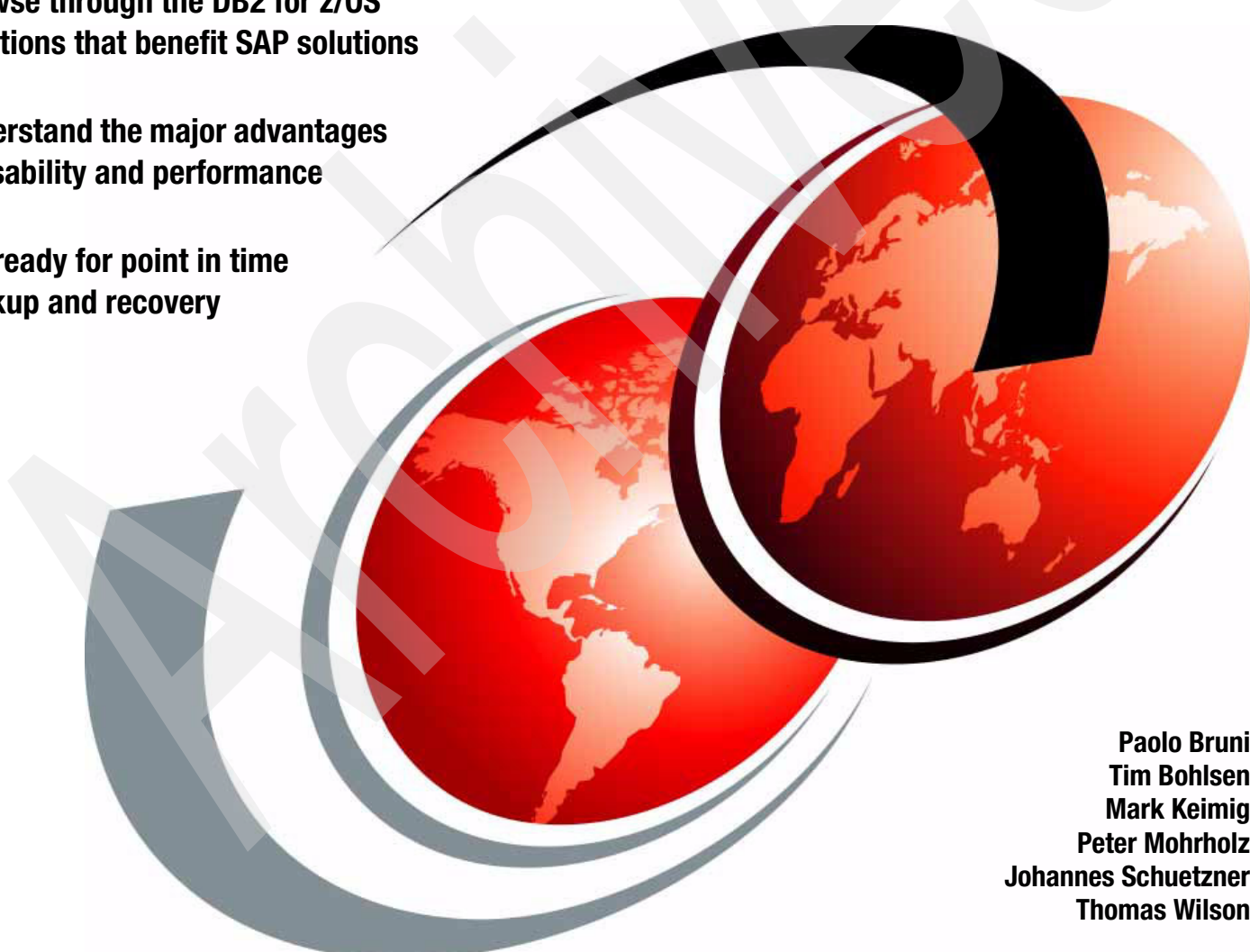


DB2 UDB for z/OS V8: Through the Looking Glass and What SAP Found There

Browse through the DB2 for z/OS
functions that benefit SAP solutions

Understand the major advantages
in usability and performance

Get ready for point in time
backup and recovery



Paolo Bruni
Tim Bohlsen
Mark Keimig
Peter Mohrholz
Johannes Schuetzner
Thomas Wilson

Redbooks



International Technical Support Organization

**DB2 UDB for z/OS V8:
Through the Looking Glass
and What SAP Found There**

December 2003

Archived

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

First Edition (December 2003)

This edition applies to SAP NetWeaver and SAP Web 6.40, the new technological release of SAP products for use with IBM DB2 Universal Database for z/OS Version 8 (program number 5625-DB2), and DB2 Utilities Suite for z/OS Version 8 (program number 5655-K63). Some considerations also apply to the current SAP Basis Release 4.6C, 4.6D, 6.10, and 6.20, as well as all SAP products based on these SAP Basis Releases.

Note: This book is based on a pre-GA version of a product and may not apply when the product becomes generally available. We recommend that you consult the product documentation or follow-on versions of this redbook for more current information.

© Copyright International Business Machines Corporation 2003. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures	vii
Tables	ix
Examples	xi
Notices	xiii
Trademarks	xiv
Summary of changes	xv
December 2003, First Edition	xv
September 2004, First Update	xv
Preface	xvii
The team that wrote this redbook	xvii
Become a published author	xix
Comments welcome	xx
Chapter 1. Introduction	1
1.1 SAP and DB2 for z/OS partnership	2
1.1.1 IBM and SAP cooperation	2
1.2 Description of functions up to DB2 V7	3
1.2.1 SAP related enhancements in DB2 V5	3
1.2.2 SAP related enhancements in DB2 V6	5
1.2.3 SAP related enhancements in DB2 V7	9
1.3 DB2 UDB for z/OS Version 8 overview	13
1.3.1 Architecture	14
1.3.2 Usability, availability, and scalability	15
1.3.3 Business Information Warehouse	16
1.3.4 Performance	17
1.3.5 Tools and administration	18
1.3.6 System level point in time backup and recovery	19
1.3.7 DB2 V8 enhancements implicit for all SAP versions	19
Chapter 2. Architecture	21
2.1 Unicode	22
2.1.1 Introduction to Unicode	22
2.1.2 SAP view of Unicode	23
2.1.3 Increased length limits	23
2.1.4 Data type mapping	24
2.1.5 Performance and storage	26
2.2 DB2 Connect as SAP database connectivity	29
2.2.1 Introduction to SAP NetWeaver	29
2.2.2 Requirements of SAP NetWeaver	31
2.2.3 Necessary features of DB2 Connect	33
2.2.4 New DB2 features and DB2 Connect	33
2.2.5 New DB2 Connect features for ICL1 compatibility	33
2.2.6 Configuration of DB2 Connect and DDF for SAP	35
2.3 Schema evolution	39
2.3.1 Online schema changes	39

2.3.2	Online schema changes overview	39
2.3.3	Data type changes	41
2.3.4	Index changes	44
2.3.5	Versioning	45
2.3.6	SAP usage considerations	48
2.3.7	New DBET states for online schema changes	48
2.3.8	Impact of online schema changes on user tasks	48
2.3.9	Dynamic partitions	49
2.4	64-bit virtual storage	51
2.4.1	Expansion of DB2 virtual storage	51
2.4.2	Enlarged storage pools	52
2.4.3	Storage monitoring	59
2.4.4	Thread storage contraction	60
2.5	Java	60
2.5.1	SAP and Java	60
2.5.2	The SAP Java components	61
2.5.3	Java and DB2 data type mapping	62
2.5.4	Java dictionary	63
2.5.5	DB2 accounting and WLM	64
	Chapter 3. Usability, availability, and scalability	67
3.1	Partitioning	68
3.1.1	Increases to table sizes	68
3.1.2	Adding new partitions and rotating partitions	68
3.1.3	Data Partitioned Secondary Indexes	69
3.1.4	Separating clustering from partitioning	70
3.1.5	Example case of an SAP table	71
3.2	Index creation enhancements	72
3.2.1	Invalidation of dynamically cached statements	72
3.2.2	Deferred indexes do not prevent table access	74
3.3	Convert column type	75
3.3.1	Supported column type changes	75
3.3.2	Table ZMAKT100I	76
3.3.3	Index ZMAKT100I~Z01	77
3.3.4	Example 1: ALTER NUMB4 to INT4	78
3.3.5	Example 2: ALTER NUMB2 to INT4	84
3.3.6	Effect of ALTER of column in primary key	86
3.4	LOB ROWID transparency	87
3.4.1	SAP and LOBs	87
3.5	Multiple DISTINCT clauses in SQL statements	88
3.5.1	The SAP usage of DISTINCT	88
	Chapter 4. SAP Business Information Warehouse	91
4.1	Star join enhancements	92
4.1.1	Sparse index in outside-in join phase	93
4.1.2	In-memory workfiles	95
4.1.3	Sparse index during inside-out join phase	95
4.1.4	Snowflake handling: Controlled materialization	96
4.2	Joins with up to 225 tables	97
4.3	Common table expressions	98
4.3.1	Recursive SQL	99
4.4	Materialized query tables	100
4.4.1	Creating MQTs	101

4.4.2	Populating and maintaining an MQT	103
4.4.3	Automatic query rewrite using MQTs	104
4.4.4	Determining if query rewrite occurred	105
Chapter 5.	Performance	107
5.1	Early tests and general expectations	108
5.2	Lock issues	109
5.2.1	Release locks at close	109
5.2.2	Reduce lock contention on volatile tables	110
5.2.3	Partition key update	116
5.3	Multi-row fetch and insert	117
5.3.1	How SAP applications benefit from multi-row operations	117
5.3.2	Implementation details	117
5.3.3	ABAP SQL statements that benefit from multi-row operations	118
5.4	Query optimization	121
5.4.1	SAP and distribution statistics	121
5.4.2	REPORT NO UPDATE NONE option	124
5.4.3	Host variables impact on SAP access paths	125
5.4.4	Index-only access path for varying-length columns	129
5.4.5	Multiple value IN lists and SAP	133
5.5	Support for backward index scan	140
5.6	Faster DSC short prepares	142
5.7	Data sharing enhancements	143
5.7.1	SAP and data sharing	143
5.7.2	CF lock propagation reduction	144
5.7.3	CF request batching	151
5.7.4	Improved LPL recovery	153
Chapter 6.	Tools and administration	155
6.1	Automatic Space Management	156
6.1.1	Challenges of data management of an SAP system	156
6.1.2	Automatic space management functions	159
6.1.3	Scenarios and benefits of Automatic Space Management	161
6.2	DSC statement ID in Explain	163
6.2.1	Description of the enhancement	163
6.2.2	New parameters and plan table changes	163
6.3	Long-running non-committing readers	164
6.4	Lock escalation alert	165
6.4.1	DB2 lock management for SAP applications	165
6.4.2	IFCID 337 reports lock escalations	166
6.5	SAP and DB2 Control Center	166
6.5.1	Stored procedures and batch programs for database administration	167
6.5.2	Stored procedures for data set manipulation	168
6.5.3	Stored procedures for submitting JCL jobs and UNIX commands	169
6.6	SAP transaction-based DB2 accounting and workload management	170
6.6.1	Rollup accounting data for DDF and RRSAF	170
6.6.2	Writing accounting records with KEEP DYNAMIC(YES)	172
6.6.3	Other DDF and RRSAF enhancements	173
6.6.4	Automatic dropping of declared temporary tables	175
6.6.5	Opportunity for SAP	175
Chapter 7.	System point in time back-up and recovery	177
7.1	The need for PITR	178
7.1.1	SAP application fundamentals	178

7.1.2 Past, present, and future for backup and recovery	180
7.2 DFSMSHsm fast replication support	181
7.2.1 The Copy Pool construct.	181
7.2.2 Copy Pool backup storage group type	182
7.2.3 DFSMSHsm commands	183
7.2.4 Preparing for DFSMSHsm fast replication.	185
7.3 The PITR DB2 functions	189
7.3.1 Suspend database writes	189
7.3.2 CI size up to 32 KB	190
7.3.3 Forward log recovery	190
7.4 Scenario 1: Restoring to a prior backup	198
7.4.1 Case 1: Restore a system using BACKUP SYSTEM	198
7.4.2 Case 2: Restore a system using SET LOG SUSPEND	198
7.5 Scenario 2: Restore to a prior arbitrary point in time.	199
7.5.1 Case 1: Arbitrary PITR using the BACKUP and RESTORE SYSTEM	199
7.5.2 Case 2: Arbitrary PITR	200
Appendix A. DB2 V8 changes to installation defaults	201
A.1 New installation default values	202
A.1.1 New buffer pool threshold values	202
A.1.2 Changes to default parameter values.	202
Abbreviations and acronyms	205
Related publications	209
IBM Redbooks	209
Other publications	209
Online resources	210
How to get IBM Redbooks	210
Help from IBM	210
Index	211

Figures

1-1	The team effort in IBM and SAP labs.	2
1-2	SAP and zSeries synergy	3
2-1	Storage comparison of SAP systems with ASCII and Unicode data	27
2-2	SAP NetWeaver flow	30
2-3	SAP Web Application Server 6.40 on zSeries.	31
2-4	DB2 Connect: network statistics and database ping	34
2-5	Online schema evolution	40
2-6	ALTER TABLE SET DATATYPE statement	42
2-7	ALTER INDEX statement.	44
2-8	Versioning information in the DB2 catalog.	47
2-9	Evolution of z/OS memory architecture	52
2-10	Data spaces before DB2 V8	54
2-11	Buffer pools exploiting 64-bit virtual storage	55
2-12	DB2 V8 EDM pool	56
2-13	DB2 V8 RID pool	57
2-14	SAP Web Application Server.	61
2-15	Table definition in Java dictionary	63
3-1	Syntax of CREATE INDEX statement	70
3-2	Example of partitioning SAP table GLPCA by posting period	71
3-3	Invalidation of cached statements: Before index creation	73
3-4	Invalidation of cached statements: After index creation.	73
3-5	Index in Rebuild Pending state: Definition	74
3-6	Index in Rebuild Pending state: Not selected by DB2 optimizer	75
3-7	SAP DDIC display of ZMAKT100I table.	76
3-8	SAP DDIC display of secondary index on table ZMAKT100I.	77
3-9	SE11 Change of data element with warning after save.	78
3-10	SE11 Change of data element after continue	79
3-11	ZMAKT100I record display before activation.	80
3-12	ZMAKT100I activation log	81
3-13	ZMAKT100I record display after activation	82
3-14	DB2 status before and after REORG of table space	83
3-15	SE11 Change of data element with warning after save.	84
3-16	SE11 Change of data element after continue	85
3-17	ZMAKT100I activation log	86
3-18	DB2 Status after ALTER involving primary key	87
4-1	Snowflake schema.	92
4-2	Star join: index key feedback.	94
4-3	Star join: Sparse index.	95
4-4	Star join: Inside-out join phase.	96
4-5	Common table expression.	98
4-6	Nested table expression	98
4-7	SAP InfoCubes: External hierarchy	99
4-8	Recursive SQL.	100
4-9	CREATE MQT: Syntax	101
5-1	SAP definition of the physical table cluster DOKCLU	112
5-2	Definition of the logical cluster table DOKTL based on table cluster DOKCLU	113
5-3	Explain output before the ALTER statement	115
5-4	Explain output after the ALTER statement.	116

5-5	RUNSTATS Distribution Statistics	123
5-6	SAP transaction ST04: Invalidate SQL from DSC	125
5-7	Index-only access: Definition of table VBDATA	132
5-8	Index-only access path with VARCHAR column	133
5-9	STAT Record for ME21 transaction	135
5-10	Poor performance from KSSK select with INLIST	136
5-11	SQL Statement and Host Variables	136
5-12	Explain showing table scan	137
5-13	Index cardinalities of KSSK Indexes	137
5-14	Data sharing topology	144
5-15	ST04 database performance monitor: subsystem activity	145
5-16	Global locking	147
5-17	Data sharing: Inter-DB2 buffer pool coherency	151
6-1	Increasing secondary allocation with sliding scale	160
6-2	Example of small index scenario	162
6-3	DB2 V8 EXPLAIN STMTCACHE syntax diagram	163
6-4	SAP transaction DB2 showing long-running not-committing readers	165
6-5	Submitting JCL and UNIX commands through Control Center stored procedure	169
6-6	DB2 installation panel DSNTIPN	171
6-7	WLM classification rule exploiting SPM for DDF	173
7-1	Breakdown of major causes of recovery actions	179
7-2	Recovery options for different failure types	180
7-3	Copy Pool structure	183
7-4	Select option 'P', Copy Pool	185
7-5	Define the "database" Copy Pool	186
7-6	Define the source storage group to the "database" Copy Pool	186
7-7	Define the "log" Copy Pool	187
7-8	Define the source storage group to the "log" Copy Pool	187
7-9	Select the Storage Group	188
7-10	Alter by Storage Group Name	188
7-11	Associating source and target storage groups	189
7-12	BACKUP SYSTEM utility execution for DSNDB0G	191
7-13	Two BACKUP SYSTEM FULL, and one DATA ONLY are taken	192
7-14	BACKUP SYSTEM utility output	193
7-15	RESTORE SYSTEM utility output	195
7-16	Recovering to an arbitrary point in time using the RESTORE SYSTEM utility	195
7-17	DSNJU004 output	197

Tables

1-1	DB2 subsystem parameters of special interest to SAP system	11
1-2	DB2 V8 enhancements that are immediately available to all SAP releases	19
2-1	Mapping of ABAP character data types to DB2 data types (non-Unicode)	25
2-2	Mapping of ABAP character data types to DB2 data types (Unicode)	25
2-3	Fields in table ZPMTEST	26
2-4	DB group	27
2-5	Binary group	27
2-6	Char group	28
2-7	Java and DB2 data types	62
5-1	Table T contents	120
5-2	ABAP table IT contents	120
5-3	Results after modify	120
5-4	Mapping of IRLM locks to XES locks	149
6-1	New default values for DDF and RRSAP related ZPARAM	174
A-1	Changed ZPARAM default settings	202

Archived

Examples

1-1	UNION ALL in VIEW	10
2-1	ABAP program snippet	24
2-2	Pseudo DDL of a Unicode R/3 system	26
2-3	A quick look at SAP table SVERS — SPUFI output	28
2-4	Supported configurations today	32
2-5	SAP NetWeaver configuration overview	32
2-6	Creation of database table from Java dictionary	63
3-1	ABAP example with LOB	87
3-2	ABAP select statement with two DISTINCTs	88
3-3	Corresponding DB2 select statement	89
3-4	Solution with two ABAP SQL statements	89
4-1	Sample create of a materialized query table	102
4-2	Converting a base table into an MQT	102
4-3	Sample REFRESH TABLE statement	103
5-1	Access to the cluster table	113
5-2	Processing each record in a loop	117
5-3	Eliminates the need for a loop	117
5-4	SELECT loop until 'not found' condition is reached	118
5-5	SELECT loop until 100 rows fetched or 'not found' condition is reached	118
5-6	SELECT into internal ABAP table	118
5-7	SELECT a single row	119
5-8	ABAP MODIFY statement	119
5-9	DB2 statements to emulate the ABAP MODIFY statement	119
5-10	ABAP INSERT statement	121
5-11	Updating column distribution statistics	124
5-12	Invalidating DSC STATS using RUNSTATS	125
5-13	Statement that caused the trouble	126
5-14	ST05 trace output of the calculation	126
5-15	ST05 Explain output of the highlighted statement — Wrong access path	127
5-16	ST05 trace output of the calculation, after implementing the ABAP hint	128
5-17	ST05 Explain output of the highlighted statement — Right access path	128
5-18	DB2 V7: Varying-length index keys and LIKE predicate	130
5-19	Create a NOT PADDED index	130
5-20	Create a PADDED index	131
5-21	Alter an index to NOT PADDED	131
5-22	IN list query before transformation	139
5-23	Transformed IN list query	140
5-24	Fast retrieval of most recent value	141
6-1	STMID parameter	164
6-2	STMTOKEN parameter	164
7-1	FRBACKUP syntax	184
7-2	FRRECOV syntax	184
7-3	FRDELETE syntax	184

Archived

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

Trademarks

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

AIX®
DB2 Connect™
DB2 Universal Database™
DB2®
DFSMSdss™
DFSMSHsm™
DRDA®
Enterprise Storage Server®
@server™

FlashCopy®
IBM®
ibm.com®
MVS™
OS/390®
QBIC®
QMF™
RACF®
RAMAC®

Redbooks™
Redbooks(logo) ™
RMF™
System/390®
Tivoli®
WebSphere®
z/Architecture™
z/OS®
zSeries®

© 2003 by SAP AG. All rights reserved. SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. MarketSet and Enterprise Buyer are jointly owned trademarks of SAP AG and Commerce One. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves information purposes only. National product specifications may vary.

The following terms are trademarks of other companies:

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

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

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

Summary of changes

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

Summary of Changes
for SG24-7088-00
for DB2 for z/OS V8 Features Benefitting SAP
as created or updated on September 16, 2004.

December 2003, First Edition

The revisions of this First Edition reflect the additions and the changes described below.

September 2004, First Update

Changed information

Change bars identify the corrected area.

- ▶ Corrected references to zLinux into the proper Linux for zSeries
- ▶ Changed references to more current products (like SAP WebAS 6.40) and documentation
- ▶ Removed paragraph LOB lock release in 1.3.1, "Architecture" on page 14.
- ▶ Corrected several statements in 2.2.6, "Configuration of DB2 Connect and DDF for SAP" on page 35.
- ▶ Removed some items in the list in 2.3.2, "Online schema changes overview" on page 39, since they are used internally or not applicable.
- ▶ Corrected the 64-bit list to be more precise in 2.4.2, "Enlarged storage pools" on page 52.
- ▶ Corrections in Chapter 5, "Performance" on page 107 to reflect recent performance documentation.
- ▶ Minor corrections in Chapter 6, "Tools and administration" on page 155

Added information

Change bars identify the added area.

- ▶ Added references to the more recent *DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ... and More*, SG24-6079.
- ▶ Added a Note box in 2.1.5, "Performance and storage" on page 26.
- ▶ Added Enabling SQL Break paragraph on "DB2 Connect configuration for SAP" on page 36.
- ▶ Added a Note box at the end of "DB2 Connect configuration for SAP" on page 36.
- ▶ Added a Note box on 6.6.1, "Rollup accounting data for DDF and RRSAP" on page 170.
- ▶ Added an Attention box on 6.6.3, "Other DDF and RRSAP enhancements" on page 173.

Archived

Preface

DB2® UDB for z/OS® Version 8 (DB2 V8) introduces a large number of new features that provide for unmatched scalability, availability, performance, and manageability to reliably handle any information needed in SAP solutions. The objective of this IBM® Redbook is to identify the new features of DB2 V8 that are particularly beneficial to SAP and to describe how SAP applications can take advantage of these new features. While some of these features can be exploited transparently by the currently available versions of DB2, for some others the next technological release of SAP will be required. The redbook introduces the new features and provides real-life examples from SAP to demonstrate the usefulness of the new features for SAP and to show how SAP can exploit them.

The considerations within this redbook apply to the whole spectrum of business solutions within the mySAP Business Suite, such as mySAP ERP and mySAP Supply Chain Management. These solutions share the common technical application platform SAP NetWeaver that includes the SAP Web Application Server and SAP Business Information Warehouse.

While this redbook specifically investigates the new features of DB2 V8 in the context of SAP, the majority of the considerations also apply to other enterprise packaged applications. The reason for this is that from a database management perspective, these applications have many similarities, such as a strong usage of dynamic SQL and a large number of database objects. For a more general description of the new features of DB2 V8, see *DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know... and More*, SG24-6079.

The redbook is structured such that it first discusses general architectural enhancements that are relevant to SAP. Then it elaborates on topics in the area of usability, availability, and scalability that affect SAP. After discussing business warehouse and performance topics, it provides details on new features that facilitate and enhance database administration. The redbook is completed by taking a look at the substantial improvements in system level backup and point in time recovery.

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.

Paolo Bruni is a certified Consultant IT Architect working as a Data Management Project Leader at the International Technical Support Organization, San Jose Center since 1998. In this capacity he has authored several redbooks on DB2 for OS/390® and related tools, and has conducted workshops and seminars worldwide. During Paolo's many years with IBM, in development, and in the field, his work has been mostly related to database systems.

Tim Bohlsen is an SAP, DB2, and z/OS Specialist. He has worked in various locations worldwide with SAP on DB2 for z/OS customers during the last six years, as well as instructing for IBM Learning Services on this topic during this time. He has eight years experience with SAP and 17 years experience in the IT industry. Prior to working with SAP R/3, Tim was an MVS™ System Programmer in Australia for five years, specializing in large system performance and automation. He holds an Honours degree in Computer Engineering from Newcastle University.

Mark Keimig is a Consulting IT Specialist in USA. He has 25 years of IT experience with large systems and has worked at IBM, Sales and Distribution Division, for the last five years. His areas of expertise include SAP on DB2 for OS/390 and Oracle, SAP Basis Administration and SAP system performance. Mark has worked on SAP on DB2 for z/OS® since the first availability of the product and has supported a variety of zSeries® customers. He has written extensively on SAP performance and SAP platform migrations.

Peter Mohrholz is a Development Architect in the SAP NetWeaver Database Platform organization at SAP's Headquarters in Walldorf. He joined SAP in 1988 and has 15 years of experience in DB2 interfacing SAP applications. He holds a masters degree in Computer Science. His areas of expertise include SAP database interface, SAP middleware, and technology development. Peter also impacts SAP's technology architecture based on his extensive database skill. He is technical focal point for all questions related to IBM DB2 and SAP's database layer with regard to architecture, development, and performance tuning.

Johannes Schuetzner is a software engineer in the IBM Boeblingen lab in Germany. He is a member of the joint SAP/IBM software development team that is in charge of the zSeries platform at SAP. He published and presented at conferences on different aspects of DB2. He studied computer science at the University of Stuttgart in Germany and at the University of Connecticut, Storrs.

Thomas Wilson is an Infrastructure Analyst with John Deere & Company in Moline, Illinois, in the U.S.A. and is currently working in support of SAP on DB2 for z/OS. He has eight years of experience in DB2 as an application programmer, and five years as a DBA and Systems Programmer. Tom holds a BS degree in Computer Science from Marycrest College in Davenport, Iowa.

A photo of the team is shown below.



Left to right: Peter, Tim, Johannes, Paolo, Mark, and Tom

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

The Rev. Charles Lutwidge Dodgson (Lewis Carroll), for the title

Yvonne Lyon
Mary Lovelace

Emma Jacobs
Deanna Polm
Bart Steegmans
International Technical Support Organization, San Jose Center, USA

Harald Duvenbeck
Namik Hrle
Andreas Mueller
Joachim Rese
Volker Schoelles
IBM @server Software Development, Boeblingen, Germany

Burkhard Diekmann
Enno Folkerts
Bernd Kohler
SAP AG, Walldorf, Germany

Alex Stevens
Yves Tolod
Debbie Yu
IBM DB2 Tools and Connectivity Development, Toronto, Canada

Thomas Beavin
Bill Bireley
Tammie Dang
Keith Howell
Chris S Leung
Dave Levish
Tom Majithia
Bruce McAlister
Joseph Ou-Yang
Ron Parrish
Terry Purcell
Akira Shibamiya
John B Tobler
Derek Tempongko
James Z Teng
Richard Vivenza
Chung Wu
Jay Yothers
IBM DB2 for z/OS Development, Silicon Valley Lab, CA, USA

Muni Bandlamoori
Naveen Chandorkar
Roger Lo
John MacKay
Manfred Olschanowsky
Helmut Roesner
Yeong Soong
IBM/SAP Integration Center, IBM Silicon Valley Lab, USA

Become a published author

Join us for a two- to seven-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge

technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

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

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

- ▶ Use the online **Contact us** review redbook form found at:

ibm.com/redbooks

- ▶ Send your comments in an Internet note to:

redbook@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. QXXE Building 80-E2
650 Harry Road
San Jose, California 95120-6099



Introduction

In this introductory chapter we describe the SAP and DB2 for z/OS partnership and the main functions that DB2 for OS/390 Version 5, DB2 UDB for OS/390 Version 6, and DB2 UDB for OS/390 and z/OS Version 7 have made available for the SAP solutions. We then introduce the DB2 UDB for z/OS Version 8 functions of interest for SAP.

1.1 SAP and DB2 for z/OS partnership

In this section we briefly describe the SAP and DB2 partnership in regard to joint development and testing.

1.1.1 IBM and SAP cooperation

IBM and SAP have a long history of cooperation in supporting the large number of enterprises that use the SAP application suite with DB2 for z/OS database server today. The partnership is made possible by teamwork for development, integration, performance analysis, and customer service. Together, they achieve excellent SAP R/3 results on the z/OS platform.

On December 7th, 1999, SAP and IBM jointly announced a closer partnership at the database level. A group of IBM DB2 experts are turning this announcement into reality at SAP. Right after the announcement, the first new SAP implementation on IBM DB2 systems was installed by IBM. In the database area, SAP's development has direct contact to IBM's DB2 development labs, and they are communicating permanently. IBM employees are working in SAP's development lab in Walldorf to constantly improve SAP on DB2. Additionally, SAP employees are located at IBM's development labs in IBM/SAP Integration and Support Centres to provide the best support possible to customers and extend the availability of the development support over a day, a principle known as *following the sun*. Additionally, SAP's Local and Regional Support is covered by SAP for every SAP combination that is released to customers. It is the goal of SAP to further utilize DB2 Universal Database™ as a development database.

The main locations involved in this world-wide cooperation are shown in Figure 1-1.

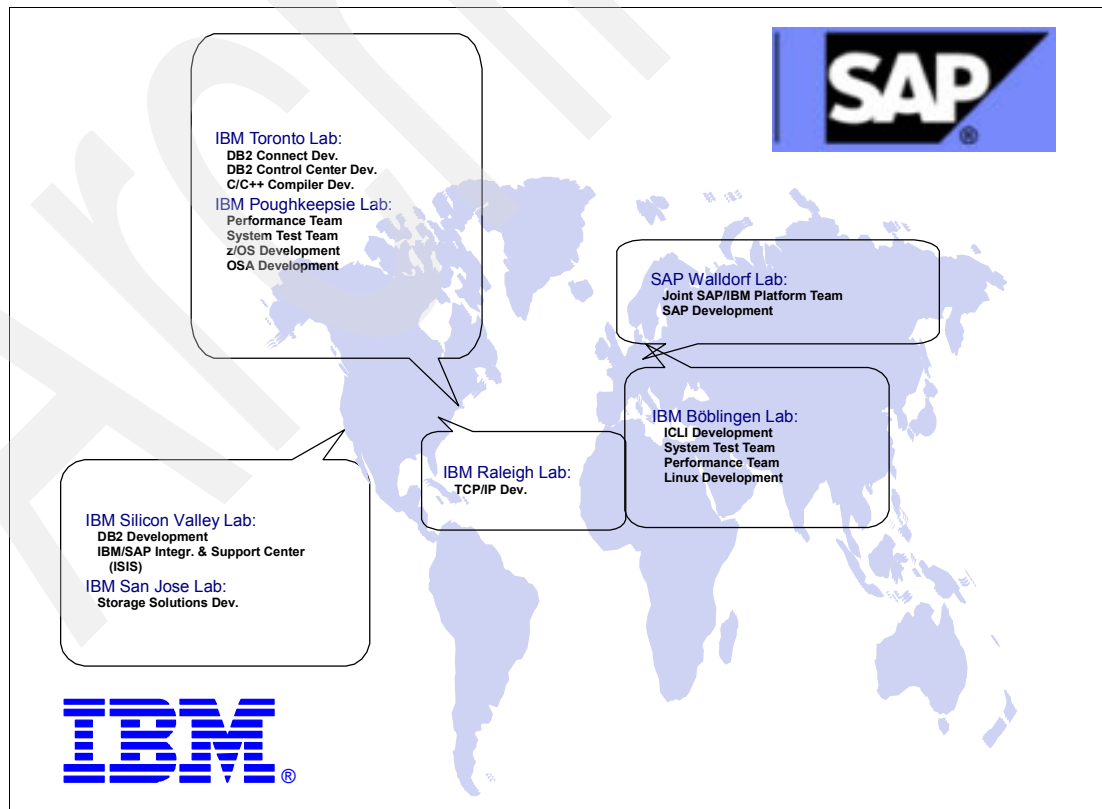


Figure 1-1 The team effort in IBM and SAP labs

The z/OS platform offers unmatched scalability, availability, performance, and manageability to reliably handle any information needed in an SAP R/3 solution using DB2 for z/OS as the database server.

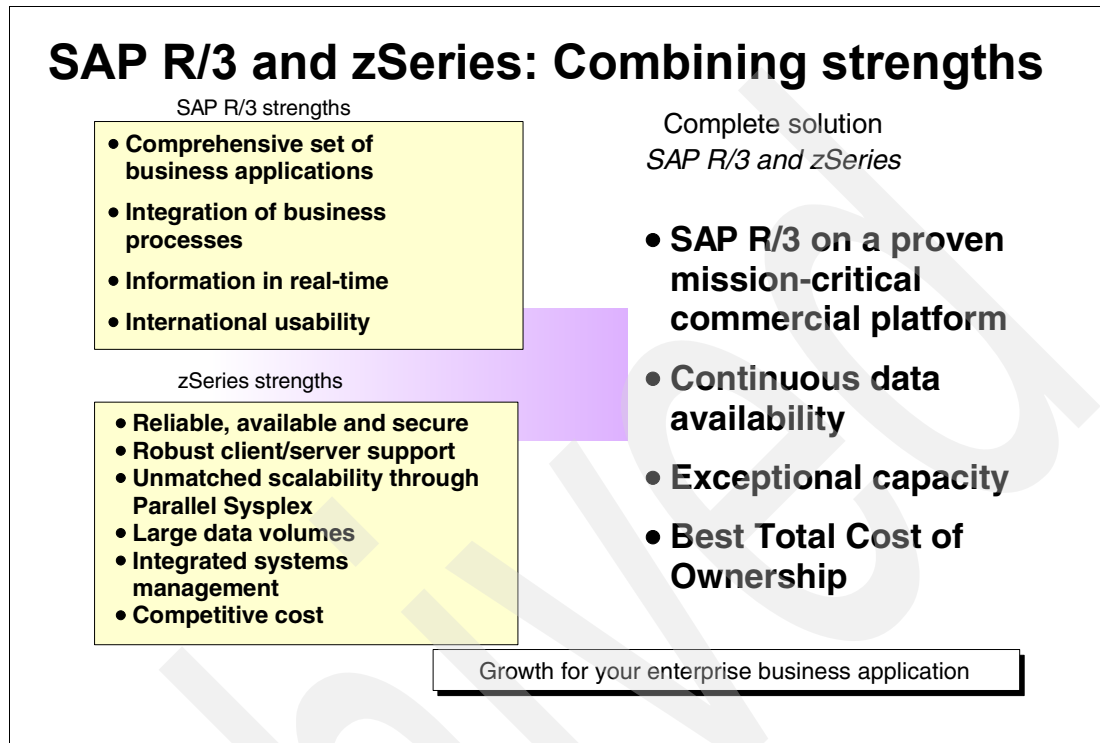


Figure 1-2 SAP and zSeries synergy

1.2 Description of functions up to DB2 V7

In this section we summarize the major DB2 SQL and system performance enhancements that benefit SAP R/3 made available by the DB2 for OS/390 V5, V6, and V7. This information confirms the DB2 commitment on supporting SAP across versions, and is included here to provide a base for comparison with the V8 enhancements.

More details are provided in the Redpaper *SAP R/3 on DB2 for OS/390: DB2 Features That Benefit SAP*, REDP0131, and its bibliography.

1.2.1 SAP related enhancements in DB2 V5

The following list shows the DB2 for OS/390 V5 SQL and system features which are used by SAP R/3.

- ▶ Dynamic statements cache
- ▶ Statement level perf indicators
- ▶ 255-char columns as short strings
- ▶ Update of partitioning key column
- ▶ Alter Table to extend column length
- ▶ Data sharing scalability improvements
- ▶ Rename table
- ▶ ASCII tables
- ▶ Reduce impact of DBD locks
- ▶ Improve recover performance

- ▶ Read stability
- ▶ Keep Update locks
- ▶ DDL concurrency: Create objects
- ▶ New client correlation identifiers
- ▶ Table/index growth monitor
- ▶ Streamline UPDATEs/DELETEs

We now briefly describe some of the most important features.

Native ASCII data storage

DB2 V5 introduces support for ASCII character data encoding in addition to the native EBCDIC encoding. By storing tables in ASCII format, SAP R/3 saves CPU time on the host by avoiding host variable conversions and column field procedures. Field procedures, previously used for collation or conversion of data, can degrade performance by as much as 30% for some transactions. SAP R/3 also saves CPU time on clients by not having to convert retrieved data.

Dynamic statement caching

Processing of dynamic SQL statements, used very frequently by SAP, requires two steps: prepare, then execute. Prepare is generally expensive: It includes parsing and syntax checking, catalog searches for tables and columns, authorization checking, access path optimization, and creation of the executable statement.

For repetitive dynamic SQL statements, the ability to cache them can significantly reduce the cost of running them.

Dynamic statement caching is activated at two levels:

- ▶ The first level is global and permits the caching in the EDM pool of all the SQL dynamic statements submitted to DB2. This level is enabled by setting the CACHE DYNAMIC SQL field to **yes** in the installation panel DSNTIP4. The corresponding ZPARM parameter is CACHEDYN. No SQL changes are needed to benefit from this function.
- ▶ The second level applies to a package/plan and is set during the bind operation by using the KEEP DYNAMIC(YES) option. The storage used by the kept statements is controlled by setting the MAX KEPT DYN STMTS field of installation panel DSNTIPE. This function implies programming modifications.

SAP R/3 has modified its own code to maximize the benefits it can get from this function. An SQL statement must have a perfect match to be able to reuse the prepared statement in the global cache; SAP R/3 uses parameter markers (or host variables), therefore the cache hit ratio can be very high. Dynamic statement caching achieves very good savings for SAP R/3 batch processes where a limited number of SQL statements get executed many times.

ORDER BY clause

Originally, DB2 required that all columns referenced in an ORDER BY clause must also be named in the SELECT list of a query.

DB2 V5 now allows you to specify columns in the ORDER BY clause that are not in the SELECT list, such as the following statement:

```
SELECT name FROM q.staff.systables ORDER BY dept, years
```

You cannot use this enhancement of ORDER BY in conjunction with UNION, UNION ALL, GROUP BY, DISTINCT, and column functions such as MAX, MIN and SUM.

This enhancement is important for SAP R/3, which generates SQL statements.

DSMAX increased from 10000 to 32767

Support has been added to permit allocation of data sets in excess of the 10,000 data set limit. This change is made in conjunction with changes introduced in OS/390 Version 2 Release 6 and later releases. Together, they enable DB2 to dynamically allocate a much larger number of concurrent data sets.

DSMAX changes from the prior OS/390 limit of 10,000 data sets to the maximum of 32,767 data sets.

Note: The increase of DSMAX has limited value for SAP R/3. SAP does not recommend specifying a DSMAX larger than 6000 due to the impact on DBM1 virtual storage below the 16 MB line.

1.2.2 SAP related enhancements in DB2 V6

In this section we describe the DB2 V6 functions that are beneficial to SAP R/3 on OS/390. Many of those functions have been retrofitted into DB2 V5 by APARs/PTFs.

The DB2 for OS/390 V6 SQL and system functions that address SAP R/3 requirements are:

- ▶ Index access on small tables
- ▶ Snowflake scheme join
- ▶ Increased number of tables in join
- ▶ Deferred dataset creation
- ▶ Switching off logging
- ▶ Local predicates in join ON clause
- ▶ Accounting Class 3 enhancements
- ▶ Non-JCL API to DB2 Utilities
- ▶ 8 KB and 16 KB page table spaces
- ▶ COPY Utility consistent backup
- ▶ DB2 logging bottleneck relief
- ▶ Table self-reference on mass insert
- ▶ Index access 'IN non-correlated subquery'
- ▶ Triggers, UDFs, UDTs
- ▶ Suspend log write activity command
- ▶ Log shortage avoidance
- ▶ Changing partitioning key ranges
- ▶ DDL concurrency: Drop database

We now briefly describe some of the most important features.

Index screening in RID list processing

Index screening predicates reference columns in the index, but are not part of the matching index columns. For example:

```
SELECT * FROM T WHERE C1 = 1 AND C3 > 4 AND C4 =6;
```

With an index on T (C1,C2,C3), C3 > 4 is an index screening predicate. It can be applied when accessing the index, but it is not a matching index predicate like C1 = 1. The value of MATCHCOLS in the PLAN_TABLE is 1. C4 = 6 is not an index screening predicate. Any column in the index that is not one of the matching columns for the query will be considered for index screening.

Prior to this enhancement, index screening was not used when RID list processing was involved. RID list processing is used by:

- ▶ List prefetch for single table access and hybrid join
- ▶ Index ANDing and ORing during multiple index access

DB2 allows index screening during RID list processing to filter out additional rows at index access time.

SAP R/3 benefits naturally from this enhancement every time an application performs a list prefetch, or an index ANDing or ORing.

Unmatched column join for VARCHAR

Suppose you want to join two tables, but the data types or the length of the join columns do not match. DB2 can perform a join in that case, but there is a performance impact. Because of the mismatch on data type or column length, the join predicate is considered Stage 2. This means all qualifying rows of the inner table are passed back to the relational data system (RDS) component of DB2 for evaluation of the join predicate. If the join columns have the same data type and column length, the Data Manager (DM) component can deal with the join predicate at Stage 1.

These enhancements have been incorporated in DB2 for nested loop and merge scan joins. Hybrid join does not support this feature.

A problem that is not being addressed by this enhancement is the mismatch of attributes on local predicates. For example, the local predicate `CHARCOL5 = 'ABCDEF'` is still stage 2 if the column length is smaller than the length of the literal.

