(9) NOT NULL,
     LRSN CHAR(6) NOT NULL)
IN tsname;

CREATE TYPE 2 UNIQUE INDEX ixname ON tbname
    (SOURCE_RID ASC, TYPE, TARGET_XRID, LRSN);

Access to the mapping table is always index-only.

The number of rows in this table is unlikely to exceed 110% of the rows in the
table or partition to be reorganized.

Each concurrent execution of REORG SHRLEVEL CHANGE should have a
separate mapping table affiliated with it. Otherwise, the utility executions may
be serialized.

5.5.6 How to Execute Online REORG

Obviously, DB2 V5 has some changes to the REORG utility control statement.
The following example shows a typical online read-write REORG:

REORG TABLESPACE DSN8D51A.DSN8S51D COPYDDN(LOCCOP)
    RECOVERYDDN(REMCCOP) SHRLEVEL CHANGE
    DEADLINE 1997-3-10-06.00
    MAPPINGTABLE MAPPTAB01
    MAXRO 200
    LONGLOG DRAIN
    DELAY 900

The example contains several new keywords:

- COPYDDN and RECOVERYDDN take inline image copies, one for the local
  site and one for the remote site.
- SHRLEVEL CHANGE makes this reorganization an online REORG with
  concurrent read-write access.
- DEADLINE 1997-3-10-06.00 gives the REORG utility a deadline for the start of
  the switch phase.
- MAPPINGTABLE provides the name of the mapping table to be used in this
  reorganization.
- MAXRO 200 means that the maximum desired time for the last log-apply
  iteration is 200 seconds (the last iteration is the part of the log phase that
  allows only read access to the data).
- LONGLOG DRAIN tells the REORG utility to drain the writer claim class after
  issuing a LONGLOG message. This action is taken if the log iteration process
  does not catch up with the applications’ writing to the log.
- DELAY 900 is the period of time the REORG utility waits after issuing the
  LONGLOG message until the draining takes place.

Note: This is an example of a REORG utility statement. For a complete
description of the new key words, see DB2 for OS/390 Version 5 Utility
Guide and Reference.
5.5.7 Measurements

The following measurements are obtained on a 10-way 9672 with 3390 DASDs.

Table 34 shows the performance measurement results of the REORG utility on a table space with no concurrent updates when the cluster ratio is 100%.

<table>
<thead>
<tr>
<th>Time</th>
<th>SHRLEVEL NONE</th>
<th>SHRLEVEL REFERENCE</th>
<th>SHRLEVEL CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Seconds</td>
<td>Seconds</td>
<td>Seconds</td>
</tr>
<tr>
<td>CPU</td>
<td>880</td>
<td>873</td>
<td>970</td>
</tr>
<tr>
<td>Elapsed</td>
<td>856</td>
<td>961</td>
<td>1242</td>
</tr>
</tbody>
</table>

Note: All measurements are done with NOSYSREC and SORTKEYS specified, and also include inline image copy.

Deltas are between the actual SHRLEVEL performed and SHRLEVEL NONE.

Table 35 shows the performance measurement results of the REORG utility on a table space with no concurrent updates when the cluster ratio is 83%.

<table>
<thead>
<tr>
<th>Time</th>
<th>SHRLEVEL NONE</th>
<th>SHRLEVEL REFERENCE</th>
<th>SHRLEVEL CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Seconds</td>
<td>Seconds</td>
<td>Seconds</td>
</tr>
<tr>
<td>CPU</td>
<td>838</td>
<td>825</td>
<td>2869</td>
</tr>
<tr>
<td>Elapsed</td>
<td>820</td>
<td>921</td>
<td>3049</td>
</tr>
</tbody>
</table>

Note: All measurements are done with NOSYSREC and SORTKEYS specified, and also include inline image copy.

Deltas are between the actual SHRLEVEL performed and SHRLEVEL NONE.

These measurements show that the cost of reorganizing your data increases, both in CPU time and elapsed time, as the cluster ratio goes down.

Note: Slightly lower CPU time for SHRLEVEL(REFERENCE) compared with SHRLEVEL(NONE) can be attributed to measurement noise.

You need to balance the increase in reorganization cost against the benefit of having data available during virtually the entire reorganization period, and the fact that your queries perform better when the data is reorganized. With prior releases of DB2, you would probably have had to reorganize when given the opportunity, not when the data needed to be reorganized.

Table 36 on page 116 and Table 37 on page 116 show measurement results of the REORG utility for both partitioning and nonpartitioning indexes.

Table 36 on page 116 shows the performance measurement results of the REORG utility on a partitioning Type 2 index without concurrent updates.
Table 37 shows the performance measurement results of the REORG utility on a nonpartitioning Type 2 index without concurrent updates.

Table 37 shows the performance measurement results of the REORG utility on a nonpartitioning Type 2 index without concurrent updates.

With no concurrent update to the data being reorganized, these measurements show that the three SHRLEVEL alternatives consume approximately the same CPU time and elapsed time.

Note: REORG TABLESPACE dbname.tsname PART n does reorganize the nonpartitioning index—that is, PCTFREE, FREEPAGE, and LEAFDIST are not optimal.

5.5.7.1 Concurrent Batch Activity

Assume that the table to be reorganized has 5 million rows in 10 partitions. Each partition contains 500 000 rows. The table has one nonpartitioning nonunique index. Concurrent with the online REORG utility, the batch job is run with sequential updates to the table space. The updates are made to nonindexed columns. The batch job is run after the unload phase of the online REORG utility.

Table 38 shows the performance measurement results of the REORG utility on a table space with two Type 2 indexes and concurrent sequential batch updates to the table after the unload phase finished. The cluster ratio is 100%.

Table 38 shows the performance measurement results of the REORG utility on a nonpartitioning Type 2 index without concurrent updates.

With no concurrent update to the data being reorganized, these measurements show that the three SHRLEVEL alternatives consume approximately the same CPU time and elapsed time.

Note: All times are in seconds.

Table 36. REORG Type 2 Partitioning Index without Concurrent Updates

<table>
<thead>
<tr>
<th>DB2 V4</th>
<th>SHRLEVEL NONE</th>
<th>SHRLEVEL REFERENCE</th>
<th>SHRLEVEL CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Time</td>
<td>Elapsed Time</td>
<td>CPU Time</td>
<td>Elapsed Time</td>
</tr>
<tr>
<td>165</td>
<td>293</td>
<td>163</td>
<td>206</td>
</tr>
</tbody>
</table>

Note: All times are in seconds.

Table 37. REORG Type 2 Nonpartitioning Index without Concurrent Updates

<table>
<thead>
<tr>
<th>DB2 V4</th>
<th>SHRLEVEL NONE</th>
<th>SHRLEVEL REFERENCE</th>
<th>SHRLEVEL CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Time</td>
<td>Elapsed Time</td>
<td>CPU Time</td>
<td>Elapsed Time</td>
</tr>
<tr>
<td>141</td>
<td>199</td>
<td>149</td>
<td>158</td>
</tr>
</tbody>
</table>

Note: All times are in seconds.

Table 38. REORG Table Space: Scenario 3. The table has two Type 2 indexes. The cluster ratio is 100%. The concurrent batch updates are sequential.

<table>
<thead>
<tr>
<th>No Updates</th>
<th>1 Million Updates</th>
<th>2 Million Updates</th>
<th>3 Million Updates</th>
<th>4 Million Updates</th>
<th>5 Million Updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Time ET</td>
<td>CPU Time ET</td>
<td>CPU Time ET</td>
<td>CPU Time ET</td>
<td>CPU Time ET</td>
<td>CPU Time ET</td>
</tr>
<tr>
<td>970</td>
<td>1242</td>
<td>1299</td>
<td>1651</td>
<td>1625</td>
<td>2044</td>
</tr>
</tbody>
</table>

Note: ET is elapsed time. All times are in seconds.
Table 38 shows the CPU time and elapsed time for five different scenarios of REORG TABLESPACE. The first is executed without any concurrent updates to the table. The second through the fifth are run with concurrent sequential updates starting with one million concurrent updates, incremented in steps of one million until five million is reached.

The differences to be noticed in this set of measurements arise because of the log phase. However, be aware of the following considerations:

- The reload phase

  The reload phase builds the mapping table index with SHRLEVEL(CHANGE). This adds one new index entry for each row that is loaded. The CPU and elapsed times for the reload phase are uniformly the same either with no concurrent updates or with 5 million concurrent updates. The CPU and elapsed times depend on cluster ratio. If the index entries are in order, they can be appended. However, the index entries may not be in order, so performance will get worse if they come in and must be inserted instead of appended. The efficiency of the building of the mapping table index is proportional to the organization of the data. The CPU and elapsed times for the reload phase are not significantly high if the cluster ratio is 100%.

- The log phase

  With no concurrent updates, the CPU and elapsed times for the log phase are negligible. With no updates during the reorganization, there is no log-apply iteration.

  Looking at Scenarios 2 through 5, each increment of one million updates to the table results in an approximate increase of 300 seconds in CPU time, and 400 seconds in elapsed time. This is time spent reading one million log records, translating RIDs, and applying the same number of changes to the shadow copy.

  It is important to remember that it is only during the last iteration in the log phase that writers are drained and access to the data is restricted to read only.

- The copy phase

  The inline full-image copy is taken during the reload phase, and the incremental image copies are taken during the last iteration in the log phase. Also, here it is important to remember that the second incremental image copy is taken after writers are drained during the last iteration in the log phase, and consequently does not add significantly to the time when the data being reorganized can be accessed only in read mode.

The only time the data being reorganized is unavailable is during the switch phase. This phase is not influenced by any concurrent update activity; it depends only on the number of physical data sets that have to be renamed.

We now look at the same scenario. However, the only change is that the cluster ratio is 83%.

Table 39 on page 118 shows the performance measurement results of the REORG utility on a table space with two type 2 indexes and concurrent sequential batch updates to the table after the unload phase finished. The cluster ratio is 83%.
Table 39. REORG Table Space: Scenario 4. The table has two Type 2 indexes. The cluster ratio is 83%. The concurrent batch updates are sequential.

<table>
<thead>
<tr>
<th>No Updates</th>
<th>1 Million Updates</th>
<th>2 Million Updates</th>
<th>3 Million Updates</th>
<th>4 Million Updates</th>
<th>5 Million Updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Time</td>
<td>ET</td>
<td>CPU Time</td>
<td>ET</td>
<td>CPU Time</td>
<td>ET</td>
</tr>
<tr>
<td>2869</td>
<td>3049</td>
<td>3091</td>
<td>3455</td>
<td>3717</td>
<td>4024</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3091</td>
<td>3455</td>
<td>3432</td>
<td>4024</td>
<td>4502</td>
<td>4995</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3455</td>
<td>3432</td>
<td>3717</td>
<td>4502</td>
<td>4995</td>
<td>4428</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3432</td>
<td>4024</td>
<td>3717</td>
<td>4502</td>
<td>4995</td>
<td>4428</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3717</td>
<td>4502</td>
<td>4995</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4502</td>
<td>4995</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4995</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4428</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5611</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: ET is elapsed time. All times are in seconds.

Table 39 shows the CPU time and elapsed time for five different scenarios of REORG TABLESPACE. The first is executed without any concurrent updates to the table. The second through the fifth represent runs with concurrent sequential updates starting with one million concurrent updates, incremented in steps of one million until five million is reached.

With no concurrent updates, the total CPU time and elapsed time are 2869 and 3094 compared with 970 and 1242 in Table 38 on page 116. This is because of the significant increase in the CPU and elapsed times in the reload phase caused by the lower cluster ratio (83%).

We conclude from the above that if you need to reorganize the data online in your environment, it is best to execute the REORG utility with SHRLEVEL(CHANGE) frequently and not wait until the cluster ratio is low. This would ensure the efficiency of both the reorganization process and concurrent batch activity. However, the number of nonpartitioning indexes on the table determines the frequency of reorganization. It is a trade-off between data availability and retrieval efficiency.

5.5.7.2 Concurrent Online Activity

Assume that the workload consists of a partitioned table with 11 partitions. The first 10 partitions contain 500,000 rows each, and the last partition contains 2.5 million rows. The table has one Type 2 clustering index and one Type 2 nonpartitioning, nonunique index. The online REORG utility is run on the last partition of the table. Concurrent with the online REORG utility, there are 25 online transactions, each inserting 10,000 rows into the last partition of the table.

The cluster ratio is 100%.

Two measurement runs were done as follows:

- The REORG utility and the online transactions were started at the same time, which means that most of the contention takes place in the unload phase.
- The online transactions were started at the end of the unload phase of the REORG utility, so that no contention takes place during the unload phase.

Table 40 on page 119 shows the performance measurement results of the REORG utility on the last partition of the partitioned table space with concurrent online transaction activity on that partition.
Table 40 shows the CPU time, elapsed time, and delta in percent for a reorganization of one partition of a partitioned table space without and with concurrent OLTP activity.

The CPU time and elapsed time for the online REORG on the table space partition with concurrent OLTP during the unload phase go up by 127% and 235%, respectively, compared with the online REORG without any concurrent OLTP for the following reasons:

- **The unload phase**
  During the unload phase I/O contention results when both this utility and the concurrent OLTP access the same data. Also, there are more records to be read by the unload phase.

  From a total of 250 000 inserts into this partition by the OLTP, 219 164 new rows are added to the partition ahead of the unload process, resulting in a total of 2 719 164 rows to read, compared with 2 500 000 rows when no concurrent activity takes place.

- **The reload phase**
  Compared with online REORG without any concurrent OLTP, the CPU time increases in the reload phase because there are more rows to process.

  In this particular case, the table can hold a maximum of 11 rows per page, and the percent free space per page is set to 5. This means that 219 164 inserts have already gone into new pages at the end of the table space. Thus, compared with the average online REORG, the CPU time and elapsed time consumed in the reload phase in this measurement job are likely to be high.

  During the reload phase, an inline full-image copy is created.

- **The build phase**
  Compared with online REORG without any concurrent online transactions, the CPU time and elapsed time increase in the build phase because there are more rows (and keys) to build indexes on.

- **The sort phase**
  SORTKEYS is the mandatory default in REORG SHRLEVEL CHANGE, and changes the process of sorting keys for the index build phase from a normal utility phase to a subtask executed in parallel with the reload and build phases. (See 5.1, “LOAD and REORG” on page 73.)
• The log phase

Compared with online REORG without any concurrent OLTP, the CPU time and elapsed time in the log phase increase significantly. In the REORG job without any concurrent activity, the log phase does no more than validate that no updates have taken place in the table being reorganized. With concurrent activity, the log phase applies changes from the log to the shadow copy in an iterative process. In addition, two incremental image copies are taken during the last iteration in the log phase and appended to the full-image copy created earlier during the reload phase.

• The copy phase

Compared with online REORG without any OLTP, the CPU time increases in the copy phase because more pages have to be copied.

• The switch phase

No difference results because the switch phase is not influenced by concurrent activity, only by the number of physical data sets to rename.

• The build2 phase

Compared to online REORG without any concurrent online transactions, the CPU time and elapsed time increase in the Build2 phase because there are more index entries to correct as the number of rows increase.

If the OLTP starts after the unload phase of the REORG utility, the percentage increase in CPU consumption and elapsed time drops significantly.

Executing online REORG utility on your data while normal activity continues against the same data imposes a cost in both increased CPU consumption and increased elapsed time. These costs must be balanced against the benefits of having better query performance on well organized data.

Table 41 shows the performance measurement results of the REORG utility on the partitioning index of the table used in Table 40 on page 119.

<table>
<thead>
<tr>
<th>Online Transactions</th>
<th>No OLTP</th>
<th>With OLTP</th>
<th>Delta %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU Time</td>
<td>Elapsed Time</td>
<td>CPU Time</td>
</tr>
<tr>
<td>During Unload</td>
<td>236</td>
<td>309</td>
<td>351</td>
</tr>
<tr>
<td>After Unload</td>
<td>236</td>
<td>309</td>
<td>284</td>
</tr>
</tbody>
</table>

**Note:** All times are in seconds.

The row named *During Unload* indicates that the REORG utility and the online transactions were started at the same time.

The row named *After Unload* indicates that the online transactions were started at the end of the unload phase.

Table 41 shows the CPU time, elapsed time, and delta in percent for reorganization of the partitioning index without and with concurrent OLTP.

The CPU time and elapsed time for the online REORG of the partitioning index with concurrent OLTP on the table space partition during the unload phase go up by 49% and 162% respectively, compared with the online REORG without any concurrent OLTP.
If the online transactions are started after the unload phase of the REORG utility, there is a significant reduction in the percentage increase in CPU consumption and elapsed time.

Table 42 shows the performance measurement results of the OLTP run on the last partition of the table:

- Without concurrent online REORG
- With concurrent online REORG on the table space partition
- With concurrent online REORG on the partitioning index.

<table>
<thead>
<tr>
<th>Table 42 (Page 1 of 2). Performance of OLTP. Each transaction inserts 10 000 rows into partition 11.</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Online Transaction</strong></td>
<td><strong>No Concurrent Online REORG</strong></td>
<td><strong>Concurrent Online REORG of Table Space Partition</strong></td>
<td><strong>Concurrent Online REORG of Partitioning Index</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>CPU Time</strong></td>
<td><strong>Elapsed Time</strong></td>
<td><strong>CPU Time</strong></td>
<td><strong>Elapsed Time</strong></td>
</tr>
<tr>
<td>1</td>
<td>89</td>
<td>897</td>
<td>94</td>
<td>1020</td>
</tr>
<tr>
<td>2</td>
<td>91</td>
<td>691</td>
<td>93</td>
<td>1022</td>
</tr>
<tr>
<td>3</td>
<td>89</td>
<td>904</td>
<td>94</td>
<td>1028</td>
</tr>
<tr>
<td>4</td>
<td>89</td>
<td>896</td>
<td>92</td>
<td>1024</td>
</tr>
<tr>
<td>5</td>
<td>89</td>
<td>898</td>
<td>94</td>
<td>1027</td>
</tr>
<tr>
<td>6</td>
<td>89</td>
<td>899</td>
<td>93</td>
<td>1022</td>
</tr>
<tr>
<td>7</td>
<td>89</td>
<td>901</td>
<td>94</td>
<td>1028</td>
</tr>
<tr>
<td>8</td>
<td>89</td>
<td>891</td>
<td>94</td>
<td>1026</td>
</tr>
<tr>
<td>9</td>
<td>88</td>
<td>897</td>
<td>94</td>
<td>1025</td>
</tr>
<tr>
<td>10</td>
<td>89</td>
<td>897</td>
<td>94</td>
<td>1027</td>
</tr>
<tr>
<td>11</td>
<td>89</td>
<td>902</td>
<td>94</td>
<td>1026</td>
</tr>
<tr>
<td>12</td>
<td>88</td>
<td>900</td>
<td>93</td>
<td>1021</td>
</tr>
<tr>
<td>13</td>
<td>89</td>
<td>892</td>
<td>93</td>
<td>1029</td>
</tr>
</tbody>
</table>
Table 42 (Page 2 of 2). Performance of OLTP. Each transaction inserts 10 000 rows into partition 11.

<table>
<thead>
<tr>
<th>Online Transaction</th>
<th>No Concurrent Online REORG</th>
<th>Concurrent Online REORG of Table Space Partition</th>
<th>Concurrent Online REORG of Partitioning Index</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU Time</td>
<td>Elapsed Time</td>
<td>CPU Time</td>
</tr>
<tr>
<td>14</td>
<td>90</td>
<td>904</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>89</td>
<td>943</td>
<td>94</td>
</tr>
<tr>
<td>15</td>
<td>88</td>
<td>898</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>935</td>
<td>95</td>
</tr>
<tr>
<td>16</td>
<td>90</td>
<td>897</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>89</td>
<td>939</td>
<td>95</td>
</tr>
<tr>
<td>17</td>
<td>89</td>
<td>905</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>939</td>
<td>94</td>
</tr>
<tr>
<td>18</td>
<td>88</td>
<td>892</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>89</td>
<td>938</td>
<td>94</td>
</tr>
<tr>
<td>19</td>
<td>89</td>
<td>905</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>89</td>
<td>949</td>
<td>93</td>
</tr>
<tr>
<td>20</td>
<td>88</td>
<td>899</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>944</td>
<td>95</td>
</tr>
<tr>
<td>21</td>
<td>89</td>
<td>894</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>89</td>
<td>947</td>
<td>95</td>
</tr>
<tr>
<td>22</td>
<td>90</td>
<td>892</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>89</td>
<td>946</td>
<td>93</td>
</tr>
<tr>
<td>23</td>
<td>90</td>
<td>895</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>89</td>
<td>944</td>
<td>94</td>
</tr>
<tr>
<td>24</td>
<td>89</td>
<td>894</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>89</td>
<td>935</td>
<td>94</td>
</tr>
<tr>
<td>25</td>
<td>89</td>
<td>896</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>946</td>
<td>93</td>
</tr>
</tbody>
</table>

Note: All times are in seconds.

The numbers for online REORG Table Space Partition and online REORG Partitioning Index in the top line correspond to the measurement done when the REORG utility and the online transactions were started at the same time.

The numbers for online REORG Table Space Partition and online REORG Partitioning Index in the bottom line correspond to the measurement done when the online transactions were started at the end of the unload phase.

5.5.8 Conclusions

Often you can find that quite normal activity against a table can result in conditions in the table space that make reorganization necessary.

For instance, in the measurement shown in Table 40 on page 119, a large number of rows are inserted into a table with 11 rows per page and PCTFREE set
to 5. The insertion of rows into new pages at the end of the table space degenerates the clustering of the table space, probably increases the I/O activity to retrieve the data, and may also create new extents in the data set. All these factors influence the effectiveness of—and thereby the response time of—online transactions.