All character strings in SAP R/3 are VARCHAR therefore SAP R/3 benefits naturally from this enhancement.

Outer join performance enhancements

DB2 V6 introduces a large number of outer join performance enhancements making outer join SQL statements perform very closely to a similar inner join statement.

In addition, the SQL syntax of the ON clause has been extended as well to allow you to “boldly” write SQL that you could not write before.

The benefits for SAP R/3 are in significantly improved performance for a large number of outer join queries, especially in DB2 V6.

Uncorrelated subquery: Indexable IN predicates

Before this enhancement, DB2 does not use a matching index when evaluating the IN predicate against the result set of a non-correlated subquery. The non-correlated IN subquery predicate is considered a stage 2 predicate. Consider the following coding:

```
UPDATE T1
SET SDATE = '01/01/1999' , STIME = '20:38:35'
WHERE PROG IN (SELECT MASTER FROM T2 WHERE INCLUDE = 'TODAY');
```

In this example:

- ▶ A unique clustering index exists on T1(PROG).
- ▶ An index exists on T2(INCLUDE, MASTER).

The non-correlated IN subquery predicate has become indexable and stage 1. The DB2 optimizer evaluates if this transformation helps performance, based on the existence of an index on the column specified on the left hand side of the IN predicate and the selectivity on the IN subquery predicates. This enhancement can be used in SELECT, UPDATE, and DELETE statements.

SAP R/3 benefits from this enhancement because these constructs are often used in SAP R/3 applications. Without this DB2 enhancement, a major performance degradation was detected. The only way to solve it was to rewrite the application logic, which was “curing” rather than “preventing” the problem.

16 terabyte tables

DB2 for OS/390 V6 greatly expands the capacity to store data in a single table space. DB2 increases the limit for storing data in a single table space to 16 terabytes. This limit is up from 1 terabyte in Version 5 and 64 gigabytes in prior releases. You can create tables that can be up to 16 terabytes in size, either in compressed or uncompressed format, assuming that sufficient disk space is available. Of particular interest to SAP R/3 is that each partition of a large table can be 4 gigabytes as opposed to 2 gigabytes in prior releases. This means that developing the partitioning key could be less complex in DB2 Version 6.

225 tables per query or view

In prior releases of DB2, the maximum number of base tables in a view was 15. In Version 6, the number of tables that a view can support is 225 (15X15). You can also specify 225 tables in SELECT, UPDATE, INSERT, and DELETE statements.

Some SAP R/3 applications are reaching the limit of 15-tables join. In the past, users developing ABAP programs had to be aware of the 15 table limit. Raising the table limit in an SQL statement from 15 to 225 tables will benefit SAP R/3.

Buffers and EDM pools in data spaces

Prior to DB2 V6, you allocate a buffer pool in either the DBM1 address space (virtual pool) or in a hiperspace (hiperpool).

The use of hiperpools helps to relieve the 2 GB addressability limitation of MVS address spaces. DB2 hiperpools reside in expanded storage only and may not contain changed pages. The total size of all hiperpools cannot exceed 8 GB.

DB2 V6 provides an option to define buffer and EDM pools in a data space.

Data spaces exploit real storage larger than the 2 GB limit when the 64-bit machine becomes available.

When using data spaces, make sure they are completely backed by processor storage. You do not want to see any paging activity when having to get to the buffers in the data space.

Buffer pool in data space

Each data space can accommodate almost 2 GB of buffers and any single buffer pool can span multiple data spaces. However, no more than one buffer pool can be in a single data space. The sum of all data space buffers cannot exceed 8 million pages. This limit is independent of the buffer size.

A hiperspace is addressable in 4 KB blocks; in other words, it is page addressable; a data space is byte addressable. You cannot put a primary buffer pool into both a hiperspace and data space.

The benefits of using buffer pools in data spaces are to:

- ▶ Improve buffer pool hit ratio. Since you can store more pages in a data space, than in a virtual pool that resides in DBM1 address space, pages can stay longer in memory.
- ▶ Allow for more parallel processing to execute prefetch I/O streams for large queries.

The main reason to choose a data space to store your virtual buffer pools is to provide relief for virtual storage constraints in the DBM1 address space and to provide greater opportunities for caching very large table spaces or indexes. If you are currently using hiperpools for read-intensive workloads and have not reached any DB2 virtual storage limit, there is no immediate benefit to moving to data spaces until processors are available that address more than 2 GB of real memory.

EDM Pool in data space

You can choose to have the part of your EDM pool that contains cached dynamic statements in a data space. By moving these “skeletons” of prepared dynamic SQL statements to a data space, you reduce the storage you require in the DBM1 address space.

If you specify YES for CACHE DYNAMIC SQL, DB2 will calculate a default value for the EDMPOOL DATA SPACE SIZE, automatically enabling the usage of a data space for cached dynamic statements.

Because SAP R/3 is primarily using dynamic SQL and this enhancement does allow for larger EDM pools, it will positively impact customers using dynamic statement caching.

Alter Index to redistribute partitions

A change to the partitioning index to shift data from one partition to another was a complex process. This process was resource intensive, especially if you only needed to modify a subset of the total number of partitions.

In DB2 V6, you now have the ability to alter indexes in order to change the partitioning key values. Access to data is only restricted when the ALTER INDEX command with the LIMITKEY parameter completes for the affected partition(s). There is a new restrictive state REORG Pending (REORP) that prevents access to these partitions until you do a REORG.

This function is especially useful for SAP R/3 where the data distribution will vary over time. Altering the key ranges without having to stop the entire table space will improve availability compared to the previous requirement.

Defer defining data sets

DB2 allows users the DEFINE NO option in the CREATE TABLESPACE and CREATE INDEX SQL statements to defer the creation of underlying VSAM data sets for the created DB2 table space or index space. The undefined table spaces or index spaces will still have a DB2 catalog entry, but are considered as empty when accessed by SELECT or FETCH operation. An existing SQLCODE +100 (sqlcode100) is returned to any application which attempts to perform a read-only operation.

Once the page set is marked with “undefined” status in the DB2 catalog (the SPACE column in SYSTABLEPART or SYSINDEXPART is set to -1), it is treated as an empty data set until the very first write operation occurs, either through SQL statements or certain DB2 Utilities (such as LOAD). At the first write, DB2 resets the “undefined” status in the catalog and creates the underlying VSAM data sets to allow the write operation. The “undefined” status stored in the DB2 catalog will not be modifiable by any DB2 ALTER command or any other third party utilities. DBAs and application package providers should consider using the DEFINE NO option if the DDL performance is critical. The DEFINE NO option provides better management relief on DD limits and data usability by deferring the VSAM DEFINE/OPEN until the very first write.

Deferring the definition of data sets is an enhancement that can be useful for customers who use only a subset of modules from the full suite of applications provided by SAP R/3 — for example, FI, CO, or SD. Currently, customers receive all application tables, regardless of

which applications they are actually going to use. This install method allows customers to add SAP R/3 modules easily after the initial install. On the other hand, it is possible for customers to have hundreds of empty tables for applications they will not use. These tables are perfect candidates to be defined using *defer define*.

SET LOG SUSPEND/RESUME command

Users have requested a way of temporarily “freezing” updates to a DB2 subsystem while the logs and database can be copied (for example, by using Enterprise Storage Server® FlashCopy® or RVA SnapShot) for remote site recover or prior point in time recovery usage. This would allow them to recovery the DB2 subsystem to a point in time without experiencing an extended recovery outage, or without having to stop or quiesce the primary system.

Avoid using this function while long-running units of recovery are active. DB2 restart time is lengthened by long-running updates.

New options are added to the -SET LOG command to be able to SUSPEND and RESUME logging for a DB2 subsystem. When a SUSPEND request is issued, a system checkpoint will be taken (in a non-data sharing environment), any unwritten log buffers will be written to disk, the BSDS will be updated with the high written RBA, and the log-write latch is obtained to prevent any further log records from being created. This will prevent any further updates to the data base until update activity is resumed with a -SET LOG RESUME request. The scope for these commands is for single subsystems only, therefore the commands will have to be entered for each member when running in a data sharing environment.

For further details on how to use this command in conjunction with SAP R/3 database backup and recovery, see SAP note 83000 on the SAP Service Marketplace Web site.

Access path selection adjustment

DB2 V6 introduced a new parameter, NPGTHRSH, which will cause the DB2 Optimizer to favor index access for tables whose statistics indicate less than a given number of pages. For a given table, if NPAGES is less than the NPGTHRSH value, index access for the table will be preferred over a table space scan. After the initial install of SAP R/3, there are many empty or small tables which could grow rapidly in size. There are also a number of tables in SAP R/3 which are very volatile, meaning the number of rows can change very quickly and in large amounts. If a RUNSTATS is run on these tables when they are small, the DB2 optimizer would favor a table space scan, which would be inappropriate when the table grows.

Prior to this enhancement, SAP R/3 recommended that the database administrator run a catalog update changing the statistics for small tables. Now SAP R/3 recommends that the customer use the new DB2 V6 parameter set as NPGTHRSH = 10. See SAP note 192320 for more detail; you can view it on the SAP Service Marketplace Web site:

<http://service.sap.com>

1.2.3 SAP related enhancements in DB2 V7

In this section we describe the DB2 V7 functions that are beneficial to SAP R/3 on OS/390 and z/OS:

- ▶ Lockout diagnostics
- ▶ Deadlocks at insert
- ▶ FETCH FIRST n ROWS ONLY
- ▶ Online REORG switch phase
- ▶ Report IRLM start parameters
- ▶ Evaluate uncommitted
- ▶ Option on timeouts for utilities

- ▶ Retained locks concern
- ▶ Simplify monitoring virtual storages usage
- ▶ Row level locking for catalog
- ▶ Statement ID for cached statements
- ▶ Real-time statistics
- ▶ Preformatting
- ▶ Business Warehouse joins

We now briefly describe some of the most important features.

Asynchronous INSERT preformatting

DB2 improves the performance of insert operations by asynchronously preformatting allocated but unformatted data pages. When a new page is used for an insert, that page is close to the end of the formatted pages, and allocated but unformatted space is available in the data set — DB2 preformats the next range of pages.

With preformatting, an insert waits less often for a page to be formatted. When the preformatted space is used and DB2 needs to extend the table space, normal data set extending and preformatting occurs.

UNION and UNION ALL operators

The scope in which UNION and UNION ALL operators can be specified has been expanded. The CREATE VIEW statement, the INSERT statement, the UPDATE statement, the DECLARE GLOBAL TEMPORARY TABLE, nested table expressions in a FROM clause, and the subquery predicate are changed to allow a fullselect where a subselect was used in previous releases.

Now, you create a view by using UNION or UNION ALL to view the data as if it were in one table. If you use UNION ALL to combine the tables, the result consists of all the rows in the tables. If you use UNION, the result is the set of all the rows in the tables without duplicate rows. Whenever possible, the following optimizations are applied to the queries referencing such views, table expressions, or subqueries:

- ▶ The joins in the queries are distributed to the subselects of the UNION ALL.
- ▶ The query predicates are distributed to the subselects of the UNION ALL.
- ▶ Aggregation in the queries is distributed to the subselects of UNION ALL.
- ▶ Subselects that are not needed to answer the queries are eliminated.

Example 1-1 illustrates creation of a view that is the UNION ALL of three fullselects, one for each month of the first quarter of 2000. The common names for the views are SNO, CHARGES, and DATE.

Example 1-1 UNION ALL in VIEW

```
CREATE VIEW DSN8710.FIRSTQTR (SNO, CHARGES, DATE) AS
  SELECT SNO, CHARGES, DATE
  FROM MONTH1
  WHERE DATE BETWEEN '01/01/2000' and '01/31/2000'
  UNION ALL
  SELECT SNO, CHARGES, DATE
  FROM MONTH2
  WHERE DATE BETWEEN '02/01/2000' and '02/29/2000'
  UNION ALL
  SELECT SNO, CHARGES, DATE
  FROM MONTH3
  WHERE DATE BETWEEN '03/01/2000' and '03/31/2000';
```

You can use the INSERT statement in the same way you use fullselects. The UPDATE statement is also changed to support row-fullselect and scalar-fullselect, where row-select and scalar-subselect were previously supported in the SET assignment clause. In the DECLARE GLOBAL TEMPORARY TABLE statement, AS (subselect) DEFINITION ONLY is changed to AS (fullselect) DEFINITION ONLY. You now can use fullselect with a basic predicate, quantified predicate, EXISTS predicate, and IN predicate.

SAP BW exploits UNION in views.

DB2 Restart Light

In data sharing environments, the new LIGHT(YES) parameter of the START DB2 command lets you restart a DB2 member in light mode. Restart Light mode means that a DB2 data sharing member restarts with a minimal storage footprint and then terminates normally after DB2 frees retained locks.

Restart Light mode is intended for a cross-system restart in the event of an MVS system failure. The reduced storage requirement makes it possible to temporarily restart a DB2 data sharing member on a system that might not have enough resources to start and stop DB2 in normal mode. Releasing the locks with a minimum of disruption promotes faster recovery and data availability. For example, applications that are running on other DB2 members have quicker access to the data for which the failed member held incompatible locks.

You can also use Restart Light mode in conjunction with the MVS Automatic Restart Manager (ARM). To have a DB2 data sharing member automatically restarted in light mode when system failure occurs, you must have an appropriately coded ARM policy. ARM does not restart the DB2 member again after a light restart is performed; the member terminates normally for the light restart.

Online system parameters

In SAP environments utilizing DB2 in a 24x7x52 mode, the need has been growing for online update of the major DB2 system parameters.

With DB2 V7 the new -SET SYSPARM command is introduced to dynamically reload the DSNZPARM load module. All parameters of the DSN6ARVP macro can be changed, and a large number of parameters from the DSN6SYSP and DSN6SPRM macros can be changed as well.

Table 1-1 lists the DB2 subsystem online parameters that are of special interest to SAP, as currently available from the *SAP 6.20 Installation Guide*. Refer to this publication, or any current follow-on version, for recommendations on the values to specify for these parameters, since they are subject to change.

Table 1-1 DB2 subsystem parameters of special interest to SAP system

ZPARM	Description	ZPARM	Description
CTHREAD	Concurrent active threads	EDMPOOL	Environmental descriptor manager (EDM) pool size
EDMBFIT	Large EDM better fit	PARAMDEG	Maximum degree of parallelism for a parallel group
MAXRBLK	Storage needed for the RID pool	CONTSTOR	Contract each thread's working storage area
DSMAX	Maximum number of open data sets	PCLOSEN	Read only switch checkpoints for page set or partition

ZPARM	Description	ZPARM	Description
PCLOSET	Read only switch time for page set or partition	DSSTIME	Data set statistics reset interval time
STATIME	Time interval between statistics collections	SYNCVAL	Statistics alignment to the hour
STARJOIN	Star join processing enablement	CHKFREQ	Checkpoint frequency in time or number of log records
URCHKTH	Checkpoint interval for message about uncommitted unit of recovery	URLGWTH	Log records written interval for message about an uncommitted unit of recovery
TIMEOUT	Idle thread timeout	UTIMOUT	Number of timeout values for utilities to be idle
RETLWAIT	lock time out in data sharing	NUMLKUS	Maximum locks a user can hold per page before escalation
RELCURHL	Release page or row lock on which a WITH HOLD cursor is positioned	SEQCACH	Sequential mode to read cached data from controller
CDSSRDEF	Current degree special register for parallelism		

Checkpoint frequency in minutes

With DB2 V7, the checkpoint frequency parameter is enhanced to optionally allow you to specify a range of minutes instead of a number of log records. Both options are available at install time and can be changed dynamically through commands.

This feature is useful in environments where the logging rate varies. You can maximize system performance by specifying the checkpoint frequency in time to avoid the performance degradation due to many system checkpoints taken in a very short period of time because of high logging rate. We recommend that you set CHECKFREQ to a value between 10 and 15.

Long-running UR warning

Prior to DB2 V7, the warning for long-running unit of recovery (UR) was based on the number of checkpoint cycles to complete before DB2 issues a warning message for an uncommitted UR. But the number of checkpoints depends on several factors that may not include the long-running job.

With DB2 V7, the warning mechanism is additionally based on the number of log records written by an uncommitted UR. The purpose of this enhancement is to provide notification of a long-running UR that may result in a lengthy DB2 restart or a lengthy recovery situation for critical tables. The warning message is repeated each additional time the threshold is reached.

The value for written log records in the message is cumulative and indicates the number of log records written since the beginning of the UR. If statistics trace class 3 is active, an instrumentation facility component identifier (ICFID) 0313 is also written.

The UR log write check threshold is set in the DB2 parameter load module DSNZPARM (DSN6SYSP URLGWTH) at install time. The value may be modified using the `-SET SYSPARM` command. We recommend that you use URCHKTH = 1 and URLGWTH = 100.

1.3 DB2 UDB for z/OS Version 8 overview

IBM DB2 UDB for z/OS Version 8 (DB2 V8 throughout this redbook) includes dozens of changes in SQL, improving family consistency in many cases, and leading the way in others. Many barriers that had been limiting our customers are now removed: using 64 bit memory, providing consistent table and column name lengths, allowing two-megabyte SQL statements, 4096 partitions, and three times the log space. These improvements include:

- ▶ Virtual storage constraints removal
- ▶ Unicode support
- ▶ Automated Prior Point in Time Recovery
- ▶ 64 bit DB2 Connect™ for Linux for zSeries
- ▶ Array fetch, insert
- ▶ Multiple DISTINCT clauses
- ▶ Lock contention on SAP cluster tables
- ▶ Use of index for backwards searches
- ▶ Transparent ROWID
- ▶ Create deferred index enhancement
- ▶ Longer table names
- ▶ Provide DSTATS functionality
- ▶ Convert column type
- ▶ Altering CLUSTER option
- ▶ Adding columns to an index
- ▶ Index-only access path for VARCHAR
- ▶ Changing number of partitions
- ▶ Partitioning nonclustering keys
- ▶ Control Center enhancement
- ▶ DRDA® performance

Key performance enhancements deliver better family consistency and run many times faster. Being able to make database changes without an outage, such as adding a partition, is a breakthrough for availability. Improvements in Java™ function, consistency, and integration with WebSphere® make z/OS a much better platform for Java. Expansions to security allow for row level granularity, helping with the security issues of Web related applications. Many of these enhancements also help in key vendor applications like PeopleSoft, SAP, and Siebel.

In this section we introduce the main enhancements available to you in New Function Mode that are applicable to the SAP application. These enhancements are grouped into categories based on the area of impact to the SAP application. These categories correspond to the chapters in this redbook, and they are:

- ▶ Architecture
- ▶ Usability, availability, and scalability
- ▶ Business Information Warehouse
- ▶ Performance
- ▶ Tools and administration
- ▶ Backup and recovery

This list is not intended to be a comprehensive list of all DB2 V8 features. A more comprehensive overview of DB2 V8 features can be currently found in *DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ... and More*, SG24-6079. DB2 V8 standard documentation, recent information and more details can be found at the Web sites listed in “Related publications” on page 209.

1.3.1 Architecture

In this section we consider some important enhancements to the DB2 architecture.

Unicode support

Architectural changes to DB2 V8 expand the DB2 catalog with full support for the Unicode catalog. This means that you can manage data from around the world. DB2 now converts any SQL statement to Unicode before parsing, and as a result, all characters parse correctly. DB2 also supports hexadecimal string constants.

DB2 Connect and DRDA

As of SAP WebAS 6.40, SAP and DB2 V8 are planned to support only DRDA as the protocol between the application servers and database server. This has made necessary to support UNICODE and other future SAP enhancements.

Universal Driver for SQLJ and JDBC

Organizations increasingly require access to data residing in multiple sources in multiple platforms throughout the enterprise. More and more companies are buying applications rather than database management systems, as database selection is being driven by interoperability, price performance, and scalability of the server platform. This enhancement provides an open and consistent set of database protocols to access data on the UNIX, Windows®, and z/OS platforms. Tools and applications can be developed using a consistent set of interfaces regardless of the platform where the data resides. End users can integrate their desktop tools and other applications in a consistent manner with whatever databases (or multiple databases concurrently) are in the enterprise. The objective of this enhancement is to implement Version 3 of the Open Group DRDA Technical Standard. It eliminates the need for gateways, improves desktop performance, and provides a consistent set of database protocols accessing data from a z/OS server as well as UNIX and Windows servers.

Schema evolution

As 24x7 availability becomes more critical for applications, the need grows for allowing changes to database objects while minimizing the impact on availability. Online schema evolution allows for table, index, and table space attribute changes while maximizing application availability. For example, you can change column types and lengths, add columns to an index, add, rotate, or rebalance partitions, and specify which index (the partitioning index or the non-partitioning index) you want to use as the clustering index.

64 bit virtual storage

This enhancement utilizes zSeries 64-bit architecture to support 64-bit virtual storage.

The zSeries 64-bit architecture allows DB2 UDB for z/OS to move various storage areas above the 2-GB bar:

- ▶ Buffer pool
- ▶ EDM pool
- ▶ Sort pool
- ▶ Bulk of the RID pool
- ▶ Compression dictionaries

A single large address space of up to 2^{64} bytes (16 exabytes) replaces hiper spaces and data spaces. As a result, managing virtual storage becomes simpler, and the scalability, availability, and performance improve as your real storage requirements and number of concurrent users increase.

Java

Starting with Web Application Server 6.30, the SAP Java database layer is ported to DB2 for z/OS. This means that every SAP Java application that is using the SAP Java database layer is able to run on DB2 for z/OS. SAP is porting every existing Java application to the new SAP Java database layer.

1.3.2 Usability, availability, and scalability

In this section we consider enhancements related to usability, availability, and scalability.

Partitioning

There are several partitioning enhancements included in DB2 V8 which are useful in SAP environments. These include the following:

More partitions

This enhancement increases the maximum number of partitions in a partitioned table space and index space past the current maximum of 254. The new maximum number of partitions is 4096. The DSSIZE value determines the maximum number of partitions that is possible.

Partitioned secondary indexes

V8 introduces data partitioned secondary indexes to improve data availability during partition level utility operations (REORG PART, LOAD PART, RECOVER PART) and facilitate fancier partition level operations (roll on/off part, rotate part) introduced by Online Schema Evolution. The improved availability is accomplished by allowing the secondary indexes on partitioned tables to be partitioned according to the partitioning of the underlying data. There is no BUILD2 phase component to REORG SHRLEVEL CHANGE when all secondary indexes are so partitioned, nor is there contention between LOAD PART jobs executing on different partitions of a table space. Query-wise, a data partitioned secondary index is most useful when the query has predicates on both the secondary index column(s) and the partitioning index column(s).

Online partitioning changes

You can add a new partition to an existing partitioned table space and rotate partitions.

Separation of partitioning and clustering

Partitioning and clustering were bundled together in versions prior to V8. Now you can have a partition without an index and can cluster the data on any index. These changes may spare one index and reduce random I/O.

REORG utility enhancements

The REORG utility is enhanced to allow you to specify that only partitions placed in Reorg Pending state should be reorganized. You do not have to specify the partition number or the partition range. You can also specify that the rows in the table space or the partition ranges being reorganized should be evenly distributed for each partition range when they are reloaded. Thus, you do not have to execute an ALTER INDEX statement before executing the REORG utility. You can specify DISCARD with SHRLEVEL CHANGE. You can avoid the BUILD2 phase during online REORG by using the new data partitioned secondary indexes.

Create index dynamic statement invalidation

Create index will now invalidate the cached statements associated with the base table contained in the dynamic statement cache without draining active statements.

Minimize impact of creating deferred indexes

Indexes created as deferred will be ignored by the DB2 optimizer.

Column type change

You can change the data type for columns. In V5 you could increase the size of varchar columns, but the changes in V8 allow you to extend numeric and character columns and to change between char and varchar.

LOB ROWID transparency

With the capability of hiding the ROWID column from DML and DDL as well, SAP can avoid the special code to handle ROWID and use the same code path of other platforms.

Longer table and column names

Architectural changes to DB2 V8 expand the DB2 catalog with support for long names. Support for longer string constants (up to 32,704 bytes), longer index keys (up to 2,000 bytes), and longer predicates (up to 32,704 bytes) make DB2 UDB for z/OS compatible with other members of the DB2 family.

SQL statements 2 MB long

Complex SQL coding, SQL procedures, and generated SQL, as well as compatibility with other platform and conversions from other products, have required the extension of the SQL statements in DB2. DB2 V8 extends the limit on the size of an SQL statement to 2 MB.

Multiple DISTINCT clauses in SQL statements

This enhancement allows the DISTINCT keyword to appear in multiple column functions with different expressions. For example, DB2 V8 now allows:

```
SELECT SUM(DISTINCT C1), AVG(DISTINCT C2) FROM T1
```

1.3.3 Business Information Warehouse

Here we discuss some enhancements affecting the Business Information Warehouse.

More tables in join

In V7 the number of tables in the FROM clause of a SELECT statement can be 225 for a star join. However, the number of tables that can be joined in other types of join is 15. V8 allows 225 tables to be joined in all types of joins.

Sparse index for star join

The star join implementation in DB2 UDB for z/OS has to deal with, potentially, a large number of work files, especially for a highly normalized star schema that can involve many snowflakes, and the cost of the sorting of these workfiles can be very expensive. DB2 V8 extends the use of a sparse index (a dynamically built index pointing to a range of values) to the star join work files and adds a new optional function of data caching on star join workfiles. The decision to use the sparse index is done based on the estimation of the costs of the access paths available.

Common table expression and recursive SQL

DB2 V8 introduces the common table expression and recursive SQL function, which extends the expressiveness of SQL and lets users derive the query result through recursion. It is also a convenient way for users to express complex queries, as using common table expressions instead of views saves both users and the system the work of creating and dropping views.

Materialized query tables

This enhancement provides a set of functions which allow DB2 applications to define, populate, and make use of materialized query tables. SAP BW will benefit from MQTs for queries against ODS objects.

1.3.4 Performance

In this section we discuss locking, RUNSTATS, and a number of other improvements.

Locking improvements

Reduced lock contention on volatile tables

Volatile (or cluster) tables, used primarily in the SAP application environment, are tables that contain groups (or clusters) of rows which logically belong together. Within each cluster, rows are meant to be accessed in the same sequence every time. Lock contention occurs when DB2 chooses different access paths for different applications operating on the same cluster table. In the absence of support for cluster tables in DB2, users have to either change system-wide parameters that will affect all tables, or change statistics for each such table to ease the lock contention.

Cluster tables are referred to as volatile tables in DB2. Adding a new keyword, VOLATILE, to the CREATE TABLE and ALTER TABLE statements signifies to DB2 which tables should be treated as volatile tables. For volatile tables, index access is chosen whenever possible, regardless of the efficiency of the available index(es). That is, a table scan is not chosen in preference to an inefficient index.

CF lock propagation reduction

This enhancement allows in a data sharing environment, parent L-locks to be granted locally without invoking global contention processing. Thereby, locking overhead due to false contention is reduced. As a result, DB2 data sharing performance is enhanced. Performance benefit varies depending on factors such as commit interval, thread reuse, number of tables accessed in a commit interval, if the SQL processing is read-only or update, etc.

Multi-row INSERT and FETCH

With this SQL enhancement, a single FETCH can be used to retrieve multiple rows of data, and an INSERT can insert one or more rows into a table. This reduces the number of times that the application and database must switch control, as well as reducing the number of network trips required for multiple fetch or insert operations for distributed requests. For some applications, this can help performance dramatically.

RUNSTATS improvements

Distribution statistics

This enhancement adds the new functionality of calculating the frequencies for non-indexed columns to RUNSTATS. The relevant catalog tables are updated with the specified number of highest frequencies and optionally with the specified number of lowest frequencies. The new functionality also optionally collects multicolumn cardinality for non-indexed column groups and update the catalog.

Fast cached SQL statement invalidation

This enhancement adds the new functionality of allowing the UPDATE NONE and REPORT NONE keywords to be used on the same RUNSTATS utility execution. This causes the utility to only invalidate statements in the dynamic statement cache without any data access or computation cost.

Host variables impact on access paths

The enhancement allows for a REOPT(ONCE) hit as an alternative to REOPT(VARS).

Index only access for varchar

This enhancement removes the key padding associated with varchar index columns. This allows the use of these columns to satisfy the results of query that can use index only access.

Backward index scan

This enhancement provides the capability for backward index scans. This allows DB2 to avoid a sort and allows customers to define fewer indexes.

Local SQL cache issues and short prepare

This enhancement will reduce the cost of a short prepare. This will lower the cost of further reductions in MAXKEEPD.

Multiple IN values

This enhancement causes the DB2 optimizer to make a wiser choice when considering index usage on single table SQL queries that involve large numbers of IN list items.

DDF performance

Performance is improved as a result of several changes across the components involved in reducing the path length of the TCP/IP Receive, and several accounting enhancements.

1.3.5 Tools and administration

Here we describe enhancements to various tools and administration features.

Automatic space management

This feature is considered very valuable for all SAP implementations. It potentially eliminates one of the main causes of failures where growth has not been completely anticipated.

DSC statement ID in Explain

This enhancement allows the DB2 Explain to report the chosen access path of an SQL statement currently in the dynamic statement cache.

Long-running non-committing readers

This enhancement results in IFCID 313 records reporting the presence of long-running, non-committing, read-only units of recovery.

Lock escalation reporting

This enhancement results in IFCID 337 records reporting the fact that a lock escalation has occurred.

Transaction based DB2 accounting and workload management

This enhancement allows SAP to provide DB2 accounting and workload management at the granularity of SAP transactions, reports, batch jobs, and end users.

DB2 Control Center

Various new DB2 Control Center features allow enhanced functionality and better usability of the SAP CCMS functionality in the area of database administration.

Migration changes

Migration is allowed exclusively from DB2 UDB for OS/390 and z/OS Version 7 subsystems. The migration SPE must have been applied and started. The migration process is changed and now consists of three distinct steps or phases:

1. **Compatibility Mode (CM):** This is the first phase, during which the user makes all the tests needed to make sure that all the applications run without problems with the new version. Fall back to V7 in case of problems is allowed.
2. **Enable New Function Mode (ENFM):** During this (possibly short) second phase, the user converts the DB2 catalog and directory to a new format by using on-line Reorg executions. No fallback to DB2 V7 is allowed once this phase is entered.
3. **New Function Mode (NFM):** This is the target third and final phase, where all new V8 functions are available.

1.3.6 System level point in time backup and recovery

The system level point in time recovery enhancement provides the capability to recover the DB2 system to any point in time, irrespective of the presence of uncommitted units of work, in the shortest amount of time. This is accomplished by identifying the minimum number of objects that should be involved in the recovery process, which in turn reduces the time needed to restore the data and minimizing the amount of log data that need to be applied. For the larger DB2 systems with more than 30,000 tables, this enhancement significantly improves the data recovery time, which in turn results in considerably shorter system downtime.

BACKUP and RESTORE SYSTEM

These two new utilities provide system level, point in time, and level of recovery. They activate new functionalities available with the new z/OS V1R5 DFSMSHsms, which allow a much easier and less disruptive way for fast volume-level backup and recovery to be used for disaster recovery and system cloning.

CI size larger than 4 KB

DB2 V8 introduces the support for CI sizes of 8, 16, and 32 KB. This is valid for user defined and DB2 defined table spaces. The new CI sizes relieve some restrictions on backup, concurrent copy, and the use of striping, as well as provide the potential for reducing elapsed time for large table space scans.

More log data sets

The maximum number of active log data sets per log copy is increased from 31 to 93. The maximum number of archive log volumes recorded in the BSDS before there is a wrap around, and the first entry is overwritten is increased from 1,000 to 10,000 per log copy.

1.3.7 DB2 V8 enhancements implicit for all SAP versions

The enhancements listed in Table 1-2 are immediately available to all SAP releases for which SAP will certify DB2 V8. They do not require SAP code changes to exploit these functions.

Table 1-2 DB2 V8 enhancements that are immediately available to all SAP releases

Area	Functions
Virtual storage	64-bit virtual storage. Thread storage enhancements
Partitioning	Adding partitions. Rotating partitions. Separate clustering from partitioning

Area	Functions
Index creation enhancements	Invalidation of cached statements. Deferred indexes do not prevent table access
SQL language	Multiple DISTINCT clauses
SAP BW	Star join enhancements (sparse indexes, in-memory workfiles). Up to 225 tables in FROM. Materialized query tables.
Locking	Reduce lock contention on volatile tables. Allow updating the partitioning key without draining partitions. Lock holder inherits WLM priority by lock waiter if higher.
Query optimization	RUNSTATS collects more distribution statistics. Easy and inexpensive invalidation of cached statements. Bind option REOPT(ONCE). Index-only access path for varying-length columns. Multiple value IN lists.
Index access	Support for backward index scan.
Dynamic statement caching	Reducing the short prepare costs.
Data sharing	CF lock propagation reduction. CF request batching. Improved LPL recovery.
Space management	Automatic space management.
System point in time back-up and recovery	Prior point in time recovery automation. Conditional Restart Enhancements. Synchronizing log points. Suspend/resume through stored procedures. Suspend database writes. CI size up to 32 KB. Forward log recovery.

Architecture

In this chapter we discuss topics in which DB2 V8 has introduced changes affecting the underlying environment where an SAP system runs. Most of these topics involve major changes impacting the preparation and installation of an SAP system, and to a lesser degree its ongoing administration. However, in most cases they do not materially affect the actual operation and execution of the SAP system in question. That is, the functionality and behavior of the SAP application tasks executed by the end users are not affected by these new features; it is the SAP system that takes advantage of the DB2 new functions.

The chapter is structured in the following sections:

- ▶ Unicode
- ▶ DB2 Connect as SAP database connectivity
- ▶ Schema evolution
- ▶ 64-bit virtual storage
- ▶ Java

2.1 Unicode

After a short introduction into Unicode, we show why DB2 V8 is now able to run an SAP UNICODE system and DB2 V7 is not. We describe the mapping of the data types and how we try to minimize the impact of Unicode regarding performance and space usage.

This section is structured as follows:

- ▶ SAP view of Unicode
- ▶ Increased length limits
- ▶ Data type mapping
- ▶ Performance and storage

2.1.1 Introduction to Unicode

Fundamentally, computers just deal with numbers. They store letters and other characters by assigning a number for each one. Before Unicode was invented, there were hundreds of different encoding systems for assigning these numbers. No single encoding system could contain enough characters. The European Union alone requires several different encodings to cover all its languages. Even for a single language like English, no single encoding was adequate for all the letters, punctuation, and technical symbols in common use.

These legacy encoding systems conflict with one another; that is, two encodings can use the same number for two different characters or use different numbers for the same character. Any given computer, especially any server, needs to support many different encodings; yet, whenever data is passed between different encodings or platforms, that data always runs the risk of corruption.

Unicode was invented to address this situation. It provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language. The Unicode Standard has been adopted by such industry leaders as Apple, HP, IBM, JustSystem, Microsoft®, Oracle, SAP, Sun, Sybase, and many others. Unicode is required by modern standards such as XML, Java, ECMAScript (JavaScript), LDAP 3.0, CORBA 3.0, WML, etc. It is the official way to implement ISO/IEC 10646. It is supported in many operating systems, all modern browsers, and many other products.

Incorporating Unicode into client-server or multitiered applications and Web sites offers significant cost savings over the use of legacy encoding systems. Unicode enables a single software product or a single Web site to be targeted across multiple platforms, languages, and countries without re-engineering. It allows data to be transported through many different systems without corruption. Even where the client applications still depend on legacy encodings, Unicode can be used as a lingua franca on the server and database side, so that the user's data can always be stored without corruption, no matter what the original encoding.

There are three different ways to encapsulate Unicode for use on a system: UTF-8, UTF-16, and UTF-32. Each of the UTFs can be useful in different environments. For systems that only offer 8-bit strings currently, but are multibyte enabled, UTF-8 may be the best choice. For systems that do not care about storage requirements, UTF-32 may be best. For systems such as Windows, Java, or ICU that use UTF-16 strings already, UTF-16 is the obvious choice. The XML specification requires that all conformant XML parsers accept both UTF-8 and UTF-16.

For more information on Unicode, refer to the Web sites:

<http://www.unicode.org/unicode/uni2book/u2.html>
<http://www.unicode.org/charts/>

2.1.2 SAP view of Unicode

To support the new environments, you need Unicode. SAP's Java strategy requires a Unicode-based database.

DB2 V8 provides the full Unicode functionality that is required by SAP; therefore, DB2 V8 is a prerequisite for an SAP Unicode installation.

While each character of modern languages has a constant size of 2 bytes in the Unicode notations UTF-16 and UTF-32, the length of a character varies in UTF-8. It is essential that the size of characters is constant, when software is designed in English or German, tested in English or German, and the Software is expected to run in Japanese, Chinese, Korean, etc.

UTF-16 vs. UTF-32

► Supported characters:

The amount of supported characters are the same for UTF-16 and UTF-32.

► Performance:

There is a claim that processing 4 byte Unicode data is faster than processing 16 bit Unicode data on some platforms. SAP has evaluated various platforms with both 16 bit and 32 bit Unicode. However, based on the fact that the UTF-32 data volume is twice as big as UTF-16 data, UTF-16 is superior to UTF-32 when performance is concerned.

► Memory consumption:

When comparing non-Unicode systems and Unicode SAP systems, the main memory consumption of an application server is approximately (the size varies with the workload):

- 150% on 16 bit Unicode system and
- 250% on 32 bit Unicode system

SAP has 1 to 2 GB of memory requirement today on large systems. Therefore 16 bit Unicode data types have a clear advantage when memory consumption is a concern.

2.1.3 Increased length limits

DB2 UDB for z/OS V7 already supports Unicode. The reasons why DB2 V7 could not run SAP with Unicode are the following:

- **Index width:** The maximum width of an index is 255 bytes.
- **Length of a character literal in the where-clause:** The maximum length of a character literal is 255 bytes.
- **Length of a character host variable in the where-clause:** The maximum length of a character host variable is 255 bytes.
- **Length of the statement text:** The maximum length of an SQL statement is 32765 bytes.

All these numbers did not change with the introduction of Unicode in DB2 V7. What SAP now needs is, that the current limits are at least doubled.

For example: There are more than 100 indexes within an SAP system that have character columns with an accumulated length of more than 127 characters. SAP is planning to use a UTF-16 character representation for DB2 for z/OS. This means, we have to multiply each character by 2 to get the length in bytes. To be able to define an index with more than 127 characters in an SAP Unicode environment, we would require 256 or more bytes as maximum index length. These indexes cannot be defined in an SAP Unicode system running on DB2 V7 because we would then exceed the maximum index width of 255 bytes. Because of this, an SAP installation would fail.

If an ABAP program is using a character literal in the WHERE clause of an SQL statement and the length of the literal exceeds 127 characters, the ABAP wouldn't work in an SAP Unicode system any more. This would result in a runtime error. The same happens if the length of a character host variable exceeds 127 characters and is used in the WHERE clause.

In an SAP Unicode system, every ABAP program that contains an SQL statement with a length greater than 16382 characters would fail in DB2 V7 because the maximum length of 32765 bytes for an SQL statement is exceeded.

In DB2 V8, all these limits are lifted (more than doubled), so there is no limit preventing an SAP Unicode system from running.

The new DB2 V8 limits are as follows:

- ▶ The maximum width of an index is 2000 bytes.
- ▶ The maximum length of SQL statement text is 2097152 bytes.
- ▶ For a hexadecimal literal (X, GX, or UX), the number of hexadecimal digits must not exceed 32704 bytes.
- ▶ Any character string literal and host variables must be short enough so that its UTF-8 representation requires no more than 32704 bytes.
- ▶ Any graphic string literal and host variable must be short enough so that its UTF-8 representation requires no more than 32698 bytes.

Some limits are described as UTF-8 limits because the DB2 parser expects a UTF-8 character representation of the SQL statement.

2.1.4 Data type mapping

DB2 V8 supports 2 Unicode types, UTF-8 and UTF-16. UTF-8 has a varying length character representation of one to six bytes. In case of SAP, a maximum of 3 bytes would be sufficient. UTF-16 has a fixed length character representation of 2 bytes (for every current modern language) or 4 bytes.

The reason why the DB2 for z/OS porting team has decided to use the UTF-16 character representation in the database is the fact that SAP has chosen to use UTF-16 for their application servers. If both the database server and the application server are using the same character representation, no conversion would be needed for:

- ▶ The SQL statement
- ▶ The character input variables (such as character values to be inserted)
- ▶ The character output variables (such as character values of columns in the select list)

Another reason against using UTF-8 is that the sorting order of UTF-16 and UTF-8 is different. For example: Suppose that an ABAP programmer wants to read a certain set of rows into an internal ABAP table. Furthermore, he wants the rows to be sorted. He does that with the ABAP statements listed in Example 2-1.

Example 2-1 ABAP program snippet

```
tables: db2jsinf.  
data: t_db2jsinf like db2jsinf occurs 0 with header line,  
      s_db2jsinf like db2jsinf.  
  
select * into table t_db2jsinf from db2jsinf order by jobname.  
loop at t_db2jsinf.  
*check for the sort order  
  if t_db2jsinf-jobname < s_db2jsinf-jobname.
```

```

write: / 'wrong sort order'.
exit.
endif.
s_db2jsinf = t_db2jsinf.

.... program logic
endloop.

```

Using UTF-8, it is possible that the check fails because the application expects a different sort order than the one delivered by DB2 using UTF-8. This will not happen if SAP uses the UTF-16 character representation within the database.

Now, the DB2 for z/OS data types we have to deal with, if we use UTF-16, are the graphic data types. The mapping of the data types looks as follows:

- ▶ For character strings with a length of less than 13000 characters, SAP uses vargraphic.
- ▶ For character strings with a length between 13000 and 32700 characters, SAP uses dbclob.