For applications where response time is critical, especially 24 x 7, it is crucial that factors that adversely affect performance be dealt with, and dealt with as soon as possible. DB2 V5, with the introduction of REORG SHRLEVEL CHANGE, provides the ability to respond quickly and makes a considerable advance toward virtually continuous availability.
Chapter 6. Client/Server Enhancements

In this chapter, we describe the following DB2 V5 performance-related enhancements in the Client/Server environment:

- Stored procedures
- Native TCP/IP support
- DRDA application performance
- Application portability.

6.1 Stored Procedures

In this section, we describe the use of multiple result sets to return data from several cursors in a single network message. We cover the use of commit on return to enable a stored procedure to issue the commit for all changes within a unit of work. We show the performance improvement from improved network utilization with the multiple result sets and commit on return options.

We describe MVS WLM enclave support for stored procedures and the additional benefits of using WLM to establish multiple address spaces. We show the performance improvement for stored procedures using subprograms.

6.1.1 Catalog Changes for SYSIBM.SYSPROCEDURES

Catalog table SYSIBM.SYSPROCEDURES contains several new columns referring to new options available in DB2 V5. Table 43 on page 126 defines the new performance-related columns added to SYSIBM.SYSPROCEDURES that we refer to.
Table 43. New Columns in SYSIBM.SYSPROCEDURES

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULT_SETS</td>
<td>SMALLINT NOT NULL</td>
<td>The maximum number of query results that can be stored by this stored</td>
</tr>
<tr>
<td></td>
<td>WITH DEFAULT 0</td>
<td>procedure. Zero indicates that there are no result sets.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMMIT_ON_RETURN</td>
<td>CHAR(1) NOT NULL</td>
<td>Indicates that the unit of work is always to be committed immediately upon</td>
</tr>
<tr>
<td></td>
<td>WITH DEFAULT “N”</td>
<td>successful return (nonnegative SQLCODE) from this stored procedure.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N The unit of work is to continue.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y The unit of work is to commit.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A null value means the same as N.</td>
</tr>
<tr>
<td>WLM_ENV</td>
<td>CHAR(18) NOT NULL</td>
<td>The name of the WLM environment to be used to run this stored procedure.</td>
</tr>
<tr>
<td></td>
<td>WITH DEFAULT</td>
<td>With a blank value, the stored procedure is run in the DB2-established</td>
</tr>
<tr>
<td></td>
<td></td>
<td>stored procedure address space.</td>
</tr>
<tr>
<td>PGM_TYPE</td>
<td>CHAR(1) NOT NULL</td>
<td>Whether the stored procedure runs as a main program or a subprogram.</td>
</tr>
<tr>
<td></td>
<td>WITH DEFAULT “M”</td>
<td>M The stored procedure runs as a main program</td>
</tr>
<tr>
<td></td>
<td></td>
<td>S The stored procedure runs as a subprogram.</td>
</tr>
</tbody>
</table>

Existing rows for stored procedures will have the defaults for these new columns when you migrate to DB2 V5. Thus you have no query result sets, DB2 rather than workload manager controls the stored procedure address space, the client is responsible for commit processing, and all procedures run as main programs.

### 6.1.2 Multiple Query Result Sets

Stored procedures can open multiple cursors and return multiple query result sets to the client that invokes the stored procedure. Figure 25 on page 127 illustrates the process with a single call from the client to the stored procedure causing two cursors, C1 and C2, to be opened, and the data is returned as two result sets in a single message.
The main advantages are these:

- Network traffic and processor costs are reduced because DB2 uses a single network message to call a stored procedure and return all the result sets.
- Control of SQL access paths is improved because the client uses a stored procedure containing efficient SQL, written by the application programmer, instead of issuing SQL directly.
- Authorization checking can be limited to the owner of the package associated with the stored procedure, rather than granting table accesses to end users. You can cache package authorizations in DB2 V5.
- Organizations having VisualAge, PowerBuilder, VisualBasic, or Lotus Approach can exploit the query result sets returned from stored procedures.

### 6.1.2.1 Returning Query Result Sets

You must meet the following conditions to return multiple result sets from a DB2 V5 stored procedure:

- The DRDA client can support result sets.
- The row in SYSIBM.SYSPROCEDURES for the stored procedure must have a value greater than zero in the RESULT_SETS column.
- The stored procedure must open one or more cursors declared with the WITH RETURN clause for each query result set to be returned to the client.
- The cursors must stay open till the program ends.
- If you specify COMMIT_ON_RETURN=Y, cursors should be declared with the WITH HOLD option.

When the stored procedure ends, then DB2:

- Returns the result set for each open cursor to the client.
• Transmits the result sets asynchronously to reduce the time the client waits for the entire request to complete.

• Returns result sets to the client in the same order that the cursors are opened by the stored procedure.

6.1.2.2 Accessing Query Result Sets

There are two ways to access result sets:

• Use ODBC or CLI calls, whether directly from C or C++, or use an application to issue the calls such as VisualAge, PowerBuilder, VisualBasic, or Lotus Approach.

• From a DB2 V5 application using new embedded SQL statements,
  1. Declare a RESULT_SET_LOCATOR variable to point at each result set.
  2. Call the stored procedure.
  3. If the stored procedure can return a variable number of result sets, use DESCRIBE PROCEDURE to obtain the number and names of the cursors.
  4. Get the result set locator value for each result set using the ASSOCIATE LOCATORS statement.
  5. Use the ALLOCATE CURSOR statement to define a cursor for each result set. Every cursor is associated with a different RESULT_SET_LOCATOR variable.
  6. If the columns in the result set can vary, use the DESCRIBE CURSOR statement to obtain column information.
  7. Fetch rows from each results set.

The programs DSN8ED1 and DSN8ED2 in the DB2 sample library are examples of stored procedures that use result sets.

Use of Temporary Tables to Return Result Sets: You can facilitate the return of nonrelational data to a DRDA client by using a temporary table to return result sets from a stored procedure. For example, you can issue an IMS transaction using MVS/APPC, insert data from the IMS reply message into a temporary table, open a cursor to select data from the temporary table, and return the data to the client when the stored procedure ends.

6.1.2.3 Measurements for Multiple Result Sets

A multithread tool was used to run four OS/2 clients concurrently. Two test runs were made, one using CLI to run the queries directly, the other using CLI to call stored procedures using the multiple result-sets option. The queries consisted of 1, 2, and 4 cursors. In the case of direct CLI, the queries retrieved either a single 32 KB block or multiple blocks per cursor depending on the quantity of data returned. Table 44 on page 129 tabulates a comparison between using CLI to run queries directly, and using CLI to run stored procedures with the multiple result set option.
Using one cursor and one block is more expensive when issued from a stored procedure. All other cases are less expensive.

The benefit derives from a single call from the client to the stored procedure, and from a single message to return the blocks of data from each cursor to the client. Maximum benefit occurs when the data from each cursor fits into a single 32 KB block. With four open cursors, returning four blocks of data (one for each cursor) in a single message saves three network messages over direct CLI.

When the data volumes become large, and when the volume of data returned for each cursor exceeds a single 32 KB block, multiple result sets still provide CPU and network reductions, but those reductions become insignificant with very large answer sets. However, client/server applications should not normally return large numbers of rows.

### 6.1.3 Commit on Return

You can use the commit on return option to reduce network communication for applications. If you specify a COMMIT_ON_RETURN value of "Y" in SYSIBM.SYSPROCEDURES, then DB2 automatically commits all work when the stored procedure completes successfully. This commit option will commit all changes within the unit of work, not just the changes made by the stored procedure, and can include commits at multiple servers.

Because DB2 releases all locks acquired by the stored procedure after the commit completes, transaction throughput can increase. The benefit of having the commit issued as soon as the stored procedure ends is that you do not need to write the application program to release locks as soon as possible.

In DB2 V4, there are two network line flows from the client, one to invoke the stored procedure and one to commit the unit of work. In DB2 V5, the commit on return option causes the commit to occur before returning to the client with the

### Table 44. Comparison of Direct CLI and Multiple Result Sets with CLI

<table>
<thead>
<tr>
<th>Query Type</th>
<th>Application Type</th>
<th>Throughput (TPS)</th>
<th>MVS CPU%</th>
<th>Throughput (Normalized)</th>
<th>Delta CPU Util/Tran (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Cursor 1 Block</td>
<td>CLI</td>
<td>18.806</td>
<td>14.67</td>
<td>128.19</td>
<td>12.94</td>
</tr>
<tr>
<td>Stored-Procedure</td>
<td></td>
<td>15.392</td>
<td>13.56</td>
<td>113.51</td>
<td></td>
</tr>
<tr>
<td>1 Cursor N Blocks</td>
<td>CLI</td>
<td>0.853</td>
<td>13.13</td>
<td>6.5</td>
<td>-5.97</td>
</tr>
<tr>
<td>Stored-Procedure</td>
<td></td>
<td>0.865</td>
<td>12.52</td>
<td>6.91</td>
<td></td>
</tr>
<tr>
<td>2 Cursors 1 Block</td>
<td>CLI</td>
<td>10.09</td>
<td>12.76</td>
<td>79.08</td>
<td>-0.13</td>
</tr>
<tr>
<td>Stored-Procedure</td>
<td></td>
<td>9.343</td>
<td>11.8</td>
<td>79.18</td>
<td></td>
</tr>
<tr>
<td>2 Cursors N Blocks</td>
<td>CLI</td>
<td>0.425</td>
<td>13.00</td>
<td>3.27</td>
<td>-5.48</td>
</tr>
<tr>
<td>Stored-Procedure</td>
<td></td>
<td>0.432</td>
<td>12.49</td>
<td>3.46</td>
<td></td>
</tr>
<tr>
<td>4 Cursors 1 Block</td>
<td>CLI</td>
<td>5.331</td>
<td>12.24</td>
<td>43.55</td>
<td>-9.33</td>
</tr>
<tr>
<td>Stored-Procedure</td>
<td></td>
<td>5.284</td>
<td>11.0</td>
<td>48.04</td>
<td></td>
</tr>
<tr>
<td>4 Cursors N Blocks</td>
<td>CLI</td>
<td>0.214</td>
<td>12.99</td>
<td>1.65</td>
<td>-6.49</td>
</tr>
<tr>
<td>Stored-Procedure</td>
<td></td>
<td>0.216</td>
<td>12.26</td>
<td>1.76</td>
<td></td>
</tr>
</tbody>
</table>
results of the stored procedure. This process eliminates 50% of the line flows as the client does not have to issue the commit. The actual saving depends on the network congestion and line speed in your environment.

If you take the commit on return option, and allow DB2 to commit work when the stored procedure returns control to the client, you should code the stored procedure to check for conditions in which DB2 must not commit units of work. If you issue a ROLLBACK from the stored procedure when such an event occurs, then the automatic COMMIT is prevented.

### 6.1.3.1 Measurements for Commit on Return
A multithread tool was used to run 20 OS/2 clients concurrently, each client calling stored procedures. Two test runs were made, one with commit on return and one without. Table 45 tabulates a comparison between running the stored procedures with and without the commit on return option.

<table>
<thead>
<tr>
<th>Without commit on return</th>
<th>With commit on return</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transactions Per Second</td>
<td>70.9</td>
</tr>
<tr>
<td>MVS CPU%</td>
<td>31.8</td>
</tr>
<tr>
<td>Transactions Per Second</td>
<td></td>
</tr>
<tr>
<td>Normalized to 100% CPU</td>
<td>223</td>
</tr>
</tbody>
</table>

The results show that use of commit on return gives a 37% increase in transactions per second throughput compared to no commit on return. Normalizing the results to assume a 100% CPU usage for each test run improves the normalized transactions per second throughput by 5% when commit on return is specified. Thus, using commit on return saves 5% CPU utilization because it eliminates network cost for the client-initiated commit.

### 6.1.4 Stored Procedures and MVS Workload Manager
In DB2 V4, you have a single stored procedures address space with multiple TCBs. Storage considerations restrict the number of concurrent TCBs below the 16 MB line. You have to select third generation language (3GL) compiler run-time options carefully to limit storage consumption, and are probably limited to between 40 and 50 concurrent TCBs. Applications use call attach facility (CAF) to connect to DB2, there is no two-phase commit, and no RACF support for non-SQL resources.

### 6.1.4.1 DB2 V5 Stored Procedures Environment
In DB2 V5, if you have OS/390 Release 3, you have a choice between a single DB2-established address space, or multiple WLM-established address spaces.

Table 46 on page 131 tabulates the main differences between stored procedures when they are run in a DB2-established stored procedures address space and when they are run in a WLM-established stored procedures address space.
Table 46. Use of DB2-Established and WLM-Established Address Spaces

<table>
<thead>
<tr>
<th>DB2-Established</th>
<th>WLM-Established</th>
</tr>
</thead>
<tbody>
<tr>
<td>There can be only one stored procedures address space.</td>
<td>There can be multiple stored procedures address spaces.</td>
</tr>
<tr>
<td>• Multiple TCBs.</td>
<td>• One or more TCBs.</td>
</tr>
<tr>
<td>• Failure in one stored procedure can affect other stored procedures running in the address space.</td>
<td>• You can run critical stored procedures in their own address spaces to protect them from failures in other stored procedures.</td>
</tr>
<tr>
<td>• The restriction in storage below the 16 MB line limits your 3GL compiler runtime options and makes it hard to support more than 50 concurrent TCBs.</td>
<td>• If you have a small number of TCBs in each address space, you have more flexibility in compiler run-time options because there is less storage restriction below the 16 MB line in each address space.</td>
</tr>
<tr>
<td>Incoming requests for stored procedures are handled in a first-in, first-out order.</td>
<td>You can prioritize requests.</td>
</tr>
<tr>
<td>Stored procedures run at the priority of the stored procedures address space.</td>
<td>Stored procedures inherit the MVS dispatching priority of the DB2 thread that issues the CALL statement.</td>
</tr>
<tr>
<td>You cannot customize the environment.</td>
<td>Each stored procedure address space is associated with a WLM environment. WLM environments can be associated with individual JCL procedures. This allows you to have different JCL STEPLIB concatenations for each stored procedure, as well as to define specific JCL DD statements for each stored procedure.</td>
</tr>
<tr>
<td>Connect to DB2 with CAF.</td>
<td>You link edit with RRSAF to connect to DB2.</td>
</tr>
<tr>
<td>You can access nonrelational data but that data is not included in your SQL unit of work.</td>
<td>RRSAF uses the new OS/390 Release 3 resource recovery services (RRS) to control commit processing. You can now access nonrelational data with two-phase commit processing from any resource managers that use OS/390 Transaction Management and RRS.</td>
</tr>
<tr>
<td>Stored procedures can access MVS protected resources with the authority of the stored procedures address space.</td>
<td>You can use the IBM-defined SERVER resource class in RACF to control access to MVS resources, using the RACF authority of the stored procedure caller. You implement RACF control of non-SQL resources by specifying &quot;Y&quot; in the EXTERNAL_SECURITY column of the SYSIBM.SYSPROCEDURES row for the stored procedure. The default is &quot;N.&quot;</td>
</tr>
<tr>
<td>Must run as a main program.</td>
<td>Can run as a main or a subprogram.</td>
</tr>
</tbody>
</table>

Figure 26 on page 132 illustrates the DB2 V5 stored procedure environment with both a DB2-established stored procedure address space with nine concurrent TCBs and three WLM-established stored procedure address spaces with TCB limits of between 1 and 3.
If you use WLM-established address spaces, you can exploit the following performance and operational enhancements to the stored procedures environment:

- Inherit MVS dispatching priority.
- Have multiple stored procedures address spaces.
- Run stored procedures as subprograms.

We strongly recommend that you use WLM-established address spaces for new applications and consider migration of existing stored procedures. To ease transition, DB2 V5 supports:

- A DB2-established stored procedures address space.
- Both DB2 and WLM-established address spaces.
- WLM-established stored procedure address spaces.

You choose which to enable using the stored procedures parameters panel DSNTIPX when you install or migrate to DB2 V5.

Existing stored procedures will continue to run in the DB2-established address space after migration to DB2 V5. To change a stored procedure to run in a WLM-established address space, you must first enable WLM-established address spaces on your system and then update the row for the stored procedure in SYSIBM.SYSPROCEDURES, changing the WLM_ENV column to the name of your WLM environment.

6.1.4.2 MVS Enclave Support

DB2 V4 introduced enclave service request block (SRB) support for Distributed Data Facility (DDF) threads, including the use of MVS period switching to change the priority of long-running requests. DB2 stored procedures must be processed using the MVS dispatching priority of the stored procedures address space. Incoming requests for stored procedures must be handled on a first-in first-out order regardless of the business importance of the stored procedure.
In DB2 V5, WLM-established address spaces enable MVS enclave support to be extended to include processing performed by DB2 stored procedure TCBs. The MVS dispatching priority of the stored procedure is now inherited from the priority of the MVS thread that issues the SQL CALL statement. DB2 and MVS now automatically assign individual MVS dispatching priorities to each DB2 stored procedure without requiring you to supply additional tuning parameters. Furthermore, MVS period switching can be used to change the priority of a stored procedure as it consumes CPU time.

### 6.1.4.3 Multiple Stored Procedure Address Spaces

When you use WLM to establish stored procedure address spaces, you can have multiple address spaces. As in the DB2-established address space, you control the number of concurrent procedures in each address space with the NUMTCB keyword. Since both multiple address spaces and multiple tasks are supported, the total number of TCBs for stored procedure processing is the sum of the NUMTCB keyword values for all stored procedure address spaces.

You could manually control the number of stored procedure address spaces using the MVS start command. Instead, however, control the number dynamically based on your response-time goals. WLM starts additional address spaces when the workload is high, provided that additional address spaces will result in improved response time, and terminates address spaces when demand is low to minimize MVS resource consumption. You select this method by specifying an MVS JCL procedure name in the WLM application environment definition.

The main advantages over the single address space in DB2 V4 are:

- WLM can automatically increase or decrease the number of stored procedure address spaces to help meet your performance objectives. This enables WLM to automatically respond to changes in your stored procedure workload.
- You can use WLM to dedicate address spaces to stored procedures of the highest business priority, to ensure there is always an address space available.
- You can configure an address space to run only one procedure at a time, by specifying NUMTCB=1, so that programming errors do not affect other procedures. This lets you prevent critically important stored procedures from affecting one another if failure occurs. This can be also be useful for new procedures when they first go into your production environment.
- You can group related stored procedures in the same address spaces to separate stored procedures associated with different business functions.
- The DB2 stored procedure function tends to be self tuning and needs little manual tuning intervention.

### 6.1.4.4 Coding Stored Procedures as Subprograms

In DB2 V4, all stored procedures run as main programs. You can code stored procedures as subprograms if you use Language Environment for MVS & VM Version 1 Release 7. You can get improved performance when running a subprogram because less initialization and termination processing is needed. However, you must make sure your applications close any files they open, and free any storage they explicitly allocate. You choose to use the subprogram option by specifying a
value of “S” in the PGM-TYPE column of the SYSIBM.SYSPROCEDURES row for the stored procedure.

Avoid the use of language facilities that can terminate the language-environment enclave: these include STOP or EXIT in PL/I and STOP RUN in COBOL. These statements cause the enclave to be terminated when a stored procedure ends, and rebuilt by the next subprogram to be executed, losing the benefit of subprograms. Try to group subprograms with the same run-time options in the same WLM environment because DB2 must terminate a language environment enclave whenever the run-time options value changes.

6.1.4.5 WLM Tuning and Operating Considerations

Limit the number of WLM environments and MVS service classes, as WLM creates at least one address space for every combination of environment and service class in the active workload.

Also, try to ensure that performance characteristics of all stored procedures within a given WLM environment are similar. A poorly performing procedure could monopolize one or more address spaces within that WLM environment, which can slow other procedures that are running without problems. A stored procedure may be known to occasionally have problems that affect other applications. WLM may allow too many of these to be scheduled at once, which could lead to DB2 lock conflicts, for example.

In general, assign any problem applications to a WLM environment reserved for large applications. This enables WLM to limit the number of problem applications that run concurrently without restricting the number of small applications that can run concurrently.

Starting and Stopping Procedures: MVS WLM controls the starting and stopping of WLM-established stored procedure address spaces, not −START PROCEDURE and −STOP PROCEDURE commands. WLM determines if an address space is already started when an SQL CALL is received and starts a replacement address space if an address space terminates abnormally. The WLM-established address spaces do not stop when a −STOP PROCEDURE(*) is issued.

6.1.4.6 Measurements for WLM-Established Address Spaces

A multithread tool was used to run 20 OS/2 clients concurrently, each client calling stored procedures. The four largest stored procedures each had their own WLM environment, while the remaining three stored procedures shared a WLM environment. WLM was set up to run in discretionary mode. Resource measurement facility (RMF) and DB2 accounting traces were turned on for the duration of each run. The following test cases were run:

- A base run with a DB2-established address space
- WLM-established address spaces running main program
- WLM-established address spaces running subprograms.

The three runs were repeated with commit on return specified.

Table 47 on page 135 tabulates a comparison between using DB2-established address spaces and using WLM-established address spaces.
The benefit of using WLM to run stored procedures as a subprogram is evident both with and without the commit on return option. WLM performs some extra queuing and latching operations to manage work queues. The figures for WLM-established address spaces in the test cases also include the initial overhead for starting the stored procedure address spaces.