The maximum length of a vargraphic column is 32704 bytes. SAP is using this artificial limit of 26000 bytes (13000 characters) to have a cushion big enough to define some other fields before reaching the maximum record limit of about 32 KB.

- ▶ For character large objects, SAP uses dbclob.

A char ABAP type with a length between 13000 and 32704 characters is always defined as dbclob(32704). The reason why the maximum length of 32704 is used and not the real length (such as 15000) is to avoid data conversion (unload/reload) during an SAP upgrade. The length of a vargraphic field can be changed by issuing an ALTER statement. But if you want to change the length of a dbclob field, you have to unload and reload the data. A string data type has an arbitrary length between 0 and 512 M characters. Therefore, SAP is using always the maximum length — which is, by the way, the maximum length for LOB values that can be logged (1 GB).

This mapping for non-Unicode data types is summarized in Table 2-1.

Table 2-1 Mapping of ABAP character data types to DB2 data types (non-Unicode)

ABAP type	ABAP length in chars	DB2 type	DB2 length in bytes
char	1 .. 32704	varchar	1 .. 32704
string	0	clob	1G

This mapping for Unicode data types is summarized in Table 2-2.

Table 2-2 Mapping of ABAP character data types to DB2 data types (Unicode)

ABAP type	ABAP length in chars	DB2 type	DB2 length in bytes
char	1 ..12999	vargraphic	1 .. 12999
char	13000 .. 32704	dbclob	32704
string	0	dbclob	512M

Let us look at an example with a test table. The data dictionary definition of the columns for table ZPMTEST is listed in Table 2-3.

Table 2-3 Fields in table ZPMTEST

Field name	Type	Length
F1	CHAR	1
F2	CHAR	255
F3	STRING	0

This results in the pseudo DDL statements listed in Example 2-2 (the underscores get replaced just before each statement gets issued).

Example 2-2 Pseudo DDL of a Unicode R/3 system

```
CREATE TABLESPACE ZPMTEST IN A100 ____
USING STOGROUP __BTD PRIQTY 40
SECQTY 40 FREEPAGE 16 PCTFREE 20
GBPCACHE CHANGED COMPRESS YES
BUFFERPOOL BP2 LOCKSIZE ROW SEGSIZE 4
LOCKMAX 1000000 CCSID UNICODE
CREATE TABLE "ZPMTEST"
("F1" VARGRAPHIC (000001) NOT NULL
"F2" VARGRAPHIC (000255) ,
"F3" DBCLOB (512M) )
IN A100____.ZPMTEST
```

You can see that the table space is defined with CCSID UNICODE and that the types of the character columns are vargraphic and dbclob. Each graphical character can store exactly one UTF-16 character.

2.1.5 Performance and storage

Because of choosing UTF-16 as Unicode character representation, SAP has now doubled the size of all characters within an SAP database. This leads to the assumption that the size of disk space and possibly the buffer pools is doubled even if the vast majority of character data fits in a single-byte ASCII encoding. This sounds very bad in terms of storage and also for performance.

Indeed, the SAP application server needs approximately 70% more storage. But if we take into account that DB2 offers data compression and that the data is even compressed in the buffer pools, we believe that we will see much less degradation. When comparing newly installed systems, initial investigations show that SAP Unicode systems, which employ data compression, require approximately 20% more storage than a comparable SAP ASCII system with uncompressed data. Figure 2-1 shows values, from an SAP BW 3.0B system, normalized to a 100% ASCII uncompressed size.

Note: With Unicode data, SAP uses compression on all table spaces

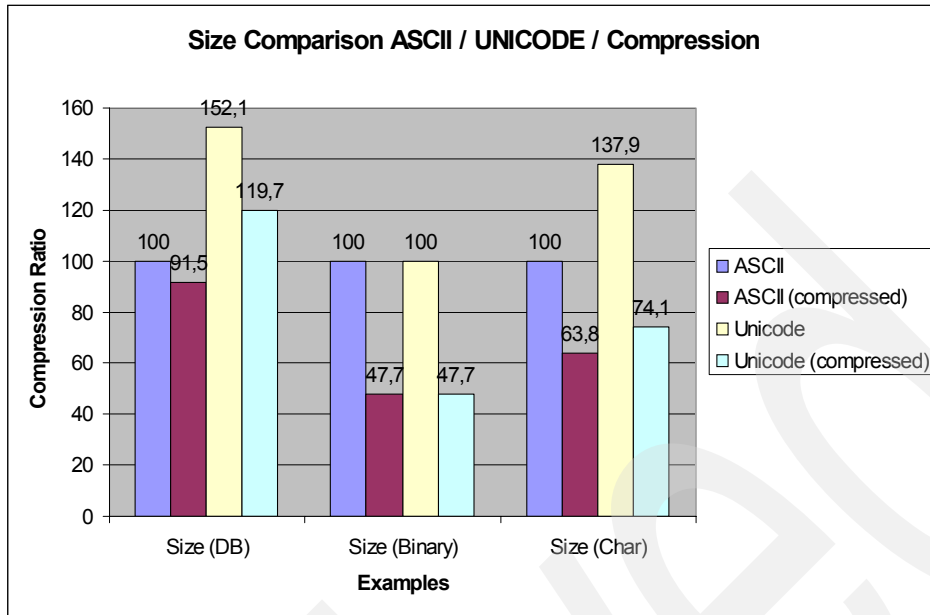


Figure 2-1 Storage comparison of SAP systems with ASCII and Unicode data

In Figure 2-1 above, the leftmost group of bars (DB) is based on the values in Table 2-4, measured after the installation of an SAP BW 3.0B system.

Table 2-4 DB group

Notation	Compression	Size in MB
ASCII	No	7.1
ASCII	Yes	6.5
UNICODE	No	10.8
UNICODE	Yes	8.5

In Figure 2-1 above, the group of bars in the middle (Binary) is based on the values in Table 2-5, measured out of the SAP cluster table RSRWBSTOR.

Table 2-5 Binary group

Notation	Compression	Size in MB
ASCII	No	134
ASCII	Yes	64
UNICODE	No	134
UNICODE	Yes	64

In Figure 2-1 above, the rightmost group of bars (Char) is based on the values in Table 2-6, measured out of the SAP table TODIR, containing a mix of INT and small VARCHAR columns.


```

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 2
DSNE621I NUMBER OF INPUT RECORDS READ IS 2
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 25
***** Bottom of Data *****
Command ==>                               Scroll ==> PAGE
  F1=Help   F3=Exit   F5=Rfind  F12=Cancel

```

2.2 DB2 Connect as SAP database connectivity

The new technology stack SAP NetWeaver requires several new features from the underlying database management system and the connectivity. After a high level introduction of SAP NetWeaver, we show the resultant requirements. We also show that DB2 Connect fulfills all these requirements and ICLI (the connectivity technology used so far) does not. DB2 Connect has been enhanced in a way that there is no functional disadvantage compared to ICLI. DB2 Connect is the IBM strategic database connectivity. All new features and enhancements will be built in into DB2 Connect, whereas ICLI will consolidate. DB2 Connect will be a more stable and reliable solution than ICLI because DB2 Connect is a connectivity standard used by many more applications and users.

2.2.1 Introduction to SAP NetWeaver

The Enterprise Services Architecture is SAP's blueprint for building, delivering, and deploying business solutions based on Web services.

The technical foundation of the SAP Enterprise Services Architecture is the next generation of mySAP™ Technology, called SAP NetWeaver™. SAP NetWeaver, shown in Figure 2-2, is the integration and application platform meant to unify and align people, information, and business processes across technologies and organizations. It embraces Internet standards such as HTTP, XML, and Web Services. A key feature of SAP NetWeaver is complete interoperability with both Microsoft.NET and IBM WebSphere.

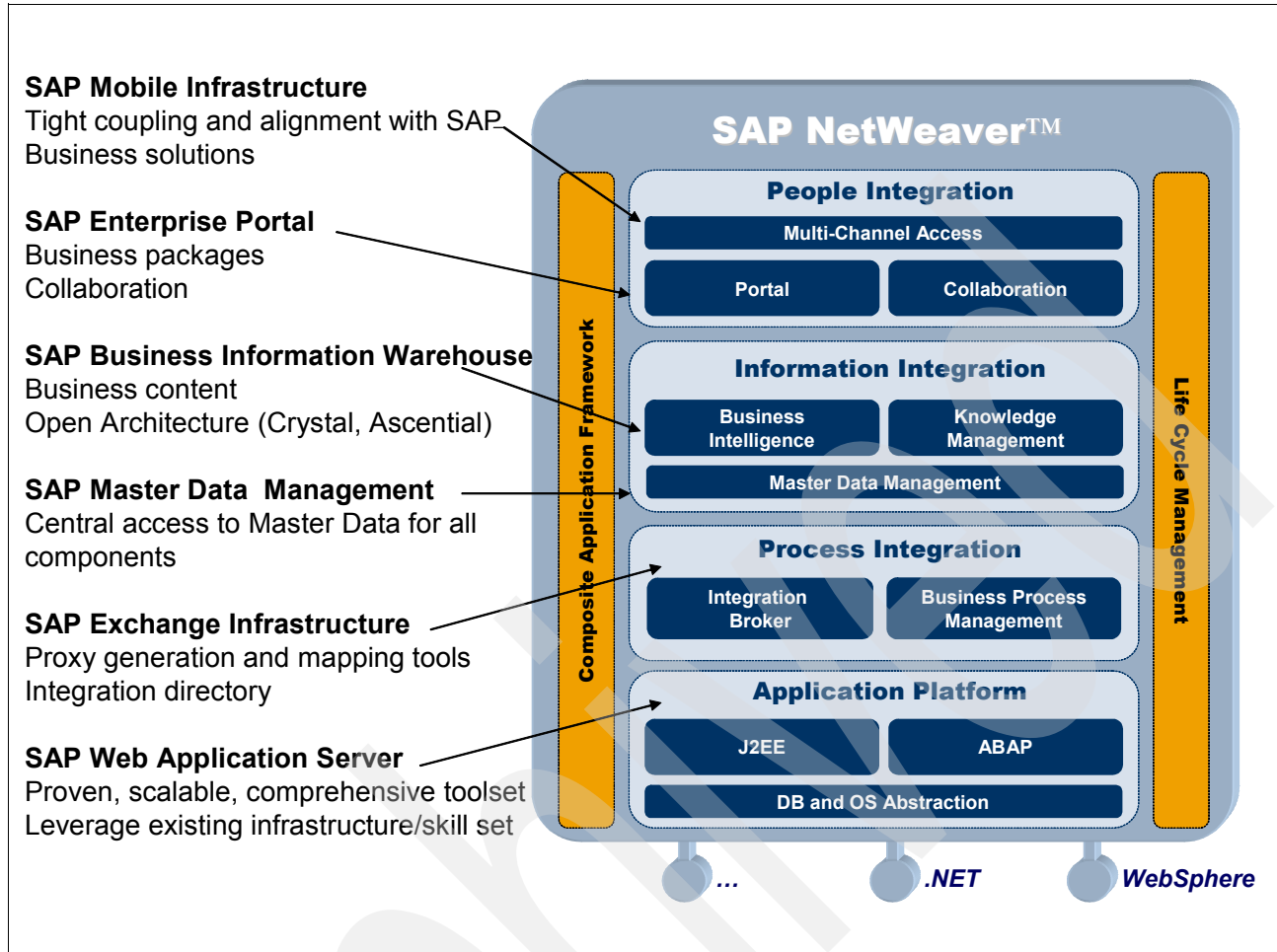


Figure 2-2 SAP NetWeaver flow

This architecture offers IT organizations the freedom of choice for the technology they prefer. Using SAP NetWeaver, they can weave their environments into a single, smooth fabric, and do so at a lower cost of ownership.

SAP solutions, such as mySAP™ Customer Relationship Management and SAP® xApps™, are already powered by SAP NetWeaver, inheriting its intrinsic virtues. In the future, all SAP solutions will be powered by the SAP NetWeaver platform. SAP NetWeaver technology enhances the scope of SAP solutions to manage the most critical business processes and adds value to every facet of the customer's organization.

The SAP NetWeaver product contains a variety of components that are synchronized in terms of the technology to provide customers a fully integrated development and runtime platform for their solutions:

- ▶ SAP Web Application Server (Web AS)
- ▶ SAP Business Information Warehouse (BW)
- ▶ SAP Exchange Infrastructure (XI)
- ▶ SAP Enterprise Portal (EP)
- ▶ SAP Master Data Management (MDM)
- ▶ SAP Mobile Infrastructure (MI)

SAP solutions currently available, such as mySAP ERP, mySAP CRM, mySAP SCM, mySAP SRM, mySAP PLM, etc., continue to run on the SAP Web Application Server technology. With future updates, however, each component will be based on the technology provided by SAP NetWeaver, making use of integration features across solutions and processes.

2.2.2 Requirements of SAP NetWeaver

Today, DB2 for z/OS customers are forced to deploy workstation platforms to run all the components of SAP NetWeaver in their system landscape. While SAP Web Application Server and SAP Business Information Warehouse are already supported on zSeries, the other components, SAP Enterprise Portal, SAP Master Data Management, SAP Mobile Infrastructure and SAP Exchange Infrastructure, do not run on the zSeries platform yet.

In the future, the SAP NetWeaver infrastructure will fully support SAP solutions on zSeries, built around DB2 for z/OS in order to meet the highest availability and scalability requirements. However, the technology necessary for a seamless integration of SAP NetWeaver and IBM zSeries needs updates on platform components like the database and the operating system.

The new SAP technology makes simultaneous use of Java and ABAP runtime environments, Unicode and non-Unicode implementations, and solutions that need large memory buffers, as depicted in Figure 2-3. The underlying platforms are required to support these technologies, and as a result, future SAP solutions on the zSeries platform require DB2 V8. Application server platforms are already accelerating the shift towards 64-bit platforms.

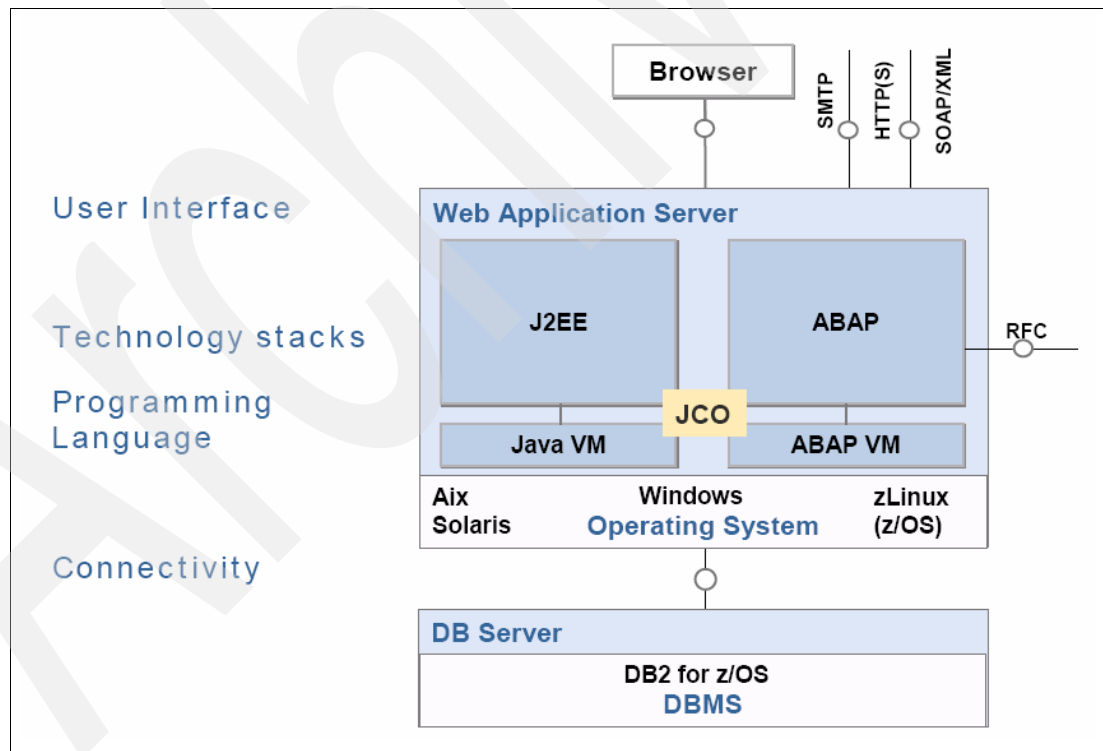


Figure 2-3 SAP Web Application Server 6.40 on zSeries

DB2 V8 has been re-engineered with the requirements of upcoming high-end business environments in mind, leading to many fundamental changes in architecture and structure. The new DB2 version runs in 64-bit mode, utilizing the power of 64-bit zSeries hardware. Limits of the past due to 31-bit addressability are now resolved. With V8, DB2 has implemented more than 50 feature requests specifically to optimize for SAP solutions on zSeries. In Example 2-4 we show the supported configurations before DB2 V8.

Example 2-4 Supported configurations today

Database Server

Hardware: S/390 G5, G6 or zSeries (z800, z900, z990)
Operating system: OS/390 2.10 or z/OS 1.1 ff.
Database: DB2 V6 or V7

Application Server

zSeries hardware: S/390 G5, G6 or zSeries (z800, z900, z990)
Workstation hardware: IBM pSeries, SUN Sparc, Intel-based server
Operating system: OS/390 2.10 or z/OS 1.2 ff. (see SAP kernel)
Linux for zSeries SLES 7 ff. (requires zSeries hardware)
AIX 5.1 ff.
Solaris 8 ff.
Windows 2000 ff.
JDK: 1.3
SAP Kernel: SAP R/3 4.6 with OS/390 2.10 ff.
SAP Web AS 6.20 with z/OS 1.2 ff.
SAP J2EE: SAP J2EE 6.30 SP1 or SP 2 supported only on
AIX & Windows and DB2 V7 and DB2 Connect V8 SP 3

In Example 2-5 we show the configurations possible with DB2 V8.

Example 2-5 SAP NetWeaver configuration overview

Database Server

Hardware: zSeries (z800, z900, z990)
Operating system: z/OS 1.4 ff.
Database: DB2 V8 ff.
DB2 Connect V8 SP 5 ff.

Application Server

Hardware: zSeries (z800, z900, z990)
Workstation hardware: IBM pSeries, SUN Sparc, Intel-based server
Operating system: z/OS 1.4 ff.
Linux for zSeries SLES 8 ff.
AIX 5.1 ff.
Solaris 8 ff.
Windows 2000 (32 bit) ff.
Windows 2003 (32/64 bit) ff.
JDK: 1.4 ff.
SAP Kernel: SAP Web AS 6.40 ff.
SAP J2EE: SAP J2EE 6.30 or SP 3 ff.

2.2.3 Necessary features of DB2 Connect

JDBC driver, Unicode enabled connectivity, XA support are mandatory features of DB2 Connect to support SAP NetWeaver. These features are all available with DB2 Connect V8. They are not available with the traditional connectivity, ICLI.

- ▶ **JDBC driver:** The JDBC driver SAP is using to access DB2 for z/OS is part of DB2 Connect. As of today, SAP is using the type 4 JDBC driver. This is a pure Java driver that needs no extra platform dependent libraries. As a result, this driver is platform independent and is used for every supported platform (see above).
- ▶ **Unicode enabled connectivity:** DB2 Connect offers a Unicode and a non-Unicode enabled application programming interface (API). The SAP database interface loads the DB2 Connect Common Client dynamically and calls the appropriate functions of the DB2 Connect API. In a non-Unicode environment, the normal character API functions are called; whereas in a Unicode environment, the corresponding wide character API functions are called. For example, the corresponding Unicode function of SQLConnect is SQLConnectW. The Unicode enabled database interface (dbdb2slib) is a different executable than the non-Unicode database interface. But both have the same name.
- ▶ **XA support:** The SAP J2EE server has the option to support distributed transactions and therefore requires the *two-phase commit* support of the JDBC driver.

2.2.4 New DB2 features and DB2 Connect

SAP benefits from new DB2 features without changing the code. Three years ago the ICLI changed the API to the SAP database interface. The new ICLI API is ODBC level 3 compliant. Since DB2 Connect is supporting the same API, no changes in the SAP code are necessitated by the change to the new connectivity. ICLI has always supported multi-row operations on the network. Because of this, the SAP database interface called the ODBC multi-row API functions in order to tell the ICLI client how many rows were affected by the next FETCH or INSERT call.

Exactly the same functions are necessary to tell DB2 Connect that the caller is now using multi-row operations. In addition to the multirow support on the network, DB2 Connect V8 FixPak 4 exploits the new multi-row features of DB2 for z/OS V8. We do not expect performance improvements on the network provided by DB2 Connect over the ICLI connection, but, because of the capability of DB2 Connect to send multiple rows through the network with one call, we expect much better performance out of the multi-row operations FETCH and INSERT. ICLI does not support the multirow functions of DB2 for z/OS V8.

2.2.5 New DB2 Connect features for ICLI compatibility

ICLI was especially designed to support SAP applications. During the lifetime of ICLI, many powerful features for problem determination in a complex SAP system were introduced. To have no functional disadvantages when compared to ICLI, DB2 Connect has implemented all missing features as compared to ICLI. As a result, the SAP customer should see no difference by changing the connectivity from ICLI to DB2 Connect.

In this section we describe some new DB2 Connect features implemented to provide equivalent functionalities to ICLI:

- ▶ **DRDA Ping Service:** The DRDA Ping Service provides the capability to measure latency performance. This feature is used in customer situations to easily check the latency of the network. IP Ping is not sufficient, because IP Ping might take a different route than the mainline application traffic, so it does not make sense to use it for evaluating application network performance. IP Ping for Windows does not differentiate turn-around times less than 10 ms, but reports them as <10. As SAP is targeting 1 ms latency, this is not good enough. The IP Ping unit of reporting is a millisecond, but a finer granulation is needed. The maximum message size of IP Ping is 8 KB (in both the NT and UNIX cases). DRDA Ping supports different request and response message sizes. The maximum message size is 40000 bytes (for both inbound and outbound).
- ▶ **Network statistics:** This feature provides the capability to monitor network performance in detail. Statistics are generated on the network performance between application servers and database servers. Several statistical values such as network turn-around time (request transmission time plus response transmission time) are measured. A snapshot is taken at defined time intervals. The default of the snapshot time interval is 300 seconds. The network statistics are collected at the granularity of an SAP application server. SAP CCMS is responsible to evaluate and store the network statistics for performance analysis and for historical trend analysis. Figure 2-4 displays the panel within the SAP transaction DB2 that allows you to start and stop collecting DB2 Connect-based network statistics.

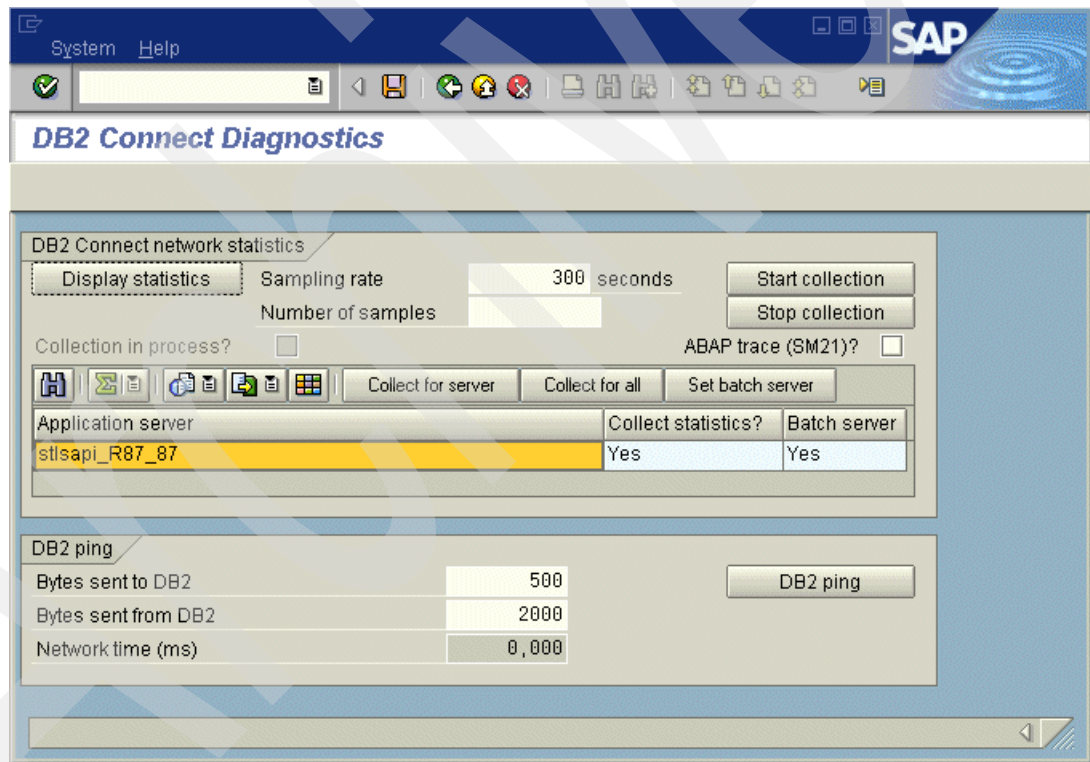


Figure 2-4 DB2 Connect: network statistics and database ping

- ▶ **Handling of TCP/IP error situations:** Concerning SAP applications, it is important that the database connectivity is able to adequately handle the TCP/IP error situation ENOBUFS. This error indicates insufficient resources available in the system to complete a TCP/IP call. Both DB2 and DB2 Connect are enabled to handle temporary buffer shortages. They implement a retry mechanism to shield this problem from applications. This function prevents SAP work from being interrupted.

- ▶ **Tracing feature:** The DB2 Connect developers or supporter are normally not familiar with SAP traces. To analyze DB2 Connect problems at the customer side, a CLI trace is necessary. DB2 Connect provides the same tracing flexibility as the ICLI. DB2 Connect is capable to trace CLI calls at the granularity of an SAP application server or on the granularity of an SAP work process.
- ▶ **ABAP statement information passed through:** There is a new CLI attribute, `SQL_ATTR_PROGRAMID`, that will be added to allow a CLI application to specify a user defined string that can be used to identify itself at the database server (DB2 V8). This new attribute can be set at the connection level by specifying the `PROGRAMID` as part of the CLI connection string or through the CLI API `SQLSetConnectAttr/SQLGetConnectAttr`. Also, it can be set at the statement level through the `SQLSetStmtAttr/SQLGetStmtAttr` APIs. CLI will associate the current connection level `PROGRAMID` with a prepared statement if no statement level `PROGRAMID` is specified. Whenever a prepare request is sent to the server, CLI will ensure that the correct `PROGRAMID` is also set at the server for that statement. The database server will then associate the `PROGRAMID` with any statements added to the dynamic SQL statement cache. SAP is using this feature to pass the ABAP statement information to DB2 for z/OS. This information is available if you look in the dynamic statement cache monitor through the SAP transactions DB2 or ST04. By just clicking a button, you can navigate to the ABAP source program and line number where the SQL statement was issued.
- ▶ **Accounting information:** DB2 Connect provides the C API function `sqlseti` to set client information specific to a connection (accounting string, transaction name, workstation name and client user ID). Starting with DB2 Connect V8 FixPak 4, DB2's correlation ID can be explicitly set through the new CLI attribute `SQL_ATTR_INFO_PROGRAMNAME`. This enables SAP to perform both DB2 accounting and to exploit WLM on the level of SAP end user IDs and SAP transactions/reports/batch jobs. Moreover, DB2 V8 introduces the feature to aggregate accounting records for certain DB2 identifiers. Also, it allows you to cut accounting records and to change the WLM enclave at commit boundaries.

2.2.6 Configuration of DB2 Connect and DDF for SAP

In this section we describe the necessary steps to install DB2 Connect to be used by SAP.

Installation of DB2 Connect V8.1

Before a customer starts an SAP Installation, SAP requires a successful installation of DB2 Connect V8.1 Unlimited Edition or any other version of DB2 Connect that satisfies the DB2 licensing conditions. For detailed documentation, see *Quick Beginnings for DB2 Connect Enterprise Edition*, GC09-4833, and *IBM DB2 Connect User's Guide*, SC09-4835.

The exact required FixPak level of DB2 Connect is kept up-to-date in SAP note 728743.

It is also assumed that a 64 bit instance was created that can be checked with the `db2level` command, and must show a response similar to this:

```
DB21085I Instance "db2inst1" uses "64" bits .....
```

To check the database manager configuration, you can use the command `db2 get dbm cfg`, which must show a response similar to this:

```
Node type = Database Server with local and remote clients
```

SAP environment variables for DB2 Connect

To run SAP using DB2 Connect, several parameters need to be set:

- ▶ Set the SAP profile parameter `db2/db2/ssid` (corresponding environment parameter: `db2_db2_ssid`) to the DB2 Connect database name (that is, SGE1).
- ▶ Set the SAP profile parameter of the RACF® user `db2/db2/user` (corresponding environment parameter: `db2_db2_user`) to R3USER, for example. Set the SAP profile parameter of the RACF user password: `db2/db2/pw` (corresponding environment parameter: `db2_db2_pw`) to SAPR3SAP, for example.
- ▶ Set the DB2 Connect environment parameter `DB2CODEPAGE` to 819 in a non-Unicode environment.

Instead of putting the user/password in the SAP profile or in the environment, you can use a secure password storage. SAP provides a feature to use a secure password storage. The settings needed to use this feature are as follows:

- ▶ Set the environment parameter `db2_db2_user` to your RACF user (`myuser`).
- ▶ Issue the UNIX command: `unsetenv db2_db2_pw` (or the corresponding command if you are using Windows). These settings must be the same as in the environment when starting R/3.
- ▶ Create a password with the SAP utility `dbdb2pwd`:
 - Issue the command: `dbdb2pwd -create pwd` to create a password. In this example, `pwd` is the password of your RACF user.
 - The passwords will be stored in `/usr/sap/<SID>/SYS/global/dbdb2pwd`.
 - To change the password, issue the command: `dbdb2pwd -create pwdnew`

DB2 Connect configuration for SAP

Binding the DBRMs

For functional reasons and optimal performance, SAP needs specialized bind options to be used when binding the DB2 Connect DBRMs on DB2 for z/OS.

First connect to the DB2 subsystem using the following command:

```
db2 connect to <database alias> user <user> using <pass word>
```

Then, you need to run the following bind command and create a new collection, in this example, called SAPDRDA:

```
db2 bind /usr/opt/db2_08_01/bnd/@ddcsmvs.1st ACTION REPLACE KEEPYNAMIC YES GENERIC
\ "DEFER PREPARE\ " REOPT ONCE COLLECTION SAPDRDA ISOLATION UR BLOCKING UNAMBIG
RELEASE COMMIT GRANT PUBLIC SQLERROR CONTINUE messages ddcsmvs.msg
```

The reason why you need to create a new collection is the following:

The CLI packages require special handling since they are shared by the JDBC driver as well, therefore many bind options are blocked on the default NULLID collection. In order to override the default bind options, you must specify a different COLLECTION which will create a new set of packages with these options. The `CurrentPackageSet` keyword then must be set for CLI to use this new collection.

The SAP database interface issues the `SQLSetConnectAttr` CLI call to be able to use the new collection using the Connection Attribute `SQL_ATTR_CURRENT_PACKAGE_SET`.

db2cli.ini

Currently, we need to set the keyword in the `db2cli.ini` file as follows:

```
KEEPYNAMIC=1
```

This is an undocumented parameter (as of FixPak 3) but it is currently necessary to be able to exploit dynamic statement caching on z/OS. This parameter will be documented with DB2 Connect V8.1 FixPak 4, and SAP database interface will issue the following calls:

```
SQLSetConnectAttr( hDbc,  
SQL_ATTR_KEEP_DYNAMIC,  
(SQLPOINTER) 1,  
SQL_IS_UIINTEGER );  
  
SQLSetConnectAttr( hDbc,  
SQL_ATTR_CURRENT_PACKAGE_SET,  
(SQLPOINTER) "SAPDRDA",  
SQL_NTS );
```

This will cause SAP to use the special collection (SAPDRDA in the example above) which was bound with KEEPDPYDYNAMIC YES and to exploit dynamic statement caching for its connection.

DB2 catalog statements

Each SAP database that you want to access needs to be catalogued in DB2 using the following set of commands. In the sample we assume that the DB2 SSID is SGE1, the remote z/OS host (which DDF is running on port 5111) is ihsap3, and the remote location (which can be queried on z/OS with DISPLAY DDF) is SGE1.

```
db2 catalog dcs database SGE1 as SGE1 PARMS \",,INTERRUPT_ENABLED\  
db2 catalog tcpip node SGE1SAP3 remote IHSAP3 server 5111  
db2 catalog database SGE1 at node SGE1SAP3 authentication server
```

The parameter value INTERRUPT_ENABLED is important because it enables the cancelling of DB2 threads on the z/OS host. Not setting this parameter has several negative effects. It prevents SAP from interrupting long-running transactions which have exceeded their maximum allowed runtime, and also prevents users from stopping a long-running transaction via an SAPGUI window.

DB2CONNECT_IN_APP_PROCESS=NO

In order to enable network monitoring for SAP, we need to set the DB2 registry variable db2set DB2CONNECT_IN_APP_PROCESS to NO.

The following command shows which DB2 registry variables are currently set:

```
db2set -all
```

During runtime of an SAP work process, the SAP database interface sets this variable as an environment variable. This is done to compensate for the possibility that a customer has not set this variable in the DB2 registry and therefore cannot do any network monitoring.

Enabling SQL Break

To enable DB2 Connect to break running SQL statements, you must set the DB2 registry variable DB2CONNECT_DISCONNECT_ON_INTERRUPT with the command:

```
db2set DB2CONNECT_DISCONNECT_ON_INTERRUPT=ON
```

NUM_POOLAGENTS=0

As a default, DB2 Connect uses Connection Pooling. This has the side-effect that if you start and stop SAP, then the DB2 threads that were created on DB2 for z/OS stay active, although the remote application (SAP!) has long terminated. This happens because, as a default, DB2 Connect uses connection pooling and keeps these threads open, so it can quickly re-use them for new DB2 Connect applications. Connection Pooling is usually only needed for

applications which do lots of connects/disconnects, which is something that SAP usually does NOT do.

Furthermore, you cannot stop your DB2 for z/OS subsystem unless you have stopped the DB2 Connect instance completely and you have no easy means for detecting (on the z/OS side) if the active DB2 threads are working on behalf of “real” SAP work processes, or if they are just hanging around inactively for Connection Pooling purposes. So we recommend that you set the DB2 Connect parameter NUM_POOLAGENTS to disable DB2 Connect Connection Pooling:

```
db2 update dbm cfg using NUM_POOLAGENTS 0
```

You can use the following command to find out the current value of the Connection Pooling parameter for your instance:

```
db2 get dbm cfg | grep NUM_POOLAGENTS
```

Note: SAP recommends to use the **db2radm** tool to configure DB2 Connect and do the bind.

DB2 DDF ZPARMS for SAP

Each of the DB2 ZPARMs described in the following sections should be set to a value different from the default.

EXTSEC=YES

If you specify YES, detailed reason codes are returned to a DRDA level 3 client when a DDF connection request fails because of security errors. Also, RACF users can change their passwords using the DRDA change password function. This support is only for DRDA requesters that have implemented support for changing passwords.

We strongly recommend that you specify a value of YES. This allows properly enabled DRDA clients to determine the cause of security failures without requiring DB2 operator support. It also allows RACF users on properly enabled DB2 clients to change their passwords. Specifying NO returns generic error codes to the clients and prevents RACF users from changing their passwords.

IDTHTOIN=0

The default of IDTHTOIN is 120 (seconds) in DB2 V8. For SAP, this parameter should be set to 0, which disables the cancellation of idle DBATs. As SAP work processes, and their corresponding DB2 DBATs are long-running, and some of them are possibly idle, it is important to disable the cancellation of idle DBATs, otherwise problems might be experienced. IDTHTOIN=0 is a requirement, because this ensures that idle DB2 threads are not automatically canceled by DB2. This parameter is only relevant if CMTSTAT is set to INACTIVE, which is the default in DB2 V8 and which is going to be recommended by SAP. Due to KEEP DYNAMIC(YES), DB2 DBATs will remain active and therefore will be susceptible to idle thread timeouts even if the transaction has ended (commit or rollback) before the dormant period began.

DDF=AUTO

It is a good idea to let DB2 automatically load DDF when it is launched. This facilitates the handling of the DDF address space.

CMTSTAT=INACTIVE

The parameter CMTSTAT should be kept at its default value, which is INACTIVE, in DB2 V8. Even with CMTSTAT=INACTIVE, the DB2 DBATs that serve SAP remain active at all times, because SAP uses dynamic statement caching (due to SAP packages being bound with KEEP DYNAMIC YES), which prevents DB2 DBATs from becoming inactive. By enabling thread reuse (CMTSTAT=INACTIVE), DB2 is able to cut accounting records and re-establish enclaves at transaction (commit or rollback) boundaries. See also 6.6.1, “Rollup accounting data for DDF and RRS AF” on page 170 and 6.6.2, “Writing accounting records with KEEP DYNAMIC(YES)” on page 172.

2.3 Schema evolution

In this section we introduce DB2 V8 online schema evolution as it applies to the SAP application. SAP customers are challenged by the need to apply upgrades, service packages, and customer developed changes in a minimally disruptive manner.

2.3.1 Online schema changes

SAP upgrades and service packages are generally applied during a service outage and can alter the attributes of numerous SAP objects. The length of time required to implement these changes depends in large part on the speed with which the changes can be activated in the database. The ability to alter database objects without unloading and reloading the data can greatly reduce the outages associated with these changes.

Customer developed changes present a slightly different challenge. These changes can be generally grouped into the following three categories:

- ▶ Emergency changes
- ▶ Maintenance changes
- ▶ Project changes

Emergency changes fix an immediate problem with a critical business function for which no workaround exists. These changes are generally implemented into a running system irrespective of current workload or time of day. The ability to alter database objects directly can reduce the transient problems associated with emergency changes.

Maintenance changes fix and enhance the customer’s existing SAP implantation. These changes are generally batched together and implemented into a running system during periods of low intensity system activity. The ability to alter database objects directly can reduce the transient problems associated with maintenance changes.

Project changes are associated with the customer’s implementation of additional function onto an existing SAP implementation. These changes often required conversion and data loading and are generally applied during a service outage. The ability to alter database objects without unloading and reloading the data can greatly reduce the outages associated with these changes.

2.3.2 Online schema changes overview

Over the last versions of DB2, significant enhancements have already been implemented to reduce the window of application unavailability:

- ▶ Easier *code maintenance* is available with the introduction of DB2 data sharing. You can stop and start individual members (DB2 subsystems) to activate maintenance (PTFs) or a

new DB2 release, while applications continue to use the active members of the data sharing group.

- ▶ Another area in which much work has been accomplished is to have the data available as much as possible. *Data* requires *maintenance* every so often. DB2 utilities have come a long way across the last releases, for example, by introducing online REORG, inline copy and statistics, and online LOAD RESUME.

Schema maintenance

Starting in V8, DB2 takes on a new challenge, that is, to reduce the unavailability window when making changes to the data definition of DB2 objects (see Figure 2-5).

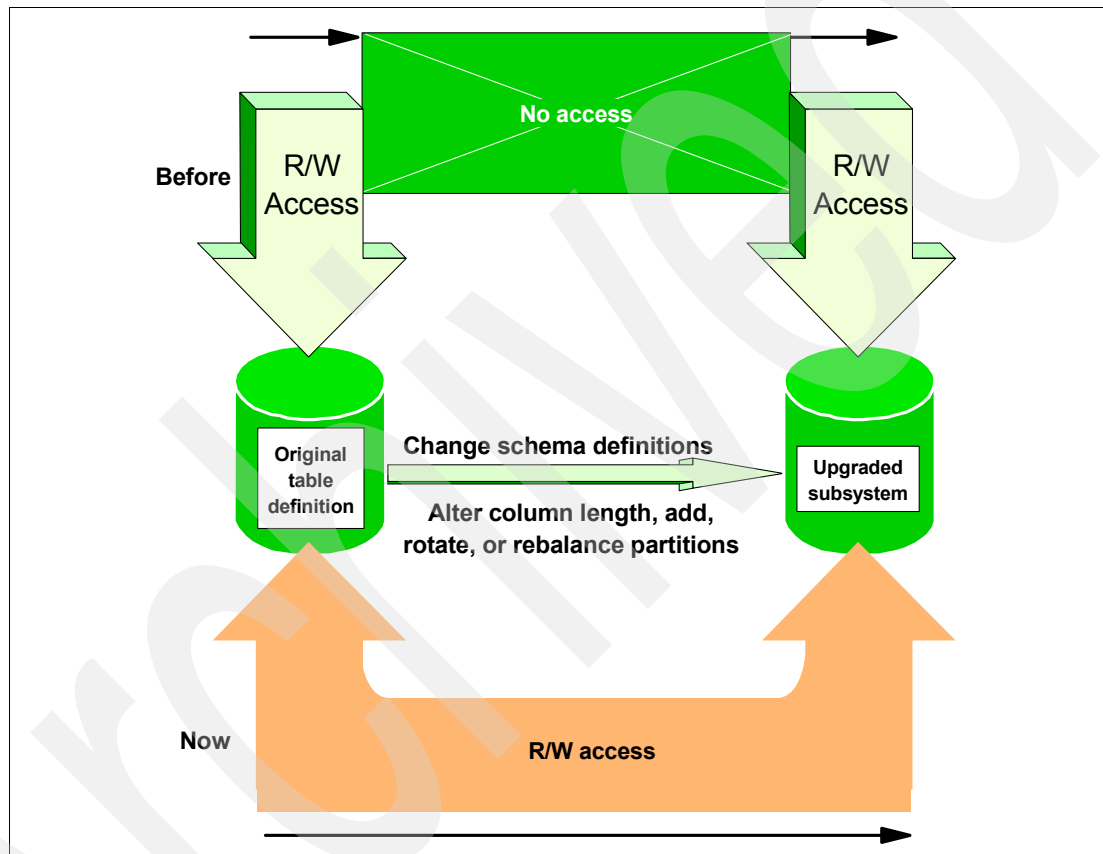


Figure 2-5 Online schema evolution

In the past, DB2 releases have implemented most DDL ALTER enhancements without actively addressing the problem of data unavailability while modifying object attributes.

Many changes to table, table space, and index schemas (DDL) in today's DB2 V7 require that SAP adhere to the following procedure to implement them:

1. Drop all dependent objects, such as tables, indexes, views, synonyms, and triggers.
2. Rename the table to QCM<table>.
3. Create the object with the new definition.
4. Reestablish authorization for the object.
5. Recreate all dependent objects, such as views and indexes, and their authorizations.
6. Copy the data from QCM<table> to <table>.
7. Test that all is OK.

However, some schema changes can already be done without having to drop and recreate an object, or stopping and starting the object, such as adding a column to a table, renaming a table, altering the primary and secondary quantity of an object, or changing partition boundaries.

As 24x7 availability becomes more critical for applications, the need grows for allowing changes to database objects reflected in the catalog and the DBD while minimizing the impact upon availability. We call this *Online Schema Evolution* (or Online Schema Changes or Online Alter). In an ideal world, this enhancement would provide support for changes to all object attributes without losing availability. DB2 V8 lays the groundwork for allowing many changes, while implementing a reasonable subset of these changes.

The following schema changes, allowed in DB2 V8, are applicable to the SAP application and are currently supported by the SAP dbsl:

- ▶ Extend CHAR(n) column lengths.
- ▶ Change type within character data types (CHAR, VARCHAR).
- ▶ Change type within numeric data types (SMALLINT, INTEGER, FLOAT, REAL, FLOAT, DOUBLE, DECIMAL).
- ▶ Change type graphic data types (GRAPHIC, VARGRAPHIC).
- ▶ Allow column data type changes for columns that are referenced within a view.
- ▶ Allow these column changes for columns that are part of an index (pending).

The following schema changes are allowed in DB2 V8 and are applicable to the SAP application, but must be implemented using database tools:

- ▶ Drop the partitioning index (or create a table without one).
- ▶ Change the clustering index.
- ▶ Add a partition to the end of a table which extends the limit value.
- ▶ Support automatic rebalancing of partitions during REORG.
- ▶ Support REORG of parts in REORG Pending states.
- ▶ Loosen the restrictiveness of indexes in recover or Rebuild Pending states.
- ▶ Add a column to an existing index.

2.3.3 Data type changes

After a table column data type is changed through an ALTER statement, the new *definition* immediately applies for all data in the associated table. No existing data is converted to the new version format. As rows are retrieved, they are materialized in the new format indicated by the catalog. Likewise, when a data row is modified or inserted, the entire row is saved using the new catalog definition.

When the object is reorganized, all rows are converted into the format of the latest version (see 2.3.5, “Versioning” on page 45).

To support changing the column data type of a column in an existing table, the SET DATATYPE clause of the ALTER TABLE ALTER COLUMN was enhanced to support these additional changes. The command is shown in Figure 2-6.

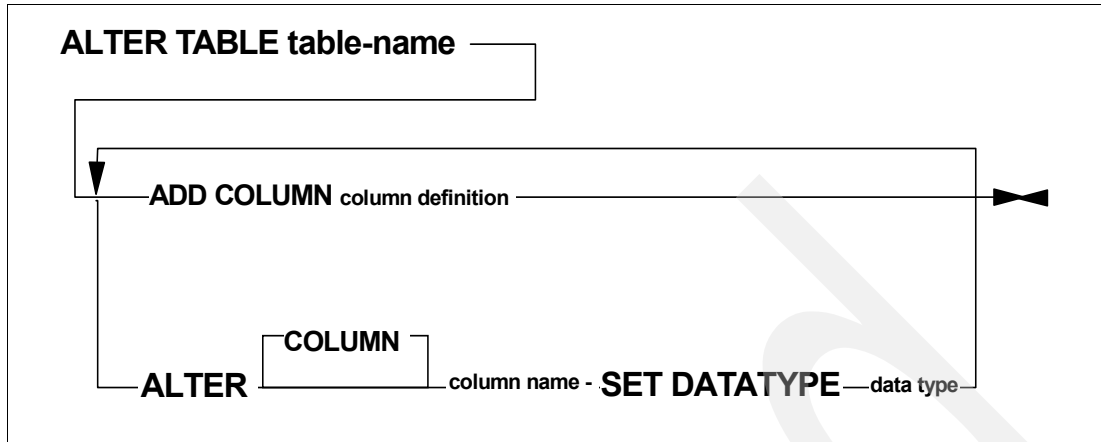


Figure 2-6 ALTER TABLE SET DATATYPE statement

A column data type may be altered if the data can be converted from the old type to the new without losing significance. This basically means that the new column definition has to allow for “larger” values than the current column definition.

Impact on dependent objects

The ALTER TABLE SET DATATYPE has different implications for different types of objects affected by it.

Table spaces

When the ALTER completes, the table space is placed in an Advisory REORG Pending (AREO*) state. See 2.3.7, “New DBET states for online schema changes” on page 48 for this new database exception state. Access to a table can continue with rows containing columns in multiple version formats, but there is a performance degradation, since altered columns have to be converted to the new format when they are accessed, and the full row is always logged, until the table space is reorganized. To reduce the performance impact, it is recommended to schedule a REORG after issuing the ALTER statement that changes the data type of a column in a table.

Indexes

When the data type or length of a column is altered on a table and that column is defined for an index, the index is altered accordingly. If a change is made to a non-indexed column, it results in a new table (and table space) version but not a new index version. For more information on versioning, see 2.3.5, “Versioning” on page 45. If a table has multiple indexes, the change of a table column results in a new table version and a new index version for each index that contains the column. Indexes created on different tables in the same table space or unchanged columns in the same table are not affected.

All new keys inserted are in the new index format. Changed columns which are included as part of an index key affect availability of the index according to the column data type. Whether or not the index is immediately available after a column in the index has incurred a data type change depends on the data type of the column being changed.

► Immediate index availability:

In DB2 V5, the ALTER TABLE statement was enhanced to provide the ability increase the length of VARCHAR columns. If an index on the altered table had a key containing altered columns, index versioning support allowed immediate access to the index.

With DB2 V8, this index versioning design is extended to support immediate access of indexes containing keys from all forms of fixed length or varying length character and graphic columns.

Changes for character data type columns result in immediate index availability. This includes columns defined as CHAR, VARCHAR, GRAPHIC, or VARGRAPHIC.

► **Delayed index availability:**

In some cases, supporting immediate changes with index versioning would result in severely degraded performance. To avoid this, the index is placed into Rebuild Pending (RBDP) instead (for both COPY NO and COPY YES indexes). Availability to the index is delayed until the index is rebuilt.

Changes for numeric data type columns are immediate with delayed index availability. This includes columns defined as SMALLINT, INTEGER, DECIMAL or NUMERIC, FLOAT, REAL, or DOUBLE. This poses a particular problem for SAP, given the integrated DWB and data dictionary. SAP cannot tolerate a change to an SAP object that causes an index to enter an RBDP state, and as of the 6.4 kernel level, these changes will not be allowed. The behavior under earlier kernels is for the activation to fail with an SQLCODE +610.

If an entire index is rebuilt from the data, all the keys are converted to the latest format. Utilities which may rebuild an entire index include:

- REBUILD INDEX
- REORG TABLESPACE
- LOAD REPLACE

If data type changes, reorganization of the entire index materializes all keys to the format of the latest version unless the index is in RBDP. In this state, access to the data is required to get the length of the key.

Scope of unavailability

To limit the scope of unavailability for dynamic SQL:

- Deletes are allowed for table rows, even if there are indexes in RBDP.
- Updates and inserts are allowed for table rows, even if their corresponding non-unique indexes are in RBDP state.
- Inserting or updating data rows that result in inserting keys into an index that is in RBDP state is disallowed for unique or unique-where-not-null indexes.
- For a select statement, DB2 does not choose an index in RBDP for an access path.

RUNSTATS

Some of the statistics get converted at the time of the ALTER (for instance, HIGH2KEY, LOW2KEY). Invalidation is done for distribution statistics in SYSCOLDISTSTATS and SYSCOLDIST, and DB2 sets the STATSTIME in SYSCOLUMNS to January 1, 0001, which signals the optimizer to ignore the distribution frequency statistics.

Cached dynamic statements

Cached dynamic statements referencing the changed table are invalidated. If auto-rebind is enabled, the plans and packages referencing the changed table space are automatically rebound during the next access if not manually rebound previously.

Views and check constraints

When a column is altered in a base table, the views that reference the column are immediately regenerated. If one of the views cannot be regenerated, then the ALTER TABLE statement fails on the first error encountered.

A change to any column within a view invalidates all plans, packages, and dynamic cached statements which are dependent on that view.

When a column data type is altered, the precision and scale of the decimal arithmetic result needs to be recalculated.

The value of the CURRENT PRECISION special register that is in effect for the ALTER TABLE is used to regenerate all the views affected by the altered column. Since a single CURRENT PRECISION setting is used for all the views, it is possible the ALTER TABLE can fail with an sqlcode -419 or complete with a precision calculated for view columns that does not work well for an application. In this case, the user has to DROP and CREATE the view in order to correct the problem.

If an ALTER TABLE fails because of a problem regenerating a view, the failing SQLCODE and tokens identifying which ALTER failed is returned and the entire ALTER TABLE statement fails.

If a check constraint is dependent on the column being altered, it is also “regenerated”. The regeneration may also fail in the case where different options are in use during the regeneration than the options in use at the time the check constraint was created. The options are the decimal point indicator and quote delimiter. The failing SQLCODE and tokens identifying which ALTER failed are returned.

2.3.4 Index changes

In DB2 V8, some index attributes can also be changed “in-flight” using an ALTER INDEX statement, without causing the index to become unavailable, as shown in Figure 2-7.

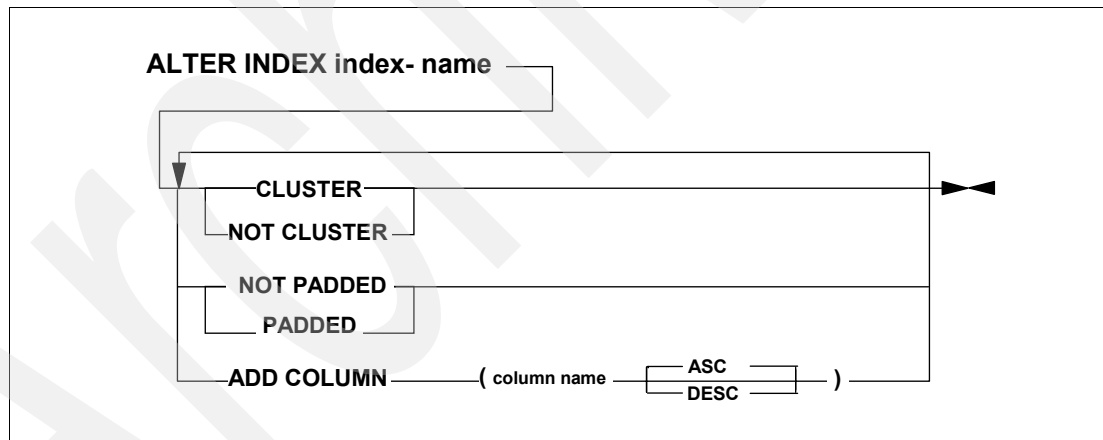


Figure 2-7 ALTER INDEX statement

Adding index columns

Columns can now be appended to the end of an existing index key with the ALTER INDEX statement. At the present time, SAP does not intend to take advantage of this capability.

Varying length index keys

In DB2 V8, varying length columns are *optionally* padded to their full length when they are part of an index key. When specifying NOT PADDED during the creation or altering of an index, padding does not occur and the keys are stored as true varying length keys. Varying length indexes are marked in the new SYSINDEXES column, PADDED, with a value of 'N'. NOT PADDED is the default for new V8 installations, while PADDED is the default when migrating from V7 for compatibility reasons. A new ZPARM, PADIX, can change the default.

From an SAP perspective, we have always set the VARCHAR from Index parameter (RETVLCFK) to NO because SAP cannot handle varchar data padded with blanks. SAP pays a performance penalty because of this. With DB2 V8, the SAP application will be able to take advantage of index-only access for VARCHAR fields in indexes once the indexes have been converted to NOT PADDED.

This conversion can be undertaken in a deliberate fashion that allows adequate time for testing and evaluation of the results. Indexes can be changed from *PADDED* to *NOT PADDED* using ALTER INDEX, as shown in the diagram in Figure 2-7. If the index has varying length columns, it is placed in Rebuild Pending state and a value of 'N' is placed in the PADDED column of SYSINDEXES. Once the index has been rebuilt, all the keys are varying length, and the pending state is reset.

Changing the table clustering

In V8, there are two enhancements related to clustering:

- ▶ Specifying the CLUSTER keyword for a secondary index in a partitioned table space. Historically, the partitioning index for partitioned tables also had to be the clustering index. These two attributes are now unbundled so that the clustering attribute can be assigned to a secondary index.
- ▶ Changing the clustering order in a partitioned or non-partitioned table space without dropping the index. The clustering attribute of an index can be modified by using the *CLUSTER* and *NOT CLUSTER* options of *ALTER INDEX*. As before, only one clustering index is allowed for any table.

If no explicit clustering index is specified for a table, the REORG utility now recognizes the first index created on each table as the implicit clustering index when ordering data rows.

If explicit clustering for a table is removed (changed to NOT CLUSTER), that index is still used as the implicit clustering index until a new explicit clustering index is chosen.

When the clustering index is changed, new INSERTs are immediately placed using the new clustering order. Preexisting data rows are not affected until the next reorganization rearranges them all to be in clustering order.

2.3.5 Versioning

To support online schema evolution, DB2 has implemented a new architecture, called *versioning*, to track object definitions at different times during its life by using versions.

Altering existing objects may result in a new format for tables, table spaces, or indexes that indicate how the data should be stored and used. Since all the data for an object and its image copies cannot be changed immediately to match the format of the latest version, support for migrating the data over time in some cases is implemented by using versions of tables and indexes. This allows data access, index access, recovery to current, and recovery to a point in time while maximizing data availability.

Versioning existed before DB2 V8 for indexes (after an indexed VARCHAR column in a table had been enlarged). It was tracked using the IOFACTOR column of SYSINDEXES. In DB2 V8, the first ALTER that creates a new index version switches to DB2 V8 versioning by setting the OLDEST_VERSION and CURRENT_VERSION columns to the existing versions in the index. And to support the table data type changes mentioned before, versioning in V8 is also implemented for tables and table spaces.

Version-generating ALTER statements

The following statements result in a new version for the affected tables and/or indexes:

```
ALTER TABLE table-name ALTER COLUMN column-name SET DATA TYPE altered-data-type
ALTER INDEX index-name NOT PADDED
ALTER INDEX index-name PADDED
ALTER INDEX index-name ADD COLUMN column-name
```

Multiple ALTER COLUMN SET DATA TYPE statements in the same unit of work are included in one new schema version.

The following ALTER statements do not result in a new version:

```
ALTER TABLE table-name ADD PARTITION ENDING AT constant
ALTER TABLE table-name ALTER PARTITION n ENDING AT constant
ALTER TABLE table-name ALTER PARTITION ROTATE FIRST TO LAST
ALTER TABLE table-name ADD PARTITIONING KEY column-name
ALTER INDEX index-name NOT CLUSTER
ALTER INDEX index-name CLUSTER
```

The following cases also do not generate a new version:

- ▶ When the table space or index was created as DEFINE NO and contains no data
- ▶ When a varying character or varying graphic column length is extended (but it can create an index version)
- ▶ When an ALTER TABLE specifies the same data type and length, so the definition is not changed
- ▶ When an ALTER TABLE ADD COLUMN of a version 0 table is specified

Version limits

A table space can have up to 256 different active versions while an index can have up to 16 different “active” versions (“active” versions include those within the pageset and all available image copies).

In regard to the range of *active versions* (which are all versions that exist for rows in the page set itself as well as the versions that exist in image copies registered in SYSCOPY) — if the maximum number of active versions is reached, the SQL statement fails with an SQLCODE -4702.

Unaltered objects remain at version 0 (zero).

Storing version information

The version information is stored in the DB2 catalog as well as inside the page set system pages.

Version information in the DB2 catalog

As can be seen in Figure 2-8, versioning information for an object is kept in the catalog tables SYSIBM.SYSTABLESPACE, SYSIBM.SYSTABLEPART, SYSIBM.SYSINDEXES, SYSIBM.SYSINDEXPART, SYSIBM.SYSTABLES, and SYSIBM.SYSCOPY.

In addition, the new catalog table SYSIBM.SYSOBDS, when there is more than one active version, contains one row for each OBD or index that can be recovered to an image copy that was made before the first version was generated for that OBD or index.

	OLDEST_VERSION	CURRENT_VERSION	VERSION	VERSION 0 DATA
SYSTABLESPACE	X	X		
SYSTABLEPART	X			
SYSTABLES			X	
SYSINDEXES	X	X	X (data version)	
SYSINDEXPART	X			
SYSCOPY	X			
SYSOBDS				X

Oldest for pageset and all available copies - updated by utilities such as MODIFY and REORG
 Used to allocate next number

- Note that versioning is tracked at table space and index level
 - ▶ Each table in a segmented table space will be assigned a different version number

Figure 2-8 Versioning information in the DB2 catalog

A table space starts out with all data in tables at version zero. When an ALTER creates a new version, it gets the next available number after the active table space CURRENT_VERSION. Once version 255 is reached, numbering starts again with version 1 if it can be reclaimed. A version of 0 indicates that a version-creating ALTER statement has never been issued for the corresponding table or table space.

Versioning information inside the page set

The version information relevant to the data is stored inside the page set. Storing the version information inside the page set makes the objects self-defining. System pages can be included in incremental image copies if the SYSTEMPAGES YES keyword is specified.

Reclaiming versions

For table spaces, and indexes defined as COPY YES, the MODIFY utility must be run to update OLDEST_VERSION for either SYSTABLEPART and SYSTABLESPACE, or SYSINDEXPART and SYSINDEXES. If there are COPY, REORG, or REPAIR VERSIONS SYSCOPY entries (ICTYPE of V) for the table space, MODIFY updates OLDEST_VERSION to be the lowest value of OLDEST_VERSION found from matching SYSCOPY rows. If no SYSCOPY rows remain for the object, MODIFY sets OLDEST_VERSION to the lowest version data row or key that exists in the active pageset.

For indexes defined as COPY NO, a REORG, REBUILD, or LOAD utility that resets the entire index before adding keys updates OLDEST_VERSION in SYSIBM.SYSINDEXES to be the same as CURRENT_VERSION.

2.3.6 SAP usage considerations

As mentioned before, DB2 V8 takes the first steps to avoid outages due to schema changes. The SAP application imposes slightly more severe restrictions than DB2 V8, such as these:

- ▶ Data types must be compatible and lengths must be the same or longer.
- ▶ Numeric columns contained in indexes cannot be altered.

To minimize any performance degradation, schedule a REORG as soon as possible after ALTER.

Schedule RUNSTATS to repopulate the catalog with accurate column and index statistics.

2.3.7 New DBET states for online schema changes

In support of online schema changes, DB2 V8, besides using the existing RBDP state, introduces a new Database Exception Tables (DBET) state, Advisory REORG (AREO*). AREO* indicates that the table space, index, or partition identified should be reorganized for optimal performance.

The DISPLAY DATABASE command now shows the new DBET state AREO* for all objects.

2.3.8 Impact of online schema changes on user tasks

In this section we discuss the effects of online schema changes on the major tasks involved in using DB2.

Database design and implementation

Designing databases and objects for applications is more forgiving than in the past. The problem with underestimating the size of objects is lessened with the ability to change column data types without losing availability to the data. When designing applications, you can give more consideration to saving space and reducing the number of data sets up front without the fear of being locked in at a future point by initial schema decisions.

Up until DB2 V7, for an application that requires inserting time based data, a separate partition is often assigned to store the data for each month. The design usually leans toward allocating the maximum number of partitions allowed up front. This probably meant allocating 254 partitions (with most of them initially with minimum space allocation) so that the application had a life span of about 20 years before running out of partitions.

With DB2 V8, it is much easier to add partitions at a later date, so the plan may be to start out with 24 partitions, and then reevaluate partition needs within the next 12 to 18 months. This results in “managing” many fewer objects that today may be preallocated without being immediately used. The use of templates and LISTDEFS in your utilities, by dynamically adding new partitions and objects, will also contribute to the overall flexibility and manageability of the schema changes.

When designing an application that requires storing the data for only a certain amount of time, such as for legal reasons, consider a rolling partition design. Now that there is the ability to easily ROTATE and reuse partitions over time, it is easier to manage a limited number of partitions that are set up based upon dates.

Operational considerations

The creation of new versions for objects can degrade performance of existing access paths. Schema changes should be planned to balance the trade-off between performance and availability expectations within a customer environment. Typically, the best time to make schema changes to minimize the poor performance impact is before a scheduled reorganization of an object.

When rotating partitions of a partitioned table, consider the time needed to complete the DDL statement. The reset operation requires that the keys for these deleted rows must also be deleted from all non-partitioned indexes. Each NPI must be scanned to delete these keys; therefore, the process can take an extended amount of elapsed time to complete as each NPI is processed serially. This is also true when deleting rows from the partition.

Additional consideration must be given for the time needed to delete data rows if processing must be done a row at a time. Individual delete row processing is required for referential integrity relationships, when DATA CAPTURE is enabled, or when there are delete triggers.

Application programming

When making schema changes, applications are usually affected. Changes in the schema must be closely coordinated between database objects and applications to avoid “breaking” existing applications. For example, if a column is extended from CHAR(*n*) to CHAR(*n+m*), the processing application truncates the last *m* bytes if the application is not changed to handle the longer column.

2.3.9 Dynamic partitions

DB2 V8 has the ability to immediately add partitions, rotate partitions, and change the partitioning key values for table-controlled partitioned tables through the ALTER TABLE statement. Here we give a brief description of these enhancements.

Adding partitions

With DB2 V8, users are able to dynamically add partitions to a partitioned table. You can add partitions up to the maximum limit, which is determined by the parameters specified when the partitioned table was initially created. When you add a partition, the next available physical partition number is used. When objects are DB2 managed (STOGROUP defined), the next data set is allocated for the table space and each partitioned index.

When objects are user managed (USING VCAT), these data sets must be predefined. The data sets for the data partition and all partitioned indexes must be defined using the VSAM access method services DEFINE command for the partition to be added (that is the value of the PARTITIONS column in SYSTABLESPACE plus one), before issuing the ALTER TABLE ADD PARTITION statement. Notice that no partition number is supplied on the ALTER TABLE ADD PARTITION statement. The part number is selected by DB2 based on the current number of partitions of the table.

The newly added partition is immediately available.

The table is quiesced, and all related plans, packages, and cached statements are invalidated. This is necessary, as the access path may be optimized to read only certain partitions. Automatic rebinds will occur (if allowed), but you may wish to issue rebinds manually.

Since you cannot specify attributes like PRIQTY, the values of the previous logical partition are used. Therefore you probably want to run an ALTER TABLESPACE statement afterwards to provide accurate space parameters, before starting to use the newly added partition.

Rotating partitions

Rotating partitions allows old data to “roll off” while reusing the partition for new data with the ALTER TABLE ALTER PARTITION ROTATE FIRST TO LAST statement. In a typical case, 13 partitions are used to continuously keep the last 12 months of data. When rotating, one can specify that all the data rows in the oldest (or logically first) partition is to be deleted, and then specify a new table space high boundary so that the partition essentially becomes the last logical partition in sequence ready to hold the data which is added. Because the data of the partition being rolled off is deleted, you may want to consider running an unload job before rotating the partition.

The partition that was rolled off is immediately available after the SQL statement is successfully executed. No REORG is necessary.

After using the new ALTER PARTITION ROTATE statement, the logical and physical order of the partitions is no longer the same. The DISPLAY command lists the status of table space partitions in a logical partition. Logical order is helpful when investigating ranges of partitions which are in REORP. It enables one to more easily see groupings of adjacent partitions that may be good candidates for reorganization. When used in conjunction with the new SCOPE PENDING keyword of REORG, a reasonable subset of partitions can be identified if one wants to reorganize REORP ranges in separate jobs.

Changing partition boundaries

In DB2 V6, the ability to modify limit keys for table partitions was introduced. The enhancement in DB2 V8 introduces the same capability for table-based partitioning with the ALTER TABLE ALTER PARTITION ENDING AT statement. The affected data partitions are placed into REORG Pending state (REORP) until they have been reorganized.

Rebalancing partitions

Rebalancing partitions is done by means of the REORG TABLESPACE utility. Specifying the REBALANCE option, when specifying a range of partitions to be reorganized, allows DB2 to set new partition boundaries for those partitions, so that all the rows that participate in the reorganization are evenly distributed across the reorganized partitions. (However, if the columns used in defining the partition boundaries have many duplicate values within the data rows, even balancing is not always possible.)

Rebalancing is ideal when no skewing of data between partitions is required, or needs to be catered for. It has an advantage over changing the partition boundaries using the ALTER TABLE ALTER PARTITION...ENDING AT statement, in that the partitions involved in the rebalancing are not put into REORP status (as in the case of the ALTER TABLE ALTER PARTITION... ENDING AT statement).

You are allowed to specify REBALANCE with REORG TABLESPACE *SHRLEVEL REFERENCE*, but you cannot specify REBALANCE with REORG TABLESPACE *SHRLEVEL CHANGE*. Also, you cannot specify partitioned table spaces with LOB columns. Also, notice that when the clustering sequence does not match the partitioning sequence, REORG must be run twice; once to move rows to the right partition; and secondly to sort in clustering sequence. DB2 leaves the table space in AREO* (Advisory REORG Pending) state after the first REORG, indicating that a second one is recommended.

Upon completion, DB2 invalidates plans, packages, and the dynamic statement cache that reference the reorganized object.

2.4 64-bit virtual storage

In this section we discuss the support of 64-bit virtual storage in DB2 V8. SAP applications tend to rely on large amounts of virtual and real storage both on the application server tier and at the database backend. Therefore, the 64-bit virtual storage exploitation by DB2 V8 is capable of providing relief for existing virtual storage constraints and enhanced scalability. Since caching data in buffer pools contributes strongly to a well-performing DB2 system, the DB2 V8 support of larger buffer pools has the potential to yield significant performance improvements to SAP systems.

This section contains the following topics:

- ▶ Expansion of DB2 virtual storage
- ▶ Enlarged storage pools
- ▶ Storage monitoring
- ▶ Thread storage contraction

2.4.1 Expansion of DB2 virtual storage

OS/390 R10 and z/OS have provided the 64-bit *real storage* addressability needed to scale in real memory addressing. While OS/390 R10 has the ability to run in either 31-bit mode or 64-bit mode on a zSeries, z/OS only runs in a 64-bit mode real storage environment. z/OS 1.2 and later releases provide *virtual storage* exploitation of the addressing range above 2 GB.

Basically, R10 has provided the initial z/Architecture™ real addressing support of up to 128 GB of central storage. Also, z/OS 64-bit real storage support has provided significant and transparent reduction of paging overhead, now only to disk, and real storage constraint relief for workloads limited by the previous maximum support of 2 GB of real storage. The elimination of Expanded Storage support has been handled by z/OS with minimal customer impact while reducing memory management overhead. For more information on 64-bit real exploitation, see the z/OS migration Web site at:

<http://www.ibm.com/servers/eserver/zseries/zos/installation/>

In z/OS V1.2, IBM has delivered the initial *64-bit virtual storage* management support. With the new z/OS 64-bit operating environment, now an application address space can have *2 to the power of 64* (or 2^{64}) virtual addresses with backing by real storage as needed. Figure 2-9 gives a pictorial representation of the evolution of the memory management from the 24-bit to the 31-bit to the 64-bit support.

DB2 V8 allows existing 31-bit DB2 applications (including those written in Assembler, PL/I, COBOL, FORTRAN, C/C++ and Java), and future DB2 applications to transparently benefit from the DB2 64-bit virtual storage support. The benefit is derived from the efficiencies with the local availability of data made possible through 64-bit data addressability. DB2 z/OS V8, which exclusively runs in 64-bit mode, can only execute on IBM @server zSeries hardware running z/OS V1R3 or later. DB2 V6 and V7 already support 64-bit real storage addressing for data space buffers, providing improved scalability and performance in a zSeries processor running in 64-bit real mode. Using 64-bit real provides a significant storage relief for most customers.

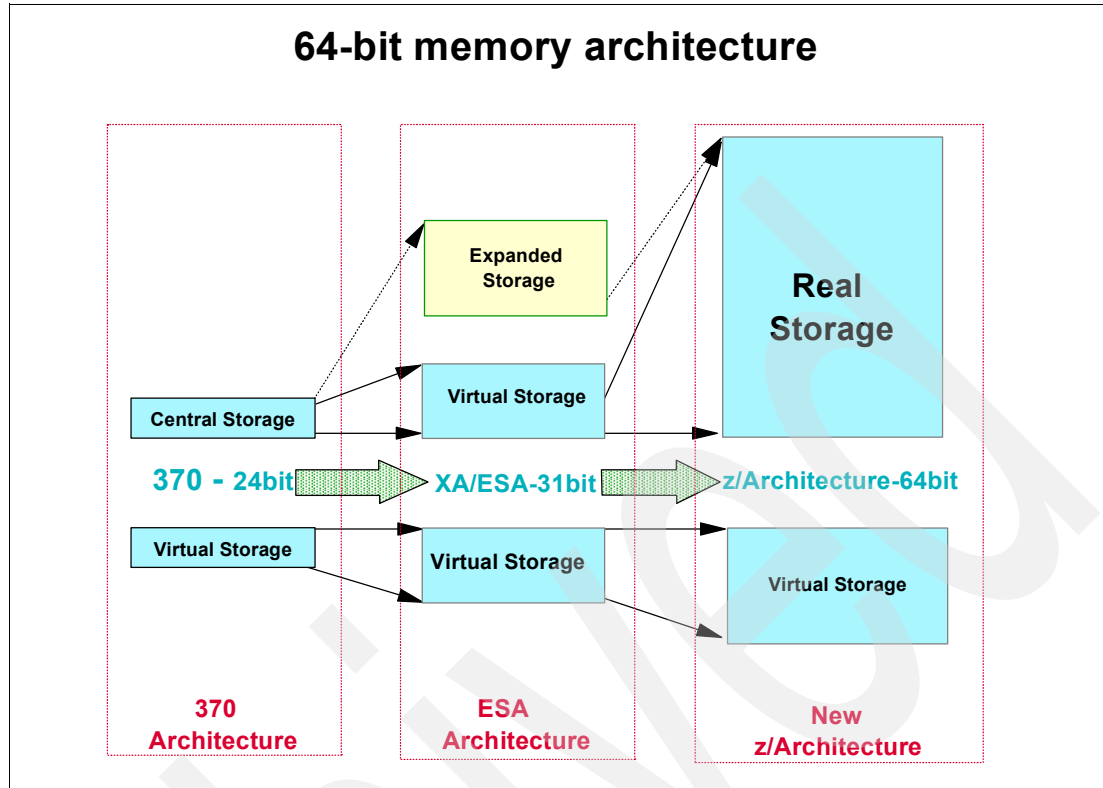


Figure 2-9 Evolution of z/OS memory architecture

Over the years, virtual storage usage has grown dramatically in DB2's DBM1 address space. This storage growth has been fueled by larger workloads, new functions, faster CPUs, and larger real storage available on mainframe processors. The latter, in particular, has allowed customers to run workloads that in the past would have been limited by paging overhead. Hence, 64-bit virtual addressing largely improves the DB2 scalability.

2.4.2 Enlarged storage pools

DB2 V8 has made massive changes to its code and now provides a solution to the current virtual storage constraint by utilizing 64-bit virtual addressing to move the following data areas above the 2 GB bar (2^{31}) in the DBM1 address space:

- ▶ Buffer pools and buffer pool control blocks
- ▶ DSC and DBD portions of the EDM pool
- ▶ RIDLIST portion of the RID pool
- ▶ Compression dictionaries
- ▶ Sort pool
- ▶ Castout buffers

Materialized LOB values were in data spaces before DB2 V8, now they are above the bar. In addition, IRLM locks and a sizeable portion of the data set control blocks are moved from below the 16 MB line to 31-bit virtual storage. Furthermore, faster short prepares enable reducing MAXKEEPD, which lifts virtual and real storage constraints.

Larger buffer pools

With the very large and ever-cheaper main memories that are available on the current and upcoming z/Architecture machines (currently 10s of GB, moving towards 100s of GB), it is becoming feasible for customers to configure very large buffer pools to gain significant

performance advantages. However, due to DBM1 virtual storage constraints, before V8, DB2 enforced maximum buffer pool sizes that were far less than the memory capacities of these machines:

- ▶ The total size of virtual pools is limited to 1.6 GB. However, in actual practice, customers typically cannot configure more than 1.0 GB due to DBM1 virtual storage constraints. VSAM control blocks and compression dictionaries are other sizeable contributors to the demands on DBM1.
- ▶ DB2 V7 limits the total size of hiperpools to 8 GB. This limit could be raised; however, hiperpools have several drawbacks which make them undesirable as a long term solution:
 - They are only page addressable, not byte addressable, and therefore buffers must be moved into the virtual pool before they can be used.
 - They can contain only clean pages.
 - You cannot do I/O directly into or out of a hiperpool.
 - The hiperpool page control blocks (HWBs) reside in the DBM1 address space and thus contribute to virtual storage constraints.
 - Hiperpools require a fairly substantial virtual pool size for effective use. Typically, the hiperpool to virtual pool size is on the order of 2:1 to 5:1. Therefore, virtual pool size ultimately limits hiperpool size.
 - A separate set of latches is used to manage hiperpool and virtual pool buffers, so as the frequency of page movement between virtual pools and hiperpools increases, the Least Recently Used (LRU) management of these pools increases, and latch contention issues can quickly arise.

Hiperpools were designed over a decade ago to exploit ESA and to make efficient use of large amounts of expanded storage.

Data spaces provided a good short term solution by exploiting the 64-bit real memory support introduced in OS/390 V2R10. DB2 V6 and DB2 V7 could place buffer pools and statement caching in data spaces, thus freeing up space for other work in the DBM1 address space. A performance penalty was paid when such buffering was not 100% backed by real storage, though.

These were some of the advantages of data spaces over hiperpools:

- ▶ Read and write cache with direct I/O to data space
- ▶ Byte addressability
- ▶ Large buffer pool sizes (32 GB for 4 KB page size and 256 GB for 32 KB page size)
- ▶ Excellent performance experienced with z900 and large processor storage
- ▶ Performance dependent upon being in 64-bit REAL mode

With the z/Architecture processors running in 64-bit addressing mode and having no expanded storage (all storage is central), hiperpools have no reason to exist.

The total size of data space virtual pools is limited to 32 GB (4 KB page size). This limit is imposed by a maximum of 8 million “page manipulation blocks” (PMBs) which reside in the DBM1 address space. Also, the lookaside pool resides in DBM1. Although data spaces provide a good short term solution for exploiting 64-bit real memory, they are undesirable as a long term solution, not only because of the size limitations, but also because of the overhead involved with copying buffers between the data spaces and the lookaside pool as they are accessed and updated. Data spaces have scalability issues, and the VSTOR limit of 2 GB for DBM1 address space remains the biggest constraint to achieving linear scalability (see Figure 2-10).

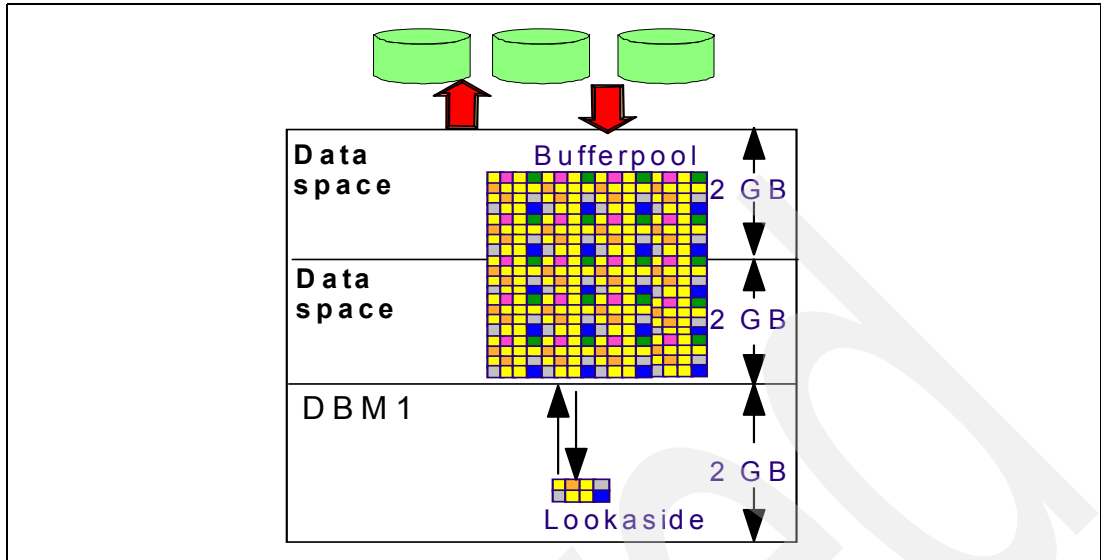


Figure 2-10 Data spaces before DB2 V8

The use of 64-bit virtual addressing greatly increases the maximum buffer pool sizes. DB2 V8 is 64-bit exclusive, and therefore always allocates the buffer pools above the 2 GB bar. This effectively eliminates the need for hiperpools and data space pools and simplifies DB2 systems management and operations tasks. Therefore, hiperpools and data space pools are no longer supported in DB2 V8. As of DB2 V8, the terms *buffer pool* and *virtual pool* become synonymous.

Buffer pools can now scale to extremely large sizes, constrained only by the physical memory limits of the machine (64-bit allows for 16 exabytes of addressability). System consolidation of identical SAP systems or by exploiting SAP MCODE (multiple components in one database) is facilitated as the buffer pools scale virtually unlimited. The recommendation still stands that buffer pools should not be over-allocated relative to the amount of real storage that is available. DB2 V8 issues the following warning messages when necessary:

- ▶ **DSNB536I:** This indicates that the total buffer pool virtual storage requirement exceeds the size of real storage of the z/OS image.
- ▶ **DSNB610I:** This indicates that a request to increase the size of the buffer pool will exceed two times the real storage, or the normal allocation of a buffer pool not previously used will cause an aggregate size which exceeds the real storage. Either request will then be limited to 8 MB (2000 pages for 4 KB, 1000 pages for 8 KB, 500 pages for 16 KB, and 250 pages for 32 KB).

DB2 limits the total amount of storage that is allocated for of buffer pools to twice the amount of real storage size in the system image.

DB2 V8 increases the maximum buffer pool sizes to the limit of the architecture, 1 TB, however the effective maximum is given by the real storage available:

- ▶ Maximum size for a single buffer pool is 1 TB.
- ▶ Maximum size for summation of all active buffer pools is 1 TB.

Figure 2-11 shows sample buffer pool settings that exploit larger buffer pools.

Database analysis Edit Goto System Help

Installation Parameters: DB2 UDB for z/OS

List format

Subsystem R870 at 12:12:54 29.09.2003 DB system DB2
 DB start 07:55:42 29.09.2003 DB release 8.1.5

Summary Buffer pools Data sharing Storage Tracing Locking Prote...

BP name	VP size	VPSEQT [%]	VPPSEQT [%]	VPXPSEQT [%]	DWQQT [%]	VDWQQT [%]	VDWQQT [p...
BP0	100.000	80	50	0	50	10	
BP1	250.000	80	50	0	50	10	
BP2	1.000.000	80	50	0	50	10	
BP3	1.500.000	80	50	0	50	10	
BP4	50.000	100	50	0	50	10	
BP40	200.000	80	50	0	50	10	
BP8K0	50.000	80	50	0	50	10	
BP16K0	50.000	80	50	0	50	10	
BP32K	75.000	80	50	0	50	10	

Figure 2-11 Buffer pools exploiting 64-bit virtual storage

When first migrating to V8, DB2 uses the following parameters to determine the size of the buffer pool:

- ▶ For data space pools and virtual pools with no corresponding hiperpool, the VPSIZE is used.
- ▶ For virtual pools with a corresponding hiperpool, VPSIZE + HPSIZE is used.
- ▶ VPSEQT, VPPSEQT, and VPXSEQT keep their previous values, even if the buffer pool size is determined by VPSIZE + HPSIZE.

DB2 V8 maintains the old V7 virtual pool and hiperpool definitions as they were at the time of migration to be used in case of fallback, and it adds new definitions of buffer pools for the catalog.

For newly installed V8 subsystems, as in prior releases, DB2 initially uses the buffer pool sizes that were specified during the installation process. Thereafter, the buffer pool attributes can be changed through the ALTER BUFFERPOOL command, and they are stored in the BSDS.

The buffer pool names (BP0, BP1, etc.) do not change. Neither do the page sizes: The options are still 4 KB, 8 KB, 16 KB, or 32 KB. The ALTER BUFFERPOOL command parameters that are no longer supported are VPTYPE, HPSIZE, HPSEQT, CASTOUT. If they are specified, just a warning message DSNB539I is issued. The other parameters remain unchanged besides new default values for DWQQT and VDWQQT. In DB2 V8 the default of the deferred write threshold (DWQQT) of a buffer pool is 30% instead of 50%. The default of the vertical deferred write threshold (VDWQQT) of a buffer pool is decreased from 10% to 5%.