<table>
<thead>
<tr>
<th>Test case</th>
<th>Transactions per Second</th>
<th>CPU%</th>
<th>Transactions per Second Normalized to 100% CPU (TPS/N)</th>
<th>TPS/N for WLM-Established Address Spaces Delta % Compared to Corresponding DB2-Established Address Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2</td>
<td>70.9</td>
<td>31.8</td>
<td>223</td>
<td>-</td>
</tr>
<tr>
<td>WLM main</td>
<td>63.7</td>
<td>30.2</td>
<td>211</td>
<td>−5.4</td>
</tr>
<tr>
<td>WLM sub</td>
<td>63.6</td>
<td>27.8</td>
<td>229</td>
<td>+2.7</td>
</tr>
<tr>
<td>DB2 COR</td>
<td>97.0</td>
<td>41.3</td>
<td>235</td>
<td>-</td>
</tr>
<tr>
<td>WLM COR main</td>
<td>83.5</td>
<td>37.8</td>
<td>221</td>
<td>−5.9</td>
</tr>
<tr>
<td>WLM COR sub</td>
<td>84.7</td>
<td>34.1</td>
<td>249</td>
<td>+5.9</td>
</tr>
</tbody>
</table>

Note:
- DB2 represents DB2-established address space
- WLM represents WLM-established address space
- main represents main program
- sub represents subprogram
- COR represents commit on return
- TPS/N represents transactions per second normalized for 100% CPU usage

The benefit of using WLM to run stored procedures as a subprogram is evident both with and without the commit on return option. WLM performs some extra queuing and latching operations to manage work queues. The figures for WLM-established address spaces in the test cases also include the initial overhead for starting the stored procedure address spaces.

### 6.1.5 Considerations for Stored Procedures

We recommend using multiple result sets to reduce network traffic in cases where a small amount of data is retrieved from each of several cursors.

The commit on return option gives useful performance savings from reduced network traffic. There are minor coding considerations, so we recommend considering commit on return for all new applications if you wish to minimize network performance overheads.

We recommend the use of WLM to establish stored procedure address spaces for the flexibility of multiple address spaces and usability options, as well as to position for the future. The enclave support for stored procedures should provide similar tuning opportunities to those given to DDF in DB2 V4. It is best to introduce multiple address spaces before the operational limitations of a single address space become a challenge. In addition, you can take advantage of subprograms to gain performance benefits, provided you make sure your applications close any files they open, and free any storage they explicitly allocate.

If you have not yet used stored procedures, consider using only WLM-established address spaces. If you currently use stored procedures, consider how soon your current operation needs the operational benefit of WLM-established procedures and aim to phase out the DB2-established address space before then.
6.2 Native TCP/IP Support

DB2 V5 provides native TCP/IP support that removes the requirement for an intermediary product, such as AnyNet, for DRDA connections to TCP/IP networks. As a result, DRDA clients on many platforms, including OS/2, AIX, Sun Solaris, HP/UX, Windows 3.11, Windows 95, and Windows NT connect to DB2 V5 without using an SNA gateway machine.

No SQL changes are associated with using native TCP/IP support. Therefore, you can choose Advanced Program-to-Program Communication (APPC) or TCP/IP, as your communication protocol is transparent to your SQL application programmers.

The DDF does not support the TCP/IP protocol for DB2 private protocol connections, TCP/IP is supported only in DRDA connections.

Existing SNA connections to Windows NT, OS/2, AIX, and other Unix systems are supported, so DB2 can work in an SNA network, a TCP/IP network, or in a network with both TCP/IP and SNA.

OS/390 Release 3 is a prerequisite for DB2 V5 to run native TCP/IP.

In this section, we explain what can influence the performance of your DB2 subsystems which connect to TCP/IP by describing the installation aspects of the TCP/IP support in 6.2.1, “Installing TCP/IP Support for DB2 V5.” We examine what level of performance you can expect from the TCP/IP support in 6.2.2, “TCP/IP Support Performance” on page 140.

6.2.1 Installing TCP/IP Support for DB2 V5

DB2 V5 installation introduces changes to prepare your subsystem for DRDA TCP/IP connections. This section addresses the important parameter settings that influence your workload throughput on TCP/IP connections.

6.2.1.1 Installation Dialog Parameters

The installation dialog has two panels that specifically address DDF and you must customize APPC support and TCP/IP before you can use the TCP/IP support. TCP/IP support requires some APPC parameters even if you plan not to use any APPC connections. The installation panels DSNTIPR and DSNTIP5 are for DDF customization.

Figure 27 on page 137 shows the DSNTIPR installation panel.
Enter data below:

1 DDF STARTUP OPTION ===> AUTO NO, AUTO, or COMMAND
2 DB2 LOCATION NAME ===> DB1G The name other DB2s use to refer to this DB2
3 DB2 NETWORK LUNAME ===>
4 DB2 NETWORK PASSWORD ===> Password for DB2's VTAM application
5 RLST ACCESS ERROR ===> NOLIMIT NOLIMIT, NORUN, or 1-5000000
6 RESYNC INTERVAL ===> 2 Minutes between resynchronization period
7 DDF THREADS ===> ACTIVE Status of a database access thread that commits or rolls back and holds no database locks or cursors
8 DB2 GENERIC LUNAME ===>
9 IDLE THREAD TIMEOUT ===> 2 180 0 or seconds until dormant server ACTIVE thread will be terminated (0-9999)
10 EXTENDED SECURITY ===> 3 YES Allow change password and descriptive security error codes. YES or NO.

PRESS: ENTER to continue RETURN to exit HELP for more information

Figure 27. The DSNTIPR Installation Panel

You must specify the DB2 Location Name and the DB2 Network Luname parameters. 1 The logical unit (LU) name is used to identify the DB2 subsystem to VTAM, and also to uniquely identify logical units of work. Even if you are planning to use only TCP/IP to communicate with remote sites, you still have to configure the LU name parameter and VTAM, because DB2 uses the network ID and the LU name to identify units of work.

2 Consider specifying a nonzero value for the idle-thread timeout parameter, to prevent TCP/IP network outages from decreasing DB2 data availability. This parameter defines the number of seconds that an active server thread is allowed to remain idle, and holding locks, before it is cancelled. Inactive and in-doubt threads are not subject to timeout. The idle-thread timeout parameter is not specific to TCP/IP. When you specify a value for this parameter, any DDF thread, using APPC or TCP/IP, is subject to being cancelled. The default value for this parameter is zero, which disables the timeout processing.

3 Extended security is a new parameter for DB2 V5 which controls the new security options, regardless of the communication protocol being used (TCP/IP or APPC). You specify YES to enable DRDA to change RACF password function. If you specify NO, RACF users are unable to change their passwords through DRDA, and generic error codes are sent to DDF clients when security errors occur.

Figure 28 on page 138 shows the DSNTIP5 installation panel that relates only to TCP/IP.
1. DRDA PORT parameter specifies the TCP/IP port number used for accepting TCP/IP requests from remote DRDA clients. To use TCP/IP as an application server (AS) or application requester (AR), specify the DRDA port 446, or a server port assigned to DB2. In a non-data-sharing environment, each DB2 subsystem in the same MVS system must have a unique DRDA port number. In a data sharing environment, all members of the data sharing group share the same DRDA port number.

2. RESYNC PORT parameter specifies the TCP/IP port number used to process requests for two-phase commit resynchronization. You must specify a server port number different from the one used for the DRDA Port parameter. Each DB2 subsystem in the same MVS system must have a unique resynchronization port. In a data sharing environment, each member of the data sharing group must also have a unique resynchronization port.

3. The TCP/IP protocol does not provide the same level of security checking as the APPC protocol for client authentication. The TCP/IP already verified parameter simulates the support provided by the VTAM. However, the support differs, as VTAM allows you to select already-verified client by client whereas TCP/IP support is networkwide and not client by client.

When using APPC, VTAM decides if a connection without a password will be accepted. When using TCP/IP, client authentication is done by the DB2 DRDA protocol during the conversation.

You do not specify an IP address (or host name) during DB2 installation. DB2 obtains this information dynamically from TCP/IP, so you cannot enter this value during installation.

6.2.1.2 DDF and MVS OpenEdition

DDF uses the OpenEdition assembler callable interface to perform TCP/IP services and some of these functions are only available in OS/390 Release 3. DDF must execute in the OpenEdition environment as a super user. For more information on how to define DB2 resources to RACF, please refer to DB2 for OS/390 Version 5 Administration Guide Volume 1.
6.2.1.3 Register DDF Port Numbers

DB2 native TCP/IP support works, even if you do not register the port numbers in the TCP/IP PROFILE data set. However, we recommend that you register the DB2 port numbers to prevent other non-OpenEdition applications from using the DB2 port numbers by mistake.

6.2.1.4 Communications Database Changes

If you plan to use DB2 only as a server, you do not need to populate the communications database (CDB). If you intend to request data, you must specify port numbers or service names in SYSIBM.LOCATIONS and add entries to the SYSIBM.IPNAMES table. You do not need a row for the local DB2 in the IPNAMES or LOCATION tables, as DB2 obtains this information dynamically from TCP/IP.

6.2.1.5 Defining TCP/IP Host Names

DB2 V5 uses TCP/IP host names both as a server and as a requester. As an AS, DB2 needs the local host name to be defined before DDF is started. As an AR, if you use host names in the column IPADDR of the CDB table SYSIBM.IPNAMES, all the host names referenced must also be defined.

To support a Sysplex environment or multiple network interfaces on a single host, we recommend that the domain name server (DNS) or the HOSTS.LOCAL data set be used to associate multiple IP address with a single host name.

6.2.1.6 TCP/IP Additional Considerations

TCP/IP communication failures can cause DDF threads to hang up by holding resources for a long time. Configuring the DDF idle-thread timeout parameter and the TCP/IP keep-alive timer, you can protect the DB2 subsystem from TCP/IP outages. The keep-alive timer can be used to notify DB2 when a network failure occurs. The keep-alive timer mechanism periodically sends a packet over a possibly idle connection and after TCP/IP receives the message and the keep-alive timer is active, TCP/IP starts accumulating the time. If the timer reaches a specified value, TCP/IP sends a packet to the remote TCP/IP. If the remote TCP/IP does not respond to the packet, the connection is ended and DB2 is notified. If any message arrives before the timer reaches the specified value, the timer is reset.

You can specify the interval TCP/IP waits to transmit the keep-alive timer packet. The default interval is 2 hours. This default implies that a DDF thread can hang up for up to 2 hours until DB2 is notified of a failure. The hung-up thread can cause many operational problems for the DB2 server, especially if the thread holds locks on critical DB2 resources.

To change the keep-alive timer interval, you have to edit the TCP/IP PROFILE data set, and include the KEEPALIVEOPTIONS parameter. We recommend that you specify a value of 5 minutes or less for the TCP/IP keep-alive timer. Note that the activation of the keep-alive timer is valid for all TCP/IP connections, not only for DB2 connections.
6.2.2 TCP/IP Support Performance

The first implementation of a TCP/IP DRDA support in the OS/390 platform is a functional feature. The preferred connection with the best performance is an SNA connection; however, we expect the performance of the TCP/IP solution to improve. DB2 V5 requires OS/390 release 3 API enhancements for TCP/IP support, which positions DB2 V5 for future TCP/IP enhancements. As newer levels of TCP/IP for OS/390 become available, we expect the performance of the TCP/IP connection to be comparable to the APPC connection level. The results we present in Table 48 show a sample measurement for a specific environment.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Elapsed Time (Seconds)</th>
<th>AIX CPU (%) (User / System)</th>
<th>MVS CPU (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP/IP</td>
<td>511.065</td>
<td>31.9 / 3.2</td>
<td>9.34</td>
</tr>
<tr>
<td>SNA (4 KB RU)</td>
<td>510.395</td>
<td>31.7 / 8.1</td>
<td>7.92</td>
</tr>
<tr>
<td>SNA (8 KB RU)</td>
<td>509.689</td>
<td>31.6 / 8.0</td>
<td>7.91</td>
</tr>
<tr>
<td>AnyNet (4 KB RU)</td>
<td>539.022</td>
<td>29.2 / 10.2</td>
<td>14.75</td>
</tr>
<tr>
<td>AnyNet (8 KB RU)</td>
<td>528.359</td>
<td>29.8 / 8.1</td>
<td>11.91</td>
</tr>
</tbody>
</table>

Table 48 shows the results from the following environment:
- A single thread query run under a DB2 batch process on an AIX system
- The query is "SELECT * FROM TABLE"
- Table has 240 000 rows of 691 bytes long (165 840 000 bytes)
- RMF and DB2 statistics trace are turned on for a 6-minute interval on OS/390
- The iostat interval on the AIX system is 6 minutes
- The TCP/IP connection is a Token Ring with a 4 KB response unit (RU) size. However, TCP/IP sends data in 1024 KB packets.

The results show the total elapsed time and the percentage of the CPU used during the 6-minute interval. The AIX CPU shows the measurements for the DB2 batch process user thread and the part that is measured for the system.

The value of this specific measurement is to show that the TCP/IP DRDA support offers improved performance over that of a configuration using AnyNet. However, the SNA connection attains the best performance.

6.3 DRDA Application Performance Enhancements

This section explains two new application performance enhancements.

6.3.1 Improved Dynamic SQL Processing

There are two possible ways from an application program to invoke a DB2 SELECT statement dynamically:

```
DECLARE C1 CURSOR FOR S1;
PREPARE S1 FROM :query;
DESCRIBE S1 INTO :SQLDA;
OPEN C1;
FETCH C1;
```
The two techniques differ only in how the PREPARE and DESCRIBE operations are performed when DEFER(PREPARE) is specified at bind time.

Figure 29 shows the DRDA flows for dynamic SQL SELECT statements.

![Diagram showing DRDA flows](image)

Figure 29. PREPARE without DEFER(PREPARE) Bind Option in DB2 V4

Figure 29 shows that in previous DB2 releases, a dynamic SQL query sent to a DRDA server results in two message exchanges, one for the PREPARE and DESCRIBE flow, and one for OPEN and the returned answer set.

**Note:** With DEFER(PREPARE) bind option, DB2 defers sending the PREPARE until the DESCRIBE statement is encountered, the message exchange is the same.

With combined PREPARE and DESCRIBE (PREPARE S1 INTO xxx FROM yyy), DB2 ignores any DEFER(PREPARE) and the message flow is as shown in Figure 29.

DB2 V5 optimizes the process for SQL PREPARE when the following conditions are satisfied:

- The connection to the server is a DRDA connection.
- The SQL statement being PREPARED is a SELECT statement.
- The SQL statement contains no parameter markers.

When these conditions are satisfied, DB2 V5 DDF automatically chains an SQL OPEN request onto the PREPARE and DESCRIBE flow. DB2 anticipates that the application will open the cursor, and the reply message from the DRDA server includes the output from the PREPARE, DESCRIBE, and OPEN operations. The SQL OPEN reply is retained in DDF storage until the application program issues...
the SQL OPEN that DB2 anticipated. At that time DDF processes the OPEN reply message and returns the answer set to the application program.

If anything other than a SQL OPEN occurs in the application program, the following action is taken, depending on what occurs:

- A new SQL PREPARE.
  - DDF chains a SQL CLOSE in front of the new PREPARE.
- The application program aborts.
  - DDF discards the OPEN reply message. The server closes the cursor during abort processing.
- The application commits.
  - With the WITH HOLD attribute, the SQL OPEN reply is retained in case of a new SQL OPEN. Without WITH HOLD, the SQL OPEN reply is discarded. The server closes the cursor at commit.

Figure 30 shows the DRDA flows for dynamic SQL SELECT statements in DB2 V5.

Figure 30 shows that in DB2 V5, a dynamic SQL query sent to a DRDA server results in only one message exchange.

**Note:** With combined PREPARE and DESCRIBE (PREPARE S1 INTO xxx FROM yyy), the message flow is as shown in Figure 30.

This new approach eliminates a VTAM send and receive operation during the SQL OPEN processing, providing the following advantages:

- The elapsed time is reduced by eliminating VTAM network delay.
- CPU time is reduced in both the requester DB2 and in the server DB2 address spaces.
6.3.2 Optimize for n Rows

In DB2 releases prior to DB2 V5, when a DB2 client uses DRDA to issue a read-only SQL query, DB2 returns enough data to fill the DRDA query block size, which can be up to 32 KB in size.

The 32 KB of query data is very often more than the application requires. For example, the application fetches only enough rows to fit the end user’s screen. If the user never looks at the next screen of data, DB2 has processed a lot of rows filling up the DRDA query block size that are never used. This can result in excessive server CPU cost and increased elapsed time.

In DB2 V5, the OPTIMIZE FOR “n” ROWS clause on the SQL SELECT statement can be used by DRDA clients to influence the DRDA network blocking used to return query answer sets. This allows the application to limit the number of data rows returned on each DRDA network transmission, so that the DB2 server does not fetch additional rows unnecessarily when the application intends to fetch only a subset of the query answer set.

The number of rows DB2 returns on each DRDA network transmission depends on the value of “n”:

- When OPTIMIZE FOR 1 ROW is specified, DB2 transmits the lesser of 16 rows and the number of rows that fit within the DRDA query block size on every network transmission.

  DB2 overrides the request for one row. This approach is being taken because OPTIMIZE FOR 1 ROW is often used to force the DB2 optimizer to select a specific access path, typically an access path where additional sorts are avoided.

  Since the CPU cost of sending one row per DRDA network transmission is very high, DB2 assumes that OPTIMIZE FOR 1 ROW is used to force a particular access path, rather than to force a network block size of one row.

- For values of “n” greater than 1, DB2 transmits the lesser of “n” rows and the number of rows that fit within the DRDA query block size on each network transmission.

6.3.3 Improved VTAM SEND/RECEIVE Processing

VTAM V4 R4 introduces two enhancements that DB2 V5 exploits to reduce the cost of VTAM send/receive operations. The APPCCMD macro provides an option, SENDRCV, to allow a VTAM SEND and a VTAM RECEIVE to be managed by a single VTAM macro invocation and an option that reduces the number of times application data is moved when a VTAM SEND or SENDRCV option is used to transmit data.

Only DRDA support uses the SENDRCV option and DB2 uses this function for both requester and server processing. The DRDA partner does not have to support the SENDRCV option in order for DB2 to use the function.

The option to reduce the number of data moves is to optimize the transmission of large messages. As DB2 DRDA requesters typically send short messages this function is available only in the DB2 DRDA server support. The amount of CPU reduction depends on the VTAM RU size and the size of the DB2 reply message.

If VTAM V4 R4 is not available then DDF send and receive processing is as in previous DB2 releases.
6.4 Application Portability

This section discusses how the DB2 V5 CLI and ASCII server support application portability.

6.4.1 Call Level Interface Support

DB2 V5 makes a significant impact on application portability with support for CLI programs.

6.4.1.1 What is CLI

DB2 Call Level Interface is IBM’s callable SQL interface to the DB2 family of database servers. It is a C and C++ application programming interface for relational database access, and it uses function calls to pass dynamic SQL statements as function arguments. It is an alternative to embedded dynamic SQL, but unlike embedded SQL, it does not require a precompile or bind process.

DB2 CLI is based on standard interfaces: ODBC and X/Open. In addition, DB2 CLI adds specific extensions to help exploiting DB2 features.

6.4.1.2 CLI Implementation

The DB2 V5 CLI support is implemented as a CLI/ODBC Driver and Driver Manager to be loaded at run time into the application address space.

The DB2 V5 CLI Driver can use either CAF or RRSAF that uses the OS/390 Resource Recovery Services (OS/390 RRS) to connect to DB2 V5. The DB2 CLI application can run as a DB2 V5 stored procedure, and as such use the attachment facility specified for stored procedures. If not run as a stored procedure, the MVSATTACHTYPE key word in the initialization file for DB2 V5 CLI determines which attachment facility to use, RRSAF or CAF.

DB2 V5 CLI applications must execute on a system on which DB2 V5 and CLI are installed, there is no support for access from a client-only machine.

Figure 31 on page 145 shows the relationship between the DB2 V5 CLI application, the DB2 V5 CLI/ODBC Driver, and the DB2 V5 subsystem.
Figure 31: Relationships between DB2 V5 CLI Components

As Figure 31 shows, DB2 V5 CLI supports access to the local DB2 V5 subsystems and any remote data source that can be accessed using DB2 V5. This includes remote DB2 subsystems using specification of alias and three-part-name, as well as remote DRDA-1 and DRDA-2 servers via LU 6.2 or TCP/IP.

You can specify a set of optional key words in the DB2 V5 CLI initialization file (DSNAOINI). This is an EBCDIC file that stores default values for various DB2 V5 CLI configuration options. You can use the initialization file to help improve the performance or usability of an application. For example, the initialization file can contain a TRACE=1 statement which externalizes a DB2 V5 CLI diagnostic trace in the trace file (DSNAOTRC), as well as an OPTIMIZEFORNROWS=2 statement to reduce the size of the answer set returned from the data source.

6.4.1.3 Advantages
DB2 V5 CLI has several key features that offer advantages over embedded SQL:

- CLI is suited for the client/server environment in which the data source is not known at the time the application is built. It provides a consistent interface for executing SQL statements, regardless of which database server the application connects to.

- CLI increases the portability of the application by removing the dependence on precompilers. Applications are distributed not as source programs with embedded SQL, which must be preprocessed by each database product, but as compiled modules in run-time libraries. The applications connect to the appropriate data source at run time.
• Individual DB2 CLI applications do not need to be bound to each new data source. Only those bind files shipped with the DB2 V5 CLI need to be bound (once) to serve all applications.

• Applications can connect to multiple data sources from the same application.

• DB2 V5 CLI eliminates the need for applications to control global data areas as SQLDA and SQLCA. DB2 V5 CLI allocates and controls the necessary data structures, and provides a handle for the application to reference them.

• CLI provides enhanced parameter input and fetching capabilities. Arrays of data can be specified on input, retrieving rows of a result set directly into that array.

• Multiple result sets can be retrieved from a call to a stored procedure.

• CLI shields the application from catalog table differences among different database servers as well as from changes across releases of database servers. Thus, DB2 V5 CLI provides a consistent interface to query catalog information.

• CLI provides extended data conversion, which requires less application code when converting information between various SQL and C data types.

• CLI incorporates both the ODBC and X/Open CLI standards, as well as conforming to the emerging ISO CLI standard.

• Application developers can apply their knowledge of industry standards directly to DB2 V5 CLI without having to know about product-specific methods of embedding SQL statements into a host language.

6.4.1.4 Selecting the Interface
Which interface to use depends on the application. DB2 V5 CLI is ideally suited for query-based applications that require portability.

The advantages listed in 6.4.1.3, “Advantages” on page 145 may make DB2 V5 CLI seem like the obvious choice for any application. However, several factors must be considered before the interface is chosen. These are among the most important:

• Performance
  – Dynamic SQL, including CLI, is prepared at run time. Static SQL is prepared at precompile time. The preparation step for dynamic SQL, except when dynamic statement caching is used in DB2 V5, requires more processing and may add network traffic. (See 3.1, “Caching Dynamic SQL Statements” on page 17 for details on dynamic statement caching.)

  It is important to notice, though, that static SQL does not always have better performance than dynamic SQL. Dynamic SQL can make use of changes, such as new indexes and updated catalog statistics, to select better access paths.

• Encapsulation and Security
  – In static SQL, authorizations for objects are associated with packages and validated at bind time. By granting execute authority on a package to a set of users, thus encapsulating their privileges in the package, you avoid the need to grant explicit access to each object.
In dynamic SQL, the authorizations are validated at run time, and on a per statement basis. Users must be granted explicit access to each object.

DB2 V5 CLI using dynamic SQL can, however, gain many of static SQL’s security capabilities:

1. Bind DB2 V5 CLI packages with DYNAMICRULES(BIND) option using a driver’s authorization ID with table privileges.
2. GRANT EXECUTE authorization on each collection.package or plan to individual users.
3. The user selects which plan or package to use with the help of the PLANNAME or COLLID key words in the private DB2 V5 CLI initialization file (DSNAOINI).
4. When issuing dynamic SQL, the statements are processed with the driver’s authorization ID, not the user’s. Users need execute authority for the package, not table privileges.

If an application requires the advantage of both static and dynamic interfaces, it is possible to make use of static SQL within a DB2 V5 CLI application by creating a stored procedure that contains static SQL. The stored procedure is called from within the DB2 V5 CLI application and is executed on the server.

### 6.4.2 ASCII Server Support

DB2 V5 introduces support for ASCII character data encoding. Many client/server configurations now have DB2 for OS/390 together with clients and servers on different platforms. System/390 systems use the EBCDIC encoding scheme and on other systems the character strings are represented using the ASCII encoding scheme. Two of the differences between EBCDIC and ASCII are the characters used for padding and the collating sequence. These differences can lead to differences in application results, depending on which platform the application uses to process the data.

To deal with these differences you can use field procedures (FIELDPROC) in previous releases of DB2. However, the CPU time required to execute these procedures can be high. The ASCII server support provides a more efficient ASCII data conversion and collation, eliminates most conversions (all for same CCSID), and collates the proper ASCII CCSID sequence. Also, storing tables in the ASCII format eliminates any need for conversion. For example, if your client/server configuration consists mainly of workstations, you can store DB2 for OS/390 tables in the same scheme as that of the workstation clients and avoid any need for conversion when transferring data between the clients and DB2 for OS/390. The choice of which coded character set identifier (CCSID) to use depends on the amount of activity in your client/server environment and whether you benefit from storing the data in DB2 for OS/390 with the same CCSID as the CCSID at the workstations. If you have several different client and server CCSIDs in your configuration, then consider selecting the most common CCSID to minimize conversion.
6.4.2.1 ASCII Support Changes
The ASCII support introduces many changes throughout DB2 V5. We briefly
describe where changes are made for ASCII server support:

- New installation parameters
  You specify whether to use EBCDIC or ASCII encoding scheme and which
coded character set to use. It is not necessary to specify a valid ASCII
CCSID during installation (default is zero), but before creating a database,
table space, or table, a valid ASCII CCSID must be set for the CCSID ASCII
option.

- Syntax changes to DDL
  The CREATE DATABASE, CREATE TABLESPACE, CREATE TABLE, and
CREATE GLOBAL TEMPORARY TABLE all have a CCSID option. You specify
either ASCII or EBCDIC. Although an encoding scheme can be specified on
these DDL statements, all data stored in a table must use the same encoding
scheme. There is no easy way to change the ASCII/EBCDIC attribute of a
database, table space, or table once it has been defined.

- Special Registers
  When a special register involving an ASCII table is referenced in an SQL
statement, the ASCII value of the special register will be used. An EBCDIC
table will use the EBCDIC value of the special register.

- Indexes
  An index has the same encoding scheme as its associated table (ASCII or
EBCDIC). For an ASCII table, all the indexes are stored in ASCII order.
Indexes must be Type 2.

- Data manipulation
  You can refer to an ASCII table in SQL data manipulation language (DML)
statements as long as all referenced tables are ASCII. However, a given
DML statement may not refer to both ASCII and EBCDIC tables in the same
statement. When running local MVS applications that expect to receive data
in EBCDIC, the data for a SELECT statement on an ASCII table returns the
data to the application in EBCDIC but in ASCII order.

- DESCRIBE and PREPARE INTO statements
  There is no general way for an application to determine the encoding
scheme of the data from the CCSID in the SQLDATA field of the SQLDA.
However, if the SQLDATA field contains an ASCII CCSID then the associated
data is encoded in ASCII.

- Datetime data
  As datetime strings use exit routines, ASCII support has separate exit
routines to format such data.

- Host variables
  To use a host variable that is encoded in ASCII, specify a USING clause with
an SQLDA in your SQL statement.

- LIKE predicate
  The LIKE predicate remains a Stage 1 predicate and can be indexed.

- Built-in scalar functions
All built-in scalar functions, except for VARGRAPHIC, are supported for ASCII data.

- Stored procedures

For stored procedures on the MVS subsystem, the data in the parameter list passed from DB2 for OS/390 and the stored procedure is in EBCDIC. So if the parameters are sent from an ASCII client up to the DB2 for OS/390 server, DB2 converts all non-EBCDIC data to EBCDIC for internal processing of the CALL statement. The answer set is blocked and sent to the client together with the CCSID of the data.

- Field, Edit and Validation procedures

These procedures must be written to match the encoding scheme of the data they process.

6.4.2.2 Implementation Considerations

Several implementation aspects introduced by the ASCII server support need consideration, as follows: The following list describes the considerations:

- Code page conversion

DB2 V4 supports code page conversion for some ASCII CCSIDs to EBCDIC CCSIDs and from some EBCDIC CCSIDs to other EBCDIC CCSIDs. In DB2 V5, the support enhances the code page conversion by adding support for conversion from some EBCDIC CCSIDs to ASCII CCSIDs and for some ASCII CCSIDs to other ASCII CCSIDs. However, if a code page conversion requests a combination of unsupported CCSIDs, then SQLCODE -332 is issued.

- Distributed data

At connect time, the requester and server systems exchange default CCSIDs (single byte character set or SBCS, mixed character set, and double byte character set or DBCS). When DB2 for OS/390 is a requester, the default CCSIDs are the EBCDIC default CCSIDs.

- Data sharing

We strongly recommend that all members of a data sharing group observe the following step when implementing ASCII support. When one member converts to DB2 V5 to provide ASCII server support, convert the remaining members as soon as possible. If the CCSIDs are changed on one member, make similar changes to the other members.

If you do not follow these recommendations, the results of SQL may be unpredictable.

- Utilities

All utilities operate on table spaces containing ASCII tables in the same way as they do for EBCDIC tables. All utility control statements and messages are in EBCDIC. However, there are some specific considerations for the LOAD and REPAIR utilities.

The LOAD utility input data may be in either ASCII or EBCDIC. You must specify ASCII or EBCDIC, as there is no way to determine if a specific CCSID is EBCDIC or ASCII. If the CCSID of the input data does not match the CCSID of the table space, then the input fields are converted to the CCSID of the table being loaded.
The REPAIR utility with DUMP option prints ASCII pages and also prints, side by side, the EBCDIC representation for usability purposes. All the support changes and considerations can influence the performance of the ASCII server support. However, the elimination of CPU-intensive field procedures provides a significant improvement in performance for clients using matching ASCII CCSIDs.

6.4.2.3 Performance
You save CPU time on clients by not having to convert retrieved data.

You save CPU time on host by avoiding host variable conversions and column field procedures. Field procedures used for collation or conversion of data can degrade performance by as much as 30% for some transactions.

The following set of results (Table 49 on page 151 and Figure 32 on page 151) are from runs to compare using ASCII CCSIDs, EBCDIC data, and EBCDIC with field procedures to convert to ASCII data. The environment for the measurements run was as follows:

- Pentium OS/2 client with ASCII CCSID (437)
- Single-user DDCS client
- Token ring connection
- 9672-R61 CMOS processor with
  - DB2 for OS/390 Version 5
  - OS/390 Release 3
  - VTAM Version 4 Release 4
- The same SQL was run on each of the following three tables to compare the overhead involved with data and host variable conversion and the field procedures.
  - Table with EBCDIC data CCSID(37) with field procedures on all columns to convert data to ASCII CCSID(437) thus ensuring the proper collating sequence. This is designated by (A) in Table 49 on page 151, Table 50 on page 152, and Table 51 on page 153.
  - Table with EBCDIC data CCSID(37). This is designated by (B) in Table 49 on page 151, Table 50 on page 152, and Table 51 on page 153.
  - Table with ASCII data CCSID(437). This is designated by (C) in Table 49 on page 151, Table 50 on page 152, and Table 51 on page 153.

Table 49 on page 151 shows the end to end elapsed times as measured from the client.
Table 49. ASCII Client: End to End Elapsed Times

<table>
<thead>
<tr>
<th>SQL Operation</th>
<th>ASCII to FIELDPROC (A)</th>
<th>ASCII to EBCDIC (B)</th>
<th>ASCII to ASCII (C)</th>
<th>Delta % (C) to (A)</th>
<th>Delta % (C) to (B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fetch 16384 rows (42 columns)</td>
<td>40.19</td>
<td>25.80</td>
<td>23.80</td>
<td>-40.78</td>
<td>-7.75</td>
</tr>
<tr>
<td>Insert 2048 rows</td>
<td>43.59</td>
<td>42.90</td>
<td>41.80</td>
<td>-4.10</td>
<td>-2.56</td>
</tr>
<tr>
<td>Update cursor 2048 rows</td>
<td>78.56</td>
<td>77.50</td>
<td>75.60</td>
<td>-3.76</td>
<td>-2.45</td>
</tr>
<tr>
<td>Delete 2048 rows</td>
<td>7.69</td>
<td>7.10</td>
<td>7.10</td>
<td>-7.67</td>
<td>0</td>
</tr>
<tr>
<td>Total Run</td>
<td>170.03</td>
<td>153.30</td>
<td>148.30</td>
<td>-9.83</td>
<td>-3.26</td>
</tr>
</tbody>
</table>

Note: All times are in seconds.

Figure 32 shows in a graphical form the results tabulated in Table 49.

End to End Elapsed Time

Figure 32. End to End Elapsed Times. This is comparing ASCII server with EBCDIC server and FIELDPROC.

End to end elapsed time as measured from the client for the ASCII server compared to FIELDPROC server shows:

- 10 to 25% improvement on fetches
- 5 to 10% improvement on inserts, updates, and deletes

End to end elapsed time as measured from the client for the ASCII server compared to EBCDIC server shows:

- 5 to 10% improvement on fetches
- 1 to 4% improvement on inserts, updates, and deletes
The savings in CPU time at the server system and the amount of conversion required on the data returned to the client, account for the improvements in elapsed time.

Table 50 shows the native class 2 CPU times as measured from the client.

<table>
<thead>
<tr>
<th>SQL Operation</th>
<th>ASCII to FIELDPROC (A)</th>
<th>ASCII to EBCDIC (B)</th>
<th>ASCII to ASCII (C)</th>
<th>Delta % (C) to (A)</th>
<th>Delta % (C) to (B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fetch 16384 rows (42 columns)</td>
<td>22.429</td>
<td>12.221</td>
<td>12.167</td>
<td>-45.74</td>
<td>-0.44</td>
</tr>
<tr>
<td>Insert 2048 rows</td>
<td>6.984</td>
<td>5.551</td>
<td>4.601</td>
<td>-34.12</td>
<td>-17.11</td>
</tr>
<tr>
<td>Update cursor 2048 rows</td>
<td>8.683</td>
<td>5.888</td>
<td>4.909</td>
<td>-43.47</td>
<td>-16.63</td>
</tr>
<tr>
<td>Delete 2048 rows</td>
<td>6.119</td>
<td>3.091</td>
<td>3.093</td>
<td>-49.45</td>
<td>-0.71</td>
</tr>
<tr>
<td>Total Run</td>
<td>44.215</td>
<td>26.751</td>
<td>24.770</td>
<td>-43.97</td>
<td>-7.41</td>
</tr>
</tbody>
</table>

**Note:** All times are in seconds.

Figure 33 shows in a graphical form the results tabulated in Table 50.

Native CPU times as measured from the client for the ASCII server compared to FIELDPROC server shows:

- 20 to 45% improvement on fetches
- 20 to 50% improvement on inserts, updates and deletes.
Native CPU times as measured from the client for the ASCII server compared to EBCDIC server shows equivalent performance.

Table 51 shows the client CPU usage.

<table>
<thead>
<tr>
<th>Fetch</th>
<th>ASCII to FIELDPROC (A)</th>
<th>ASCII to EBCDIC (B)</th>
<th>ASCII to ASCII (C)</th>
<th>Delta % (C) to (A)</th>
<th>Delta % (C) to (B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16384 Rows</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>42 Columns/Row</td>
<td>41</td>
<td>30</td>
<td>31</td>
<td>−24.39</td>
<td>3.33</td>
</tr>
<tr>
<td>Elapsed time on client (seconds)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average % CPU occupancy</td>
<td>69.34</td>
<td>92.87</td>
<td>72.54</td>
<td>4.61</td>
<td>−21.89</td>
</tr>
<tr>
<td>Normalized to 30 seconds</td>
<td>94.76</td>
<td>92.87</td>
<td>74.96</td>
<td>−20.90</td>
<td>−19.29</td>
</tr>
</tbody>
</table>

Figure 34 shows in a graphical form the results tabulated in Table 51.

![Client CPU Occupancy](image)

Figure 34. Client CPU Usage. This is comparing ASCII server with EBCDIC server and FIELDPROC.

You see that overall the client CPU usage is reduced by 20% when ASCII server is used.
Chapter 7. Data Sharing Enhancements

DB2 V5 provides enhancements to functions in the data sharing environment. The coupling facility in the Parallel Sysplex stores structures for DB2. In DB2 V4, the GBP structures must be built manually, but the use of automation helps to simplify the rebuild process. DB2 V5 allows you to either manually rebuild the structure for planned coupling facility or GBP reconfiguration or dynamically rebuild the structure when a loss of connectivity occurs. We examine the implications of rebuild in 7.1, “Group Buffer Pool Rebuild.”

In DB2 V4, when a GBP fails, manual intervention is required to recover lost data. We describe the ability for DB2 V5 to recover GBPs automatically in 7.2, “Group Buffer Pool Automatic Recovery” on page 161.

In 7.3, “Selective Locking of Partitions in Partitioned Table Spaces” on page 164, we examine the processing of locks for partitioned table spaces.

DB2 V5 introduces several improvements to reduce the cost of data sharing. We assess the value of these enhancements in 7.4, “Reducing Data Sharing Costs” on page 165.

7.1 Group Buffer Pool Rebuild

DB2 V4 supports the rebuild function for the SCA; the IRLM supports the rebuild function for the lock structure in the coupling facility. DB2 V5 supports rebuilding the GBP structure. In this section, we describe the rebuild function and examine the performance implications of GBP rebuilding in “Group Buffer Pool Rebuild Performance” on page 157.

To use the GBP rebuild function, you must understand which parameter settings influence when the rebuild will occur. The CF provides the intersystem caching and locking services that DB2 uses to assure that data accessed throughout a Sysplex will not be corrupted and that the data will be consistent among all sharing users. The CF provides the medium in which CF structures are rebuilt should a failure occur. The rebuild process can be initiated either by command or automatically by policy definitions when a loss of connectivity occurs from one or more MVS CPCs to a coupling facility.

MVS determines the actions to take in the case of failure of a coupling facility or coupling facility link. The policy definitions in the Coupling Facility Resource Manager (CFRM) and the Sysplex failure management (SFM) coupled data sets specify the actions. Figure 35 on page 156 shows a scenario for a dynamic rebuild of a GBP structure.
Figure 35. Group Buffer Pool Rebuild: Scenario of a Rebuild of Structure DB0G_GBP3 in CF2.

1 The data sharing group DB0G has GBP structures in CFs CF1 and CF2. The CFRM policy is used by MVS when the structures are allocated.

2 The structure specification builds two structures (DB0G_GBP0 and DB0G_GBP3) when there is inter-DB2 read-write interest on page sets that use buffer pools BP0 and BP3. The PREFLIST parameter specifies an ordered list of CF names from which the system is to choose when allocating a structure in a CF. MVS allocates DB0G_GBP0 in CF2 and DB0G_GBP3 in CF1. In this scenario, DB0G_GBP0 has no relevance.

Member DB1G in the data sharing group DB0G has a link failure to coupling facility CF1.

MVS determines whether to perform a rebuild of the structure in CF1.

3 The active SFM policy is used to calculate the system’s loss-of-connectivity percentage.

4 This percentage is compared with the REBUILDPERCENT value in the structure definition. The weights for the systems are used in the calculation. CPC1 has a weight of 40 and CPC2 and CPC3 have 30.

\[ \text{REBUILDPERCENT} = \left( \frac{A}{B} \right) \times 100 \]

- \( A \) is the total weight for the systems that have lost connectivity, in this scenario CPC1 and therefore \( A = 40 \).
• $B$ is the total weight for all systems that have connectivity prior to the failure, in this scenario CPC1, CPC2, and CPC3 and therefore $B = (40 + 30 + 30) = 100$

• The REBUILDPERCENT is $(40/100) \times 100 = 40$.

The value 40 is greater than one, and MVS will rebuild the structure DB0G_GBP3. The recommendation is to set low REBUILDPERCENT values so that DB2 always rebuilds on a loss of connectivity; a value of 1 means always rebuild. The rationale for this recommendation is that rebuilding the structure is usually preferable to the alternative, which is for the member to disconnect from the GBP and to add pages to the LPL.

5 The GBP rebuild must ensure that all changed pages in the original structure in CF1 are either cast out to disk or written to the new structure in CF2. MVS allocates the structure using the value of INITSIZE to allocate the new structure. DB2 members DB2G and DB3G, connected to the GBP, then participate in the rebuild process. The rebuild of the GBP will be in coupling facility CF2. The GBP rebuild is accomplished in two phases:

1. The Quiesce phase quiesces all GBP activity. Also, the ownership of the structure is given to one of the active DB2 members if the structure owner member loses connectivity. In our example, the structure owner is DB2G, so there is no need to switch ownership.

2. The Connect phase connects each DB2 member to the new structure in CF2. The castout owners that still have access to the original structure in CF1 read changed pages from the DB0G_GBP3 structure in CF1 and write to the new DB0G_GBP3 structure in CF2. If there are too few data entries in the new DB0G_GBP3, then the castout owners will cast out the pages to disk. When the castout activity is complete, the new structure owner does final sweep processing. A final sweep means that the DB2 member that is the GBP structure owner ensures that any remaining changed pages are cast out to disk or written to the new structure. All locally cached shared data is invalidated, so if a member wants to access a previous locally cached data after the rebuild, the member must refresh the data from the rebuilt GBP in CF2 or from disk.

The scenario in Figure 35 on page 156 explains the dynamic rebuild. The manual rebuild carries out the same set of actions, except there is no involvement of the SFM policy to determine whether to rebuild and no final sweep processing from the structure owner is necessary, unless there are failed members. Issuing the MVS command SETXCF START,REBUILD,STRNAME=DB0G_GBP3,LOC=OTHER initiates an equivalent rebuild of the GBP.

**Group Buffer Pool Rebuild Performance:** In DB2 V4, if you specify REBUILDPERCENT in your GBP structure specification in CFRM, then it is ignored. DB2 V5 provides the capability to rebuild the structure and not shut down the data sharing group to deallocate the original structure. Threads are suspended while manual or dynamic rebuild is in progress. The rebuild event is transparent to the application programs; however end users may notice a brief pause while rebuild completes. The GBP rebuild enhancement is an availability improvement; in addition, DB2 avoids writing pages to the logical page list when there is a link failure. This support allows applications to complete their writes to the GBP when the rebuild is complete.
We assess the performance of this enhancement to assist you in determining the use of rebuild in your installation. The set of results in Table 52 on page 159, Table 53 on page 159, and Figure 36 on page 160 show that the size of the GBP to be rebuilt and the amount of other workload activity in the system both influence the performance of the rebuild process. The results are the elapsed times for the rebuild process for manual and dynamic rebuild, with and without additional workload activity.

7.1.1 Group Buffer Pool Manual Rebuild

The measurement environment for the manual rebuild for the following workload activity is as follows:

- An LPAR set up on a 9672 RX3 CMOS processor with two five-way partitions
- Two 9674 C03 CFs with single links
- 100 MB, 300 MB, and 500 MB of GBP space.
  