DB2 V8 adds the new optional keyword PGFIX to the ALTER BUFFERPOOL command. If it is set to YES, then the buffer pool is long-term fixed in real storage. Page buffers are fixed when they are first employed since the buffer pool is allocated or expanded. By saving CPU cycles, this results in a performance improvement. In case PGFIX is set to NO, which is the default value, z/OS can page out buffer pool pages. Page buffers are fixed and unfixed in real storage across each I/O and GBP operation. To avoid situations where long-term fixed buffer pools exceed the real storage capacity of the z/OS image, DB2 ensures that these buffer pools do not exceed 80% of this capacity.

EDM pool

In DB2 V8 the EDM pool always consists of three separate pools. A new dynamic statement cache is created above the 2 GB bar. If dynamic statement caching is activated before DB2 V8, the statements are cached in the data space, if one is defined, or in the normal EDM pool if a data space is not defined. Now, cached statements are always cached in the EDM dynamic statement (DSC) cache pool above the 2 GB bar; see Figure 2-12. The DB2 ZPARM EDMSTMTC, which defaults to 102400 KB, specifies its size. As SAP relies heavily on dynamic statement caching, this approach ensures that there is always sufficient virtual storage available. Moreover, it facilitates the tuning of this pool, because no other objects compete with cached statements for storage in this pool.

Also, a new EDM DBD cache is created above the 2 GB bar. This gives the DBDs the needed space to grow and relieves contention with other objects in the EDM pool. The size of the EDM DBD cache is defined by the ZPARM EDMDBDC, which defaults to 102400 KB. Again, due to the large number of objects that comprise SAP systems, this feature particularly benefits SAP applications.

Plans and packages remain in the EDM “main pool” below the 2 GB bar. This does not pose a problem to SAP, as it exclusively employs dynamic SQL statements. Therefore, its plans and packages are miniscule. The ZPARM that controls the size of the EDM pool is, as before, EDMPOOL. The new default of EDMPOOL is 32768 KB.

The default values of these three pools are large enough so that they promise to be good initial values for SAP systems, which enhances DB2 usability for SAP customers.

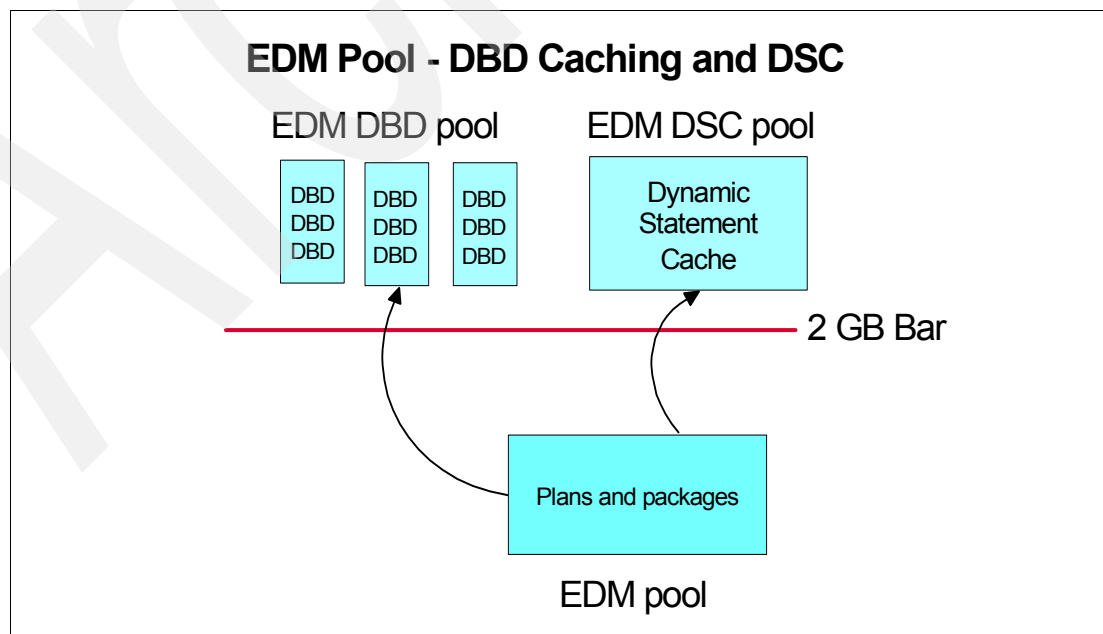


Figure 2-12 DB2 V8 EDM pool

RID pool

The RID pool has proven to be important for SAP system. The general recommendation is to initially set its size to 100 MB for DB2 subsystems that serve SAP rather than the default value of 4 MB (DB2 ZPARM MAXRBLK).

DB2 V8 splits the RID pool into two parts. A small part of the RID pool remains below the 2 GB bar and the majority (about 75%) is moved above; see Figure 2-13. The RID pool below the 2 GB bar stores the RID maps which are small in number, and the RID pool above the 2 GB bar contains the RID lists which comprise the bulk of the RID pool storage. Therefore, the amount of storage that the RID pool requires below the 2 GB bar is small.

Because of the changes, there are some slight modifications in estimating the size for the RID pool. The same size RIDMAPs hold half as many RIDLISTs as in DB2 V6 or DB2 V7. The RIDMAP size is doubled to accommodate the same number of 8 byte RIDLISTs, and each RIDLIST now holds twice as many RIDs. Each RIDBLOCK is now 32 KB in size.

Here is the new RIDPOOL calculation:

- ▶ Each RIDMAP contains over 4000 RIDLISTs.
- ▶ Each RIDLIST contains 6400 RID entries.
- ▶ Each RIDMAP/RIDLIST combination can then contain over 26 million RIDs, versus roughly 13 million in previous DB2 versions.

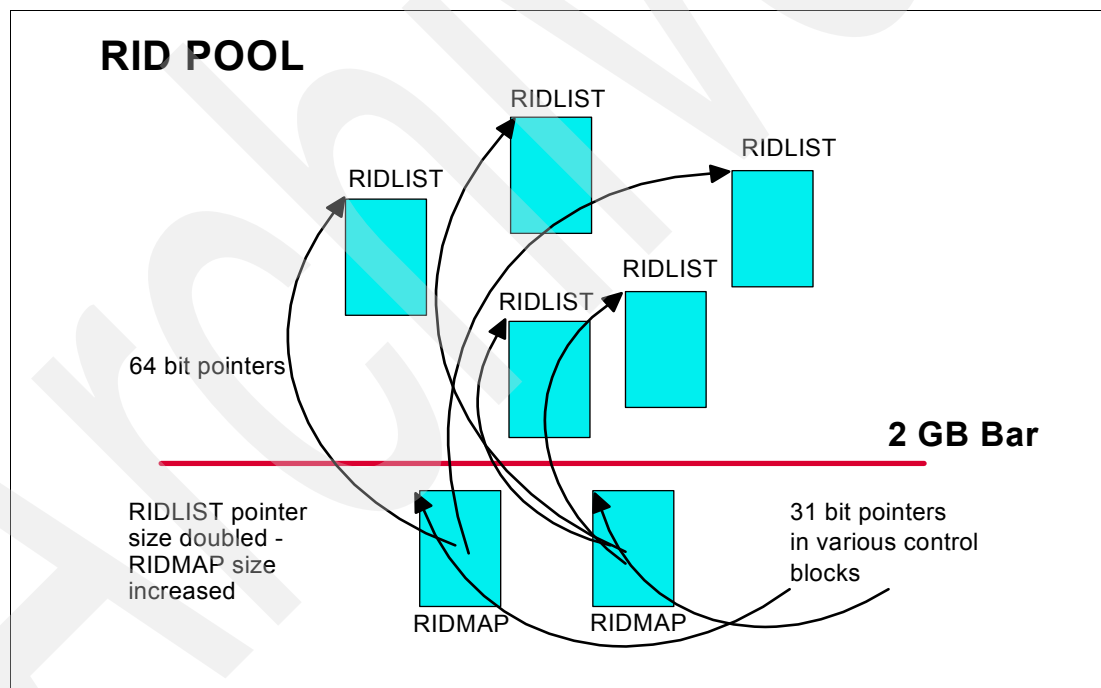


Figure 2-13 DB2 V8 RID pool

Compression dictionaries

The compression dictionary for a compressed table space or partition is loaded into virtual storage for each compressed table space or partition as it is opened. Even for tablespaces not accessed frequently, it occupies a good chunk of storage while the data set is open. A compression dictionary can occupy up to 64 KB bytes of storage per data set (sixteen 4-KB pages); therefore, moving the dictionary above the 2 GB bar provides significant storage relief for many customers. Due to the thousands of objects that often are concurrently open at SAP systems, this is particularly relevant to SAP systems.

For some customers, those who have a large number of compressed table spaces, the compression dictionaries can use up as much as 500 MB. This can further increase compression dictionary storage requirement for some systems, depending upon how many of these data sets contain compressed table spaces. DB2 V8 also implements support for 4096 partitions for a single table or index, this is another driver for moving compression dictionaries above the 2 GB bar. With 4096 partitions, customers might choose to have a larger number of smaller partitions resulting in a corresponding increase in the total number of compression dictionaries in those partitioned database.

The compression dictionary is loaded above the bar after it is built. All references to the dictionary now use 64-bit pointers. Compression uses standard 64-bit hardware compression instructions. Standalone utilities still load the dictionary below the bar.

Sort pool

Sorting requires a large amount of virtual storage, as there can be multiple copies of the data being sorted at a given time. Two kinds of storage pools are used for DB2 sort to store various control structures and data records. One pool is an agent-related local storage pool, and the other is a global sort pool. To take advantage of the 64-bit addressability for a larger storage pool, some high level sort control structures remain in agent-related storage below the 2 GB bar, but these structures contain 64-bit pointers to areas in the global sort pool above the 2 GB bar. The sort pool above the 2 GB bar contains sort tree nodes and data buffers. Therefore, it consumes less storage below the 2 GB bar and contributes to virtual storage constraint relief in this critical storage area.

LOB data

LOBs are now materialized, depending on the application requirements, above the 2 GB bar in DBM1 address space, and allocated in areas limited by the system parameters previously used for allocating data spaces:

- ▶ **LOBVALA (the size per user):** In DB2 V8, the default is raised to 10240 KB. The limit value remains to be 2097152 KB.
- ▶ **LOBVALS (the size per system):** The default is 2048 MB, the limit value is 51200 MB.

Data set control blocks

Each data set that DB2 opens prior to V8 requires some storage below the 16 MB line, which is critical storage. Due to the large number of objects in SAP systems, the amount of storage required for this purpose may add up significantly. Also, the maximum number of concurrently open data sets, which is governed by the DB2 ZPARM DSMAX, is restricted to 32767. This limit is imposed by system-generated DDNAMEs, which DB2 uses before DB2 V8.

DB2 V8 raises the maximum number of open data sets to 100000 with a new default value of 10000. It generates its own DDNAMEs, so the dynamic allocation limit of 32767 no longer applies. With z/OS 1.5 and above, the data set control blocks no longer reside below the 16 MB line, which provides more headroom below the 16 MB line. They are then located in 31 bit virtual storage. This enhancement ensures that the number of open data sets does not pose a storage problem. With z/OS 1.3 and 1.4, the data set control blocks still need to be below the 16 MB line.

The number of open data sets that are closed when DSMAX is approached is now the minimum of 3% of DSMAX and 300. This avoids bursts of data set close activity. Furthermore, if a large number of data sets are already open, data set open performance is substantially improved compared to previous DB2 releases.

Other 64-bit virtual storage changes

Other changes have been made to complement the 64-bit virtual support. There are 64-bit serviceability enhancements such as the 64-bit dump formatter and continued IPCS support. The locks now reside above the 2 GB bar. The default for IRLM has changed in V2.2. It was PC=NO; now PC=YES is enforced, which matches SAP's recommendation.

2.4.3 Storage monitoring

With DB2 V8 exploitation of 64-bit virtual storage, the following capabilities are possible:

- ▶ Buffer pool monitoring and tuning becomes simpler:
 - Hiperpools and data space pools are eliminated, thus reducing complexity. There is now only one type of buffer pool. Dynamic statement caching pool and LOB data spaces have been eliminated.
 - The ssnmDBM1 virtual storage constraints are no longer a key consideration in determining the optimum sizes for buffer pools.
- ▶ This may allow installations to increase the number of concurrent active threads (CTHREAD). ECSA allocation may need to be increased if CTHREAD is raised.
- ▶ A single DB2 subsystem is able to run larger workloads. This may cause some installations to defer going to a data sharing environment for capacity reasons (since data sharing is still required for the highest scalability and availability), or to consolidate the data sharing groups to fewer members.
- ▶ To handle the expected increases in workload, the maximum number of deferred write and castout engines are increased in order to decrease *engine not available* conditions.

You can use IFCIDs 0217 and 0225 to monitor ssnmDBM1 virtual storage usage above and below 2 GB.

VSTOR information is collected in SMF by RMF™ in record type 78-2. RMF can produce:

- ▶ Common storage summary and detail reports
- ▶ Private area summary and detail reports

The report are requested as follows:

- ▶ Specify S, either explicitly or by default: RMF produces summary reports.
- ▶ Specify D: RMF produces both summary reports and detail reports.

These are the available options:

- ▶ REPORTS(VSTOR(D)):
 - This produces a summary and detail report for common storage.
- ▶ REPORTS(VSTOR(D, ssnmDBM1)):
 - This produces a summary and detail report for common storage and a summary and detail report for the private area of the ssnmDBM1 address space.
- ▶ REPORTS(VSTOR(MYJOB)):
 - This produces a summary report for common storage and a summary report for the private area of the MYJOB address space.

More information on setting up and monitoring 64-bit is contained in the technical bulletin, *z/OS Performance: Managing Processor Storage in an all "Real" Environment*, available from:

<http://www.ibm.com/support/techdocs>

2.4.4 Thread storage contraction

Related to the exploitation of 64-bit virtual storage is a DB2 V8 enhancement in thread storage management. Since SAP uses long-running DB2 threads, the amount of storage that these threads acquire may accumulate notably unless freed storage segments are garbage collected on a regular basis.

New storage contraction thresholds

Already before V8, DB2 provides the installation parameter CONTSTOR that allows you to turn on the periodic contraction of the working storage area of each thread. This helps in relieving storage constraints in the DBM1 address space. Before V8, two hidden ZPARMs are associated with CONTSTOR that govern when storage is contracted. However, the default values of these hidden ZPARMs are too high for SAP systems. To prevent thread storage from growing too large, SAP recommends setting CONTSTOR to YES and also setting the hidden ZPARM SPRMSTH to a smaller value (1 MB).

DB2 V8 changes the thresholds that trigger thread storage contraction if storage contraction is activated. Storage is contracted for a thread after it has completed 10 transactions and when the storage that is consumed by a thread reaches 1 MB. This criteria matches the characteristics of SAP applications. For SAP applications, there is hence no need to manipulate the internal contraction thresholds of DB2 anymore. This results in less manual intervention and enhanced usability.

Reduced storage consumption by threads serving SAP

There are also improvements in the amount of storage that DB2 threads serving SAP typically consume. This is due to the fact that DDF threads in DB2 V8 require less user thread storage than DB2 V7 threads that use ICL1 as database connectivity. As SAP recommends it, CONTSTOR is assumed to be YES in both cases. This improvement contributes to virtual storage constraints relief.

Short prepares from the global dynamic statement cache are faster in DB2 V8 (see 5.6, “Faster DSC short prepares” on page 142). This reduces the pressure on MAXKEEPD. A stronger reliance on the global statement cache means a smaller local statement cache, which is part of thread storage. As a net result the required thread storage is reduced even further.

2.5 Java

In this section we describe how the SAP Java applications are using DB2 for z/OS to store their data.

2.5.1 SAP and Java

Starting with Web Application Server 6.30, shown in Figure 2-14, the SAP Java database layer is ported to DB2 for z/OS. This means that every SAP Java application that is using the SAP Java database layer is able to run on DB2 for z/OS. There is a big push within SAP to port every existing Java application to the new SAP Java database layer.

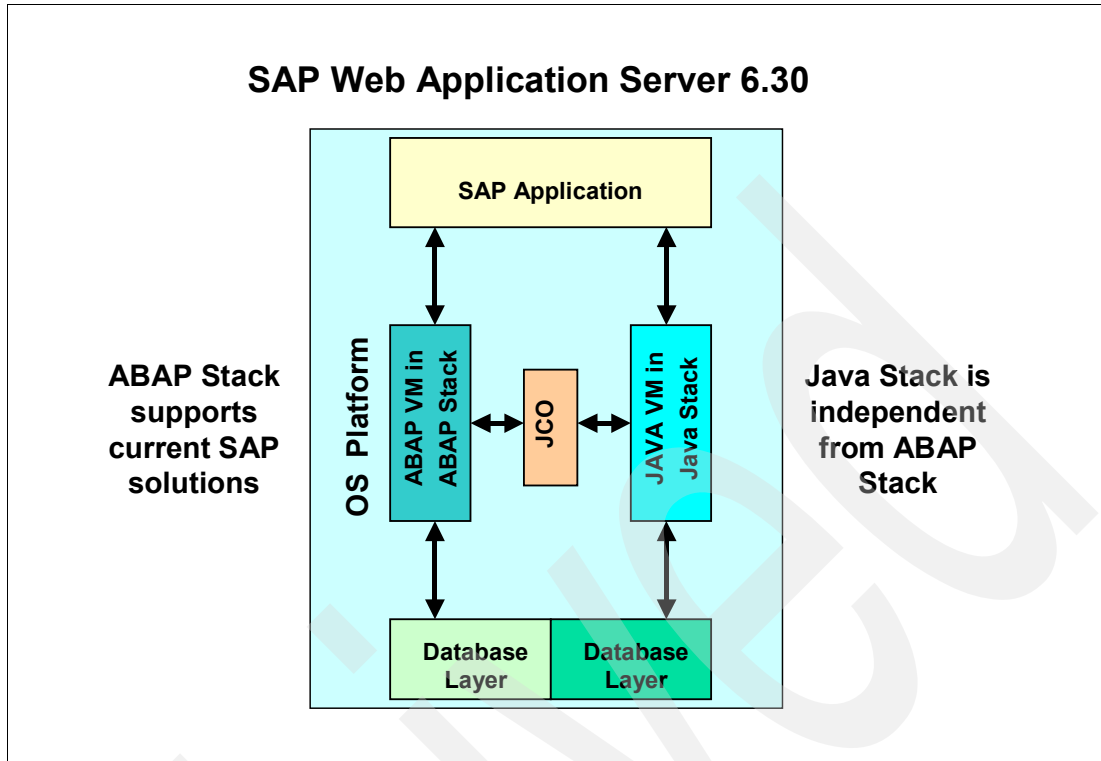


Figure 2-14 SAP Web Application Server

Figure 2-14 shows the high level architecture of SAP Web Application Server 6.30. An SAP application is either written in ABAP or in Java. An ABAP application cannot use the Java stack, and a Java application cannot use the ABAP stack. At installation time the customer has the choice to install the J2EE engine (including the Java stack) as add-in to the ABAP engine (including the ABAP stack). This means that both the ABAP engine and the J2EE engine will be started if the customer starts SAP. It also means that an ABAP program is able to communicate with a Java program via the SAP Java Connector (JCO) and vice versa. If a customer installs the J2EE engine standalone, these features are not available without re-configuration of the J2EE engine. The ABAP stack and the Java stack are using different database layers to access the database. The SAP Java database layer supports only Unicode as encoding scheme; whereas there are both a Unicode version and a non-Unicode (ASCII) version of the ABAP stack.

2.5.2 The SAP Java components

The SAP Java persistence is based on Unicode. Therefore the minimum requirement is the JDBC driver db2jcc which is delivered with DB2 Connect and DB2 for z/OS V8.

With Web Application Server 6.30, SAP delivers already a J2EE application server using the JDBC driver db2jcc and DB2 for z/OS V7. Because of the restrictions of DB2 V7 regarding Unicode, the J2EE server based on DB2 V7 has the same restrictions. These are:

- ▶ Maximum index length is 255 bytes
- ▶ Maximum statement length is 32K bytes
- ▶ Maximum character length in a statement predicate is 255 bytes

To circumvent the maximum index length of DB2 V7, the SAP Java database interface tries to use UTF-8 in case the index exceeds the DB2 V7 limit. There are no SAP applications in 6.30 that cannot be handled by the SAP Java database interface. If customers want to write their

own applications, it is necessary to stay within the DB2 V7 limits. To circumvent the visible ROWID column if a table has LOB columns, the SAP Java database interface hides the ROWID from the application. The SAP Java database interface creates a projection view. In this view, all columns of the table are defined except the ROWID column. The view has exactly the same name as the original table. The table gets renamed in '#<table name'.

With the Unicode enhancements in DB2 V8, all these restrictions are lifted. Eventually, defined UTF-8 columns will be converted to UTF-16. With DB2 V8, the SAP Java database interface make also use of the invisible ROWID column. After migration to V8, all unnecessary views will be dropped and the tables will be renamed to their original names.

2.5.3 Java and DB2 data type mapping

Table 2-7 shows the set of data types that are supported by the SAP Open SQL persistence framework. One of the main goals of Open SQL is portability across the database platforms supported by SAP. In order to achieve the portability goal, Open SQL only supports a subset of the JDBC types defined in java.sql.Types.

Table 2-7 Java and DB2 data types

SAP JDBC type	DB2 for z/OS	SAP supports online extension
VARCHAR(N) 1 <= N <= 127	VARGRAPHIC(N) CCSID 1200	Yes
LONGVARCHAR(N) 1 <= N <= 1333	VARGRAPHIC(N) CCSID 1200	Yes
BINARY(N) 1 <= N <= 255	CHAR(N) FOR BIT DATA	No
LONGVARBINARY(N) 1 <= N <= 2000	VARCHAR(N) FOR BIT DATA	Yes
SMALLINT	SMALLINT	Yes
INTEGER	INTEGER	Yes
BIGINT	NUMERIC(19)	No
DECIMAL (P,[S])	DECIMAL(P,[S])	Yes
REAL	DOUBLE	No
DOUBLE	DOUBLE	No
DATE	DATE	No
TIME	TIME	No
TIMESTAMP	TIMESTAMP	No
CLOB	DBCLOB CCSID 1200	No, 1
BLOB	BLOB	No, 2
1- SAP always defines DBCLOB(512M), it would be Yes		
2- SAP always defines BLOB(1G), it would be Yes		

Columns with numeric data types that are part of an index are not altered, because the index would be put in REORG Pending state.

2.5.4 Java dictionary

Similar to the DDIC functionality in ABAP, SAP's aim is to provide a database dictionary for Java. This dictionary supports all database systems that are certified for SAP. Figure 2-15 shows a sample Java dictionary panel.

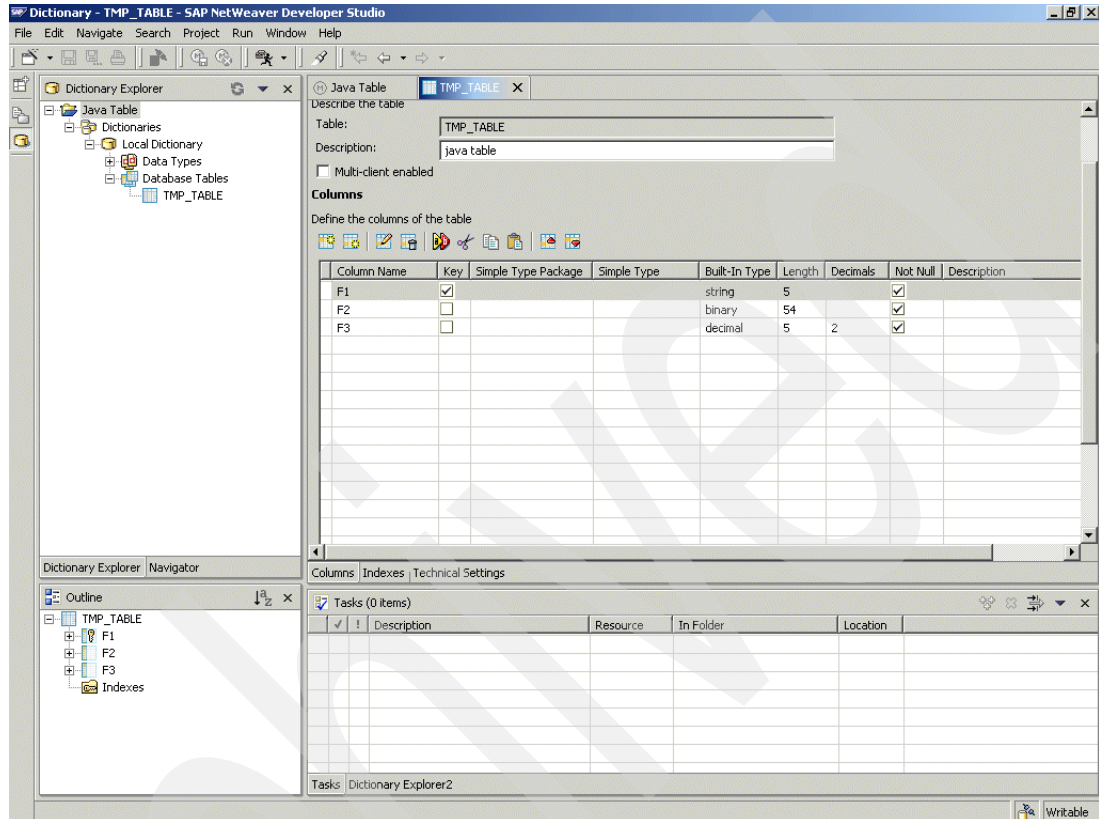


Figure 2-15 Table definition in Java dictionary

When the table in Figure 2-15 is to be created, the SAP common JDBC layer submits the corresponding DDL statements. Example 2-6 displays the log entries that the JDBC layer generates when creating a database table.

Example 2-6 Creation of database table from Java dictionary

```
----- Starting deployment -----
...
Starting to execute deployment action Java Table
...
15:49:09 2003-10-08 db2-Info: DatabaseMetaData encountered exception
com.sap.dictionary.database.dbs.JddException.
Aktion: CREATE
CREATE TABLESPACE "TABLE"
IN "TMPXX8NX"
USING STOGROUP SAPBCUDB
PRIQTY 40
SECQTY 40
FREEPAGE 20
PCTFREE 16
GBPCACHE CHANGED
DEFINE YES
BUFFERPOOL BP2
```

```

LOCKSIZE ROW
LOCKMAX 1000000
CLOSE YES
COMPRESS YES
MAXROWS 255
SEGSIZE 20
CCSID UNICODE
CREATE TABLE "TMP_TABLE"
(
"F1" VARGRAPHIC(5) NOT NULL,
"F2" CHAR(54) FOR BIT DATA NOT NULL,
"F3" DECIMAL(5,2) DEFAULT 0 NOT NULL
)
IN TMPXX8NX.TABLE
CCSID UNICODE
CREATE
UNIQUE INDEX
"#TMP_TABLEN9I"
ON "TMP_TABLE"
(
"F1" ASC
)
USING STOGROUP SAPBCUDB
PRIQTY 40
SECQTY 40
FREEPAGE 20
PCTFREE 16
GBPCACHE CHANGED
DEFINE YES
CLUSTER
BUFFERPOOL BP2
CLOSE YES
DEFER YES
COPY YES
PIECESIZE 2097152 K
ALTER TABLE "TMP_TABLE" ADD PRIMARY KEY
(
F1
)
Deployment action Java Table executed.
Deployment successful for Java Table
----- Deployment was successful -----

```

2.5.5 DB2 accounting and WLM

In order for SAP to be able to provide transaction based DB2 accounting and workload management for its Java applications, it needs to be able set DB2 client identifiers using JDBC.

The DB2 Java Universal Driver, which is the SAP JDBC driver of choice, implements the DRDA protocol like CLI. Therefore, DB2 treats CLI clients and Java Universal Driver clients uniformly. The DB2 client identifiers that are the basis of accounting data rollup can be altered at transaction boundaries.

The Universal Driver extends the Connection interface from the JDBC standard to also provide the following methods, which set client identifier values:

- ▶ `DB2Connection.setDB2ClientUser(String user)`
- ▶ `DB2Connection.setDB2ClientWorkstation(String name)`
- ▶ `DB2Connection.setDB2ClientApplicationInformation(String info)`
- ▶ `DB2Connection.setDB2ClientAccountingInformation(String info)`

There are equivalent methods to query these identifiers. For more details on enhanced accounting capabilities and workload management support of DB2 V8, see 6.6, “SAP transaction-based DB2 accounting and workload management” on page 170.

Archived



Usability, availability, and scalability

In this chapter we describe the following topics:

- ▶ Partitioning
- ▶ Index creation enhancements
- ▶ Convert column type
- ▶ LOB ROWID transparency
- ▶ Multiple DISTINCT clauses in SQL statements

3.1 Partitioning

In this section we describe enhancements that are introduced in DB2 V8 in the area of table partitioning. This will be of interest to many SAP users, where the high volume of data in particular tables requires the use of partitioning functionality. Most commonly, partitioning is considered as tables start to approach the size of 64 GB for the table space. Partitioning can improve the manageability of the data, through more effective REORG, data management, and backup/recovery strategies, as well as potentially improving performance of programs using this data.

DB2 V8 also introduces the concept of Data Partitioned Secondary Indexes, where secondary indexes on tables can also be partitioned into separate partitions, in this case corresponding to the set of rows in the corresponding table space partition.

SAP has allowed partitioning to be used in all releases, but explicit support within the inbuilt database administrative functions has been available from SAP Release 4.5. This support is provided by SAP transaction SE14. It allows almost all functions to be performed through the SAP interface, and reduces the need to manually deal with creating and executing data definition language (DDL).

The enhancements in partitioning features in DB2 V8 will be gradually implemented in the SAP SE14 transaction with the SAP releases starting in release WebAS 6.40.

3.1.1 Increases to table sizes

With DB2 V8, the maximum number of partitions has been increased from 254 to 4096, allowing more flexibility in choosing partitioning schemes. This increases the theoretical maximum size for a partitioned table to 16 - 128 TB, depending on the page size. A 32 KB page size table in a partitioned table space can grow to the maximum size of 128 TB. If SAP tables have a key structure of an existing or added index that allows a suitable partitioning scheme to be chosen, this results in improvements to the manageability of large objects.

Having maximum flexibility in DB2 V8 is particularly important for enterprise packages such as SAP, since the structure of the data storage schemas are fixed, and traditional means of adapting the data structures to suit the environment are often not possible.

3.1.2 Adding new partitions and rotating partitions

With DB2 for z/OS Version 7, it was possible to redistribute the data among partitions by altering the limit keys, and conducting reorganization utilities on the tables in question. However, it was not possible to change the number of partitions. DB2 V8 implements the ability to add a partition with the **ALTER TABLE ADD PARTITION** statement. Issuing this command results in DB2 creating a new partition at the end of the current partitions.

Rotating partitions allows old data to “roll off” while reusing the partition for new data with the **ALTER TABLE ALTER PARTITION ROTATE FIRST TO LAST** statement. A common case in an SAP system is where (n+1) partitions are used to continuously keep the last n periods of data. When rotating, one can specify that all the data rows in the oldest (or logically first) partition is to be deleted, probably after archiving activity has removed the data. Then you specify a new table space high boundary so that the partition essentially becomes the last logical partition in sequence ready to hold the data which is added. Because the data of the partition being rolled off is deleted, if long term retention of the data is required, you would certainly want to consider running an SAP Archiving process before rotating the partition, as any remaining rows will be deleted by the process. Confirmation of successful SAP archiving would be done by checking that no rows remain in the partition.

The partition that was rolled off is immediately available after the SQL statement is successfully executed. No REORG is necessary.

After using the new **ALTER TABLE ALTER PARTITION ROTATE FIRST TO LAST** statement, the logical and physical order of the partitions is no longer the same. The `display` command lists the status of table space partitions in logical partition. Logical order is helpful when investigating ranges of partitions which are in REORP. It enables one to more easily see groupings of adjacent partitions that may be good candidates for reorganization. When used in conjunction with the new **SCOPE PENDING** keyword of REORG, a reasonable subset of partitions can be identified if one wants to reorganize REORP ranges in separate jobs.

The availability of these two additional ALTER statements allows for continuous availability, even where the growth of data requires adjustments to the storage layout used. For many SAP tables, the nature of growth of data may not be able to be determined in advance, or anticipated by the project team implementing the software. The enhancements to partition controls described here allow a more dynamic approach to data management, and can mean that unanticipated data growth in an SAP system is able to be addressed with minimal impact to users, greatly increasing availability. Previously, if a chosen partitioning scheme did not have the ability to handle the data volumes in a system, a significant amount of “downtime” may be needed to unload the data, and increase the number of partitions. Now, it is possible to increase the theoretical data capacity “on the fly” without any significant impact to availability.

The ability to rotate and “roll off” data in tables is also highly applicable to many SAP data types. With suitable planning, and involvement with the other employees planning SAP Archiving activities, a customer can implement an extremely efficient mechanism to deal with long term data growth. By designing the partitioning scheme to match the criteria for SAP Archiving, such as financial period or document numbering schemes, the execution of an SAP Archiving delete run can then be followed by the rotation of the partition to be usable for future data.

3.1.3 Data Partitioned Secondary Indexes

DB2 V8 introduces the ability to physically partition secondary indexes. The partitioning scheme introduced is the same as that of the table space. That is, there are as many index partitions in the secondary index as table space partitions, and the index keys in partition 'n' of the index reference only the data in partition 'n' of the table space. Such an index is called a *Data Partitioned Secondary Index (DPSI)*.

You create a DPSI by specifying the new *PARTITIONED* keyword in the CREATE INDEX statement, shown in Figure 3-1, and defining keys on columns that do not coincide with the partitioning columns of the table. When defining a DPSI, the index cannot be created as **UNIQUE** or **UNIQUE WHERE NOT NULL**. This is to avoid having to search each part of the DPSI to make sure the key is unique.

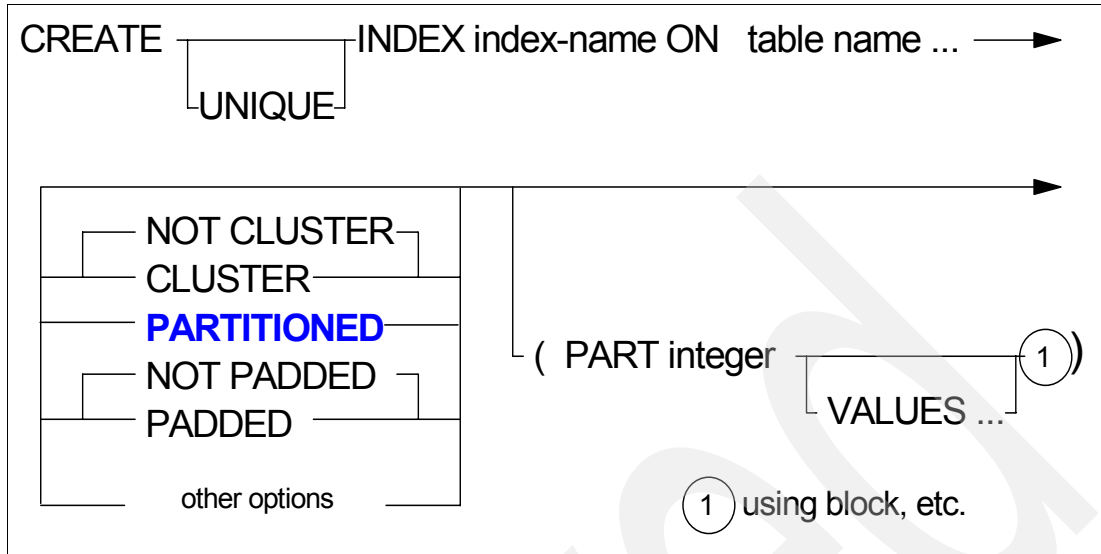


Figure 3-1 Syntax of `CREATE INDEX` statement

3.1.4 Separating clustering from partitioning

With DB2 V7, partitioning a table always involved defining a partitioning index, whose limit keys effectively determined which partition a particular record was placed in. This is referred to as index-controlled partitioning, and required that an index was created to address the partitioning scheme, and the actual limit key values to separate the data content of each partition were defined in the index creation.

The data model used in an SAP system means that every table is created with a clustering index based on the primary access path expected for the table. Many tables also have one or more additional secondary indexes created to service alternative access paths. For example, in an SAP R/3 Enterprise 4.7 system, 84% of the indexes that exist in the system are primary clustering indexes, and the remaining 16% are additional secondary indexes. As an SAP customer receives both the entire set of application code, and database schema (or metadata) from the one source, generally the system functions very well with no adjustments.

In the case where performance or throughput may be enhanced through additional indexes being created, tools are supplied as part of the package to facilitate this action. This action is often performed where additional functions are coded and implemented by a customer, or the system is used in a manner not expected in the default setup of the supplied application. In many cases the SAP application is supplied with definition data in the SAP Data Dictionary for indexes which are not created by default. These may be then be created easily in the database, for cases where specific functionality that is not universally used by all SAP implementations takes place in an SAP system.

The result of these two design constraints of the DBMS and the application package (prior to DB2 V8) previously meant that the clustering and partitioning indexes were usually one and the same, and — as the primary clustering index is predominantly used as the access path for SQL statements — in most customer cases, this works well. The one remaining consideration, in this case, is where the key range of the primary clustering index does not lend itself to an efficient scheme of partitioning. For example, the primary key distribution may be very “lumpy”, and change activity is not evenly distributed, or concentrated unevenly across the key ranges. This results in added overhead in maintaining good data distribution, for example, with `REORG PART` utilities. Also, it can result in inefficient usage of the available underlying disk, if the chosen partitioning scheme is less than ideal.

3.1.5 Example case of an SAP table

An example of the separation of clustering and partitioning is shown in Figure 3-2 on page 71, where the table GLPCA is used. For this example, there is a column in the table which suits the purpose of partitioning to address size limitations, POPER. This column is the financial period of the records, and allows an even distribution of data, and a very structured roll-on roll-off process to be followed for long term data management.

In this case, there is no index containing this column. Prior to DB2 V8, the database administrator would need to have created an additional index on the table to allow the partitioning to take place. The creation of the index would have all of the negative impacts associated with an additional index, such as additional disk requirements, performance penalty associated with change activities, etc. Presumably an index based on this column would have no dramatically positive performance impact for the set of SQL statements executed on the table, else it would have already been created.

In the example in Figure 3-2, the clustering index on column GL_SIRID remains, and is a Data Partitioned Secondary Index, or DPSI. There are a number of other indexes within an SAP system on table GLPCA, one of which is shown here, on column DOCNR. In this case it has been created as a non-Partitioned Secondary Index, but could be created as a DPSI, providing it is not defined as a unique index.

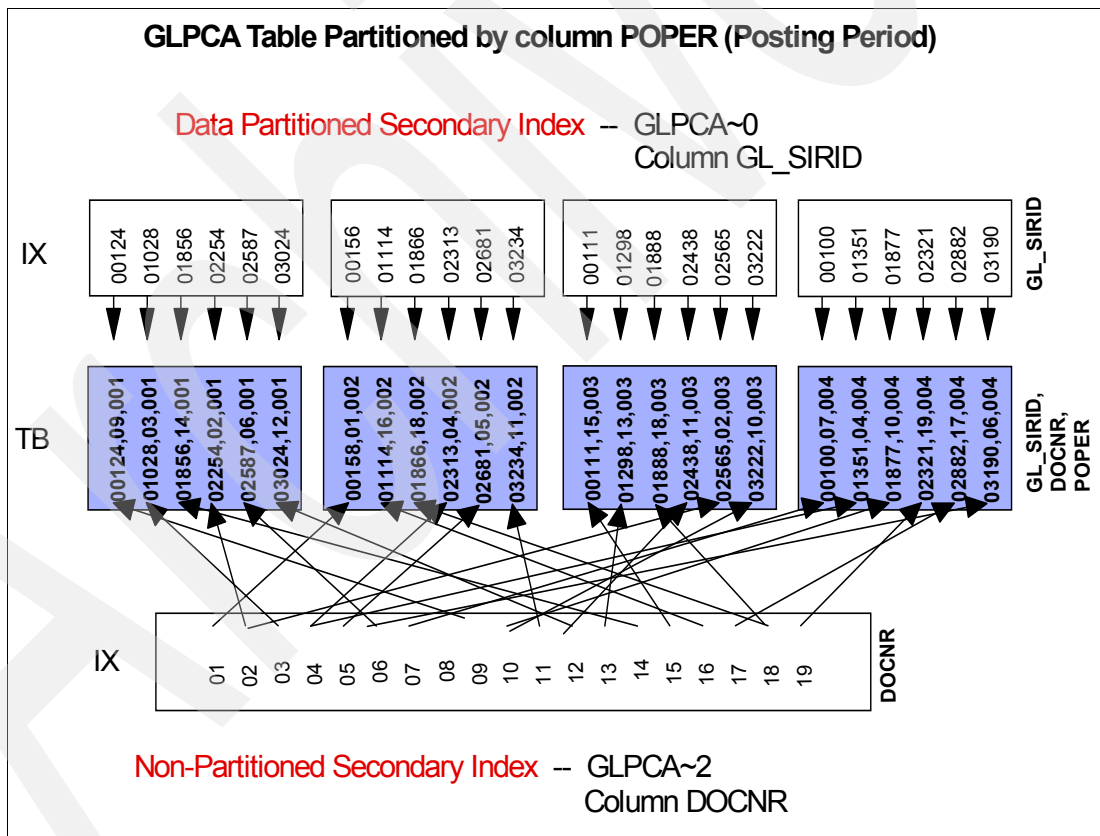


Figure 3-2 Example of partitioning SAP table GLPCA by posting period

3.2 Index creation enhancements

In this section we discuss two enhancements concerning the creation of indexes, which are relevant for SAP systems:

- ▶ Invalidation of dynamically cached statements occurs.
- ▶ Deferred indexes do not prevent table access.

3.2.1 Invalidation of dynamically cached statements

Once an SQL statement encounters performance problems, a typical remedy is to create an appropriate index that the DB2 optimizer can choose to speed up the processing of this statement. However, if the statement is cached in the dynamic statement cache (DSC), the newly created index is ignored for this statement, prior to DB2 V8, in single subsystem environments. It keeps the access path that the DB2 optimizer has previously determined. The optimizer can consider the index for the statement only after that statement has been removed from DSC, either because it was displaced from DSC or because the database administrator explicitly invalidated it.

While adding to operational complexity, there is a circumvention for this problem that addresses known indexes. The statements can be manually invalidated after a new index is created, for example, by running the RUNSTATS utility with the options UPDATE NONE REPORT NO in DB2 V8. For indexes, however, that are created as part of applying SAP maintenance, this is virtually impossible. These indexes are automatically created by SAP. Therefore, their existence is not obvious to the database administrator. The existing cached statements often cannot take advantage of the new indexes for a considerable amount of time.

In data sharing environments, the creation of an index invalidates cached statements that refer to the base table of the new index already before DB2 V8. In this case, the cached statements are even quiesced, which means that they cannot continued to be used within active transactions.

DB2 V8 introduces analogous behavior for both data sharing and non-data sharing. If a new index is created, the affected cached statements are invalidated, but not quiesced. That means that statements that are used in active transactions can still be employed until these transactions complete. Indexes that are created with the option DEFER YES, which means that the index tree is not immediately built, also invalidate cached statements.

This new feature enhances the usability of DB2. It makes DB2 require less manual interaction and hence contributes to a less error-prone approach.

In Figure 3-3 we show the dynamic statement cache statistics on a statement that refers to table TATOPA.

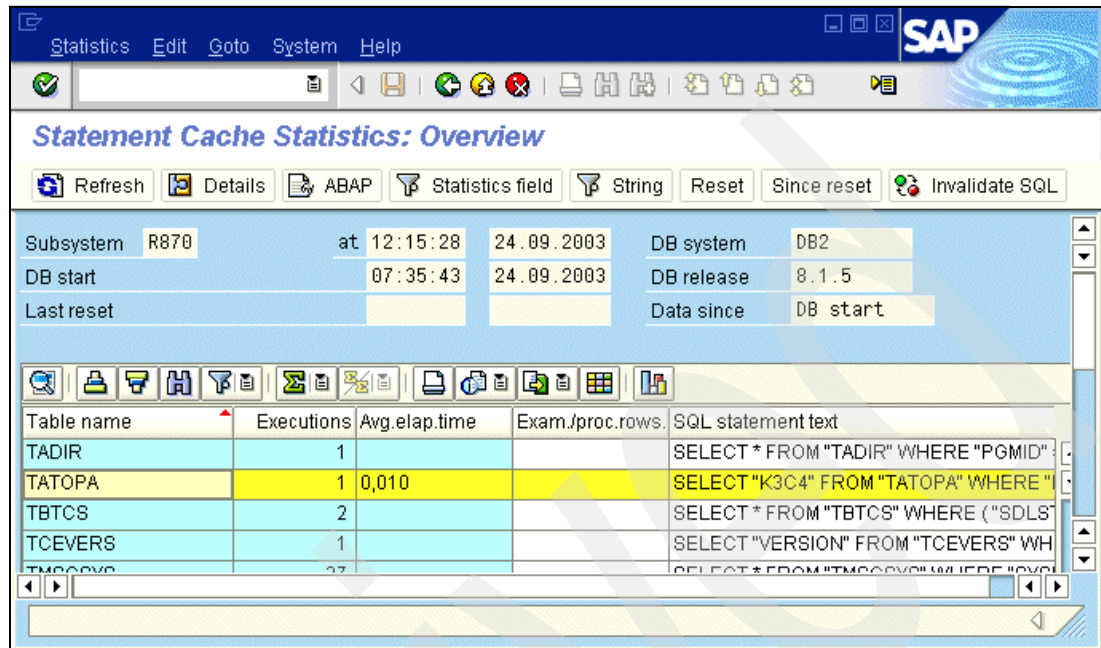


Figure 3-3 Invalidation of cached statements: Before index creation

The creation of a new index on table TATOPA invalidates the cached statement. As the statement is not part of active transactions, it is immediately removed from DSC. Figure 3-4 demonstrates that it was removed from DSC.

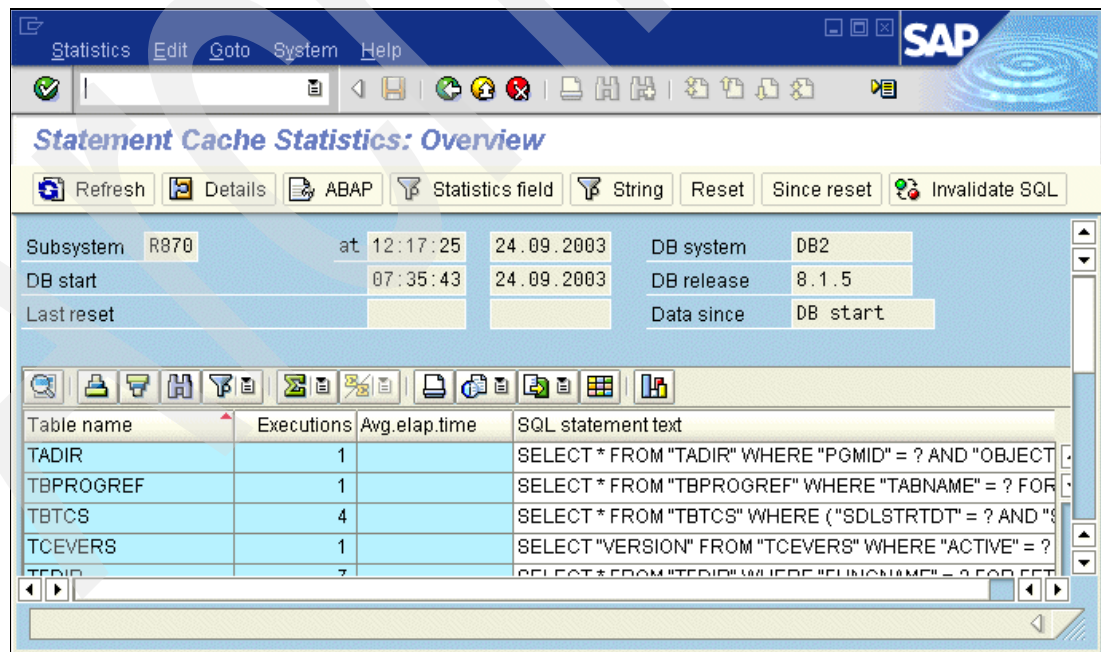


Figure 3-4 Invalidation of cached statements: After index creation

3.2.2 Deferred indexes do not prevent table access

Prior to V8, DB2 already allows you to create deferred indexes. Deferred indexes are indexes that are created using the DEFER YES clause. When creating a deferred index, the description of the index and its index space is added to the catalog, but it is not built. It is placed in Rebuild Pending status. The only exception to this is if the table on which the index is defined is empty. The advantage of deferring the creation of the index tree is that multiple indexes, which are defined on the same table, can be built in a single run rather than scanning the table multiple times. For example, SAP BW considerably makes use of deferring index creation.

The problem with deferred indexes is that, prior to DB2 V8, they block table access if the optimizer selects an access path that uses the deferred index for a given statement. The execution of the statement results in SQL code -904 and reason code 00C900AE, which indicates that an attempt was made to access an index that is in Rebuild Pending state.

With DB2 V8, the optimizer does not consider indexes in Rebuild Pending state during the dynamic prepare of a statement. It avoids indexes in advisory Rebuild Pending state for index-only access. To take advantage of deferred indexes as soon as possible, cached statements, which refer to the base table of a deferred index, are invalidated if the index is rebuilt and reset from Rebuild Pending state.

This enhanced behavior improves the usability of creating deferred indexes. It eliminates the exposure of having the optimizer select an index that is not yet ready to be used.

Figure 3-5 shows an example where a deferred index (TATOPA~1) is available that perfectly matches the predicate of a statement.

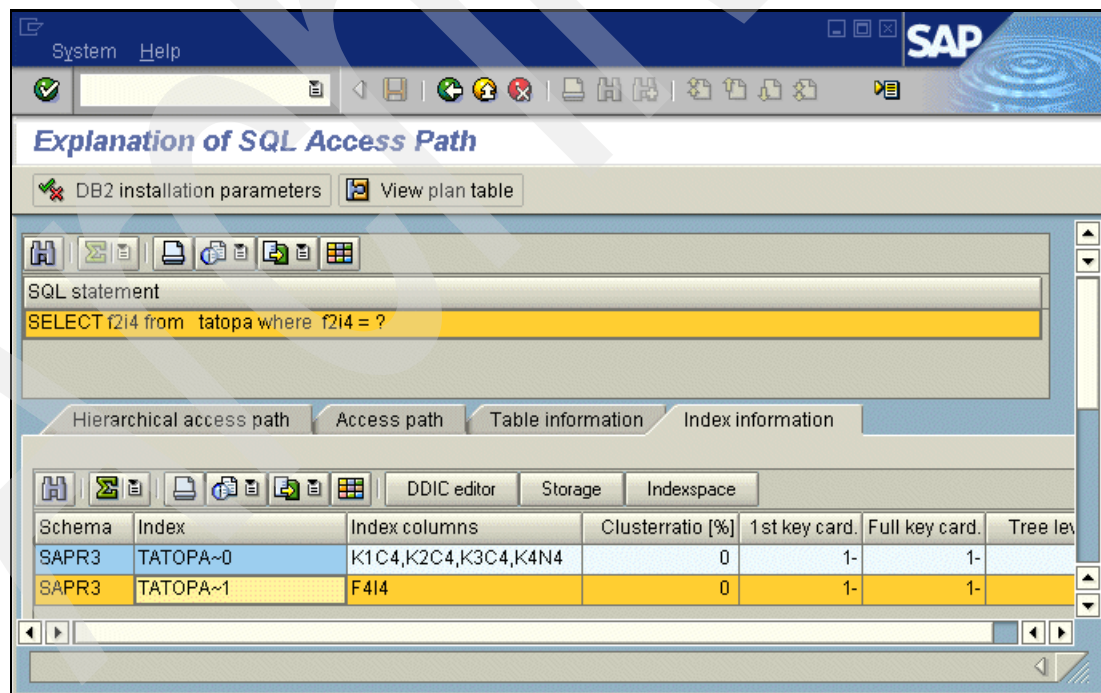


Figure 3-5 Index in Rebuild Pending state: Definition

As Figure 3-6 indicates, the optimizer refrains from exploiting TATOPA~1, since this index is in Rebuild Pending state.

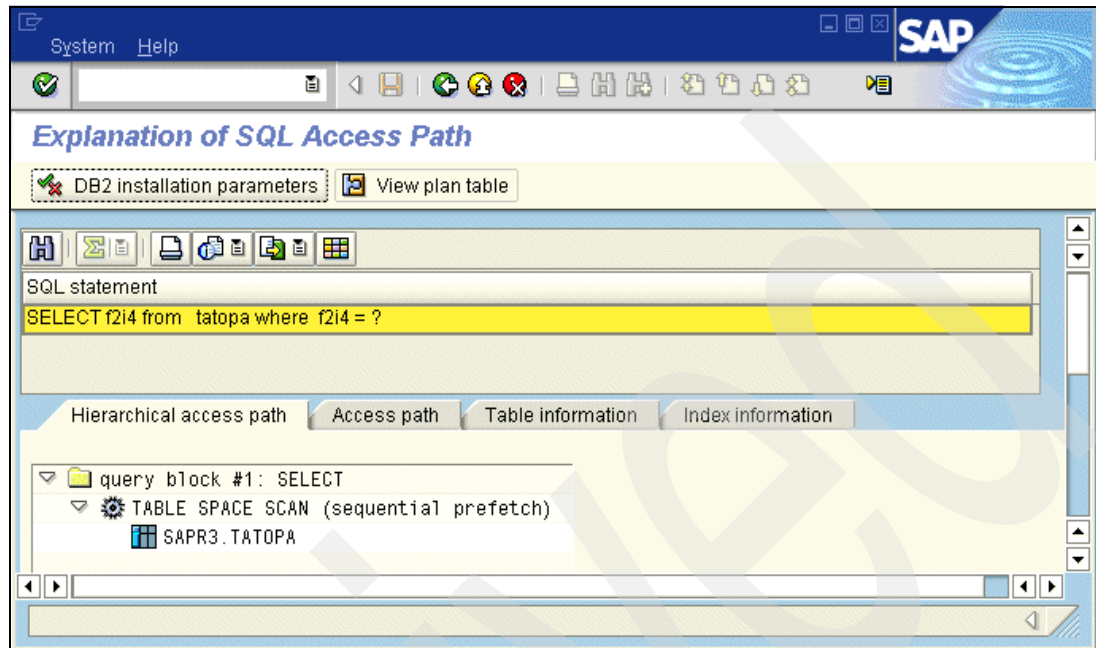


Figure 3-6 Index in Rebuild Pending state: Not selected by DB2 optimizer

3.3 Convert column type

In this section we describe the manner in which the SAP supports the alteration of data types within the SAP data dictionary environment.

3.3.1 Supported column type changes

DB2 V8 only supports column type changes between compatible data types. This simply means that changes are not supported between the dissimilar field categories of numeric, character, and graphic. The following schema changes allowed in DB2 V8, are applicable to the SAP application and will be supported by SAP.

- ▶ Extend CHAR(n) column lengths
- ▶ Change type within character data types (CHAR, VARCHAR)
- ▶ Change type within numeric data types (SMALLINT, INTEGER, FLOAT, REAL, FLOAT, DOUBLE, DECIMAL)
- ▶ Change type graphic data types (GRAPHIC, VARGRAPHIC)
- ▶ Allow column data type changes for columns that are referenced within a view
- ▶ Allow column changes for columns which are part of an index definition (VARCHAR, CHAR, GRAPHIC, VARGRAPHIC)

The alteration of numeric column types (SMALLINT, INTEGER, FLOAT, REAL, FLOAT, DOUBLE, DECIMAL), which are part of an index definition, is not supported by SAP.

Altering numeric columns contained in an index results in the index being placed into an RBDP (Rebuild Pending) state, making them unavailable for data access. This is particularly problematic for primary key indices enforcing a UNIQUE constraint, because unavailability of the primary index effectively prevents INSERT statements and UPDATE statements from changing key values.

Furthermore, online schema evolution is most valuable for large tables, meaning that the loss of indexes might cause severe performance degradation in the SAP environment. The following three examples demonstrate the successful execution of a numeric online schema change and an unsuccessful execution because of an index.

3.3.2 Table ZMAKT100I

Figure 3-7 shows the structure of table ZMAKT100I table. This table was created in the database and populated with approximately 500,000 rows.

Field	Key	Initial	Data element	Data T...	Length	Deci...	Short Description
SPRS1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	SPBAS	LANG	1		Language Key
ARBGB	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	ZMARK	CHAR	26		Mark Test Text Element
NUMBR	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	ZMARKI	INT2	5		mark test integer
TEXT	<input type="checkbox"/>	<input type="checkbox"/>	NATXT	CHAR	73		Message text
NUMB2	<input type="checkbox"/>	<input type="checkbox"/>	ZMARKI	INT2	5		mark test integer
NUMB3	<input type="checkbox"/>	<input type="checkbox"/>	ZMARKI	INT2	5		mark test integer
NUMB4	<input type="checkbox"/>	<input type="checkbox"/>	ZMARKI	INT2	5		mark test integer

Figure 3-7 SAP DDIC display of ZMAKT100I table

3.3.3 Index ZMAKT100I~Z01

Figure 3-8 shows the structure of a secondary index on table ZMAKT100I.

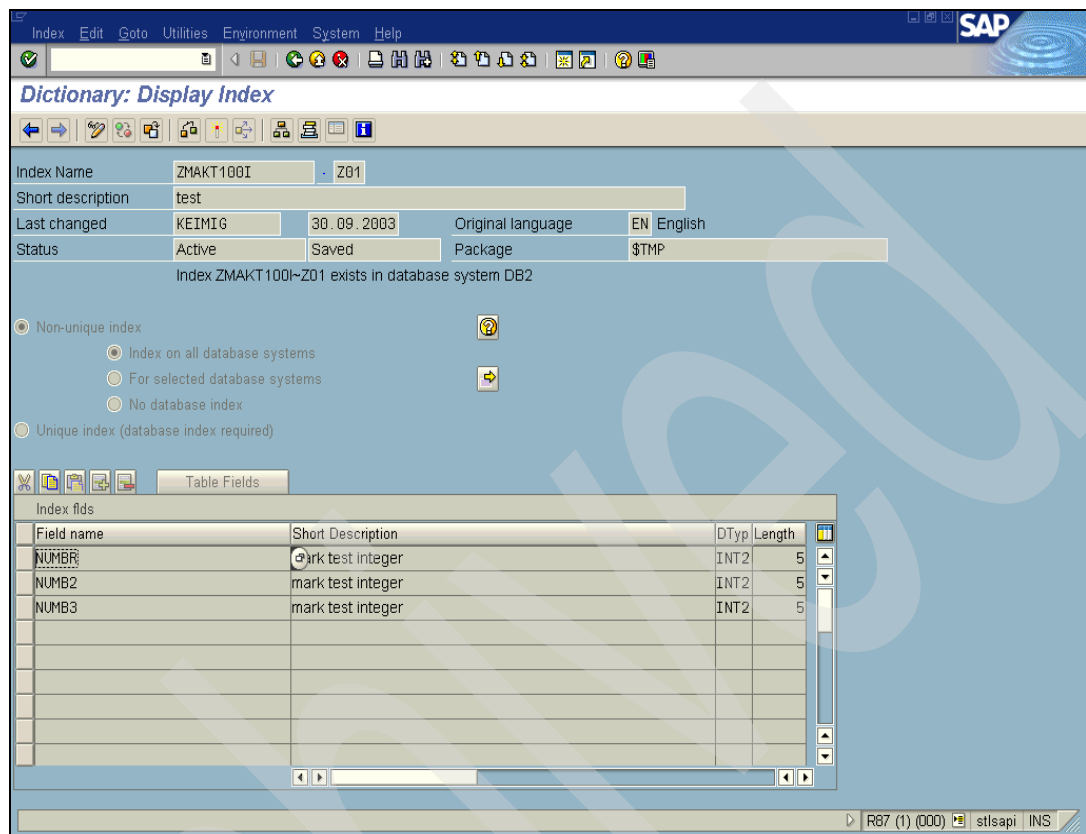


Figure 3-8 SAP DDIC display of secondary index on table ZMAKT100I

3.3.4 Example 1: ALTER NUMB4 to INT4

This example consists of the use of the SE11 tool to change the data element for field NUMB4 from ZMARKI (INT2) to ZMARKI4 (INT4).

SAP object change

Figure 3-9 shows the warning message following the save, giving an opportunity to abandon the change.

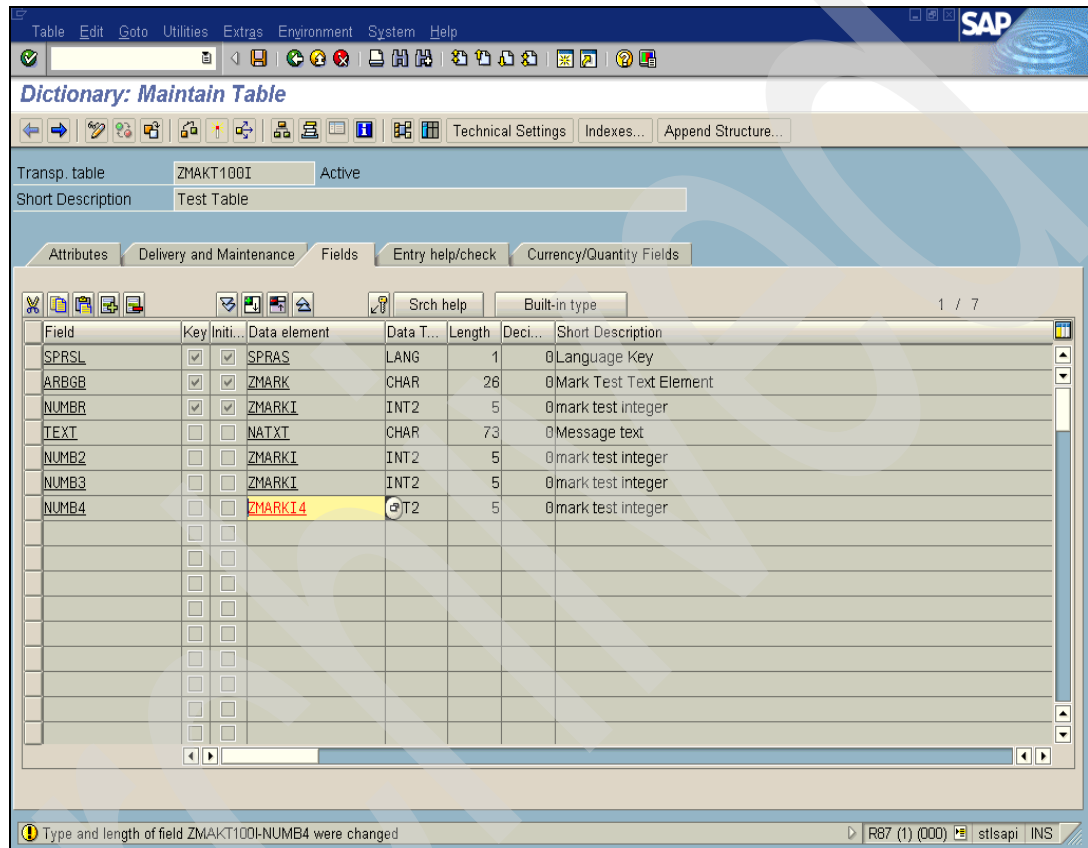


Figure 3-9 SE11 Change of data element with warning after save

Figure 3-10 shows the SE11 display following the save of the change to the ZMAKT100I object.

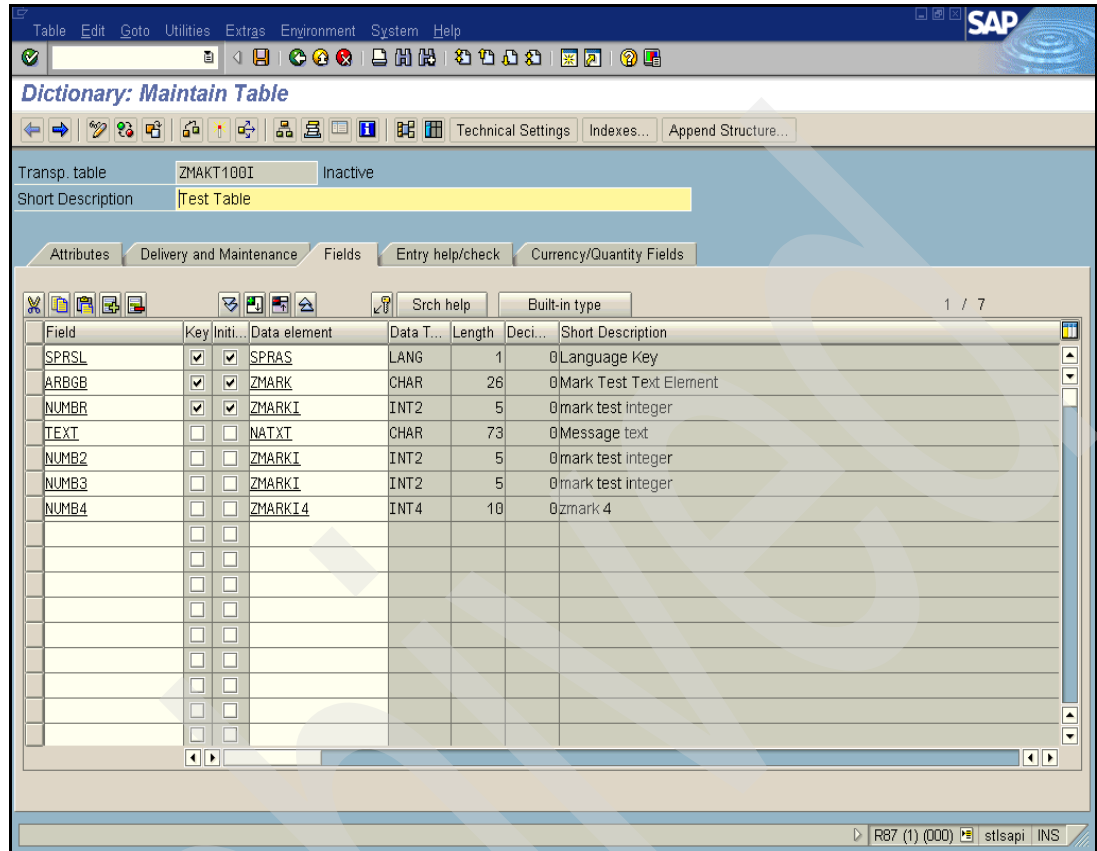


Figure 3-10 SE11 Change of data element after continue

SAP object activation

Figure 3-11 is the SE16 display of the record structure prior to change activation.

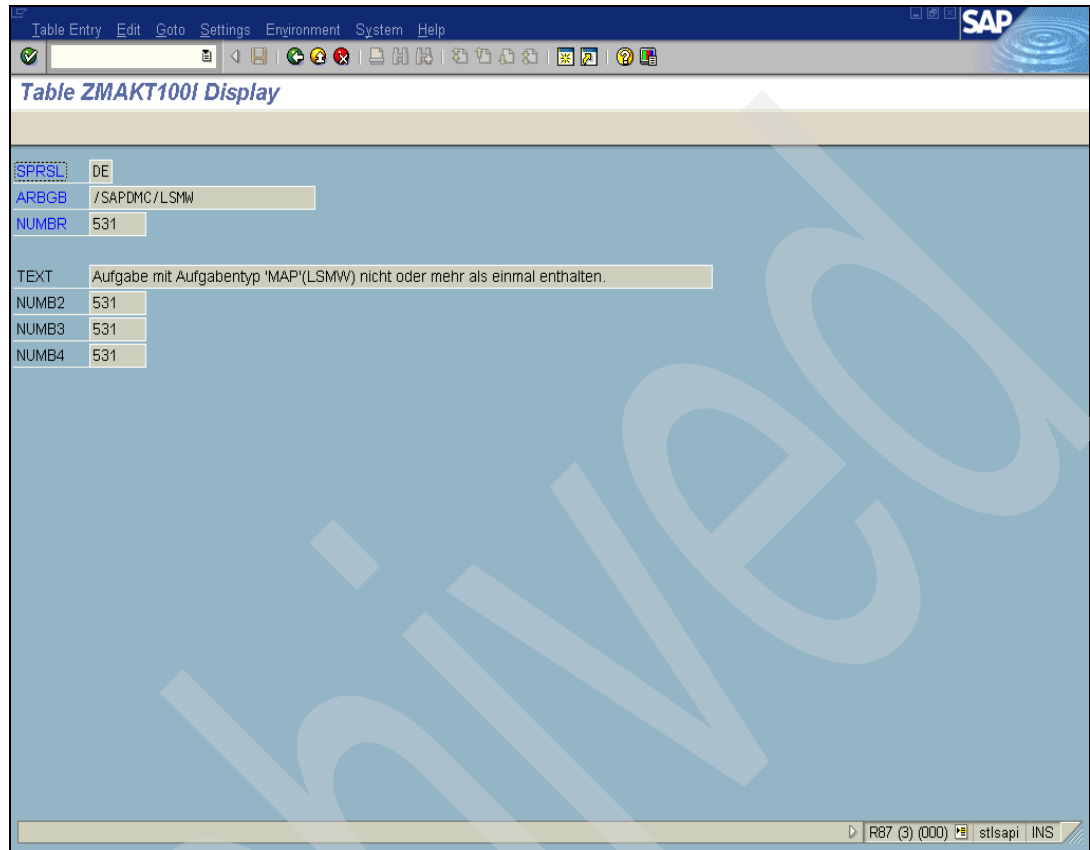


Figure 3-11 ZMAKT100I record display before activation

Figure 3-12 shows the ZMAKT100I activation log.

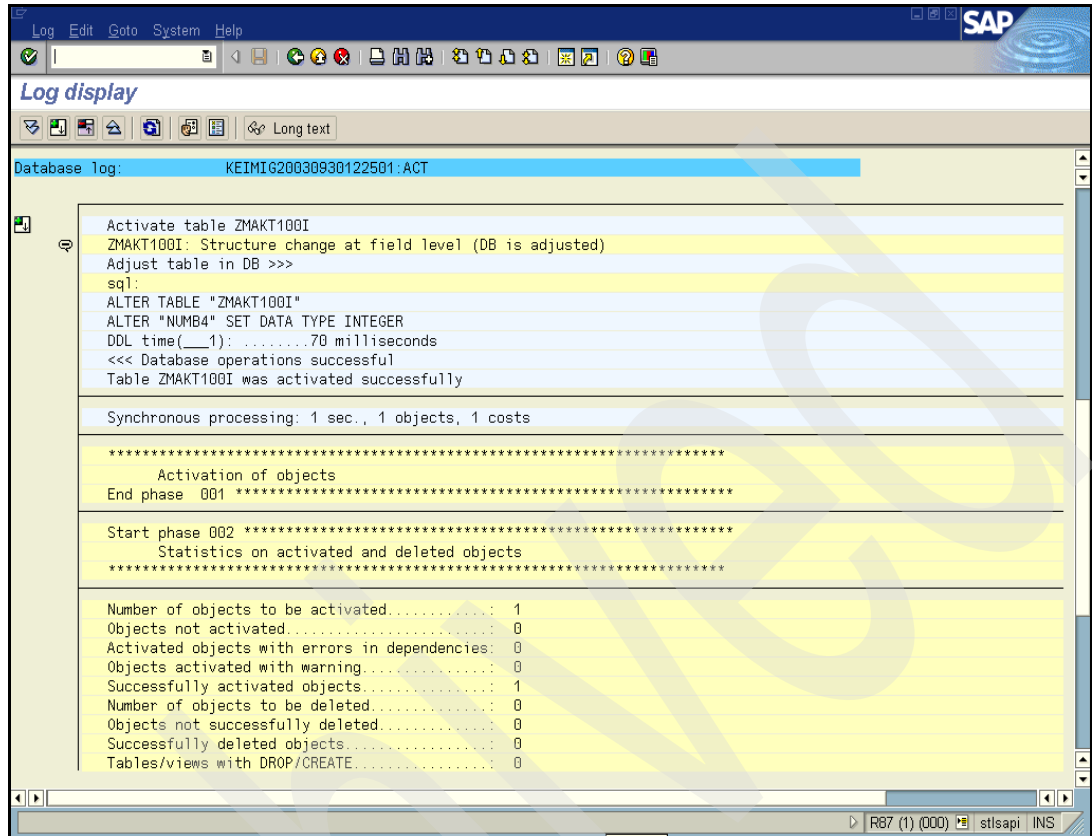


Figure 3-12 ZMAKT100I activation log

Figure 3-13 is the SE16 display of the record structure after change activation.

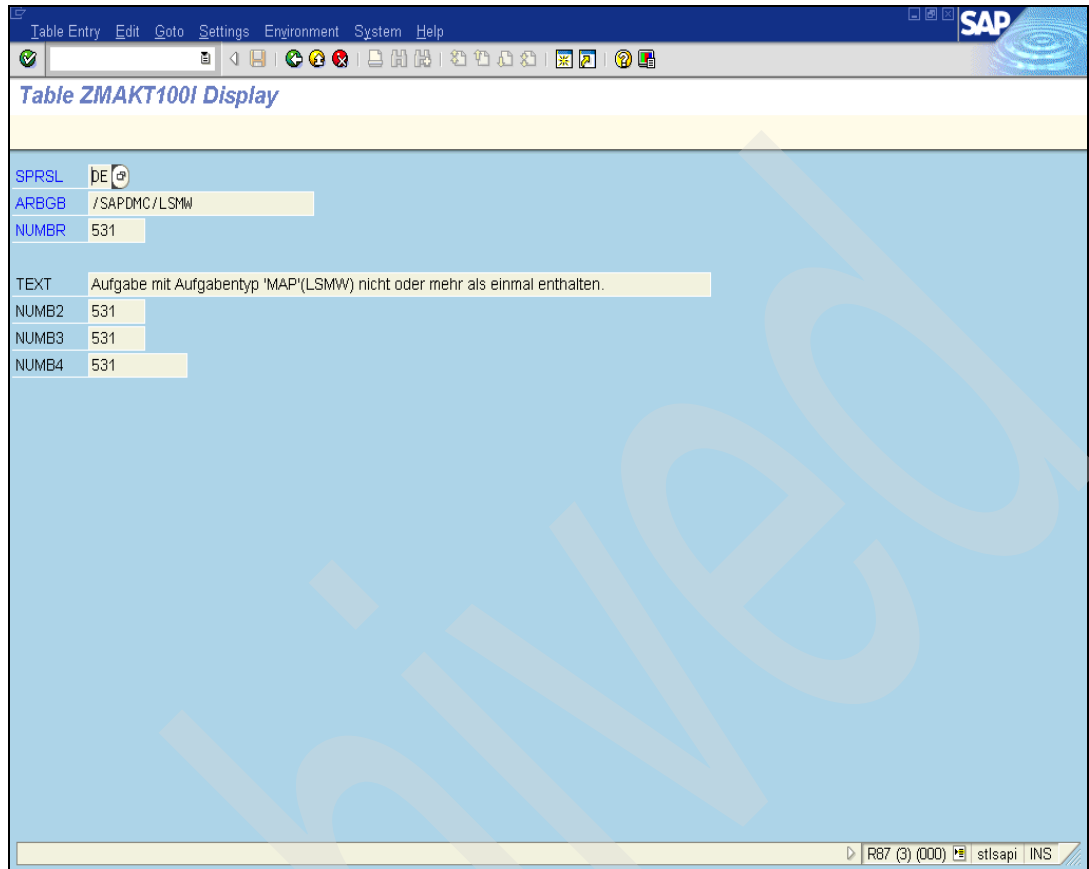


Figure 3-13 ZMAKT100I record display after activation

DB2 object status after activation

Figure 3-14 shows the status of the objects in the DB2 subsystem before and after a REORG of the table space. The status of the table space after activation is AREO*, which is an Advisory REORG status. This status indicates that some ALTER statement has been issued against an object in the table space which has the potential to reduce performance. This status is a recommendation that the table space requires a REORG at the earliest convenient time and does not impact data availability.

Following the execution of a REORG, the AREO* status is reset.

Tablespace status after activation

1. Show AREO* Status

```
-R870 DISPLAY DB(U040XT3U) SPACE(*)
```

NAME	TYPE	PART	STATUS
ZMAKT10	TS		RW,AREO*
ZMAK1Z6I	IX		RW
ZMAKT100	IX		RW
2. REORG TABLESPACE (U040XT3U.ZMAKT10) UNLDDN (UNLD) WORKDDN(WORK)
3. Show RW Status

```
-R870 DISPLAY DB(U040XT3U) SPACE(*)
```

NAME	TYPE	PART	STATUS
ZMAKT10	TS		RW
ZMAK1Z6I	IX		RW
ZMAKT100	IX		RW

Figure 3-14 DB2 status before and after REORG of table space

Figure 3-16 shows the SE11 display following the save of the change to the ZMAKT100I object.

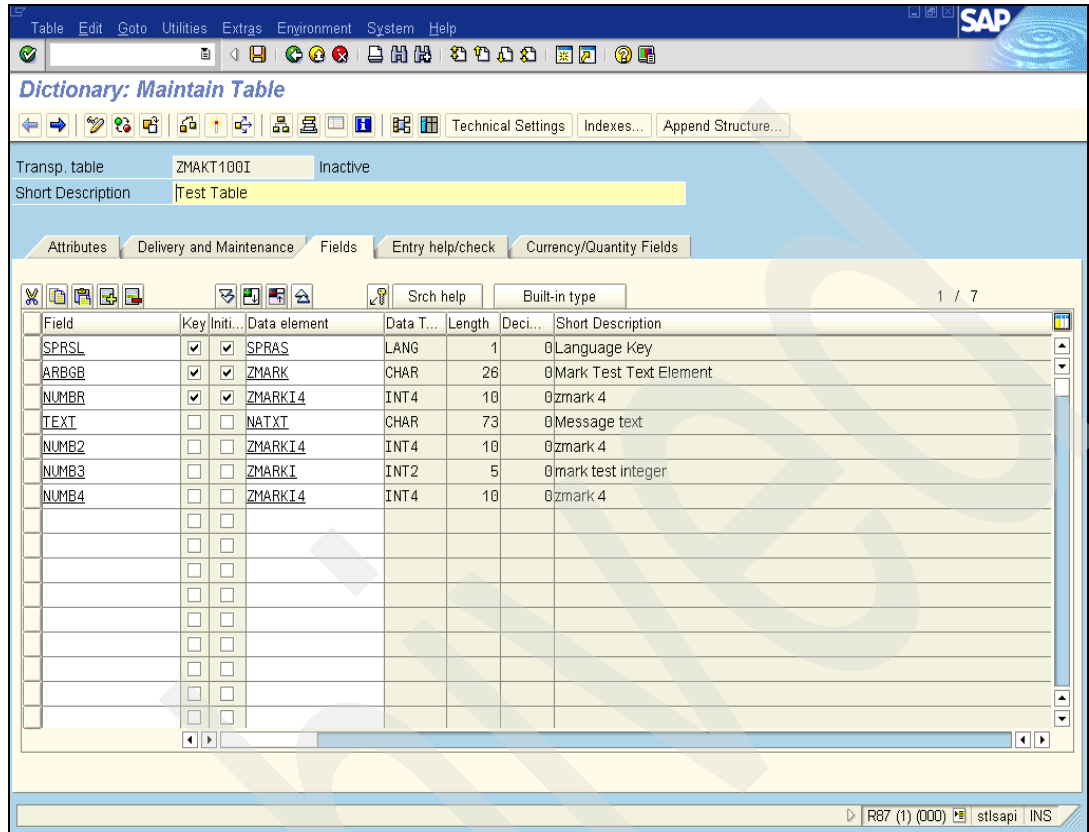


Figure 3-16 SE11 Change of data element after continue

SAP object activation failure

Figure 3-17 shows the behavior of the SAP 6.20 Kernel. The activate fails with a sqlcode of +610, a rollback is issued, and the DB2 database is unchanged. There is no means by which the required index rebuilds can be driven from the DDIC activation process, and rather than leave an index in Rebuild Pending state, a rollback occurs.

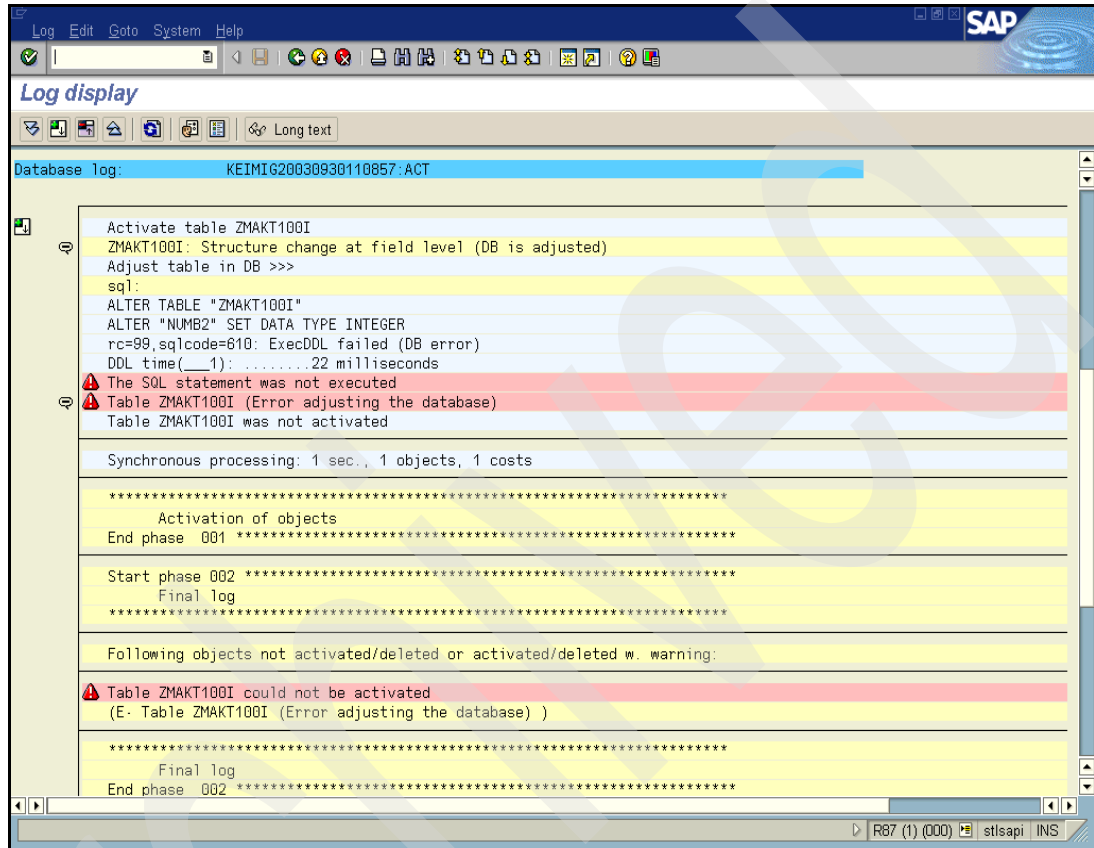


Figure 3-17 ZMAKT100I activation log

3.3.6 Effect of ALTER of column in primary key

Even though the SAP activation logic prevents column alteration when indexes are involved, it is useful to explore these effects. Figure 3-18 shows the effect of issuing an ALTER against the NUMBR column. This column is used in both the primary and secondary indexes of the ZMAKT100I table. When we issue an ALTER, we see the same SQLCODE of +610. Notice that, although the table space is now in AREO* status, both the indexes are in Rebuild Pending (RBDP) status.