  Each 100 MB of data is cached in a separate local buffer pool and GBP (for example, GBP1, GBP2, GBP3, GBP4, and GBP5 are used in the 500 MB measurement). The GBPs are large enough to hold 100 MB of data and the directory entries.
- Each 100 MB resides in a table space of three partitions
- Page set rebuild ownership is spread across the two members, DB1G and DB2G
- 25452 pages are rebuilt for each GBP rebuild
- Batch programs are used to update the data, and each program processes specific partitions. Job scheduling spreads the page set or partition castout ownership between DB1G and DB2G
- DB2 specifies structure size to ensure that the normal rebuild method is used.

  When rebuilding the GBP structure, DB2 ensures that the new structure has at least 10% more data entries than the number of changed pages in the old structure. If this 10% margin is not met, then DB2 builds an empty new structure and casts out to disk the entries from the old structure.

The degree of parallelism during the manual rebuild is determined by the number of members times the number of GBP structures. For example, in the 300 MB measurement, there are three GBPs. Table 52 on page 159 shows the distribution of castout ownership between members DB1G and DB2G.
Table 52. Castout Ownership for 300 MB measurement

<table>
<thead>
<tr>
<th>Group Buffer Pool and Partition</th>
<th>DB1G Castout Owner</th>
<th>DB2G Castout Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group buffer pool GBP1, Partition P1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Group buffer pool GBP1, Partition P2</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Group buffer pool GBP1, Partition P3</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Group buffer pool GBP2, Partition P1</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Group buffer pool GBP2, Partition P2</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Group buffer pool GBP2, Partition P3</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Group buffer pool GBP3, Partition P1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Group buffer pool GBP3, Partition P2</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Group buffer pool GBP3, Partition P3</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

In the rebuilding of GBP1, member DB1G moves data for two partitions and member DB2G moves data for one partition.

Table 53 shows the distribution of the rebuilt pages across the castout owners.

Table 53. Rebuilt Pages by Castout Owner

<table>
<thead>
<tr>
<th>Group Buffer Pool</th>
<th>DB1G Owner Rebuild</th>
<th>DB2G Owner Rebuild</th>
<th>Total Pages Rebuilt</th>
</tr>
</thead>
<tbody>
<tr>
<td>GBP1</td>
<td>17271</td>
<td>8181</td>
<td>25452</td>
</tr>
<tr>
<td>GBP2</td>
<td>8181</td>
<td>17271</td>
<td>25452</td>
</tr>
<tr>
<td>GBP3</td>
<td>8181</td>
<td>17271</td>
<td>25452</td>
</tr>
</tbody>
</table>

The number of GBPs can influence the elapsed time of the rebuild process as the GBP structure determines the amount of parallelism. You may consider having several GBPs instead of one very large GBP if the rebuild elapsed time is an important issue in your data sharing environment. The distribution of the castout ownership assists in improving the elapsed time for the rebuild. Figure 36 on page 160 shows the elapsed time for the GBP rebuild.
7.1.2 Group Buffer Pool Manual Rebuild Under IRWW Workload

The measurement environment for the manual rebuild under the IBM Relational Warehouse Workload (IRWW) differs from the manual rebuild discussed in 7.1.1, “Group Buffer Pool Manual Rebuild” on page 158 as the 9672 and 9674 models used in the measurements are not as powerful. The IRWW measurement environment details are discussed in DB2 V4 Data Sharing Performance Topics (SG24-4611). The speeds of the CPs and CF are the main factors in the elapsed times for rebuild.

Table 54 shows the effect of the manual rebuild on the external throughput rate under the IRWW workload.

<table>
<thead>
<tr>
<th>Environment</th>
<th>DB1G External Throughput Rate</th>
<th>DB2G External Throughput Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 minute interval for workload with no interruption</td>
<td>18243</td>
<td>18304</td>
</tr>
<tr>
<td>10 minute interval for workload with manual rebuild</td>
<td>16892</td>
<td>17176</td>
</tr>
</tbody>
</table>

The IRWW workload produces 60 transactions per second and the rebuild for eight GBPs takes an elapsed time of 45 seconds. For the measurement period of 10 minutes, the throughput of the additional workload diminishes by about 6% when the rebuild is in progress.

7.1.3 Group Buffer Pool Dynamic Rebuild Under IRWW Workload

The measurement results show the throughput of the workload with a dynamic GBP rebuild is larger than that with the manual rebuild. The reason for this difference is that dynamic rebuild is not performed serially; for example, the final sweep processing of the GBPs can overlap.
Table 55 on page 161 shows the effect of the dynamic rebuild on the external throughput rate under the IRWW workload.

<table>
<thead>
<tr>
<th>Environment</th>
<th>DB1G External Throughput Rate</th>
<th>DB2G External Throughput Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 minute interval for workload with no interruption</td>
<td>18243</td>
<td>18304</td>
</tr>
<tr>
<td>10 minute interval for workload with dynamic rebuild</td>
<td>17194</td>
<td>17282</td>
</tr>
</tbody>
</table>

You can conclude from these measurement results with concurrent workload activity that the degradation in throughput is acceptable for this environment.

7.2 Group Buffer Pool Automatic Recovery

DB2 V5 has the ability to recover GBPs automatically, thus expediting the GBP recovery process and reducing DB2 data outages across a GBP failure. To recover from a GBP failure, DB2 must recover the lost data from all the log records of all the members that may have updated the data. DB2 merges the log records into the correct time sequence order and then applies the changes to the disk copy of the data.

Damage assessment is the process of determining which page sets and partitions were using the failed GBP and marking those objects as “GBP recovery pending” (GRECP). Damage assessment is initiated either when a structure fails or when all members lose connectivity to the structure. Also, if updating applications try to write to the failed GBP, DB2 adds pages to the LPL to be recovered. When the damage assessment completes, there are probably multiple sets of page sets and partitions in GRECP state and many pages in their logical page lists.

In DB2 V4, the operator issues START DATABASE commands to recover pages in the logical page list and page sets in GRECP status. DB2 V5 can automatically initiate the recovery of the GRECP page sets and partitions, and the LPL pages after a GBP failure. A new option AUTOREC for GBPs (default value is YES) allows initiation of a recovery for the failing GBP. This occurs only if all the connecting members in the data sharing group are at DB2 V5 level and at least one member is actively connected.

7.2.1 Operational Considerations

We recommend that you be aware of the following operational considerations:

- Log data set configuration is crucial.
  - It is important to have good chance of DASD cache hits.
  - The APAR PN90695 enables sequential caching for log reads. DFSMS Version 1.3 is a prerequisite to enable this.
  - Set INBUFF and OUTBUFF ZPARM values to maximum.
- Threads that arrive during recovery are not suspended; they receive SQLCODE -904. You may wish to design your applications to be sensitive to this and retry after an interval.
• Autorecovery can generate substantial console write to operator (WTO) traffic. Therefore, you should configure your MVS console environment with this in mind.

7.2.2 Performance Results

In all, 718 IRWW page sets were recovered. Member DB1G recovered 367 page sets and member DB2G recovered 351 page sets.

Figure 37 shows the GBP automatic recovery results.

![Group Buffer Pool Automatic Recovery](image)

Figure 37. Group Buffer Pool Automatic Recovery: Scenario with Different GBP Checkpoint Intervals

With GBP checkpoint interval of 8 minutes, 2 minutes, and 1 minute, recovery took about 12 minutes, less than 5 minutes, and a little over 2 minutes, respectively.

Table 56 shows the cost to achieve the GBP automatic recovery performance using the default GBP checkpoint interval of 8 minutes.

<table>
<thead>
<tr>
<th>Item Measured</th>
<th>DB1G</th>
<th>DB2G</th>
</tr>
</thead>
<tbody>
<tr>
<td>External throughput rate</td>
<td>18191</td>
<td>18163</td>
</tr>
<tr>
<td>Internal throughput rate</td>
<td>24466</td>
<td>25180</td>
</tr>
<tr>
<td>Percent CPU utilization</td>
<td>74.35</td>
<td>72.13</td>
</tr>
<tr>
<td>Group buffer pool writes</td>
<td>116251</td>
<td>110373</td>
</tr>
<tr>
<td>Castouts</td>
<td>70670</td>
<td>51274</td>
</tr>
</tbody>
</table>

Table 57 on page 163 shows the cost to achieve the GBP automatic recovery performance using the GBP checkpoint interval of 2 minutes.
Table 57. GBP Automatic Recovery: GBP Checkpoint Interval 2 Minutes

<table>
<thead>
<tr>
<th></th>
<th>DB1G</th>
<th>DB2G</th>
</tr>
</thead>
<tbody>
<tr>
<td>External throughput rate</td>
<td>17055</td>
<td>17942</td>
</tr>
<tr>
<td>Internal throughput rate</td>
<td>22985</td>
<td>24105</td>
</tr>
<tr>
<td>Percent CPU utilization</td>
<td>74.20</td>
<td>74.43</td>
</tr>
<tr>
<td>Group buffer pool writes</td>
<td>106690</td>
<td>107097</td>
</tr>
<tr>
<td>Castouts</td>
<td>92405</td>
<td>70280</td>
</tr>
</tbody>
</table>

Table 58 shows the cost to achieve the GBP automatic recovery performance using the GBP checkpoint interval of 1 minute.

Table 58. GBP Automatic Recovery: GBP Checkpoint Interval 1 Minute

<table>
<thead>
<tr>
<th></th>
<th>DB1G</th>
<th>DB2G</th>
</tr>
</thead>
<tbody>
<tr>
<td>External throughput rate</td>
<td>17385</td>
<td>17907</td>
</tr>
<tr>
<td>Internal throughput rate</td>
<td>23044</td>
<td>23368</td>
</tr>
<tr>
<td>Percent CPU utilization</td>
<td>75.44</td>
<td>76.63</td>
</tr>
<tr>
<td>Group buffer pool writes</td>
<td>105461</td>
<td>114401</td>
</tr>
<tr>
<td>Castouts</td>
<td>94466</td>
<td>72715</td>
</tr>
</tbody>
</table>

The analysis reveals the following:

- The internal throughput rate drops by 5.2% when using 2 minute GBP checkpoint interval compared with the 8 minute interval. The marginal cost of going to a 1 minute GBP checkpoint interval is small.
- The castout rate increases as the GBP write hit ratio declines, a result of more frequent GBP checkpoints. You can also see some imbalance between castout work done by each member. You should expect to see this in your environment, depending on the workload.
- The external throughput rate fluctuations you see may be due to interference such as DASD contention, and spikes in coupling facility link utilization. There is a strong correlation between external throughput rate and castout rate, depending on the GBP checkpoint interval.

We recommend that you consider the following guidelines:

- Spread your GBPs across more than one CF.
- Use only very short GBP checkpoint intervals for critical page sets.
- Vary GBP checkpoint interval by time of the day. For example, use 1 minute interval only during online day.
- Allocate sufficient local buffer pool memory to improve the write hit rate during recovery, ensuring DASD data set placement, and cache effectiveness.
7.3 Selective Locking of Partitions in Partitioned Table Spaces

DB2 V5 introduces selective partition locking. Only the partitions being accessed are locked. This applies to dynamic SQL and plans bound with ACQUIRE(USE), as well as static SQL. In DB2 V4, all the partitions of a partition table space are locked, and therefore with up to 254 partitions in a table space, selective locking improves the concurrency of accesses to the partitioned tables. However, a program temporary fix (PTF) to DB2 V4 introduces locking only the last partition, although the whole table is considered to be locked.

The main use of selective partition locking is to allow DB2 to support explicit hierarchy locking at the partition level in a data sharing environment. This allows page or row locks to be granted locally if the accessed partition is accessed only by transactions within a single member. If more than one partition is accessed then selective partition locking increases the cost of parent locking, but can reduce the propagation of locks to XES.

If lock escalation occurs, it affects only the accessed partitions when using selective partition locking, and does not affect other partitions accessed by different transactions. Figure 38 illustrates an example of selective partition locking.

![Figure 38. Selective Partition Locking in DB2 V5 Compared with Parent L-locking in DB2 V4](image)

The left-hand side of Figure 38 shows how DB2 V4 handles parent L-locking with affinity routing and the right-hand side depicts the same scenario for DB2 V5.

Consider a partitioned table space, TS1, with two partitions, Part1 and Part2, where member DB1G accesses only Part1 and member DB2G accesses only Part2. Also assume page-level locking.
In DB2 V4, parent L-locking for partitioned table spaces is at the table space level and not at the partition level. Therefore, 1 the parent lock for Part1 page locks TS1, and similarly the parent lock for the Part2 page also locks TS1. As transactions access Part1 on DB1G and Part2 on DB2G, both DB1G and DB2G hold parent L-locks on TS1. 2 When member DB1G has to update, the page locks for Part1 and Part2 are propagated to XES and 3 the lock structure in the coupling facility is accessed to assist in determining what action is taken.

In DB2 V5 with selective partition locking, DB2 can now do partition-independent L-locking on the partitions. So in the same scenario, 4 DB1G holds the parent lock on Part1 and not at table space level on TS1. DB2G does not hold a lock on Part1. Similarly, DB2G holds the parent L-lock on Part2 and DB1G does not hold a lock on Part2. As DB1G takes page locks for pages belonging to Part1, 5 IRLM can grant the page locks locally because there is no inter-DB2 interest on the Part1 parent. DB2 now avoids incurring the additional overhead of propagating the locks to XES. The same saving is applicable to the page locks taken by DB2G for Part2.

Selective partition locking provides true partition independence for L-locks, avoiding inter-DB2 read/write interest when each member processes a different partition. You can specify the LOCKPART YES option of CREATE TABLESPACE and ALTER TABLESPACE and then each locked partition is a separate lock parent. DB2 and IRLM can detect when no inter-DB2 read-write interest exists on that partition and therefore do not propagate child locks unnecessarily, which reduces CPU times by reducing false contentions.

The use of the LOCKPART option also provides support for coexistence of data sharing members at different DB2 levels.

The combination of DB2 V5 selective partition locking and the existing GBP dependency tracking mechanisms yielded a 17% ITR improvement in IRWW workload modified to exploit affinity routing.

### 7.4 Reducing Data Sharing Costs

DB2 V5 introduces many enhancements that reduce the cost of using data sharing:

- Conditional P-locks for data pages with row locks
- Specifying the maximum number of rows in a page
- Dynamic inter-DB2 interest tracking improvement
- Reduced IRLM latch contention from P-lock unlock
- Reduced other P-lock negotiation
- Hashing algorithm changed to reduce false contention
- Default change for the global commit log sequence number
- Space-map page update frequency and member-private space map pages.
7.4.1 Conditional Page P-Locks for Data Pages with Row Locks

Data sharing in DB2 V4 introduced a new type of lock, called a page physical lock (page P-lock) which plays an important role in how DB2 tracks inter-DB2 interest. For example, in DB2 V4, if one DB2 member requests a page P-lock that another DB2 member holds, the mode of the page P-lock can be negotiated. This process involves the first DB2 member relinquishing the page P-lock by forcing a write to the active log, a write to a GBP, and an unlock of the page P-lock. Now, DB2 V5 uses a conditional page P-lock for insert so the requesting member will move to a new page instead of negotiating the page P-lock.

Table 59 shows the results of the distribution of resources for Workload A, for Workload B without the use of conditional page P-lock, and for Workload B’ with the use of conditional page P-lock.

<table>
<thead>
<tr>
<th>Statistics for Each Commit</th>
<th>Workload A</th>
<th>Workload B</th>
<th>Workload B’</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total SQL Calls</td>
<td>28</td>
<td>124</td>
<td>124</td>
</tr>
<tr>
<td>Select (%)</td>
<td>58</td>
<td>53</td>
<td>53</td>
</tr>
<tr>
<td>Insert and Delete (%)</td>
<td>19</td>
<td>33</td>
<td>33</td>
</tr>
<tr>
<td>Update (%)</td>
<td>23</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Page P-Lock lock requests</td>
<td>1.5</td>
<td>25</td>
<td>28</td>
</tr>
<tr>
<td>IRLM global contention rate</td>
<td>0.03</td>
<td>4.2</td>
<td>1.5</td>
</tr>
<tr>
<td>Page P-Lock negotiation rate</td>
<td>0.03</td>
<td>3.9</td>
<td>1.4</td>
</tr>
<tr>
<td>XES global contention rate</td>
<td>0</td>
<td>0.01</td>
<td>0</td>
</tr>
<tr>
<td>MSTR SRB time (ms)</td>
<td>10</td>
<td>28</td>
<td>34</td>
</tr>
<tr>
<td>DBM1 SRB time (ms)</td>
<td>18</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>IRLM SRB time (ms)</td>
<td>1.4</td>
<td>41</td>
<td>16</td>
</tr>
</tbody>
</table>

Because of the workload characteristics for Workload A, there is very little IRLM global contention and therefore the conditional page P-lock would not show any significant benefit.

Workload B has a high IRLM global contention and a high page P-lock contention. Workload B’ shows that the conditional page P-lock capability reduces the IRLM global contention and page P-lock contention. The SRB time for IRLM drops dramatically from 41 ms to 16 ms.

Conditional page P-Lock provides a vast improvement in workload environments where there are many data page P-locks because of row locks in conjunction with a high percentage of inserts.

7.4.2 Specifying the Maximum Number of Rows in a Page

In DB2 V5, you can specify the MAXROWS option with CREATE TABLESPACE and ALTER TABLESPACE to control the number of rows to be placed in a page. This maximum number persists through inserts or deletes and when data is loaded or reorganized. The default is now 255 rows per page, so it is no longer necessary to use compression to attain 255 rows per page.
For applications requiring the concurrency of row locks with the lower cost of page locks, you can use this option to ensure that only one row is placed in a page. However, you still need to specify LOCKSIZE PAGE even if MAXROWS (1) is specified to avoid the data page P-lock requests. This helps in eliminating the data page P-lock request when row locking is used in a data sharing environment.

7.4.3 Dynamic Inter-DB2 Interest Tracking Improvement

In DB2 V4, when a member has read-write interest in a page set or partition and the inter-DB2 interest on the page set or partition then goes away, the page set or partition remains GBP-dependent until the last updating member switches to read-only. DB2 V5 removes the GBP-dependent status earlier by switching to non-GBP-dependent status as soon as the reader does a physical close, instead of waiting for a pseudo-close by the updater. You need to specify CLOSE YES to allow DB2 to use PCLOSESET and PCLOSEN to monitor whether a “not-in-use” GBP-dependent page set or partition can be physically closed by a read-only accessed member. When a DB2 member of a data sharing group is stopped, DB2 V5 also switches to non-GBP-dependent status without waiting for a pseudo-close. This improvement assists in reducing the number of GBP-dependent resources in the GBPs.

7.4.4 Reduce IRLM Latch Contention from P-lock Unlock

In DB2 V5, because of code optimization, you should see a reduction in IRLM latch contention from P-lock unlock. Both accounting and statistics indicate this.

7.4.5 Reduce Other P-Lock Negotiation

DB2 V5 reduces “Other P-lock negotiation” by freeing Type 2 index tree P-locks at the end of tree modification, instead of downgrading to a share lock. This enhancement helps improve performance of applications that are INSERT intensive, particularly when one or more indexes are affected by the insert activity. Statistics confirm this.

7.4.6 Hashing Algorithm Change to Reduce False Contention

A change to the hashing algorithm in DB2 V5 reduces contention for page set or partition L-locks. This enhancement helps improve performance of transactions that use RELEASE(COMMIT) or do not reuse thread.

Both accounting and statistics confirm this improvement.

7.4.7 Default Change for the Global Commit Log Sequence Number

The GCLSN update timer default is 1 second in DB2 V5, reduced from 10 seconds in DB2 V4. This change provides more effective lock avoidance. Frequent commits of all processes in each member in a data sharing group enables the GCLSN to move forward and improve lock avoidance for GBP-dependent page sets or partitions. The CLSN update timer updates each members CLSN entry in the SCA.
7.4.8 Space Map Page Update and Member-Private Space Map Pages

DB2 V5 introduces a reduction in the frequency of updates to space map pages and allow members in a data sharing environment to have private space map pages to reduce the space map page P-lock contention. This enhancement works in conjunction with the preformatting of data. See 5.1.1, “PREFORMAT” on page 74 for more information.

DB2 tries to isolate inserts from each DB2 member to its own set of data and space map pages. By isolating inserts to different areas for each data sharing member, DB2 will not be able to store data in the clustering key sequence even if a clustering index is defined. However, for an application retrieving a single row using an index, the loss of the clustering sequence does not create a problem.

By ignoring the clustering for inserts, the space allocation is controlled by the space map pages only. In a data sharing environment, a space map page can be updated by a member only when a page P-lock is held in X mode. To allow each DB2 member to make inserts into its own set of data pages, the space map page P-lock controls the ownership of the space map page and its covering data pages. To prevent a member from using another member's space map pages for inserts, DB2 uses conditional P-lock requests for space maps on inserts. The space management scheme applies only to nonsegmented table spaces.

DB2 V5 provides a (NO_CLUSTERING) option for the LOAD REPLACE and REORG to use this space management scheme.

7.5 Sysplex Coupling Data Set Configuration Recommendations

This section is not directly realted to data sharing enhancements. We have included this section to provide some recommendations. In addition to all of the regular non-data-sharing and data sharing tuning wisdom, we recommend that you consider the following guidelines for the Sysplex coupling data set configuration:

- Place the primary and alternate CFRM coupling data sets on separate volumes.
- Place the primary and alternate Sysplex coupling data sets on either separate volumes, or on the same volumes as the CFRM coupling data sets but in reverse order (that is, one volume contains primary Sysplex and alternate CFRM, other volume contains alternate SYSPLEX and primary CFRM).
- Place the CFRM coupling data sets behind a cached control unit (with DASD Fast Write, if possible).
- Place the CFRM coupling data sets on quiet volumes with little activity or contention.
- Place the Sysplex and CFRM coupling data sets on volumes with absolutely no RESERVE or RELEASE activity.
- Avoid overspecifying the number of coupling data set items supported in the CFRM coupling data sets at format time. You can specify the number of structures, connectors, policies, etc.; specifying them higher than really necessary can increase the amount of CFRM coupling data set I/O that has to be done.
Appendix A. Estimates When SORTKEYS Is Used

When SORTKEYS is specified, there are some changes to the formulas used to calculate values used in the LOAD utility.

A.1.1.1 Number of Keys to Sort
The following formula is used to calculate the estimated number of keys to sort when SORTKEYS is specified:

- Count 1 for each index.
- Count 1 for each foreign key that is not exactly indexed (that is, where foreign key and index definition do not correspond exactly).
- For each foreign key that is exactly indexed,
  - Count 0 for the first relationship in which the foreign key participates.
  - Count 1 for the subsequent relationship in which the foreign key participates (if any).
- Multiply count by number of rows to be loaded.

If more than one table is to be loaded in same job, repeat the steps above for each table and sum the results.

Example 1.
Assume that there are 22000 rows to load into DSN8510.DEPT table. This table has three indexes and two foreign keys. The foreign keys are exactly indexed.

The calculation of the number of keys to sort is:

$$(3 + 0) \times 22000 = 66000$$

The LOAD statement is:

```
LOAD DATA REPLACE INDDN INPUT
SORTKEYS 66000
INTO TABLE DSN8510.DEPT
```

Example 2.
Assume that one of the foreign keys is not exactly indexed.

The calculation of the number of keys to sort is:

$$(3 + 1) \times 22000 = 88000$$

A.1.1.2 Size of Work Data Sets
The following formulas are used to calculate the estimated size of the SYSUT1 and SORTOUT data sets when SORTKEYS is specified.

1. If SORTKEYS is used and an estimate of the number of keys to be sorted is omitted or specified as 0:
   - The SYSUT1 data set is required. The method used to estimate the size is $\max(k,e)$ for a simple table space or $\max(k,e,m)$ if the table space is partitioned or segmented. Here,
- $k$ is the key calculation \((\text{max}(\text{longest index key} + 13, \text{longest foreign key} + 13) \times \text{number of keys extracted})\).

- $e$ is the error calculation (see “Work Data Set Calculation” in The DB2 Command and Utility Reference).

- $m$ is the map calculation \((17 \times \text{number of rows produced by LOAD job})\).

- For keys extracted,
  - Count 1 for each index.
  - Count 1 for each foreign key that is not exactly indexed (that is, where foreign key and index definition do not correspond exactly).
  - For each foreign key that is exactly indexed,
    - Count 0 for the first relationship in which the foreign key participates.
    - Count 1 for the subsequent relationship in which the foreign key participates (if any).
  - Multiply count by number of rows to be loaded.

**Example 3.**

Assume that there are 22,000 rows to load into DSN8510.DEPT table.

This table has three indexes
- on column DEPTNO (3 bytes)
- on column MGRNO (3 bytes)
- on column ADMRDEPT (3 bytes)

and two foreign keys
- on column ADMRDEPT (3 bytes)
- on column MGRNO (3 bytes)

The calculation of the size of the SYSUT1 data set when the estimate of the number of keys to be sorted is omitted or specified as zero is:

\[
\text{max}(k, e)
\]

\[
K = \text{max}(3 + 13, 3 + 13) \times ((3 + 0) \times 22000)
\]

\[
K = 16 \times 66000
\]

\[
K = 1056000 \text{ bytes} \Rightarrow 1032 \text{ KB}
\]

If we assume that $k$ is the greater of $k$ and $e$, this would be the size of the SYSUT1 data set.

- The SORTOUT data set is required. The method used to estimate the size is $\text{max}(f, e)$ where:
  - $f$ is foreign key calculation \(((\text{longest foreign key} + 13) \times \text{number of keys extracted})\).
  - $e$ is error calculation (see “Work Data Set Calculation” in The DB2 Command and Utility Reference).
Example 4.

Assume the same table as in Example 3.

The calculation of the size of the SORTOUT data set is:

\[
\max(f, e) = (3 + 13) \times ((3 + 0) \times 22000) \\
= 16 \times 66000 \\
= 1056000 \text{ bytes} \Rightarrow 1032 \text{ KB}
\]

If we assume that \( f \) is the greater of \( f \) and \( e \), this would be the size of the SORTOUT data set.

If the table to be loaded does not have foreign keys, the SORTOUT data set is still needed to handle errors encountered during the reload phase. The size of the SORTOUT data set in this case is difficult to predict. It must be large enough to hold an 80-byte record for every error encountered, which normally is not many, but in worst case can be one for each record to load.

2. If SORTKEYS is used and an estimate of the number of keys to be sorted is specified (not 0):

- The SYSUT1 data set is required. The method used to estimate the size is \( \max(f, e) \) for a simple table space, or \( \max(f, e, m) \) if the table space is partitioned or segmented.
  - \( f, e, \) and \( m \) are described above in example 4.

Example 5.

Assume the same table as in Example 3.

The calculation of the size of the SYSUT1 data set when an estimate of the number of keys to sort is provided (not 0) is:

\[
\max(f, e) = (3 + 13) \times ((3 + 0) \times 22000) \\
= 16 \times 66000 \\
= 1056000 \text{ bytes} \Rightarrow 1032 \text{ KB}
\]

If we assume that \( f \) is the greater of \( f \) and \( e \), this would be the size of the SYSUT1 data set.

If the table to load does not have foreign keys, the SYSUT1 data set is still needed to handle errors encountered during the reload phase. The size of the SYSUT1 data set in this case is difficult to predict. It must be large enough to hold an 80 bytes record per error encountered, which normally is not many, but in worst case can be one per record to load.

- The SORTOUT data set is required. The method used to estimate the size is \( \max(f, e) \) as described in Example 4.
Example 6.

Assume the same table as in Example 3. The calculation of the size of the SORTOUT data set is:

\[ \text{max}(f, e) \]

\[ f = (3 + 13) \times ((3 + 0) \times 22000) \]

\[ f = 16 \times 66000 \]

\[ f = 1056000 \text{ bytes} \Rightarrow 1032 \text{ KB} \]

If we assume that \( f \) is the greater of \( f \) and \( e \), this would be the size of the SORTOUT data set.

If the table to be loaded does not have foreign keys, the SORTOUT data set is still needed to handle errors encountered during the reload phase. The size of the SORTOUT data set in this case is difficult to predict. It must be large enough to hold an 80-byte record for every error encountered, which normally is not many, but in worst case can be one for each record to load.
Appendix B. Example of Key Column Correlation Statistics

This appendix gives an example of the entries that are recorded in SYSIBM.SYSCOLDIST, or in SYSIBM.SYSCOLDISTSTATS for partitioned table spaces, when you use KEYCARD and FREQVAL keywords to collect statistics. The table and data we use are similar to those shown in Chapter 5 of DB2 for OS/390 Version5 Administration Guide Volume 2. We show how you extract the catalog information, and demonstrate the value of the CASE expression.

B.1.1.1 Table and Index Definitions

Table 60 shows our test table DB2RES7.CREWINFO. Columns CITY and STATE are highly correlated, while columns DEPTNO and SEX are entirely independent.

<table>
<thead>
<tr>
<th>Table 60. Table CREWINFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>CITY</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>FRESNO</td>
</tr>
<tr>
<td>FRESNO</td>
</tr>
<tr>
<td>FRESNO</td>
</tr>
<tr>
<td>FRESNO</td>
</tr>
<tr>
<td>NEW YORK</td>
</tr>
<tr>
<td>NEW YORK</td>
</tr>
<tr>
<td>MIAMI</td>
</tr>
<tr>
<td>MIAMI</td>
</tr>
<tr>
<td>LOS ANGELES</td>
</tr>
<tr>
<td>LOS ANGELES</td>
</tr>
</tbody>
</table>

The following DDL creates the test table:

```
CREATE TABLE DB2RES7.CREWINFO
  (CITY CHAR (14) NOT NULL,
   STATE CHAR (2) NOT NULL,
   DEPTNO CHAR (4) NOT NULL,
   SEX CHAR (1) NOT NULL,
   EMPNO CHAR (5) NOT NULL,
   ZIPCODE CHAR (5) NOT NULL)
  IN TSSID0003.CREWINFS
```

There are two indexes defined on the table:

- INDEX1 on table columns 1, 2, and 6.
- INDEX2 on table columns 3 and 4.

The following DDL creates the indexes:

```
CREATE INDEX INDEX1
  ON DB2RES7.CREWINFO
  (CITY, STATE, DEPTNO, SEX)
  CLUSTER

CREATE INDEX INDEX2
  ON DB2RES7.CREWINFO
  (DEPTNO, SEX)
```

For this example we loaded only the ten rows shown into our test table to enable an easier explanation of the statistics that RUNSTATS collects.
B.1.1.2 RUNSTATS
We ran RUNSTATS using the following utility control statement:

RUNSTATS TABLESPACE (TSSD0003,CREWINFS)
  TABLE (CREWINFS)
  INDEX (DB2RES7.INDEX1 KEYCARD -- CARDINALITY
    FREQVAL NUMCOLS 1 COUNT 10
    FREQVAL NUMCOLS 2 COUNT 10
    FREQVAL NUMCOLS 3 COUNT 10,
    DB2RES7.INDEX2)

Note that DB2 V5 allows you to place comments in utility control statements.
This statement collects the following statistics:

• FIRSTKEYCARDF and FULLKEYCARDF for INDEX1 and INDEX2 in
  SYSIBM.SYSINDEXES.

• Cardinality on the concatenation of the first and second key columns of
  INDEX1.

• Frequent values, by default, for the ten most frequent values of DEPTNO, the
  first key column in INDEX2.

• Frequent values for:
  – The ten most frequent values of CITY, the first key column in INDEX1.
  – The ten most frequent values of CITY concatenated with STATE, the first
    two key columns in INDEX1.
  – The ten most frequent values of CITY concatenated with STATE
    concatenated with ZIPCODE, the first three key columns in INDEX1.

In practice you might not collect statistics for NUMCOLS 3, as only the first two
key columns of INDEX1 are correlated.

B.1.1.3 Catalog Queries
The queries we show here are appropriate to our test environment, but you need
modify them for your own use.

We extract the values of FIRSTKEYCARDF and FULLKEYCARDF for INDEX1 and
INDEX2 from SYSIBM.SYSINDEXES using QUERY A.

---QUERY A

SELECT NAME AS INDEX,
  INTEGER(FIRSTKEYCARDF) AS FIRSTKEYCARDF,
  INTEGER(FULLKEYCARDF) AS FULLKEYCARDF
FROM SYSIBM.SYSINDEXES
WHERE NAME IN ("INDEX1","INDEX2")
  AND CREATOR = "DB2RES7"
ORDER BY INDEX

We next show the results of running QUERY A.

<table>
<thead>
<tr>
<th>INDEX</th>
<th>FIRSTKEYCARDF</th>
<th>FULLKEYCARDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDEX1</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>INDEX2</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>
We extract FIRSTKEYCARD and FULLKEYCARD values in integer form for easy reference.

We extract statistics about the frequent values for the first key column in indexes INDEX1 and INDEX2, the concatenations of the key columns in INDEX1, and the cardinality values on concatenated index key columns in INDEX1 using QUERY B.

---QUERY B

```
SELECT NAME, TYPE AS T, INTEGER(CARDF) AS CF,
CASE
  WHEN NUMCOLUMNS = 2 THEN -- list the 2 associated columns
    HEX(SUBSTR(COLGROUPCOLNO,1,4))
  WHEN NUMCOLUMNS = 3 THEN -- list the 3 associated columns
    HEX(SUBSTR(COLGROUPCOLNO,1,6))
  WHEN NUMCOLUMNS = 4 THEN -- list the 4 associated columns
    HEX(SUBSTR(COLGROUPCOLNO,1,8))
  WHEN NUMCOLUMNS = 5 THEN -- list the 5 associated columns
    HEX(SUBSTR(COLGROUPCOLNO,1,10))
  ELSE NULL
END AS COLGRPCOLNO,
  NUMCOLUMNS AS NUMCOL,
CASE
  WHEN FREQUENCYF = -1 THEN -- report % of rows in table
    INTEGER(FLOAT(FREQUENCYF)*100) -- for each frequent value
  ELSE -1
END AS FQY,
  COLVALUE
FROM SYSIBM.SYSCOLDIST
WHERE TBNAME = CREWINFO -- ensure that query
  AND TBOWNER = DB2RES7 -- is a matching index scan
ORDER BY NAME, T, NUMCOL, COLGRPCOLNO, FQY DESC, COLVALUE
```

COLGROUPCOLNO in SYSIBM.SYSCOLDIST is VARCHAR(254). If the statistics in a row are for a concatenation of columns, then the field is an array of SMALLINT columns with a dimension equal to the value in NUMCOLUMNS. We use the CASE expression to extract the SMALLINT values based on the actual value in NUMCOLUMNS. The query caters for up to five concatenated key columns, with an output length of 20 bytes for COLGRPCOLNO, and can easily be modified for more key columns.

The second use of case is to translate the FREQUENCYF column values into the percentage of rows in the table for each value of COLVALUE, by multiplying FREQUENCYF by 100, except when the value in FREQUENCYF is −1. We show the percentage values in the FQY results column.

We next show the results of running QUERY B.
The following points are of interest:

- Consider where the value in column T (TYPE) is C, indicating the row contains cardinality statistics:
  - The value of NAME shows that CITY is the first column of the set of columns associated with the statistics.
  - The NUMCOL (NUMCOLUMNS) value of 2 shows the row is for a concatenation of two columns.
  - The value of 00010002 in COLGROUPNO shows that the concatenated columns are the first two columns of the table.
  - The CF (CARDF) value for CITY is 4, and is therefore the number of distinct values for the first two key columns of INDEX1.
    The value is the same as FIRSTKEYCARDF for INDEX1 and indicates key column correlation is highly likely in the first two columns.
    The value is less than FULLKEYCARDF and indicates correlation is restricted to the first two key columns.

- Consider where the value in column T (TYPE) is F, indicating the row contains statistics about frequent values:
  - NUMCOL (NUMCOLUMNS) of 1 indicates statistics for the frequent values for the table column referred to by the NAME column. We have four frequent values for CITY, the first key column in INDEX1, and four frequent values for DEPTNO the first key column in INDEX2.
  - NUMCOL (NUMCOLUMNS) of 2, COLGROUPNO of 00010002 and NAME of CITY indicate statistics for the frequent values for the concatenation of the first two key columns of INDEX1.
  - NUMCOL (NUMCOLUMNS) of 3, COLGROUPNO of 000100020006 and NAME of CITY indicate statistics for the frequent values for the concatenation of the first three key columns of INDEX1.
These examples show how to interpret cardinality and statistics about frequent values from the DB2 catalog and show examples of conditional logic using the CASE expression.
Appendix C. Special Notices

This publication is intended to help managers and professionals understand and evaluate DB2 for OS/390 Version 5 performance. The information in this publication is not intended as the specification of any programming interfaces that are provided by DB2 for OS/390 Version 5. See the PUBLICATIONS section of the IBM Programming Announcement for DB2 for OS/390 Version 5 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM’s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM’s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer’s ability to evaluate and integrate them into the customer’s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:
The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Microsoft, Windows, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Java and HotJava are trademarks of Sun Microsystems, Inc.

Other trademarks are trademarks of their respective companies.
Appendix D. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook:

- **DB2 for OS/390 Version 5 What is New?, GC26-8971**
- **DB2 for OS/390 Version 5 Release Guide, SC26-8965**
- **DB2 for OS/390 Version 5 Data Sharing: Planning and Administration, SC26-8961**
- **DB2 for OS/390 Version 5 Installation Guide, GC26-8970**
- **DB2 for OS/390 Version 5 Administration Guide, SC26-8957**
- **DB2 for OS/390 Version 5 Command Reference, SC26-8960**
- **DB2 for OS/390 Version 5 Utility Guide and Reference, SC26-8967**
- **DB2 for OS/390 Version 5 Application Programming and SQL Guide, SC26-8958**
- **DB2 for OS/390 Version 5 SQL Reference, SC26-8966**
- **DB2 for OS/390 Version 5 Call Level Interface Guide and Reference, SC26-8959**
- **OS/390 MVS Planning: Workload Management, GC28-1761**

D.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How To Get ITSO Redbooks” on page 183:

- **DB2 for MVS/ESA Version 4 Data Sharing Performance Topics, SG24-4611**
- **DB2 for MVS/ESA Version 4 Non-Data-Sharing Performance Topics, SG24-4562**
- **DB2 for MVS/ESA Version 4 Data Sharing Implementation, SG24-4791**
- **Getting Started with Stored Procedures: Give Them a Call through the Network, SG24-4693-00**
- **Distributed Relational Database: Cross Platform Connectivity and Application, SG24-4311-02**
- **DB2 Quick Upper Bound Estimate: An Application Design Methodology, SG24-2549**
- **DB2 PM Usage Guide Update, SG24-2584**
- **DB2 V3 Performance Topics, GG24-4284**
- **System/390 MVS Parallel Sysplex Performance, SG24-4356-02**
### D.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

<table>
<thead>
<tr>
<th>CD-ROM Title</th>
<th>Subscription Number</th>
<th>Collection Kit Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>System/390 Redbooks Collection</td>
<td>SBOF-7201</td>
<td>SK2T-2177</td>
</tr>
<tr>
<td>Networking and Systems Management Redbooks Collection</td>
<td>SBOF-7370</td>
<td>SK2T-6022</td>
</tr>
<tr>
<td>Transaction Processing and Data Management Redbook</td>
<td>SBOF-7240</td>
<td>SK2T-8038</td>
</tr>
<tr>
<td>AS/400 Redbooks Collection</td>
<td>SBOF-7270</td>
<td>SK2T-2849</td>
</tr>
<tr>
<td>RS/6000 Redbooks Collection (HTML, BkMgr)</td>
<td>SBOF-7230</td>
<td>SK2T-8040</td>
</tr>
<tr>
<td>RS/6000 Redbooks Collection (PostScript)</td>
<td>SBOF-7205</td>
<td>SK2T-8041</td>
</tr>
<tr>
<td>Application Development Redbooks Collection</td>
<td>SBOF-7290</td>
<td>SK2T-8037</td>
</tr>
<tr>
<td>Personal Systems Redbooks Collection</td>
<td>SBOF-7250</td>
<td>SK2T-8042</td>
</tr>
</tbody>
</table>

### D.3 Other Publications

These publications are also relevant as further information sources:

- *IBM DATABASE 2 Performance Monitor for OS/390 Version 5 Report Reference, Volume 1*, SC26-8985
How To Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at URL http://www.redbooks.ibm.com/redbooks.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type GOPHER.WTSCPOK.ITSO.IBM.COM
- **Tools disks**
  
  To get LIST3820s of redbooks, type one of the following commands:
  