In this example (see Figure 3-18), the primary index enforces a unique constraint and the RBDP state effectively prevents insert and key update SQL statements against the table.

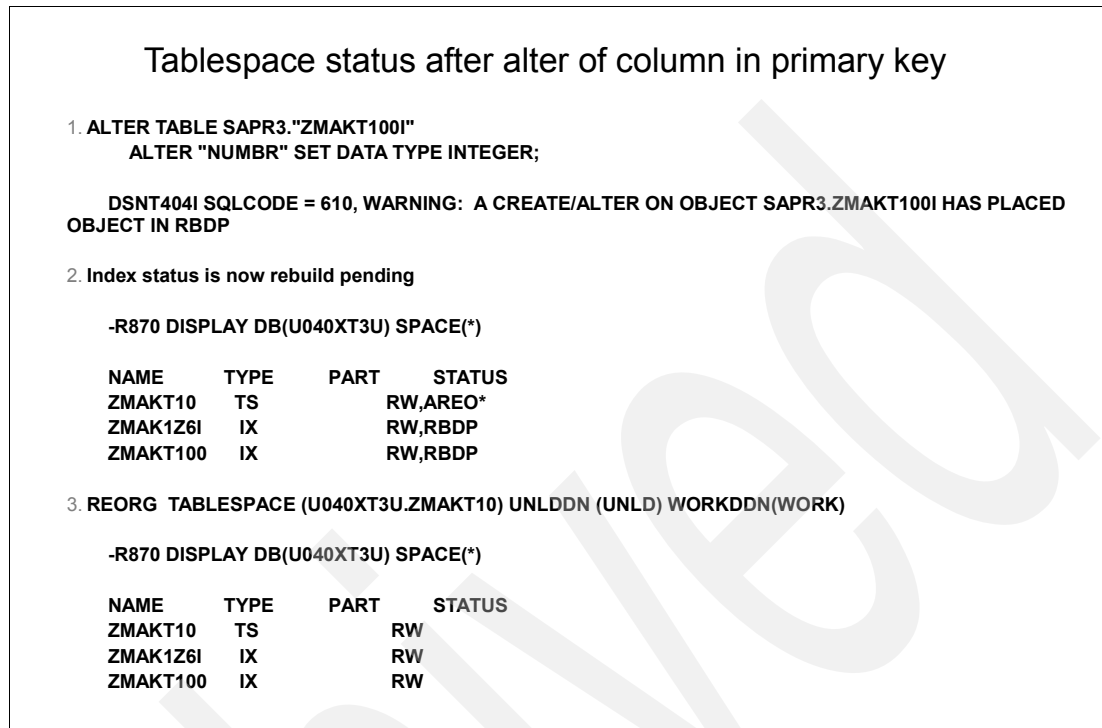


Figure 3-18 DB2 Status after ALTER involving primary key

3.4 LOB ROWID transparency

In this section, we describe how SAP takes advantage of the ROWID transparent definition for LOBs introduced with DB2 V8.

3.4.1 SAP and LOBs

DB2 for z/OS supports LOBs starting with V6. SAP has been using LOBs since Web AS 6.10. If a table has at least one LOB column, you have to have an additional ROWID column. But you can only have one ROWID column per table regardless of how many LOB columns are defined for the table. SAP has its own active dictionary. This means that every database object defined in the SAP dictionary must have a corresponding entry in the DB2 catalog. The additional ROWID column is unknown to the dictionary. This causes additional special coding in every area where DDL is used.

For the convenience of an ABAP programmer, the ABAP processor provides an initial layout of a work area. The ABAP processor also checks if the work area and the SAP dictionary definition are compatible. In Example 3-1 we show an ABAP example where we access a table that consists of three columns: f1 char(1); f2 clob(1G); myrowid.

Example 3-1 ABAP example with LOB

Let's assume a table lob_test is defined in the SAP dictionary as:

f1: type character with the length of 1

f2: type string

ABAP program snippet:

```

data: begin of wa,
      f1(1) type c,
      f2   type string,
end of wa.

select * into wa from lob_test
...
endselect.

```

In DB2, the additional ROWID column has to be defined because the dictionary type *string* is mapped to the CLOB data type. In Example 3-1 the ABAP processor generates a layout without ROWID. In the database interface, this layout is used as I/O area for the record to be fetched. Now, if SAP issues a **select * from lob_test**, DB2 will return the ROWID column as well. But the ROWID would not fit into the I/O area. The solution SAP has chosen is to change the statement text. The * gets replaced by the full select list known by the SAP dictionary (field list without ROWID column) in the database interface instead of changing each ABAP program individually. To hide the ROWID column from the application, a change had to be made in the database independent layer of SAP. A similar effort is needed for the SAP J2EE application server.

With the capability of hiding the ROWID column from DML and DDL as well, SAP can throw away this error-prone special code and use exactly the same code path used for all other database platforms. For SAP customers, this should result in a more robust environment.

The argument above is also valid for any application using LOBs. Prior to V8, these applications had to change their code in order to run on DB2 for z/OS; now, no code change is needed just because LOBs are being used. So the feature of hiding the ROWID makes much easier the porting of existing LOB applications to DB2 for z/OS.

3.5 Multiple DISTINCT clauses in SQL statements

In this section we describe the enhancement multiple DISTINCT clause within one subselect.

3.5.1 The SAP usage of DISTINCT

In the ABAP SQL syntax, it is allowed to specify multiple DISTINCTs on different columns in a subselect. Prior to DB2 V8, this statement caused an SQL error with SQLCODE -127 (DISTINCT is specified more than once in a subselect) if the DISTINCT referred to different columns.

Let us look at the ABAP select statement in Example 3-2.

Example 3-2 ABAP select statement with two DISTINCTs

```

tables: db2jsinf.
data: cnt1 type i,
      cnt2 type i.
select count( distinct jobname ) count( distinct creator ) into (cnt1,cnt2) from db2jsinf.
....

```

The corresponding DB2 code is listed in Example 3-3.

Example 3-3 Corresponding DB2 select statement

```
SELECT COUNT( DISTINCT "JOBNAME" ) , COUNT( DISTINCT "CREATOR" ) FROM "DB2JSINF"  
FOR FETCH ONLY WITH UR
```

The only possibility to solve the problem with DB2 versions prior to V8 is to change the SAP application. Instead of one SQL statement, multiple SQL statements are necessary. In our case, we need two statements to get the same result. See Example 3-4.

Example 3-4 Solution with two ABAP SQL statements

```
select count( distinct jobname ) FROM db2jsinf.  
  
select count( distinct creator ) FROM db2jsinf.
```

But there is an additional disadvantage besides the necessity of a code change. It is much more efficient to scan the table only once and build the two counters in parallel instead of scanning the table twice. Both problems are solved by DB2 V8 supporting multiple DISTINCTs on different columns in one subselect.

Archived

SAP Business Information Warehouse

In this chapter we describe DB2 V8 features that are particularly beneficial for SAP Business Information Warehouse (BW). SAP BW is a scalable data warehouse system that integrates and analyzes business information and supports decision processes. It is one of the corner blocks of SAP NetWeaver. In contrast to traditional SAP systems, the fundamental trait of SAP BW is OLAP (online analytical processing) with fairly complex queries. This is the reason why before V8 DB2 subsystems needed to be tuned differently if they serve SAP BW systems rather than other SAP systems.

Through transfer rules from external systems, operational data enters an SAP BW system and is stored in the persistent staging area of the system. After being consolidated and cleansed, data is transferred to ODS (operational data store) objects or to InfoCubes. ODS objects support reporting functionality only. InfoCubes, on the other hand, are the base for multi-dimensional analytical processing. They are implemented by sets of tables that form a star or snowflake schema.

On a regular basis, large amounts of data are transferred from source systems to SAP BW systems. To cope with this data, SAP BW takes advantage of DB2's partitioning capabilities. Hence, the DB2 V8 enhancements in partitioning, that is the ability to rotate partitions, directly benefit SAP BW. These enhancements are described in 3.1, "Partitioning" on page 68.

This chapter covers the following topics:

- ▶ Star join enhancements
- ▶ Joins with up to 225 tables
- ▶ Common table expressions
- ▶ Materialized query tables

4.1 Star join enhancements

SAP BW models InfoCubes with dimension tables surrounding a centralized fact table, which is typically very large and may contain billions of data rows. The data model is highly normalized and therefore encompasses many tables. The dimension tables contain a finite number of descriptions of the event occurrences that are stored in the fact table. This avoids maintaining redundant descriptive data in the fact table. There is a high correlation among dimensions, which leads to a sparse nature of data in the fact table.

While the relationship between dimension tables and fact table is a parent-child relationship, SAP BW does not define foreign keys to manifest this. If dimensions consists of single tables, the schema is called a star schema. If they are made up of multiple tables, the schema resembles a snowflake and is therefore called a snowflake schema (see Figure 4-1). Unlike OLTP queries where a large number of short duration queries execute, OLAP queries involve a large number of tables and an immensely large volume of data to perform decision making tasks. Hence, OLAP queries are expected to run much longer than OLTP queries, but are less frequent.

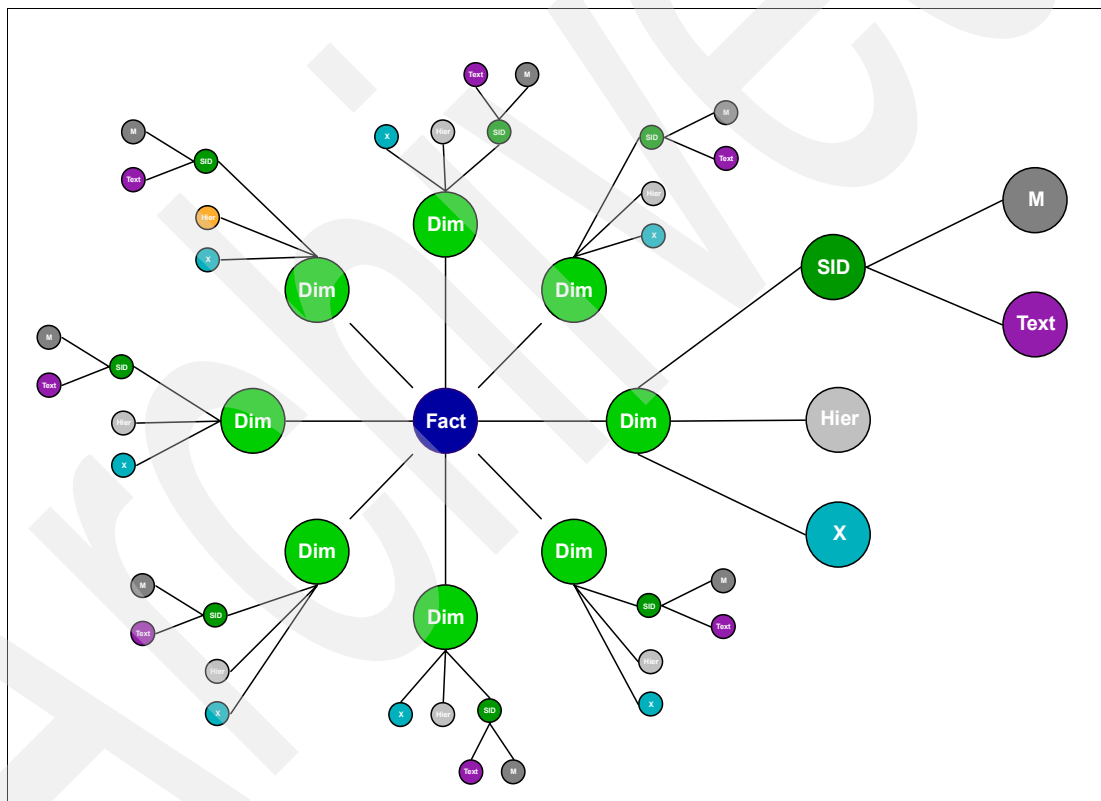


Figure 4-1 Snowflake schema

DB2 tackles star or snowflake schemas with the described attributes using the star join access path. Efficient star join processing is crucial to satisfactory SAP BW performance. Star join access involves logical Cartesian joins of dimension tables and joining the cartesian product to the fact table. In more detail, the general approach is as follows:

- **“Selectively” join from the outside-in:** Purely doing a cartesian join of all dimension tables before accessing the fact table may not be efficient if the dimension tables do not have filtering predicates applied, or there is no available index on the fact table to support all dimensions. The optimizer should be able to determine which dimension tables should be accessed before the fact table to provide the greatest level of filtering of fact table rows.

- ▶ **Efficient “cartesian join” from the outside-in:** A physical cartesian join generates a large number of resultant rows based on the cross product of the unrelated dimensions. A more efficient cartesian type process is required as the number and size of the dimension tables increase, to avoid an exponential growth in storage requirements. Index key feedback technique is useful for making cartesian joins efficient.
- ▶ **Efficient “join back”, inside-out:** The join back to dimension tables that are accessed after the fact table must also be efficient. Non-indexed or materialized dimensions present a challenge for excessive sort merge joins and workfile usage. To meet this challenge, DB2 V7 provided partial support with the introduction of sparse index on workfiles for the inside-out join phase only. DB2 V8 extends this further with controlled materialization and also in memory workfiles.
- ▶ **Efficient access of the fact table:** Due to the generation of arbitrary (or unrelated) key ranges from the cartesian process, the fact table must minimize unnecessary probes and provide the greatest level of matching index columns based on the pre-joined dimensions.

4.1.1 Sparse index in outside-in join phase

For an efficient cartesian process, DB2 employs a logical, rather than physical cartesian join of the dimension tables. Each dimension that the optimizer chooses to access before the fact table has all local predicates applied with the result sorted into join column order and is materialized into its own separate workfile. DB2 simulates a cartesian join by repositioning itself within each workfile during nested loop join processing to potentially join all possible combinations to the central fact table. Notice that the nested loop join is pushed down to Data Manager (stage 1) in this phase of star join processing. The sequence of this simulated cartesian join respects the column order of the selected fact table index.

The sparseness of data within the fact table implies that a significant number of values of the cartesian product do not match any fact table records. To optimize execution time, DB2 avoids joining unnecessarily derived rows to the fact table. It accomplishes this by introducing an index key feedback loop to return the next highest key value whenever a not-found condition is encountered. A matching record in the fact table returns the record. A miss returns the next valid fact table key so that DB2 can reposition itself within the dimension workfiles, thus skipping composite rows with no possibility of obtaining a fact table match. Figure 4-2 visualizes this process.

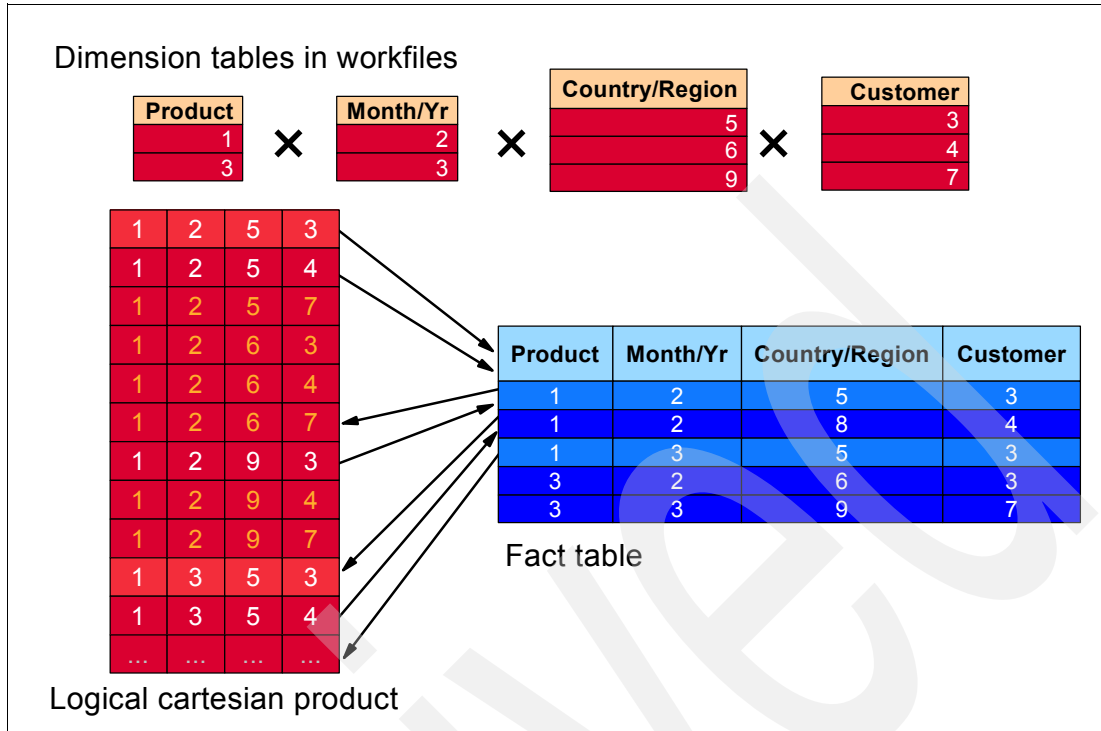


Figure 4-2 Star join: index key feedback

This approach allows significant skipping in the cartesian join, but skipping the index keys is not free. As there are no indexes on workfiles used in the outside-in phase of star join processing, these workfiles are scanned sequentially. If a dimension table is not at a leading position within the cartesian product, it is likely that the workfile containing it is scanned repetitively. In this case, the cost of skipping workfile entries may grow significantly.

DB2 V8 introduces sparse indexes on snowflake workfiles that are used during outside-in processing to overcome this problem. These sparse indexes reside in memory and consume up to 240 KB. If the total number of workfile entries is small enough, then all entries can be represented in the index. This provides a one-to-one relationship between index entries and workfile records. The index only becomes sparse if the number of index entries cannot be contained within the space allocated. The structure of a sparse index is flat rather than a b-tree structure of standard table indexes (see Figure 4-3). The index is probed through an equi-join predicate and a binary search of the index is utilized to find the target portion of the workfile. Then a sequential search of the identified workfile portion is performed to find the corresponding record.

The benefit of these sparse indexes is that they allow fast access to arbitrary records of snowflake workfiles that are used in the outside-in phase of star join processing. This ability is important when considering the index key feedback mechanism to process the virtual cartesian product.

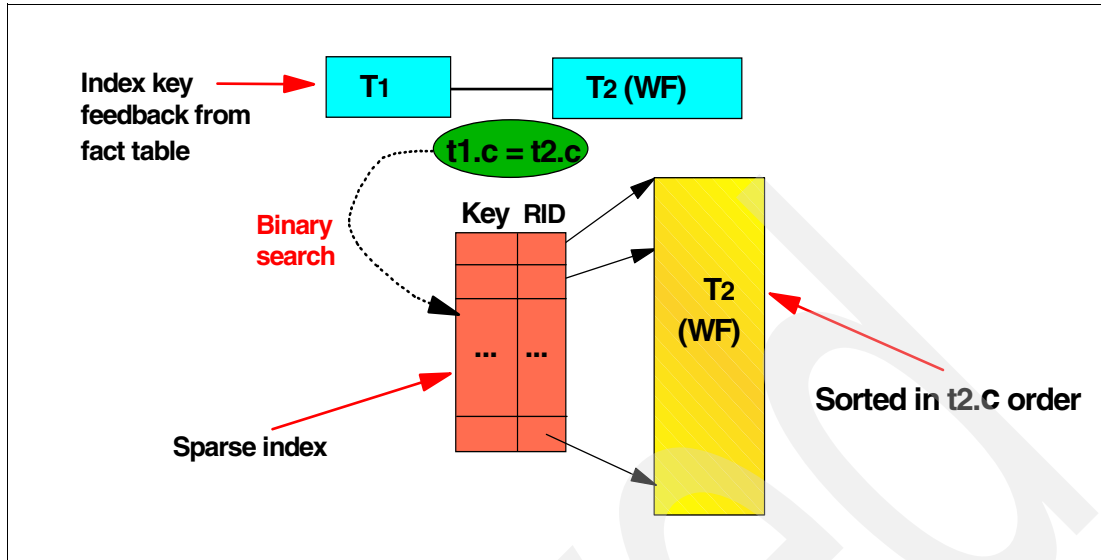


Figure 4-3 Star join: Sparse index

4.1.2 In-memory workfiles

While the sparse indexes avoid sequential scans of the workfiles, they do not eliminate potentially large random I/O activity. Therefore, DB2 V8 goes one step further and additionally supports in-memory workfiles. The in-memory workfiles contain all the columns of a workfile that are necessary to satisfy a query. These columns are the join column and the selected columns. It is a dense index in the sense that it contains an entry for each workfile record.

As the in-memory workfiles potentially save a large number of I/O operations against workfiles, they promise a considerable performance gain. Also, they may reduce contention of the workfile buffer pool, because they are cached in a separate storage pool. This is especially beneficial if concurrent sort operations are performed.

The new storage pool that is dedicated to holding in-memory workfiles is called star join pool. The DB2 ZPARM SJMXPOOL specifies its maximum size, which defaults to 20 MB. It resides above the 2 GB line and is only in effect when star join processing is enabled through ZPARM STARJOIN. When a query that exploits star join processing finishes, the allocated blocks in the star join pool to process the query are freed.

The use of the star join pool is not compulsory. If it is not created, star join processing takes place without using in-memory workfiles. Also, if the allocation of space for a workfile in the star join pool fails, because SJMXPOOL is reached, then processing falls back to using the new sparse index.

4.1.3 Sparse index during inside-out join phase

After joining the logical cartesian product of selective dimension tables to the fact table, the intermediate result (composite table) may still be large. The lack of indexes on workfiles used during the inside-out phase of star join processing makes the optimizer choose sort merge join to subsequently join the remaining dimension tables to the composite table. This may involve sorting the large composite table with excessive CPU and I/O consumption and increased parallelism overhead. Therefore, DB2 V7 introduced the support of sparse indexes on the snowflake workfiles of the inside-out phase. This allows the optimizer to consider nested loop join, which is more efficient in many cases.

Tip: APAR PQ61458 provides sparse indexes on the snowflake workfiles that are used in the inside-out join phase for DB2 V7.

DB2 V8 provides further enhancements in the inside-out join phase by being able to select supporting nested loop join (rather than merge scan join) and in-memory workfiles in this phase (see Figure 4-4). These in-memory workfiles behave the same as the in-memory workfiles from the outside-in phase and also populate the star join pool. Moreover, DB2 V8 accomplishes controlled materialization of snowflakes in this phase.

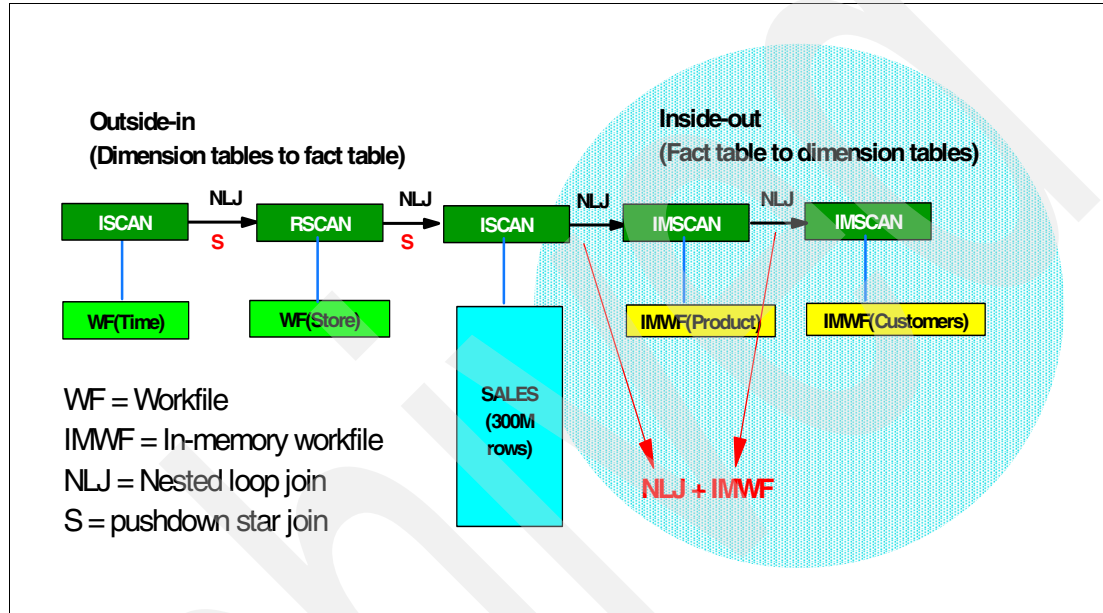


Figure 4-4 Star join: Inside-out join phase

4.1.4 Snowflake handling: Controlled materialization

Prior to DB2 V8, all snowflakes were materialized. This provided the benefit of simplified access path selection by reducing the overall number of tables joined.

For the inside-out join phase (post fact table), relatively small snowflakes, or snowflakes that provide adequate filtering, are good candidates to be materialized. With the introduction of in-memory workfiles and/or sparse index on workfiles, the snowflake, which may contain many tables, is resolved once and fact table rows are joined to a much smaller result set using an efficient join method that can take advantage of the in-memory or sparse index.

For large or non-filtering snowflakes, the materialization overhead may dominate the overall query time, and is therefore detrimental to query performance. For in-memory workfile and sparse index on workfile, the result must be sorted to allow a binary search to locate the target row. Sorting a large result can be expensive. If the memory is available for a very large result, the binary search for in-memory workfiles may result in multiple iterations to find the target row. If fallback occurs to sparse index, then the index may be too sparse, and therefore each locate in the workfile may still require a large sequential scan.

V8 introduces controlled materialization. The filtering of each snowflake is ranked, and only those snowflakes that provide adequate filtering compared to the base table size will be materialized.

The choice not to materialize can overcome the sort and workfile allocation overhead, and rather than requiring an index to be built on the workfile, the indexes on the underlying snowflake tables can be used for efficient joins after the fact table.

Besides the star join enhancements already described, DB2 V8 provides an improved cost estimation algorithm that better estimates the filtering effect of dimensions. This results in a better table join sequence and can yield a significant performance improvement.

4.2 Joins with up to 225 tables

The nature of queries against BW InfoCubes is that they are highly complex and easily involve dozens of tables. The access path that the DB2 optimizer selects for these queries is typically star join. The general limit of 15 tables, which prior versions of DB2 imposed, is therefore too low. To overcome this problem, DB2 V6 and DB2 V7 already raised the limit to 225 tables in the FROM clause if a query qualifies for star join processing. To completely get around the 15 table limit, customers can make use of the “hidden” ZPARM MXTBJOIN so that their queries can run. This parameter is hidden because, in general, there is a need for extra storage and processor time when dealing with these complex queries.

The reason the default limit on the number of tables joined has stayed at 15 is because there has been a risk that a large query could cause DB2 to consume extra amounts of resources (storage and CPU) when evaluating the cost of each possible join sequence.

In DB2 V8 the default limit is changed from 15 to 225 tables to be joined. This means that users can more easily join more than 15 tables. It also means that DB2 can join this many tables without restriction.

A number of enhancements have been implemented in DB2 V8 to reduce the amount of resources needed for the optimization process. This allows you to join more tables using less resources. A new functionality can recognize common query patterns (like star schema) and optimize large joins very efficiently.

The number of possible join permutations grows exponentially with the number of tables. To avoid excessive resource consumption during query optimization, DB2 V8 has enhanced the internal monitoring of how much storage and CPU is being consumed by the optimization process. If it exceeds certain thresholds, then curbs are put in place to force the optimization process to complete quickly. When excessive resources have been consumed by the optimization process, the goal changes — from selecting the “optimal” plan, to selecting a “reasonable” plan, in a minimal amount of time.

The resource threshold used is expressed in terms of storage (number of megabytes, MAX_OPT_STOR), CPU (number of seconds, MAX_OPT_CPU), and elapsed time (number of seconds, MAX_OPT_ELAP). The threshold is large enough so that most existing queries are not impacted, but small enough so that they prevent severe resource shortages. To guard against regressing existing queries, the threshold is only applied when the number of tables joined is greater than 15, the limit prior to DB2 V8. This limit can be changed through the hidden ZPARM TABLES_JOINED_THRESHOLD. SAP currently recommends to set this parameter to 10.

Tip: The ZPARM MXQBCE, which was introduced with DB2 V7, is still supported. If both MXQBCE and MAX_OPT_STOR/MAX_OPT_CPU/MAX_OPT_ELAP are set, DB2 V8 employs the more restrictive value. To avoid confusion, you should only use one option.

4.3 Common table expressions

DB2 V8 introduces common table expressions (CTEs), a new SQL feature that acts like a temporary view that is defined and used for the duration of a statement execution. There can be many CTEs in a single SQL statement, but each must have a unique name. Each CTE can be referenced many times in the statement, even by other CTEs, and all references to a CTE in a statement execution share the same result table. This differentiates them from regular views or nested table expressions (NTE), which are derived each time they are referenced. CTEs are introduced by the keyword WITH and occur at the beginning of the query.

Figure 4-5 contains a sample SQL statement that makes use of a CTE. The first CTE in the statement is the query for table expression E, which determines employee number, last name, salary, and hiring decade for all employees of the EMPLOYEE table. The columns of the associated result table are those named in the SELECT statement. For all available decades, the CTE M determines the minimum salary paid to the employees that were hired during the appropriate decade.

```
WITH
  E AS
  (
    SELECT EMPNO, LASTNAME, SALARY,
           SUBSTR (CHAR (HIREDATE, ISO), 1, 3) CONCAT '0 - 9'
           AS HIREDECADE
    FROM EMPLOYEE
  ),
  M (HIREDECADE, MINIMUM_SALARY) AS
  (
    SELECT HIREDECADE, MIN (SALARY)
    FROM E
    GROUP BY HIREDECADE
  )
SELECT E.EMPNO, E.LASTNAME, E.HIREDECADE,
       E.SALARY, M.MINIMUM_SALARY
FROM E INNER JOIN M
      ON E.HIREDECADE = M.HIREDECADE
```

Figure 4-5 Common table expression

For comparison, as shown in Figure 4-6, an NTE can be used to produce the same result set.

```
SELECT E.EMPNO, E.LASTNAME, E.HIREDECADE, E.SALARY, M.MINIMUM_SALARY
FROM
  (
    SELECT EMPNO, LASTNAME, SALARY,
           SUBSTR (CHAR (HIREDATE, ISO), 1, 3) CONCAT '0 - 9'
           AS HIREDECADE
    FROM EMPLOYEE
  ) AS E
INNER JOIN
  (
    SELECT S.HIREDECADE, MIN (S.SALARY) AS MINIMUM_SALARY
    FROM
      (
        SELECT SUBSTR (CHAR (HIREDATE, ISO), 1, 3)
              CONCAT '0 - 9' AS HIREDECADE,
              SALARY
        FROM EMPLOYEE
      ) AS S
    GROUP BY S.HIREDECADE
  ) AS M
ON E.HIREDECADE = M.HIREDECADE
```

Figure 4-6 Nested table expression

As the example shows, the column names of the CTE are specified in parentheses and follow the name of the CTE. This is the same technique as used in naming the columns of a view. The SELECT follows the common table expressions. Since it can refer to the CTEs, the SQL statement is more comprehensible compared to the use of NTEs.

The introduction of CTEs gives SAP BW on zSeries more flexibility, which is important in this fairly dynamic and complex field with challenging queries. Tasks for which SAP BW currently employs NTEs might be revisited in the future and solved by CTEs to take advantage of its benefits.

4.3.1 Recursive SQL

By means of CTEs, DB2 V8 introduces recursive SQL. Recursive SQL is very useful to retrieve data from tables that contain component breakdowns where each component is broken down into subcomponents and each subcomponent is broken down again into sub-subcomponents, etc. For example, BW InfoCubes support the definition of external hierarchies on columns of InfoCubes. External hierarchies span dimensions that allow multi-level grouping of data records.

Figure 4-7 shows a hierarchy that groups countries at different granularities. Hierarchy definitions are stored in single tables. To determine the countries in EMEA, for example, in a first run, the table first needs to be accessed to retrieve the subnodes Europe and Africa. Recursively, in a second run, the countries that are assigned to the nodes Europe and Africa are retrieved to establish the final result set.

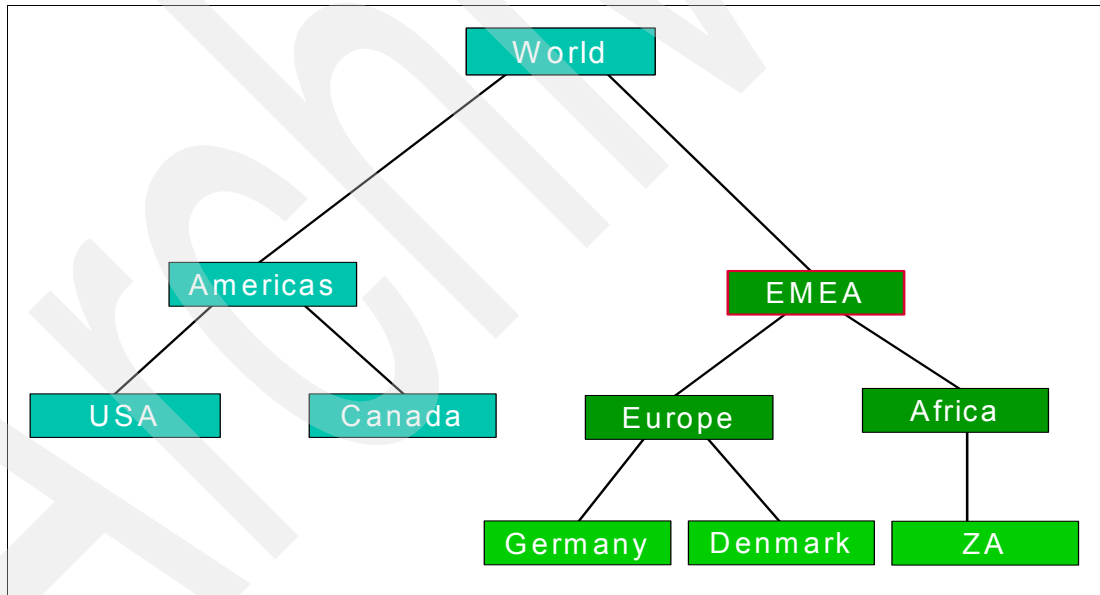


Figure 4-7 SAP InfoCubes: External hierarchy

The described query could be satisfied using the recursive statement from Figure 4-8. Recursive SQL involves defining a CTE that references itself. The initialization SELECT is executed only once. It controls the starting point of the recursion. The iterative SELECT is executed for all qualifying records and then repetitively for all further records that qualify.

While SAP BW today employs a database-independent approach to hierarchy processing, DB2 V8 allows for an alternative approach that might be considered in the future.

```

WITH
  RECURSIVETABLE (NAME, ISLEAF) AS
  (
    Initialization Select
    SELECT ROOT.NAME, ROOT.ISLEAF
      FROM HIERARCHYTABLE ROOT
      WHERE ROOT.PARENT = 'EMEA'
    UNION ALL
    Iterative Select
    SELECT CHILD.NAME, CHILD.ISLEAF
      FROM RECURSIVETABLE PARENT, HIERARCHYTABLE CHILD
      WHERE PARENT.NAME = CHILD.PARENT
        AND PARENT.ISLEAF = 'NO'
  )
    Main Select
  SELECT NAME
    FROM RECURSIVETABLE
    WHERE ISLEAF = 'YES'

```

Figure 4-8 Recursive SQL

4.4 Materialized query tables

The nature of queries in SAP BW is to access a significant amount of rows of very large fact tables, which may contain billions of rows, that are joined with several dimension tables. A typical query selects based on dimensions, aggregates on a few dimension columns, and applies column functions on the records of interest. Due to the large amount of data to be processed, these queries can take up a considerable elapsed time to process. In order to improve the performance and reduce the elapsed time of these queries, DB2 can exploit query parallelism.

A complementary approach is to somehow save (precompute and materialize) the results of prior queries and reuse these common query results for subsequent queries. DB2 V8 allows you to materialize and save these results for later use and avoid recomputation of the same result set, thus potentially reducing the elapsed time from hours down to minutes or seconds. The materialized result sets reside in so-called *materialized query tables (MQTs)*. If there are MQTs available that can be used to satisfy parts of a query, DB2 automatically rewrites the query. As a side-effect, the optimization of queries that reference tables on which MQTs are defined may increase due to the catalog accesses and processing during the automatic query rewrite phase.

SAP BW employs its own mechanism to precompute queries and materialize the result set. The result sets are stored in so-called aggregate tables. Aggregates are always associated with an InfoCube. Aggregates are computed by request or automatically after new data has been loaded. They are one of the corner blocks of SAP BW. As MQTs implement the same concept as aggregates and therefore would not add additional value, it does not make sense to exploit them for InfoCubes. However, aggregates cannot be defined for ODS objects. Hence, MQTs have the potential to accelerate ODS queries.

The design and use of materialized query tables involves trade-off between conflicting design objectives. On one hand, MQTs that are specialized to a particular query or set of queries lead to the largest performance benefits. However, this approach can lead to a proliferation of MQTs, because many are needed to support a wide variety of queries. Then the maintenance costs of MQTs can become an issue. On the other hand, MQTs that are more generic and that support a larger number of queries often tend to provide less performance improvement. Since there are fewer MQTs, the costs of maintaining them are reduced.

The following steps are necessary to exploit MQTs:

- ▶ Creating MQTs
- ▶ Populating and maintaining an MQT
- ▶ Enable the MQT for query optimization

4.4.1 Creating MQTs

As we mentioned previously, a materialized query table contains pre-computed data. The pre-computed data is the result of a full select on existing tables.

You can either:

- ▶ Create MQTs from scratch using CREATE TABLE statements.
- ▶ Change existing tables into an MQTs using ALTER TABLE statements.

Creating an MQT from scratch

The CREATE TABLE statement syntax has been enhanced to allow you to create an MQT. The new syntax is shown in Figure 4-9.

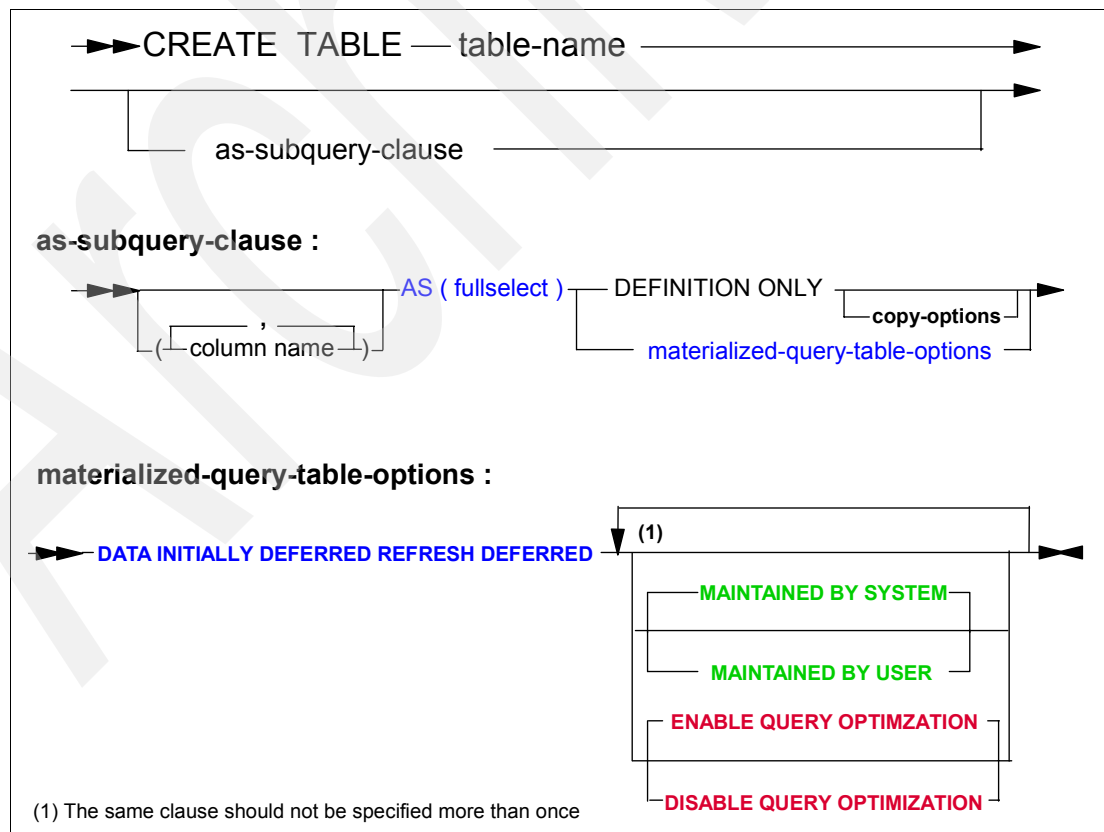


Figure 4-9 CREATE MQT: Syntax

Creating an MQT is similar to creating a view. In both cases you specify a fullselect to define its contents. The difference is that a view is only a logical definition, while a materialized query table contains materialized data of the query result on disk. For this reason, an MQT is also called a *materialized view*. Example 4-1 shows a CREATE TABLE statement to create an MQT. The fullselect, together with the DATA INITIALLY DEFERRED and REFRESH DEFERRED clauses, defines the table as a materialized query table.