  ```
  TOOLS SENDTO EHONE4 TOOLSYS REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLZ REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get lists of redbooks:
  
  ```
  TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
  TOOLS SENDTO USDIST MKTOOLS MKTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTOOLS MKTOOLS GET LISTSERV PACKAGE
  ```

  To register for information on workshops, residencies, and redbooks:
  
  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  ```

  For a list of product area specialists in the ITSO:
  
  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Home Page on the World Wide Web**
  

- **IBM Direct Publications Catalog on the World Wide Web**
  

  IBM employees may obtain LIST3820s of redbooks from this page.

- **ITSO4USA category on INEWS**

- **Online** — send orders to: USIB6FPL at IBMMAIL or DKBMBMSH at IBMMAIL

- **Internet Listserv**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserv. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.
How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** (Do not send credit card information over the Internet) — send orders to:

<table>
<thead>
<tr>
<th>In United States:</th>
<th>IBMMAIL</th>
<th>Internet</th>
</tr>
</thead>
<tbody>
<tr>
<td>In Canada:</td>
<td>caibmbkz at ibmmail</td>
<td><a href="mailto:lmannix@vnet.ibm.com">lmannix@vnet.ibm.com</a></td>
</tr>
<tr>
<td>Outside North America:</td>
<td>bookshop at dkibmbsh at ibmmail</td>
<td><a href="mailto:bookshop@dk.ibm.com">bookshop@dk.ibm.com</a></td>
</tr>
</tbody>
</table>

- **Telephone orders**

  | United States (toll free) | 1-800-879-2755 |
  | Canada (toll free) | 1-800-IBM-4YOU |
  | Outside North America | (long distance charges apply) |
  | (+45) 4810-1320 - Danish | (+45) 4810-1020 - German |
  | (+45) 4810-1420 - Dutch | (+45) 4810-1620 - Italian |
  | (+45) 4810-1540 - English | (+45) 4810-1270 - Norwegian |
  | (+45) 4810-1670 - Finnish | (+45) 4810-1120 - Spanish |
  | (+45) 4810-1220 - French | (+45) 4810-1170 - Swedish |

- **Mail Orders** — send orders to:

  | IBM Publications | IBM Publications | IBM Direct Services |
  | Publications Customer Support | 144-4th Avenue, S.W. | Sortemosevej 21 |
  | P.O. Box 29554 | Calgary, Alberta T2P 3N5 | DK-3450 AallerT3d |
  | Raleigh, NC 27626-0570 | Canada | Denmark |
  | USA | |

- **Fax** — send orders to:

  | United States (toll free) | 1-800-445-9269 |
  | Canada (toll free) | 1-800-267-4455 |
  | Outside North America | (+45) 48 14 2207 (long distance charge) |

- **1-800-IBM-4FAX (United States) or (+1) 415 855 43 29 (Outside USA)** — ask for:

  - Index # 4421 Abstracts of new redbooks
  - Index # 4422 IBM redbooks
  - Index # 4420 Redbooks for last six months

- **Direct Services** — send note to softwareshop@vnet.ibm.com

- **On the World Wide Web**

  | Redbooks Home Page | http://www.redbooks.ibm.com/redbooks |

- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).
IBM Redbook Order Form

Please send me the following:

<table>
<thead>
<tr>
<th>Title</th>
<th>Order Number</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• Please put me on the mailing list for updated versions of the IBM Redbook Catalog.

First name  
Last name

Company

Address

City  
Postal code  
Country

Telephone number  
Telefax number  
VAT number

• Invoice to customer number

• Credit card number

Credit card expiration date  
Card issued to  
Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

DO NOT SEND CREDIT CARD INFORMATION OVER THE INTERNET.
## Index

### A
- access path reoptimization 8
- active log preformatting 10
- Advanced Program-to-Program Communication
  - See APPC
- AIX 11, 136, 140
- ALTER BUFFERPOOL command 58
- AnyNet 11, 136, 140
- APAR
  - PN8570 54
  - PN90695 161
- APPC 136, 137, 138
- application portability
  - ASCII server support 147
  - CLI support 144
- ASCII server support
  - changes 148
  - implementation considerations 149
  - performance 150
  - table support 12
- authorization caching 10
- authorized program analysis report
  - See APAR
- availability
  - data sharing 6
  - DDL concurrency 10
  - REORG 5

### B
- BSAM striping 9
- buffer pool thresholds
  - VPPSEQT 58
  - VPSEQT 58
  - VPXPSEQT 58
- BUILD phase of online REORG 109
- BUILD2 phase of online REORG 110

### C
- C language 128
- C++ 2, 128
- cache dynamic SQL statements
  - applications that can benefit 18
  - controlling 20
  - criteria for prepared statement caching 19
  - criteria for prepared statement retrieval 19
  - EDM pool considerations 20
  - performance measurements 22
  - why necessary 17
- CAF 130, 131, 144
- call attach facility
  - See CAF
- call level interface
  - See CLI
- capacity
  - data sharing 6
  - large partitioned tables 8
- CASE expressions 8
- catalog tables
  - SYSIBM.SYSCOLDIST 28, 40, 105, 175
  - SYSIBM.SYSCOLDISTSTATS 40, 105
  - SYSIBM.SYSCOLSTATS 91
  - SYSIBM.SYSCOLUMNS 91
  - SYSIBM.SYSINDEXES 40, 100
  - SYSIBM.SYSINDEXSTATS 100
  - SYSIBM.SYSPACKAGE 139
  - SYSIBM.SYSLOCATIONS 139
  - SYSIBM.SYSPACKSTMT 33
  - SYSIBM.SYSPLAN 33
  - SYSIBM.SYSPROCEDURES 125, 131
  - SYSIBM.SYSSTMT 33
- CCSID
  - code page conversion 149
  - data sharing environment 149
  - determining the encoding scheme in an application 148
  - distributed data 149
  - measurement data 150
  - specifying encoding scheme 148
  - used in client/server environment 147
  - used with LOAD utility 149
- CFRM
  - coupling data set 168
  - policy 156
- CHANGELIMIT option 86
- CICS 64
- CLI
  - advantages 145
  - implementation 144
  - selecting the interface 146
- client/server enhancements
  - application portability 13, 144
  - CLI support 144
  - DRDA application performance 12, 140
  - native TCP/IP support 11, 136
  - package authorization 12
  - security 13
  - stored procedures 11, 125
- coded character set identifier
  - See CCSID
- compatibility mode 59, 60, 70
- conditional P-locks 6
- COPY
  - automating the process 88
  - CHANGELIMIT option 86
COPY (continued)
performance enhancements 85
correlated index columns 8
coupling facility resource manager
See CFRM
CURRENT DEGREE special register 20, 51
CURRENT RULES special register 20
CURRENT TIMESTAMP special register 28
CURRENTDATA bind option 20
cursor WITH HOLD
effect on parallelism 54
example 18, 26
locking overhead 17
status of cursor at commit 19, 26
used in a DRDA application 142
used with DECLARE CURSOR statement 24
used with stored procedures 127
Customer Information Control System
See CICS

D

data sharing enhancements
coeexistence support 6
conditional P-locks 6
dynamic inter-DB2 interest tracking 7
group buffer pool automatic recovery 6, 161
group buffer pool rebuild 6, 155
maximum rows in a page 7
reducing data sharing costs 6, 165
selective partition locking 7, 164
database descriptor
See DBD
DataJoiner 2, 11
DB2 Estimator 2, 15
DB2 installer 14
DB2 PM
accounting information 35
as a server solution 2
common use of record trace 33
new fields to monitor EDM pool 21
OS/2 version of online monitor 15
record trace to format records 32
usage guide update redbook 181
use with EXPLAIN 31
using for real-time online monitoring 15
DB2-managed data set 74, 110
DBCS 149
DBD
locking with cursor WITH HOLD 17, 24
monitoring EDM pool 22
DCE security 13
DDL concurrency 10
DEFER(PREPARE) bind option 29, 141, 142
DEGREE(ANY) bind option 51
DESCRIBE statement
considerations for applications 30
effect of using KEEPDYNAMIC(YES) 25
used with DEFER(PREPARE) 141
DESCRIBE statement (continued)
used with reoptimization 30
desktop tools 11
directory tables
DBD01 78
SYSLGRNX 10
DISPLAY BUFFERPOOL command 70
distributed computing environment security
See DCE security
double byte character set
See DBCS
DRDA application performance enhancements
ASCII table support 12
high volume data transfer 12
improved dynamic SQL processing 140
improved VTAM processing 143
OPTIMIZE for n rows 12, 143
reduced network traffic 12
DRDA-1 server 145
DRDA-2 server 145
DSNTEP2 18
dynamic SQL statement caching 7
DYNAMICRULES(BIND) bind option 20, 147
DYNAMICRULES(RUN) bind option 20

E

EDM pool
caching dynamic SQL 17, 18
considerations 20
how to monitor 21
impact on size 23
environmental descriptor management pool
See EDM pool
EXECUTE statement
considerations for applications 30
reoptimization 28
reoptimization of dynamic SQL 30
used with REOPT(VARS) 29

F

false contention 7, 165, 167
FIRSTKEYCARD 42, 91, 175
FIRSTKEYCARDF 40, 91, 101, 174
FULLKEYCARD 40, 91, 175
FULLKEYCARDF 43, 91, 102

G

GBPCACHE 58
GCLSN 7, 167
global commit log sequence number
See GCLSN
goal mode 59, 62, 70
group buffer pool automatic recovery
operational considerations 161
performance results 162
Index

group buffer pool rebuild performance
  dynamic rebuild under IRWW workload 160
  manual rebuild 158
  manual rebuild under IRWW workload 160

H
  HIGH2KEY 91, 93
  HP/UX 11, 136

I
  IBM Database 2 Performance Monitor
    See DB2 PM
  IBM Relational Warehouse Workload
    See IRWW
  IFCID
    0002 21
    0003 21
    0022 32, 35
    0148 21
    0221 35, 71
    0222 35
    0231 35, 71
    0247 32
  image copy enhancements 14
  IMS 11, 64, 128
  Information Management System
    See IMS
  inline image copy
    considerations 82
    measurements 83
    with LOAD 81
    with REORG 81
  instrumentation facility component ID
    See IFCID
  integrated resource lock manager
    See IRLM
  IRLM
    estimating extra storage required for parallel tasks 59
    global contention 166
    group buffer pool rebuild 155
    increased storage requirements 56
    latch contention 7
    MAXCSA parameter 59
    reduced IRLM latch contention 165, 167
    SRB time 166
    Sysplex query parallelism 56, 59
  IRWW
    dynamic rebuild 160
    external throughput rate for dynamic rebuild 161
    external throughput rate for manual rebuild 160
    manual rebuild under IRWW workload 160
    measurement environment 160
    modified workload for affinity routing 165
  isolation
    cursor stability 54
    read stability 71

K
  KEEPALIVEOPTIONS parameter 139
  KEEP_DYNAMIC(NO) 24, 29
  KEEP_DYNAMIC(YES)
    effect on storage 27
    recommendations for use 27
    use with REOPT(VARS) 29
    with prepared statement cache 25
    without prepared statement cache 24
  key column correlation statistics
    considerations for RUNSTATS 106
    cost of RUNSTATS to collect statistics 105
    definition 39
    how queries can benefit 40
    how RUNSTATS collects statistics 100
    how to influence access path selection 41
    index screening 41
    limitations of RUNSTATS 105
    matching predicates 40
    performance measurement 44
    table joins 41
    where DB2 stores statistics 103

L
  large partitioned tables 8
  LOAD
    inline image copy 81
    performance enhancements 9, 73
    PREFORMAT 74
    SORTKEYS 75
  LOG phase of online REORG 109
  Lotus Approach 11, 127, 128
  LOW2KEY 91, 93

M
  mapping table 113, 114, 117
  MAXCSA parameter 59
  mixed character set 149
  multiple query result sets
    accessing query result sets 128
    measurements 128
    returning query result sets 127
    use of temporary tables 128

N
  NODEFER(PREPARE) bind option 29, 33
  noncolumn expressions
    impact with REOPT(VARS) 49
    optimizer enhancements 48
  nonpartitioning indexes 10, 45
nonuniform distribution statistics  49

O

ODBC  13, 128, 144, 146
online REORG
  how to execute  114
  locks in effect in different phases  110
  mapping table  113
  measurements with concurrent batch activity  116
  measurements with no concurrent activity  115
  measurements with concurrent online activity  118
  performance enhancements  106
  phases of the utility  109
  shadow data sets  107
  SHRLEVEL  106
Open Database Connectivity
  See ODBC
OpenEdition  138
OPTIMIZE for n rows  12
optimizer enhancements
  enabling parallel access  51
  noncolumn expressions  48
  nonuniform distribution statistics  49
  scalar functions  49

P

P-lock negotiation  165, 166, 167
package authorization  12
parallelism enhancements  53
parameterized SQL  18
partition scan
  enhancements  46
  impact with REOPT(VARS)  47
partitioning for optimal performance  56, 62
performance
  access path reoptimization  8
  authorization caching  10
  BSAM stripe  9
  CASE expressions  8
  client/server  11, 125
  correlated index columns  8
  data sharing enhancements  6, 155
  DDL concurrency  10
  deferred close by partition  10
  dynamic SQL statement caching  7
  expectations  3
  keeping update locks  10
  LOAD with inline copy  9
  LOAD with SORTKEYS  9
  measurements  2
  multiple service tasks  10
  nonpartitioning index pieces  45
  nonpartitioning indexes  10
  preformatting active log  10
  preformatting of data  9
  query parallelism  53
  RECOVER index  9

performance (continued)
  REORG with inline copy  9
  REORG with SORTKEYS  9
  RUNSTATS with sampling  9
  SQL performance  17
  Sysplex query parallelism  5
  use of indexes  10
  user productivity enhancements  14
  utilities  9
  pieces for nonpartitioning indexes  45
  PowerBuilder  2, 11, 127, 128
  PREFORMAT  74
  preformatting of data  9
  PREPARE statement
    caching dynamic SQL  17
    how to keep the prepared statement past commit  24
    keeping the prepared statement past commit  7
    why keep the prepared statement past commit  24
  prepared statement caching  19
  prepared statement retrieval  19
  productivity enhancements
    DB2 installer  14
    image copy  14
    OS/390 server solution  15
    RENAME TABLE statement  14
    Visual Explain  14
  program temporary fix
    See PTF
  PTF  164

Q

QMF
  dynamic statement caching  18
  use of EXPLAIN  31
query management facility
  See QMF
query parallelism
  enhancements  53
  Sysplex  54

R

RACF
  control end user’s access to non-SQL resources  2, 11
  enabling DRDA to change password function  137
  IBM-defined SERVER resource class  131
  PassTickets  13
  profiles  13
  support with CAF  130
  REBUILDPERCENT  156, 157
  RECOVER INDEX enhancements  84
  reducing data sharing costs
    conditional page P-locks  166
    default change for the GCLSN  167
    hashing algorithm change to reduce false contention  167
reducing data sharing costs (continued)
reduce IRLM latch contention from P-lock
unlock 167
space map page update and member-private space
map pages 168
specifying maximum number of rows in a
page 166
tracking dynamic inter-DB2 interest 167
RELOAD phase of online REORG 109
RENAME TABLE statement 14
REORG
inline image copy 81
performance enhancements 9, 73
PREFORMAT 74
SORTKEYS 75
Resource Access Control Facility
See RACF
resource limit specification table
See RLST
Resource Recovery Services Attachment Facility
See RRSAF
RLST
control dynamic SQL queries 63
scope of resource limits 63
setting limits 63
RRSAF 12, 131, 144
RUNSTATS
key column correlation statistics 100
measurements 93
need for RUNSTATS 90
sampling 91
S
sampling 9
SBCS 149
SCA 58
scalar functions 49
selective partition locking 7, 164, 165
SFM
coupled data sets 155
policy 156, 157
shadow data sets
dB2-managed objects 107
naming convention 108
user-managed objects 108
shared communications area
See SCA
shared data 1
shared disk 57
SHRLEVEL 106
single byte character set
See SBCS
SNA 12, 136, 140
SORT phase of online REORG 109
SORTKEYS
advantages 77
considerations 78
measurements for LOAD 78, 81
SORTKEYS (continued)
measurements for REORG 79
processing 76, 77
SPUFI 18, 28, 31
SQL performance
caching dynamic SQL statements 17
enhanced limited partition scan 46
keeping prepared statements past commit 23
nonpartitioning index pieces 45
optimizer enhancements 48
SQL reoptimization 28
SQL reoptimization
dynamic SQL statements 30
how to monitor reoptimization 32
how to reoptimize 29
how to use EXPLAIN 30
key column correlation statistics 39
performance measurements 35
static SQL statements 29
why required 28
SQLRULES bind option 20
stored procedures
catalog changes 125
commit on return 129
improved environment 11
multiple DB2 address spaces 11
multiple query result sets 11, 126
using MVS workload manager 130
Sun Solaris 11, 136
SWITCH phase of online REORG 109
SYS1.PARMLIB 70
Sysplex coupling data set configuration recommendations 168
Sysplex failure management
See SFM
Sysplex query parallelism
configuring for 56
dedicated query processing configuration 65
introduction 54
IRLM storage requirements 59
number of online CPs 58
partitioning for optimal performance 62
performance measurements 65
recommendations 70
RLST 63
separate query processing configuration 64
setting buffer pool allocation thresholds 57
setting MVS workload management goals 59
work files on shared disk 57
system parameter
See ZPARM
Systems Network Architecture
See SNA
Index 191
TCP/IP
communications database changes 139
considerations 139
DDF and MVS OpenEdition 138
defining TCP/IP host names 139
installation dialog parameters 136
installing 136
networks 11
performance 140
register DDF port numbers 139
Type 1 index
LOAD utility 78
REORG utility 79
Type 2 index
LOAD utility 79
LOAD utility with inline copy 83
online REORG on a nonpartitioning index 116
online REORG on a partitioning index 115
online REORG on a partitioning index with concurrent OLTP 120
online REORG on a table space 115
online REORG on a table space partition with concurrent OLTP 118
online REORG on a table space with concurrent batch updates 116, 118
RECOVER INDEX utility with SYSUT1 84
RECOVER INDEX utility without SYSUT1 85
REORG utility 80
REORG utility with inline copy 83

UNIX 13, 136, 180
UNLOAD phase of online REORG 109
update locks 10
user-managed data set 74, 108
UTILINIT phase of online REORG 109
utility performance 73
UTILTERM phase of online REORG 110

VALIDATE(BIND) bind option 31, 32
VALIDATE(RUN) bind option 28, 29, 31, 34, 35
virtual storage access method
See VSAM
virtual telecommunications access method
See VTAM
Visual Explain 14
VisualAge 2, 11, 13, 127, 128
VisualBasic 2, 11, 127, 128
VPXPSEQT 58, 64
VSAM 11, 73, 108
VTAM
enhancements introduced in V4 R4 143
send and receive operations 12, 142

Windows

3.11 136
95 2, 180
NT 136

WLM
classification rules example 61, 62
compatibility mode 59, 60, 70
discretionary goals 60
example of multiple performance period goals 60
execution velocity goals 59
goal mode 59, 62, 70
measurements 134
multiple stored procedure address spaces 133
MVS enclave support 132
response time goals 59
setting goals 59
stored procedures as subprograms 133
stored procedures environment 130
tuning and operating considerations 134
workload manager
See WLM

X
X/Open 13, 180

Z
ZPARM
CACHEDYN 20
CACHEPAC 12
INBUFF 161
MAXKEEPD 27
OUTBUFF 161
PCLOSEN 167
PCLOSESET 167
ITSO Redbook Evaluation

DB2 for OS/390 Version 5 Performance Topics
SG24-2213-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. Please complete this questionnaire and return it using one of the following methods:

• Use the online evaluation form found at http://www.redbooks.com
• Fax this form to: USA International Access Code + 1 914 432 8264
• Send your comments in an Internet note to redeval@vnet.ibm.com

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _______________________

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____
If no, please explain:
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________

What other redbooks would you like to see published?
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________
_____________________________________________________________________________________________________

© Copyright IBM Corp. 1997

193
### Artwork Definitions

<table>
<thead>
<tr>
<th>id</th>
<th>File</th>
<th>Page</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITLOGO</td>
<td>2213SU</td>
<td>i</td>
<td>i</td>
</tr>
<tr>
<td>ITLOGOS</td>
<td>2213SU</td>
<td>i</td>
<td>i</td>
</tr>
</tbody>
</table>

### Table Definitions

<table>
<thead>
<tr>
<th>id</th>
<th>File</th>
<th>Page</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>DYNST1</td>
<td>2213CH3</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>DYNST2</td>
<td>2213CH3</td>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>PLANEX</td>
<td>2213CH3</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>STMT</td>
<td>2213CH3</td>
<td>33</td>
<td>34</td>
</tr>
<tr>
<td>TB43A</td>
<td>2213CH3</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>TB43B</td>
<td>2213CH3</td>
<td>36</td>
<td>37</td>
</tr>
<tr>
<td>TB43X</td>
<td>2213CH3</td>
<td>37</td>
<td>37</td>
</tr>
<tr>
<td>TB43C</td>
<td>2213CH3</td>
<td>37</td>
<td>38</td>
</tr>
<tr>
<td>TB43D</td>
<td>2213CH3</td>
<td>38</td>
<td>38</td>
</tr>
<tr>
<td>TB414A</td>
<td>2213CH3</td>
<td>42</td>
<td>43</td>
</tr>
<tr>
<td>TBCORR2</td>
<td>2213CH3</td>
<td>44</td>
<td>44</td>
</tr>
<tr>
<td>TB41705</td>
<td>2213CH3</td>
<td>50</td>
<td>51</td>
</tr>
<tr>
<td>TB6105</td>
<td>2213CH5</td>
<td>78</td>
<td>78</td>
</tr>
<tr>
<td>TB6107</td>
<td>2213CH5</td>
<td>79</td>
<td>79</td>
</tr>
<tr>
<td>TB6110</td>
<td>2213CH5</td>
<td>79</td>
<td>79</td>
</tr>
<tr>
<td>TB6111</td>
<td>2213CH5</td>
<td>79</td>
<td>80</td>
</tr>
<tr>
<td>TB6115</td>
<td>2213CH5</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>TB6116</td>
<td>2213CH5</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>TB6120</td>
<td>2213CH5</td>
<td>83</td>
<td>83</td>
</tr>
<tr>
<td>TB6205</td>
<td>2213CH5</td>
<td>84</td>
<td>84</td>
</tr>
<tr>
<td>TB6210</td>
<td>2213CH5</td>
<td>85</td>
<td>85</td>
</tr>
<tr>
<td>TB6215</td>
<td>2213CH5</td>
<td>85</td>
<td>85</td>
</tr>
<tr>
<td>FATTAB</td>
<td>2213CH5</td>
<td>91</td>
<td>91</td>
</tr>
<tr>
<td>TB641</td>
<td>2213CH5</td>
<td>94</td>
<td>94</td>
</tr>
<tr>
<td>TB642</td>
<td>2213CH5</td>
<td>94</td>
<td>95</td>
</tr>
<tr>
<td>TB649</td>
<td>2213CH5</td>
<td>95</td>
<td>95</td>
</tr>
<tr>
<td>TB643</td>
<td>2213CH5</td>
<td>95</td>
<td>95</td>
</tr>
<tr>
<td>TB644</td>
<td>2213CH5</td>
<td>96</td>
<td>96</td>
</tr>
<tr>
<td>TB645</td>
<td>2213CH5</td>
<td>96</td>
<td>97</td>
</tr>
<tr>
<td>TB646</td>
<td>2213CH5</td>
<td>97</td>
<td>97</td>
</tr>
<tr>
<td>TB647</td>
<td>2213CH5</td>
<td>98</td>
<td>98</td>
</tr>
<tr>
<td>TB64A</td>
<td>2213CH5</td>
<td>98</td>
<td>99</td>
</tr>
<tr>
<td>TB64B</td>
<td>2213CH5</td>
<td>103</td>
<td>104</td>
</tr>
<tr>
<td>THD</td>
<td>2213CH5</td>
<td>105</td>
<td>105</td>
</tr>
<tr>
<td>TB6605A</td>
<td>2213CH5</td>
<td>105</td>
<td>105</td>
</tr>
<tr>
<td>id</td>
<td>File</td>
<td>Page</td>
<td>References</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>------</td>
<td>------------</td>
</tr>
<tr>
<td>4611XMP</td>
<td>2213VARS</td>
<td>i</td>
<td></td>
</tr>
</tbody>
</table>

### Figures

<table>
<thead>
<tr>
<th>id</th>
<th>File</th>
<th>Page</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>4501</td>
<td>2213CH3</td>
<td>45</td>
<td>1, 45</td>
</tr>
<tr>
<td>4502</td>
<td>2213CH3</td>
<td>46</td>
<td>2, 45</td>
</tr>
<tr>
<td>5201</td>
<td>2213CH4</td>
<td>55</td>
<td>3, 55, 55</td>
</tr>
<tr>
<td>5202</td>
<td>2213CH4</td>
<td>56</td>
<td>4, 55</td>
</tr>
<tr>
<td>S5201</td>
<td>2213CH4</td>
<td>57</td>
<td>5, 56, 55</td>
</tr>
<tr>
<td>5203</td>
<td>2213CH4</td>
<td>58</td>
<td>6, 57</td>
</tr>
<tr>
<td>VEL1</td>
<td>2213CH4</td>
<td>60</td>
<td>7, 60</td>
</tr>
<tr>
<td>WLMAC1</td>
<td>2213CH4</td>
<td>61</td>
<td>8, 61, 62</td>
</tr>
<tr>
<td>WLMAC2</td>
<td>2213CH4</td>
<td>62</td>
<td>9, 61, 62</td>
</tr>
<tr>
<td>RSLTPG</td>
<td>2213CH4</td>
<td>63</td>
<td>10</td>
</tr>
<tr>
<td>Reference</td>
<td>Field</td>
<td>Page(s)</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>CA01PG</td>
<td>2213CH4</td>
<td>64, 64</td>
<td></td>
</tr>
<tr>
<td>CA02PG</td>
<td>2213CH4</td>
<td>65, 65</td>
<td></td>
</tr>
<tr>
<td>XQ17F</td>
<td>2213CH4</td>
<td>66, 66, 67</td>
<td></td>
</tr>
<tr>
<td>XQ18F</td>
<td>2213CH4</td>
<td>67, 66, 67</td>
<td></td>
</tr>
<tr>
<td>XQ13F</td>
<td>2213CH4</td>
<td>67, 67</td>
<td></td>
</tr>
<tr>
<td>XQ14F</td>
<td>2213CH4</td>
<td>68, 67</td>
<td></td>
</tr>
<tr>
<td>XQ15F</td>
<td>2213CH4</td>
<td>69, 67</td>
<td></td>
</tr>
<tr>
<td>XQ16F</td>
<td>2213CH4</td>
<td>69, 69</td>
<td></td>
</tr>
<tr>
<td>PS605</td>
<td>2213CH5</td>
<td>76, 76</td>
<td></td>
</tr>
<tr>
<td>PS610</td>
<td>2213CH5</td>
<td>77, 76</td>
<td></td>
</tr>
<tr>
<td>PS6605</td>
<td>2213CH5</td>
<td>111, 110</td>
<td></td>
</tr>
<tr>
<td>PS6610</td>
<td>2213CH5</td>
<td>111, 110</td>
<td></td>
</tr>
<tr>
<td>PS6615</td>
<td>2213CH5</td>
<td>112, 110</td>
<td></td>
</tr>
<tr>
<td>PS6620</td>
<td>2213CH5</td>
<td>113, 110</td>
<td></td>
</tr>
<tr>
<td>FIG71A</td>
<td>2213CH6</td>
<td>127, 126</td>
<td></td>
</tr>
<tr>
<td>FIG71B</td>
<td>2213CH6</td>
<td>132, 131</td>
<td></td>
</tr>
<tr>
<td>7201</td>
<td>2213CH6</td>
<td>137, 136</td>
<td></td>
</tr>
<tr>
<td>7202</td>
<td>2213CH6</td>
<td>138, 137</td>
<td></td>
</tr>
<tr>
<td>FG73105</td>
<td>2213CH6</td>
<td>141, 141, 141</td>
<td></td>
</tr>
<tr>
<td>FG73110</td>
<td>2213CH6</td>
<td>142, 142, 142</td>
<td></td>
</tr>
<tr>
<td>FG74105</td>
<td>2213CH6</td>
<td>145, 144, 145</td>
<td></td>
</tr>
<tr>
<td>2213AS1</td>
<td>2213CH6</td>
<td>151, 150, 151</td>
<td></td>
</tr>
<tr>
<td>2213AS2</td>
<td>2213CH6</td>
<td>152, 152</td>
<td></td>
</tr>
<tr>
<td>2213AS3</td>
<td>2213CH6</td>
<td>153, 153</td>
<td></td>
</tr>
<tr>
<td>CFINT</td>
<td>2213CH7</td>
<td>156, 155, 157</td>
<td></td>
</tr>
<tr>
<td>DS02</td>
<td>2213CH7</td>
<td>160, 158, 159</td>
<td></td>
</tr>
<tr>
<td>2213GBP</td>
<td>2213CH7</td>
<td>162, 162</td>
<td></td>
</tr>
<tr>
<td>8105</td>
<td>2213CH7</td>
<td>164, 164</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>File</td>
<td>Page</td>
<td>References</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>INTRO</td>
<td>2213CH1</td>
<td>1</td>
<td>Chapter 1, Introduction</td>
</tr>
<tr>
<td>CHP2</td>
<td>2213CH2</td>
<td>5</td>
<td>Chapter 2, Overview of DB2 for OS/390 Version 5 Enhancements</td>
</tr>
<tr>
<td>MSTASK</td>
<td>2213CH2</td>
<td>10</td>
<td>2.1.10.4, Multiple Service Tasks</td>
</tr>
<tr>
<td>CHP4</td>
<td>2213CH3</td>
<td>17</td>
<td>Chapter 3, SQL Performance</td>
</tr>
<tr>
<td>CH41</td>
<td>2213CH3</td>
<td>17</td>
<td>3.1, Caching Dynamic SQL Statements</td>
</tr>
<tr>
<td>CH41RET</td>
<td>2213CH3</td>
<td>19</td>
<td>3.1.2.2, Criteria for Prepared Statement Retrieval</td>
</tr>
<tr>
<td>CH42</td>
<td>2213CH3</td>
<td>23</td>
<td>3.2, Keeping Prepared Statements Past Commit</td>
</tr>
<tr>
<td>CH413</td>
<td>2213CH3</td>
<td>28</td>
<td>3.3, Reoptimization at Run Time</td>
</tr>
<tr>
<td>CH44</td>
<td>2213CH3</td>
<td>39</td>
<td>3.4, Use of Key Column Correlation Statistics</td>
</tr>
<tr>
<td>CH415</td>
<td>2213CH3</td>
<td>45</td>
<td>3.5, Nonpartitioning Index Pieces</td>
</tr>
<tr>
<td>CH416</td>
<td>2213CH3</td>
<td>46</td>
<td>3.6, Enhanced Limited Partition Scan</td>
</tr>
<tr>
<td>CH417</td>
<td>2213CH3</td>
<td>48</td>
<td>3.7, Optimizer Enhancements</td>
</tr>
<tr>
<td>4172</td>
<td>2213CH3</td>
<td>48</td>
<td>3.7.1, Noncolumn Expressions</td>
</tr>
<tr>
<td>4171</td>
<td>2213CH3</td>
<td>49</td>
<td>3.7.2, Scalar Functions</td>
</tr>
<tr>
<td>QUERY</td>
<td>2213CH4</td>
<td>53</td>
<td>Chapter 4, Query Parallelism</td>
</tr>
<tr>
<td>PLE</td>
<td>2213CH4</td>
<td>53</td>
<td>4.1, Parallelism Enhancements</td>
</tr>
<tr>
<td>XQCH</td>
<td>2213CH4</td>
<td>54</td>
<td>4.2, Sysplex Query Parallelism</td>
</tr>
<tr>
<td>XQINT</td>
<td>2213CH4</td>
<td>54</td>
<td>4.2.1, Introduction</td>
</tr>
<tr>
<td>XQCONF</td>
<td>2213CH4</td>
<td>56</td>
<td>4.2.2, Configuring for Sysplex Query Parallelism</td>
</tr>
<tr>
<td>POPP</td>
<td>2213CH4</td>
<td>62</td>
<td>4.2.2.7, Partitioning for Optimal Parallel Performance</td>
</tr>
<tr>
<td>XQCA</td>
<td>2213CH4</td>
<td>64</td>
<td>4.2.3, Data Sharing Coordinator and Assistant Configurations</td>
</tr>
<tr>
<td>XQPS</td>
<td>2213CH4</td>
<td>65</td>
<td>4.2.4, Performance Measurements</td>
</tr>
<tr>
<td>XQMR</td>
<td>2213CH4</td>
<td>66</td>
<td>4.2.4.2, Measurement Results</td>
</tr>
<tr>
<td>XQCONC</td>
<td>2213CH4</td>
<td>70</td>
<td>4.2.5, Conclusions and Recommendations</td>
</tr>
<tr>
<td>CH61</td>
<td>2213CH5</td>
<td>73</td>
<td>5.1, LOAD and REORG</td>
</tr>
<tr>
<td>CH611</td>
<td>2213CH5</td>
<td>74</td>
<td>5.1.1, PREFORMAT</td>
</tr>
<tr>
<td>CH62</td>
<td>2213CH5</td>
<td>84</td>
<td>5.2, RECOVER INDEX</td>
</tr>
<tr>
<td>CH63</td>
<td>2213CH5</td>
<td>85</td>
<td>5.3, COPY</td>
</tr>
<tr>
<td>CH6305</td>
<td>2213CH5</td>
<td>88</td>
<td>5.3.4, How to Automate the Image Copy Process</td>
</tr>
<tr>
<td>CH64</td>
<td>2213CH5</td>
<td>90</td>
<td>5.4, RUNSTATS</td>
</tr>
<tr>
<td>H64NOR</td>
<td>2213CH5</td>
<td>98</td>
<td>5.4.4.4, Comparison of No RUNSTATS and 1% Sampling</td>
</tr>
<tr>
<td>id</td>
<td>File</td>
<td>Page</td>
<td>References</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>------</td>
<td>------------</td>
</tr>
<tr>
<td>DYNST1</td>
<td>2213CH3</td>
<td>22</td>
<td>1</td>
</tr>
<tr>
<td>DYNST2</td>
<td>2213CH3</td>
<td>23</td>
<td>2</td>
</tr>
<tr>
<td>PLANEX</td>
<td>2213CH3</td>
<td>32</td>
<td>3</td>
</tr>
<tr>
<td>STMT</td>
<td>2213CH3</td>
<td>34</td>
<td>4</td>
</tr>
<tr>
<td>TB43A</td>
<td>2213CH3</td>
<td>36</td>
<td>5</td>
</tr>
<tr>
<td>TB43B</td>
<td>2213CH3</td>
<td>37</td>
<td>6</td>
</tr>
<tr>
<td>TB43X</td>
<td>2213CH3</td>
<td>37</td>
<td>7</td>
</tr>
<tr>
<td>TB43C</td>
<td>2213CH3</td>
<td>38</td>
<td>8</td>
</tr>
<tr>
<td>TB43D</td>
<td>2213CH3</td>
<td>38</td>
<td>9</td>
</tr>
<tr>
<td>TB414A</td>
<td>2213CH3</td>
<td>43</td>
<td>10</td>
</tr>
<tr>
<td>TBCORR2</td>
<td>2213CH3</td>
<td>44</td>
<td>11</td>
</tr>
<tr>
<td>TB41705</td>
<td>2213CH3</td>
<td>51</td>
<td>12</td>
</tr>
<tr>
<td>TB6105</td>
<td>2213CH5</td>
<td>78</td>
<td>13</td>
</tr>
<tr>
<td>TB6107</td>
<td>2213CH5</td>
<td>79</td>
<td>14</td>
</tr>
<tr>
<td>TB6110</td>
<td>2213CH5</td>
<td>79</td>
<td>15</td>
</tr>
<tr>
<td>TB6111</td>
<td>2213CH5</td>
<td>80</td>
<td>16</td>
</tr>
<tr>
<td>TB6115</td>
<td>2213CH5</td>
<td>80</td>
<td>17</td>
</tr>
<tr>
<td>TB6116</td>
<td>2213CH5</td>
<td>80</td>
<td>18</td>
</tr>
<tr>
<td>TB6120</td>
<td>2213CH5</td>
<td>83</td>
<td>19</td>
</tr>
<tr>
<td>TB6205</td>
<td>2213CH5</td>
<td>84</td>
<td>20</td>
</tr>
<tr>
<td>TB6210</td>
<td>2213CH5</td>
<td>85</td>
<td>21</td>
</tr>
<tr>
<td>TB6215</td>
<td>2213CH5</td>
<td>85</td>
<td>22</td>
</tr>
<tr>
<td>FATTAB</td>
<td>2213CH5</td>
<td>91</td>
<td>23</td>
</tr>
<tr>
<td>TB641</td>
<td>2213CH5</td>
<td>94</td>
<td>24</td>
</tr>
<tr>
<td>TB642</td>
<td>2213CH5</td>
<td>95</td>
<td>25</td>
</tr>
<tr>
<td>TB649</td>
<td>2213CH5</td>
<td>95</td>
<td>26</td>
</tr>
<tr>
<td>TB643</td>
<td>2213CH5</td>
<td>96</td>
<td>27</td>
</tr>
<tr>
<td>TB644</td>
<td>2213CH5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code</td>
<td>Page Range</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>TB645</td>
<td>2213CH5</td>
<td>97-28, 96, 96</td>
<td></td>
</tr>
<tr>
<td>TB646</td>
<td>2213CH5</td>
<td>97-29, 96, 97</td>
<td></td>
</tr>
<tr>
<td>TB647</td>
<td>2213CH5</td>
<td>98-30, 98</td>
<td></td>
</tr>
<tr>
<td>TB64A</td>
<td>2213CH5</td>
<td>99-31, 98, 99</td>
<td></td>
</tr>
<tr>
<td>TB64B</td>
<td>2213CH5</td>
<td>104-32, 103</td>
<td></td>
</tr>
<tr>
<td>TB6605A</td>
<td>2213CH5</td>
<td>115-34, 115</td>
<td></td>
</tr>
<tr>
<td>TB6605B</td>
<td>2213CH5</td>
<td>115-35, 115</td>
<td></td>
</tr>
<tr>
<td>TB6610</td>
<td>2213CH5</td>
<td>116-36, 115, 115</td>
<td></td>
</tr>
<tr>
<td>TB6615</td>
<td>2213CH5</td>
<td>116-37, 115, 115, 116</td>
<td></td>
</tr>
<tr>
<td>TB662A</td>
<td>2213CH5</td>
<td>116-38, 115, 116, 116</td>
<td></td>
</tr>
<tr>
<td>TB662B</td>
<td>2213CH5</td>
<td>118-39, 116, 117, 118</td>
<td></td>
</tr>
<tr>
<td>TB6625</td>
<td>2213CH5</td>
<td>119-40, 118, 119, 120, 122</td>
<td></td>
</tr>
<tr>
<td>TB6630</td>
<td>2213CH5</td>
<td>120-41, 120, 120</td>
<td></td>
</tr>
<tr>
<td>TB6635</td>
<td>2213CH5</td>
<td>121-42, 121</td>
<td></td>
</tr>
<tr>
<td>TB71A</td>
<td>2213CH6</td>
<td>126-43, 125</td>
<td></td>
</tr>
<tr>
<td>TB71C</td>
<td>2213CH6</td>
<td>129-44, 128</td>
<td></td>
</tr>
<tr>
<td>TB71B</td>
<td>2213CH6</td>
<td>130-45, 130</td>
<td></td>
</tr>
<tr>
<td>CH71G</td>
<td>2213CH6</td>
<td>131-46, 130</td>
<td></td>
</tr>
<tr>
<td>TB71E</td>
<td>2213CH6</td>
<td>135-47, 134</td>
<td></td>
</tr>
<tr>
<td>TCPTAB</td>
<td>2213CH6</td>
<td>140-48, 140, 140</td>
<td></td>
</tr>
<tr>
<td>7421</td>
<td>2213CH6</td>
<td>151-49, 150, 150, 150, 150, 150, 151</td>
<td></td>
</tr>
<tr>
<td>7422</td>
<td>2213CH6</td>
<td>152-50, 150, 150, 152, 152</td>
<td></td>
</tr>
<tr>
<td>7423</td>
<td>2213CH6</td>
<td>153-51, 150, 150, 153, 153</td>
<td></td>
</tr>
<tr>
<td>CH85TAB</td>
<td>2213CH7</td>
<td>159-52, 158, 158</td>
<td></td>
</tr>
<tr>
<td>CH89TAB</td>
<td>2213CH7</td>
<td>159-53, 158, 159</td>
<td></td>
</tr>
<tr>
<td>CH8YTAB</td>
<td>2213CH7</td>
<td>160-54, 160</td>
<td></td>
</tr>
<tr>
<td>CH8ZTAB</td>
<td>2213CH7</td>
<td>161-55, 161</td>
<td></td>
</tr>
<tr>
<td>CH8RT1</td>
<td>2213CH7</td>
<td>162-56, 162</td>
<td></td>
</tr>
<tr>
<td>CH8RT2</td>
<td>2213CH7</td>
<td>162</td>
<td></td>
</tr>
</tbody>
</table>
Processing Options

Runtime values:
Document fileid ......................................................... SG242213 SCRIPT
Document type ............................................................ USERDOC
Document style ......................................................... REDBOOK
Profile ................................................................. EDFPRF30
Service Level ......................................................... 0029
SCRIPT/VS Release ............................................... 4.0.0
Date ........................................................................ 97.07.02
Time ........................................................................ 19:17:44
Device ....................................................................... 3820A
Number of Passes .................................................. 4
Index ........................................................................ YES
SYVAR D ..................................................................... YES
SYVAR G .................................................................. INLINE
SYVAR S ..................................................................... OFF

Formatting values used:
Annotation ................................................................... NO
Cross reference listing .............................................. YES
Cross reference head prefix only .............................. NO
Dialog ....................................................................... LABEL
Duplex ....................................................................... YES
DVCF conditions file ............................................... (none)
DVCF value 1 ........................................................... (none)
DVCF value 2 ........................................................... (none)
DVCF value 3 ........................................................... (none)
DVCF value 4 ........................................................... (none)
DVCF value 5 ........................................................... (none)
DVCF value 6 ........................................................... (none)
DVCF value 7 ........................................................... (none)
DVCF value 8 ........................................................... (none)
DVCF value 9 ........................................................... (none)
Explode ....................................................................... NO
Figure list on new page ............................................ YES
Figure/table number separation ............................. YES
Folio-by-chapter separation ..................................... NO
Head 0 body text ..................................................... Part
Head 1 body text ..................................................... Chapter
Head 1 appendix text .............................................. Appendix
Hyphenation ........................................................... NO
Justification ........................................................... NO
Language ............................................................. ENGL
Layout ....................................................................... OFF
Leader dots ........................................................... YES
Master index .......................................................... (none)
Partial TOC (maximum level) ................................. 4
Partial TOC (new page after) ................................. INLINE
Print example id’s ................................................... NO
Print cross reference page numbers ........................ YES
Process value ......................................................... (none)
Punctuation move characters .................................... .
Read cross-reference file ........................................ (none)
Running heading/footing rule ................................. NONE
Show index entries ................................................ NO
Table of Contents (maximum level) ......................... 3
Table list on new page ............................................. YES
Title page (draft) alignment ................................... RIGHT
Write cross-reference file ....................................... (none)
<table>
<thead>
<tr>
<th>Page</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2213SU</td>
</tr>
<tr>
<td>0</td>
<td>2213VARS</td>
</tr>
<tr>
<td>i</td>
<td>REDB$ED1</td>
</tr>
<tr>
<td>i</td>
<td>2213EDNO</td>
</tr>
<tr>
<td>i</td>
<td>REDB$ED2</td>
</tr>
<tr>
<td>xi</td>
<td>2213ABST</td>
</tr>
<tr>
<td>xi</td>
<td>2213ACKS</td>
</tr>
<tr>
<td>xii</td>
<td>REDB$COM</td>
</tr>
<tr>
<td>xii</td>
<td>2213IMBD</td>
</tr>
<tr>
<td>3</td>
<td>2213CH1</td>
</tr>
<tr>
<td>3</td>
<td>2213CH2</td>
</tr>
<tr>
<td>15</td>
<td>2213CH3</td>
</tr>
<tr>
<td>52</td>
<td>2213CH4</td>
</tr>
<tr>
<td>71</td>
<td>2213CH5</td>
</tr>
<tr>
<td>123</td>
<td>2213CH6</td>
</tr>
<tr>
<td>153</td>
<td>2213CH7</td>
</tr>
<tr>
<td>168</td>
<td>2213APP</td>
</tr>
<tr>
<td>179</td>
<td>2213SPEC</td>
</tr>
<tr>
<td>179</td>
<td>REDB$SPE</td>
</tr>
<tr>
<td>180</td>
<td>2213BIBL</td>
</tr>
<tr>
<td>181</td>
<td>REDB$BIB</td>
</tr>
<tr>
<td>182</td>
<td>REDB$ORD</td>
</tr>
<tr>
<td>192</td>
<td>REDB$EVA</td>
</tr>
</tbody>
</table>