Example 4-1 Sample create of a materialized query table

```
CREATE TABLE TRANSCNT (ACCTID, LOCID, YEAR, CNT)
  AS ( SELECT ACCTID, LOCID, YEAR, COUNT(*)
      FROM TRANS
      GROUP BY ACCTID, LOCID, YEAR )
  DATA INITIALLY DEFERRED
  REFRESH DEFERRED
  MAINTAINED BY SYSTEM
  ENABLE QUERY OPTIMIZATION;
```

The column names of an MQT can be explicitly specified or be derived from the fullselect associated with the table.

You have the option to disable a MQT from being considered for automatic query rewrite. If it is disabled, then there are fewer restrictions on the fullselect that defines an MQT.can be specified for it.

Registering existing tables as MQT

Customers may have already used regular tables as a form of materialized query tables and implemented their own populating mechanisms. With DB2 V8, to take advantage of the automatic query rewrite for these tables, you can register these existing tables as materialized query tables. This can be achieved by using a new clause of the ALTER TABLE statement.

The statement in Example 4-2 registers a table TRANSCOUNT as a materialized query table with the associated subselect to DB2. The data in the table will remain the same as indicated by DATA INITIALLY DEFERRED, and will still be maintained by the user, as specified by the MAINTAINED BY USER clause. The user can continue to LOAD, INSERT, UPDATE, or DELETE data in the table TRANSCOUNT. ALTER TABLE can also change a materialized query table into a base table.

Example 4-2 Converting a base table into an MQT

```
ALTER TABLE TRANSCOUNT ADD MATERIALIZED QUERY
  (SELECT ACCTID, LOCID, YEAR, COUNT(*) as cnt
   FROM TRANS
   GROUP BY ACCTID, LOCID, YEAR)
  DATA INITIALLY DEFERRED
  REFRESH DEFERRED
  MAINTAINED BY USER;
```

The ALTER TABLE statement can be used to enable — which is the default — or to disable an existing materialized query table for consideration by automatic query rewrite. Altering a table to change it to an MQT with query optimization enabled makes the table immediately eligible for use in query rewrite. When altering a table this way, it is important to pay attention to the accuracy of the data in the table. If the contents of a table does not yet properly reflect the contents of corresponding source table, then the table should be altered to an MQT with query optimization disabled. In a subsequent step the table should be refreshed and enabled with query optimization.

You can also switch materialized query table types between system-maintained and user-maintained with the ALTER TABLE statement.

4.4.2 Populating and maintaining an MQT

The time when a materialized query table is populated with the pre-computed data depends on the definition of DATA INITIALLY DEFERRED or REFRESH DEFERRED.

DATA INITIALLY DEFERRED means that when a MQT is created, the MQT is not populated by the result of the query.

REFRESH DEFERRED means the data in the MQT is not refreshed immediately when its source tables are updated. However the data can be manually refreshed at any time, for example by using the REFRESH TABLE statement.

Important: With DB2 V8, all MQTs have to be defined as DATA INITIALLY DEFERRED and REFRESH DEFERRED. This means the user has to ensure that the data currency meets the user requirements to avoid using outdated data and that the user is responsible to keep the data in the MQT up to date.

The MAINTAINED BY option indicates how the data in the MQT is to be refreshed:

- ▶ **MAINTAINED BY SYSTEM**, which is the default, indicates that the MQT is system-maintained. The only way to refresh the data in a system-maintained MQT is by using the **REFRESH TABLE** statement. A system-maintained MQT cannot be updated by using the LOAD utility, INSERT, UPDATE or DELETE SQL statements. Therefore, a system-maintained MQT is read-only. If a view or a cursor is defined on a system-maintained MQT, it becomes read-only.
- ▶ Alternatively, **MAINTAINED BY USER** can be specified to define a user-maintained MQT. A user-maintained MQT can be updated by the LOAD utility, INSERT, UPDATE or DELETE SQL statements, as well as the REFRESH TABLE statement. Therefore, a user-maintained MQT is updatable.

Note: With DB2 UDB for Linux, UNIX, and Windows the REFRESH TABLE is only allowed on system-maintained tables.

REFRESH TABLE

The REFRESH TABLE statement can be used to populate an MQT. See Example 4-3 for a sample REFRESH TABLE statement.

Example 4-3 Sample REFRESH TABLE statement

```
REFRESH TABLE mq_table;
```

The REFRESH TABLE statement:

1. Deletes all rows in the MQT using mass delete, if the tablespace was defined as segmented.
2. Executes the fullselect in the MQT definition to recalculate the data from the source tables that are specified in the fullselect with the isolation level for the materialized query table (as recorded in the catalog).
3. Inserts the calculated result into the MQT.
4. Updates the catalog for the refresh timestamp and cardinality of the MQT.

Even though the REFRESH TABLE statement involves delete and insert, it is a single commit scope. All changes made by the REFRESH TABLE statement are logged.

The REFRESH TABLE statement is an explainable statement. The Explain output contains rows for INSERT with the fullselect in the MQT definition.

4.4.3 Automatic query rewrite using MQTs

A major advantage of MQTs is that the optimizer understands them. While applications always reference the source table, the optimizer takes a look at the original query during access path selection and determines whether the referenced tables can be replaced by an MQT to reduce the query cost.

The process of recognizing when a materialized query table can be used in answering a query, deciding whether one or more MQTs should actually be used in answering a query, and rewriting the query accordingly, is done by a DB2 function called *automatic query rewrite* (AQR).

Automatic query rewrite is based on the fact that the submitted query may share a number of common sub-operations specified in the fullselect of a materialized query table definition. Therefore, the result of the submitted query can be derived from or can directly use the result of one or more MQTs. In other words, the automatic query rewrite process analyzes the user query to see if it can take advantage of any of the existing MQTs, by “proving” that the contents of a materialized query table overlaps with the content of a query, and compensating for the non-overlapping parts. When such an overlap exists, the query and the MQT are said to match. After discovering a match, the query is rewritten to access the matched materialized query table instead of one or more source tables, originally specified in the query.

The automatic query rewrite process searches for matched MQTs that result in an access path with the lowest cost after rewrite. The costs of the rewritten query and the original query are compared and the one with the lowest cost is chosen. If the final query plan comes from a rewritten query, the PLAN_TABLE shows the name of the matched MQTs and the access path using them. For more information on the information about MQTs in the PLAN_TABLE, see 4.4.4, “Determining if query rewrite occurred” on page 105. No authorization on an MQT is required for it to be used in automatic query rewrite.

There are a number of options and settings that affect whether or not an MQT is considered by AQR. Some of these options and settings are considered in the following.

DDL options

You can specify the following options on the CREATE or ALTER TABLE statement that affect whether or not DB2 will consider an MQT during automatic query rewrite.

- ▶ *ENABLE QUERY OPTIMIZATION*, which is the default, specifies that the MQT can be exploited by automatic query rewrite.
- ▶ When the *DISABLE QUERY OPTIMIZATION* clause is specified, the MQT is not considered by the automatic query rewrite process.

In addition, automatic query rewrite only considers a system-maintained MQT if a REFRESH TABLE has occurred. When using user-maintained MQTs, you may wish to create the MQT with the *DISABLE QUERY OPTIMIZATION* option, and ALTER it later to *ENABLE QUERY OPTIMIZATION*, once the table has been (re)populated.

Special registers

The new special registers `CURRENT REFRESH AGE` and `CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION` also control whether or not an MQT can be considered by automatic query rewrite for a dynamically prepared query.

- ▶ *CURRENT REFRESH AGE*. The value in this special register represents a refresh age. The refresh age of an MQT is the time between the current timestamp and the time that the MQT was last refreshed using the `REFRESH TABLE` statement. The latter information is recorded in the `REFRESH_TIME` column of the `SYSVIEWS` system catalog table.

In DB2 V8, only `CURRENT REFRESH AGE` of 0 or `ANY` is supported:

- 0 means only MQTs that are kept current with the source tables are considered by automatic query rewrite. Since DB2 V8 does not support immediately refreshed MQTs, specifying 0 means that AQR will not consider any MQTs.
- *ANY* represents the maximum duration, meaning all MQTs are considered by automatic query rewrite.

A subsystem default value for `CURRENT REFRESH AGE` can be specified in the `CURRENT REFRESH AGE` field on panel `DSNTIP4` at installation time, `DSNZPARM REFSHAGE`.

- ▶ Besides the `REFRESH TABLE` statement, user-maintained MQTs can be updated using `INSERT`, `UPDATE`, or `DELETE` SQL statements, or through the `LOAD` utility. Therefore, the refresh age of a user-maintained MQT can no longer truly represent the freshness of data in the MQT. Hence, the new special register *CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION* is used to determine which type of MQTs — system-maintained or user-maintained — are considered by automatic query rewrite.

- **ALL** Indicates that all MQTs are considered by automatic query rewrite.
- **NONE** Indicates that no MQTs are considered.
- **SYSTEM** Indicates that (under the assumption that `CURRENT REFRESH AGE` is set to `ANY`), only system-maintained MQTs are considered.
- **USER** Indicates that (under the assumption that `CURRENT REFRESH AGE` is set to `ANY`), only user-maintained MQTs are considered.

A subsystem default value for `CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION` can be specified in the `CURRENT MAINT TYPES` field on panel `DSNTIP4` at installation time, `DSNZPARM MAINTYPE`.

MQTs created or altered with `DISABLE QUERY OPTIMIZATION` specified are not eligible for automatic query rewrite, thus, are not affected by the above special registers. If a system-maintained MQT has not been populated with data, then the MQT is not considered by automatic query rewrite. For a user-maintained MQT, the refresh timestamp in the system catalog table is not maintained.

4.4.4 Determining if query rewrite occurred

You can use the `SQL EXPLAIN` statement to determine if DB2 has rewritten a user query to use an MQT. When DB2 rewrites a query, the `PLAN TABLE` shows the name of the employed MQT in the `TNAME` column instead of the table you specified in the query. The value of the `TABLE_TYPE` column is set to "M", which indicates that the table in the `TNAME` column is an MQT.

This information is also available in mini-plan performance trace record (IFCID 0022).

Archived



Performance

In this chapter we describe the following performance related enhancements:

- ▶ Early tests and general expectations
- ▶ Lock issues
- ▶ Multi-row fetch and insert
- ▶ Query optimization
- ▶ Support for backward index scan
- ▶ Faster DSC short prepares
- ▶ Data sharing enhancements

5.1 Early tests and general expectations

The workload of large DB2 users has been growing tremendously in the last two decades. Complex and large systems have been reaching bottlenecks within the DB2 subsystem that limit the exploitation of the zSeries architecture. The major limit to be removed to allow continuous growth is related to the 2 GB virtual storage per address space in a 31-bit architecture, which in turn limits the utilization of the available larger real storage. Real storage usage has been growing at the rate of about 20 to 30% a year in line with the growth of threads and storage pools. With DB2 V7, the DBM1 address space size would not be able to scale performance and utilize the CPU speed and the 128 GB of real storage now available with the z990 machine, because it can prevent full CPU usage by limits in buffer pools and concurrent threads.

A large amount of new code has therefore been introduced in DB2 V8 in order to align with the zArchitecture and start exploiting the 64-bit virtual addressing.

Extra code means larger module sizes and larger control blocks, which in turn imply an increase in real storage utilization. Some CPU time increase is also unavoidable to execute the extra code necessary for the 64 bit virtual support, the increase in size for names and keys, the increase in size and complexity of the SQL statements, and the Unicode catalog support.

The bottom line is that, in order to put DB2 in a position to continue growing beyond the current limits, there is a delta to be paid in real storage and CPU utilization. For estimates on the virtual and real storage requirements of DB2 subsystems that serve SAP systems, see the SAP documentation *SAP on IBM DB2 UDB for z/OS: Database Administration Guide - SAP NetWeaver Version 6.40* or SAP note 717215

Once positioned with the requirements (z/OS 1.3 or later, and zSeries machine) you will be able to continue to grow linearly and take advantage of the performance enhancements of DB2 V8.

In terms of expectations, the major performance enhancements can be associated to the following three groups:

- ▶ 10 to 100 times possible performance improvements
 - MQT
 - More indexable predicates
 - DSTATS
- ▶ 2 to 10 times possible performance improvements
 - Star join work file index and in memory work file
 - DBM1 virtual storage constraint relief
 - Partition load/reorg/rebuild with dpsi
- ▶ Up to 2 times (more in distributed systems)
 - Multi-row fetch, cursor update, cursor delete, insert

See 1.3, “DB2 UDB for z/OS Version 8 overview” on page 13 and related bibliography for a description of the foregoing functions.

5.2 Lock issues

In this section we describe typical locking problems within an SAP system and show how the new features in DB2 V8 for z/OS will help to solve these problems.

Locking problems within an SAP system, especially if the system is under heavy load, can cause the unavailability of the system for every user. If this happens, the responsible support personnel have to resolve the problem in a very short time — the pressure on them is very high. On the other hand, these are the kind of problems that are not easy to find. So, what the support person usually does, is to find out who is blocking the system and just kill this process. This helps the system to run again, but it does not solve the problem. To see exactly what caused the locking problem, heavy traces are needed. Because these traces produce a large amount of output, it is very helpful to reproduce the lock situation in a small, controlled environment. Unfortunately, very often the lock situation cannot be reproduced easily and only occurs if the system is under heavy load. In this case, a heavy trace can affect the system badly and also produce large output, which is hard to analyze.

You can attack these problems in two ways:

- ▶ By providing better tools to analyze locking problems
- ▶ By avoiding locking problems at all

The second way is obviously much more attractive. DB2 for z/OS offers considerable help for an application such as SAP to avoid locking problems.

In general, avoiding or eliminating lock situations provides relief in many areas, such as these:

- ▶ The system is no longer unusable because a crucial resource is blocked.
- ▶ Better overall system performance is experienced.
- ▶ Better single transaction performance occurs.
- ▶ There are no abnormal endings of transactions because of timeouts.
- ▶ Administration is easier because the objects are not blocked.
- ▶ There are fewer complaints by end users.

5.2.1 Release locks at close

The SAP applications make extensively use of business objects such as documents. A single object can easily exceed the maximum record size (32 KB) of DB2. Because of this, SAP splits each object into pieces, orders them by a sequence number, and stores each piece in a database row. All rows together make up the business object. To rebuild the business object out of the database rows, SAP reads all rows belonging to one object in the order of the sequence number. The SAP database layer must ensure that it reads a consistent object.

To read several rows consistently, the SAP database layer sets the isolation level to read stability (RS). DB2 takes a share lock for each row it returns. Unlike the isolation level, cursor stability (CS), the share locks stay if the cursor moves forward. The isolation RS guarantees that nobody can update a row that was read with RS until COMMIT. This is more than SAP needs. After reading the rows consistently, the SAP application doesn't care about changes to these rows.

Attention: The SAP enqueue mechanism is not used when an ABAP program reads a business object. To read a business object consistently, SAP relies on the DB2 locking mechanism. SAP is using this enqueue mechanism to protect a business object that has been changed in the dialog process (the change has only been done in memory and then logged in a special table just before commit). The enqueue gets released by the asynchronous update process after the real database update of the business object has been performed.

The combination of holding shared locks until COMMIT and not committing long-running batch programs very often causes lock contentions within an SAP system. The batch program reads rows with isolation level RS and runs for hours without issuing a COMMIT. Each transaction that tries to update such a row will fail with an SQLCODE -913 and REASON CODE 0C9008E (timeout).

With the exploitation of the *release locks at close* capability of DB2 V8, shared locks can be released at close cursor time. SAP exploits this feature starting with release 6.40 of WebAS. This kind of problem should not occur any more.

5.2.2 Reduce lock contention on volatile tables

A volatile table is a table whose contents can vary from zero to very large at run time. In an SAP system this kind of table is used, for instance, to store business data created during the day and cleaned up after the business day.

Problem description and solution

DB2 often does a table space scan or non-matching index scan when the data access statistics indicate that a table is small, even though matching index access is possible. This becomes a problem if the table is small or empty when statistics are collected, but the table is large when it is queried. In that case, the statistics are not accurate and this can lead DB2 to pick an inefficient access path. Favoring index access may be desired for tables whose size can vary greatly.

So, the table starts with 0 records and gets filled during the day. If DB2 picks a table space scan as access path, you can see a dramatically increasing response time during the day. Running a RUNSTATS if the table is big enough would certainly help. But in an SAP system, it is a huge effort to watch each volatile table and check if the table is big enough and index access is needed. The installation of a hot package or a change in the customizing can easily change the number of volatile tables in the system.

Prior to V8, two methods could be used to address the challenge, but neither of them without deficiencies. Setting ZPARM NPGTHRSH to -1 practically disables cost based optimization. An index access is used for every statement. As the cost based optimization is one of the DB2 “jewels” this approach is strongly discouraged. Notice that setting NPGTHRSH to some positive value limits the number of tables for which the index access is preferred to those with NPAGES lower than the NPGTHRSH setting. Therefore, this ZPARM is intended to ensure the index access path on tables that fluctuate in their cardinality, but it does not cover all the volatile tables (those with NPAGES higher than the NPGTHRSH value).

Another approach is to update the catalog statistics so that the required access path is likely to be selected, but this is both unreliable (there is no guarantee that the specific access path will be actually selected) and difficult to administer. Every RUNSTATS removes the updates (so the RUNSTATS must be avoided for these tables). Performing the updates creates contention with Prepare statements and DDL.

SAP is using NPGTHRSH to ensure that unknown tables that quickly change their sizes are also accessed via an index. From SAP's point of view, this is an ideal solution for accessing volatile tables. It is much better to get a little less performance than possible if there are only a few rows in the table and the best possible performance if there are millions of rows in the table, instead of being a bit faster when the table has just a few rows in it, but getting the worst possible access path with a horrible performance if there are millions of rows in the table. The volatile table enhancement in DB2 V8 provides a way to tell DB2 the special characteristic of the table. If an application accesses a table marked as volatile, DB2 is using the index whenever possible. Unfortunately, the volatile feature of DB2 V8 would not help in this case because it is only applicable if you know the volatile tables up-front. Therefore the volatile feature is not used by SAP to replace NPGTHRSH.

SAP stores logical business objects in SAP cluster tables. The size of a logical object can easily exceed 32 KB. Because of this, a logical object traditionally has to be stored in multiple records. These stored objects need to be accessed in a strictly prescribed way — for example, the first row in a multi-row object, followed by the second row, etc. This can be achieved only if the same access path is always used when accessing the tables, irrespective of the catalog statistics and other factors that influence the optimizer. In other words, these tables need to be accessed according to the following rules:

- ▶ If there is an index that has at least one matching column for given statement's predicates, a tablespace scan should be avoided.
- ▶ If more than one such index exists, choose the one with the largest number of matching columns.
- ▶ If there are still more than one qualifying, then choose the one that provides ordering (if applicable).
- ▶ No list prefetch nor sequential prefetch should be selected.

If this access pattern is not used major performance and contention problems (typically deadlocks) can be expected.

DB2 V8 addresses the problem by introducing a new table attribute, VOLATILE. If specified, the access path of any statement that includes that table is predetermined: it will be an index scan with as many matching columns as possible, with no prefetch and, if possible with no sort. A tablespace scan is used only if there is no index with at least one matching column.

The new attribute can be specified at table creation, but also, the existing tables can be altered to be VOLATILE. Although the name suggests differently, the main usage of the option is not for the tables with volatile cardinality (the existing ZPARAM NPGTHRSH addresses that issue well enough), but for the multi-row object tables and queues. You can identify tables for which the VOLATILE attribute has been set by querying SYSIBM.SYSTABLES and looking for a 'Y' in the SPLIT_ROWS column.

SAP volatile table example

In Figure 5-1 we show one example of a physical cluster table.

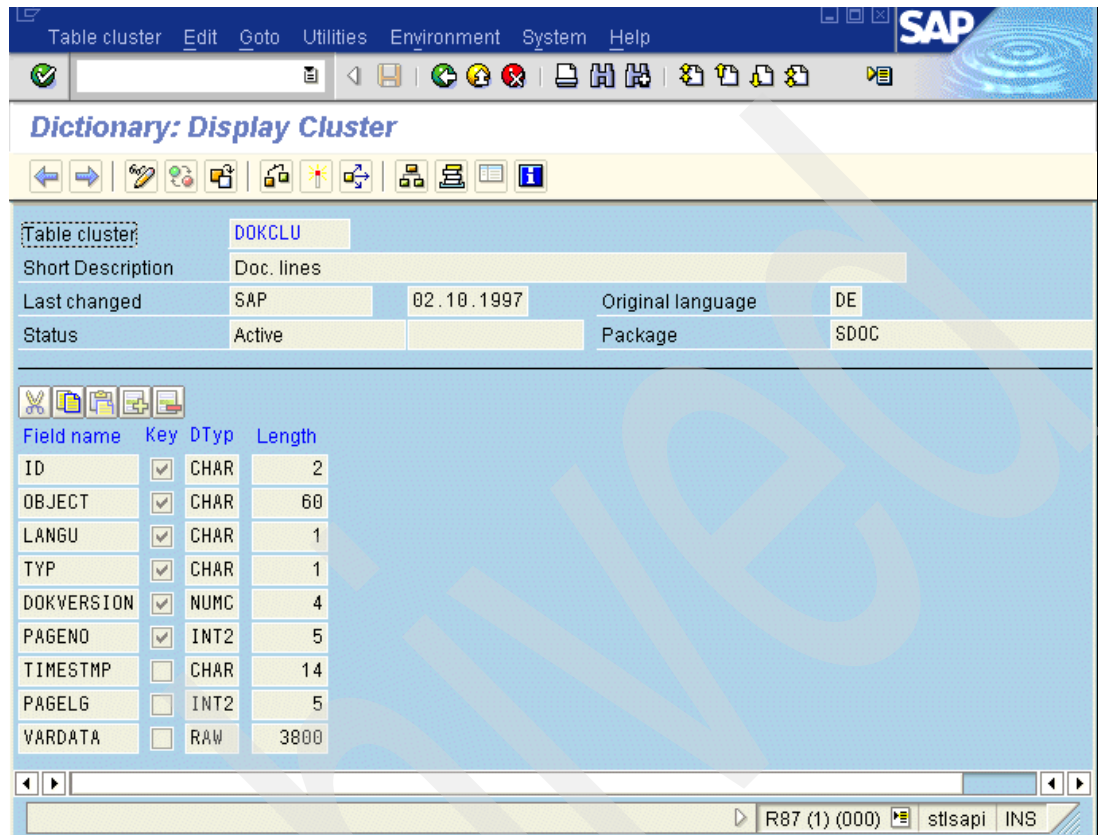


Figure 5-1 SAP definition of the physical table cluster DOKCLU

Figure 5-2 shows the corresponding logical table.

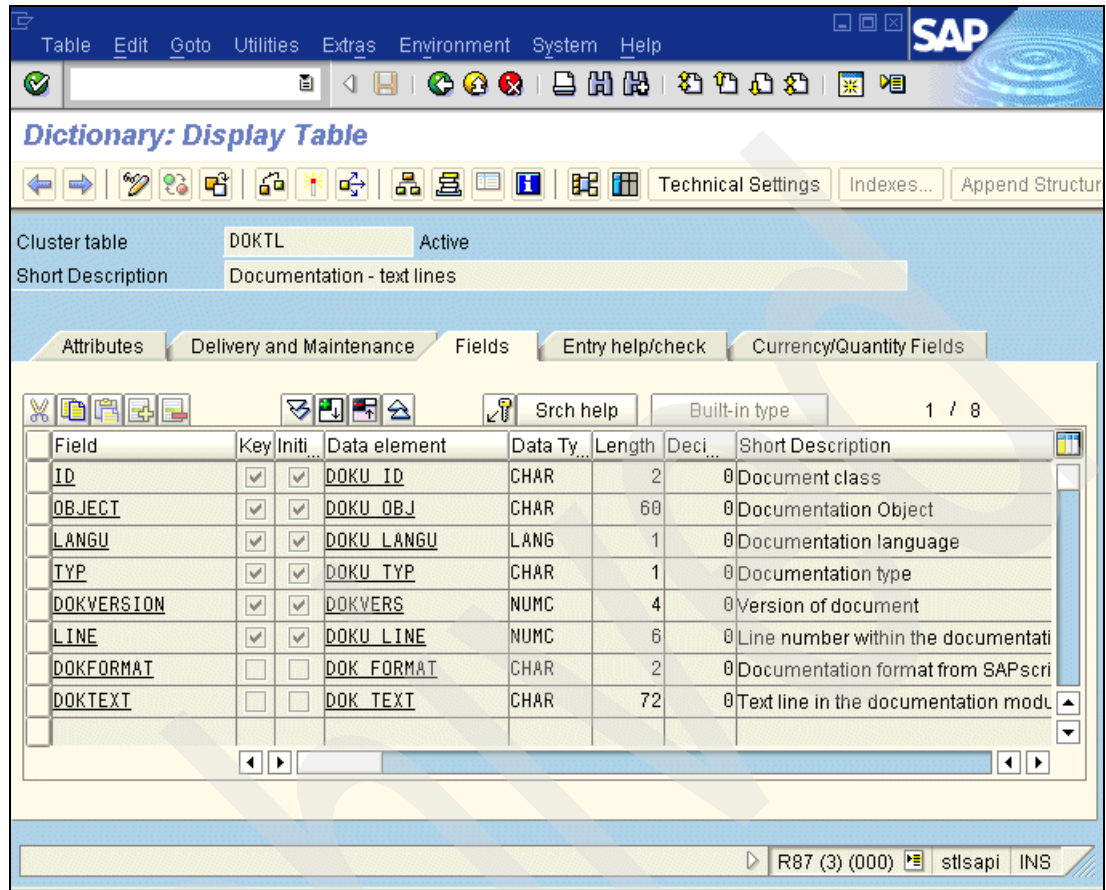


Figure 5-2 Definition of the logical cluster table DOKTL based on table cluster DOKCLU

In Example 5-1 we show the SQL accessing the cluster table.

Example 5-1 Access to the cluster table

The following ABAP statement:

```

select * from doktl
where id = 'FU'
and object = 'ADDRESS_INTRO_PRINTFORM'
and langu = 'D'
and typ = 'T'
and dokversion = 1.
write: / doktl-line, doktl-doktext.
endselect.

```

gets converted to the following DB2 statement (ST05 output):

SQL Statement

```

SELECT
  "ID" , "OBJECT" , "LANGU" , "TYP" , "DOKVERSION" , "PAGENO" , "TIMESTAMP" ,
  "PAGELG" , "VARDATA"
FROM
  "DOKCLU"

```

```

WHERE
  "ID" = ? AND "OBJECT" = ? AND "LANGU" = ? AND "TYP" = ? AND "DOKVERSION" = ?
ORDER BY
  "ID" , "OBJECT" , "LANGU" , "TYP" , "DOKVERSION" ,
  "PAGENO" FOR FETCH ONLY WITH RS

```

Variable

```

A0(CH,2) = FU
A1(CH,60) = ADDRESS_INTRO_PRINTFORM
A2(CH,1) = D
A3(CH,1) = T
A4(NU,4) = 0001

```

In Example 5-1, the SAP cluster interface returns 400 records from the logical table DOKTL back to the ABAP program, but it reads only 3 rows from the physical table cluster DOKCLU. The ABAP programmer always expects to read a consistent object. To ensure that, the SAP cluster interface reads the physical records always in the ascending order of the field 'pageno'. A reader of a logical object would get a corrupted object if a concurrent writer is able to overtake the reader.

To avoid reading corrupted data, the SAP cluster interface updates a logical object in ascending order of the field 'pageno' (one update per record). In addition, the database dependent interface for DB2 sets the isolation level to RS. Taken together, this guarantees that the SAP cluster interface always reads consistent logical objects. But this solution only runs fine if the index of the table cluster is always used when the application reads a logical object. Using the table cluster index is the only way to force DB2 to read the logical object in ascending order beginning with pageno=0. If the index is not used for any reason, the danger of getting timeouts and deadlocks increases dramatically. Unfortunately, it turns out that this is sometimes the case at the customer side.

To get rid of lock contention on SAP cluster tables, SAP is using the two features of DB2 V8:

- ▶ Releasing lock at close (see 5.2.1, "Release locks at close" on page 109)
- ▶ Defining the table as volatile to always guarantee index access

DB2 V8 adds two new keywords to the CREATE TABLE and ALTER TABLE statements; VOLATILE (to force index access whenever possible) and NOT VOLATILE (to allow any type of access to be used). Notice that DB2 uses the CLUSTER keyword for other purposes.

- ▶ **VOLATILE:** Specifies that for SQL operations, index access is to be used on this table whenever possible. However, be aware that by specifying this keyword, list prefetch and certain other optimization techniques are disabled.
- ▶ **NOT VOLATILE:** Specifies that SQL access to this table will be based on the current statistics. This is the default.
- ▶ **CARDINALITY:** An optional keyword expressing the fact that the table can have frequently changing cardinality; it can have only a few rows at times, and thousands or millions of rows at other times. This keyword is allowed for DB2 family compatibility, but will serve no additional function in DB2 for z/OS.

Explain

In Figure 5-3 we show a simple example of Explain for the SQL statement accessing the ZPMTEST table.

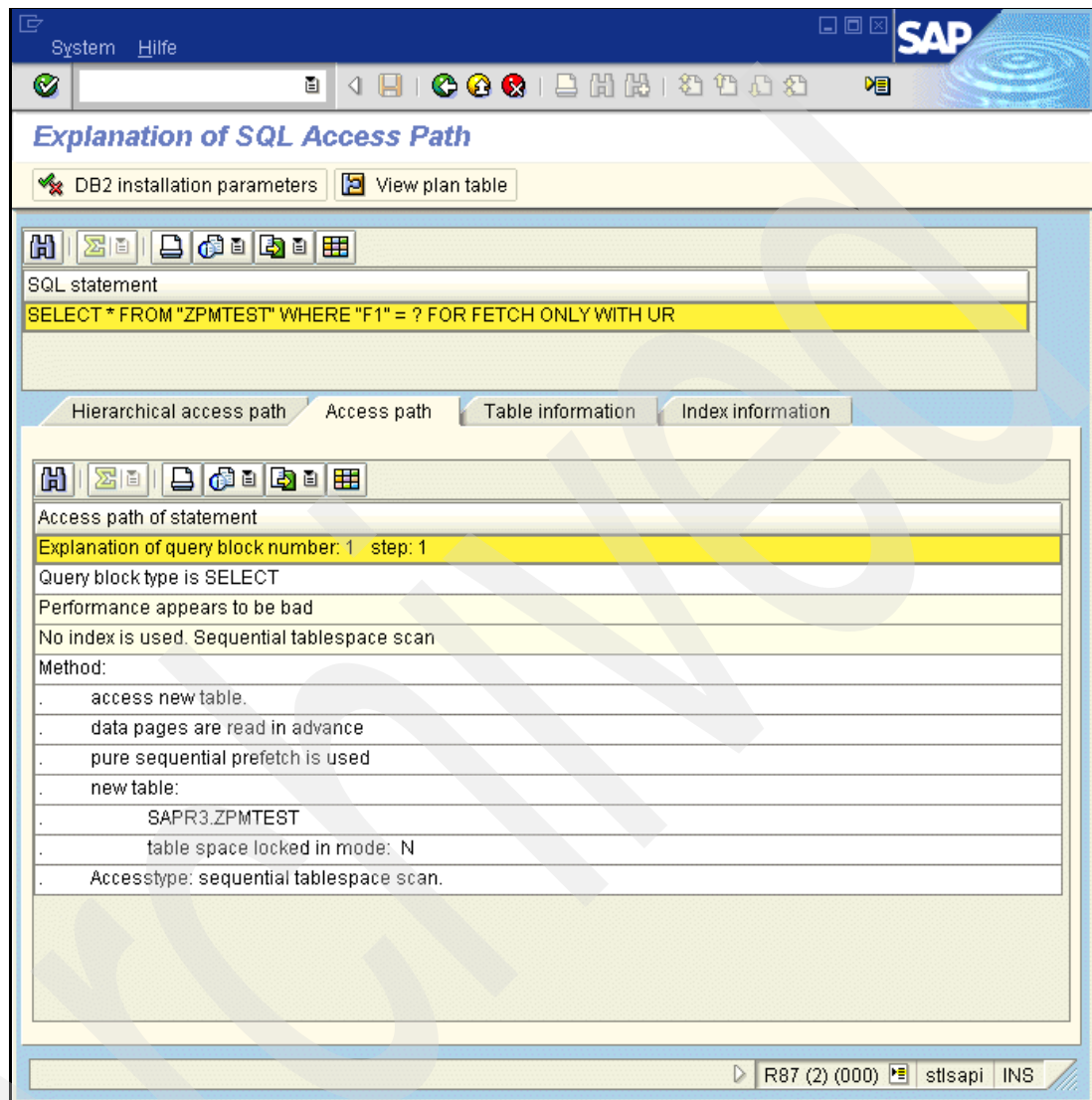


Figure 5-3 Explain output before the ALTER statement

We now issue the following ALTER statement:

```
ALTER TABLE ZPMTEST VOLATILE
```

The new Explain output is listed in Figure 5-4.

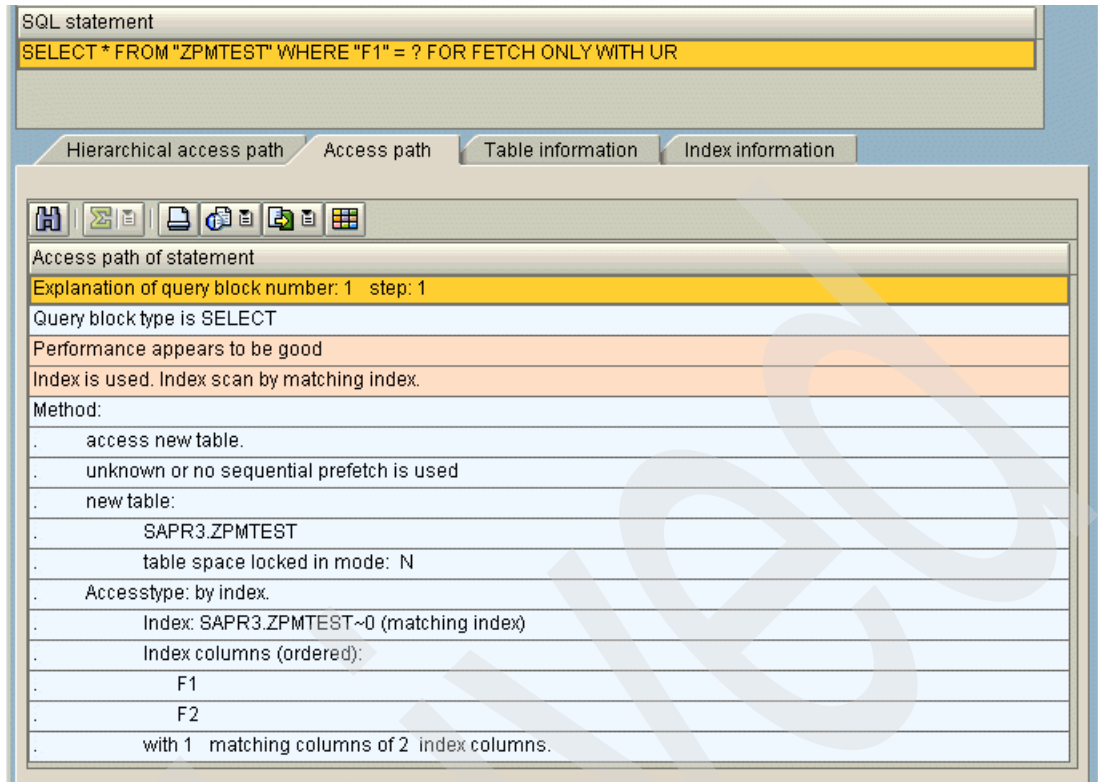


Figure 5-4 Explain output after the ALTER statement

Once the cluster table is defined as volatile in DB2, access by index is selected by the optimizer.

5.2.3 Partition key update

Within the SAP ABAP language it is allowed to update any field even if it is part of the key. DB2 V7 already allows you to update the partitioning key. But DB2 V7 does not allow concurrency if an update of a partitioning key changes the partition to which the row belongs. If the update requires moving the data row from one partition to another, DB2 V7 tries to take exclusive control of the objects to perform the update by acquiring DRAIN locks. Because of this, no other application can access the range of partitions affected by the update of values in the partitioning key columns. Following is the list of objects on which DB2 V7 takes exclusive control to perform the update:

- ▶ The partition of the table space from which the row is moving, the partition of the table space to which the row is moving, and all partitions in between
- ▶ The partition of the partitioning index from which the partitioning key is moving, the partition of the partitioning index to which the partitioning key is moving, and all partitions in between
- ▶ Non-partitioning indexes defined on the table space

With the DB2 V8 capability of NOT taking exclusive control of the objects to perform the update, concurrency is allowed if an update of a partitioning key changes the partition to which the row belongs.

5.3 Multi-row fetch and insert

In this section we describe the usage of multi-row fetch and multi-row insert in an SAP environment. Prior to DB2 V8, SAP already used multi-row operations, but only through the network, in order to save network trips. On the server side, SAP had to process row by row. With this new feature, SAP can reduce the number of DB2 calls dramatically. DB2 V8 itself benefits a lot. It is no longer necessary to go through the whole software stack for each row. The benefit is obvious. The usage of multi-row operations saves cross-memory trips and significantly reduces elapsed time and CPU cost on the server.

5.3.1 How SAP applications benefit from multi-row operations

SAP has been using multi-row operations since the first version of SAP R/3. The whole software stack down to the database interface supports multi-row operations. A simple change in the database layer is enough to exploit multi-row operations for every single SAP application. Because a multi-row operation for a single row has no disadvantages compared to the single-row operation, SAP is always using multi-row operations.

5.3.2 Implementation details

In general, a loop gets replaced by a single call on the server side.

The old code, needed prior to DB2 V8, is shown in Example 5-2. The DB2 call sequence of an ABAP INSERT operation is:

```
EXECUTE .... EXECUTE
```

The DB2 call sequence of an ABAP READ operation is:

```
: OPEN; FETCH ... FETCH; CLOSE
```

Example 5-2 Processing each record in a loop

```
for ( i = 0; i < row_count; i++ )
{
    /* process single row */
    /* insert case */
    EXEC SQL EXECUTE S_0000 USING DESCRIPTOR :*db2da_p;
    /* fetch case */
    EXEC SQL FETCH C_0000 USING DESCRIPTOR :*db2da_p;

    if ( SQLERROR and NOT SQLCODE == DUPLICATE_RECORD)
        exit
}

```

Attention: A duplicate record at insert is not treated as an error.

In Example 5-3 we show the new code for DB2 V8.

Example 5-3 Eliminates the need for a loop

```
/* process multiple rows; nr is set to row_count*/
/* insert case */
EXEC SQL EXECUTE AS_0000 USING DESCRIPTOR :*db2da_p FOR :nr ROWS;
/* update case */
EXEC SQL FETCH NEXT ROWSET AC_0000 FOR :nr ROWS USING DESCRIPTOR :*db2da_p;

```

Attention: All multi-row INSERT statements are prepared with the NOT ATOMIC attribute set. This is necessary to prevent the multi-row execute statement from failing after a duplicate record popped up. Because of this, SAP has to analyze all SQLCODES for serious SQL errors after the call, and report the most serious one to the SAP error handler.

5.3.3 ABAP SQL statements that benefit from multi-row operations

In this section we show examples of SQL statements that benefit from multi-row operations.

SELECT statements

In the following examples we show ABAP SELECT statements that cause multi-row FETCHes in DB2.

The row count for the select statement in Example 5-4 is: <buffer size> / record length. The buffer size is usually 32 KB. If the record length is 100 bytes, you will see multi-row fetches with a rowset size of about 32 records (row count=32).

Example 5-4 SELECT loop until 'not found' condition is reached

```
SELECT * FROM DB2JOB.  
    *process row  
    ...  
endselect.
```

The row count for the statement in Example 5-5 is the minimum of <buffer size> / record length and 100.

Example 5-5 SELECT loop until 100 rows fetched or 'not found' condition is reached

```
SELECT * FROM DB2JOB UP TO 100 ROWS.  
    * process row  
    ...  
endselect.
```

If the row count is less than 100, the multi-fetch statement is repeated as often until 100 rows are fetched. In this case it is very likely that the last multi-row fetch has a row count less than <buffer size> / record. For example if <buffer size> / record = 30, you can see three multi-row fetch calls with a record count of 30 and one with a record count 10. You might see fewer calls if the result set of the SELECT statement has less than 100 rows.

With the ABAP SELECT statement shown in Example 5-6, you can lower the calculated row count. If the calculated row count is greater than PACKAGE SIZE, then the PACKAGE SIZE is used as the row count; otherwise, the row count does not change.

Example 5-6 SELECT into internal ABAP table

```
data: t_db2job like db2job occurs 0 with header line.  
...  
select * into table t_db2job package size 10 from db2job order by jobname.  
...  
endselect.
```

Example 5-7 shows the ABAP SELECT with a fully qualified primary key.

Example 5-7 SELECT a single row

```
select single * from db2job
where jobname = 'UPDATE_CATALOG'
and creator = 'SAPR3'
and type = 'B'
and cardno = 1.
```

This statement always returns 0 or 1 row. It causes a multi-fetch operation with a row count of 1.

MODIFY statement

The ABAP MODIFY statement, shown in Example 5-8, is used very often within SAP applications. It is very convenient and used for modifying many records with one statement.

Example 5-8 ABAP MODIFY statement

Assumption:

F1, ..., Fn are non key fields and K1, ..., Km are key fields of table T

ABAP snippet:

```
data: IT like T occurs 0 with header line.
* fill up the internal table IT
...
modify T from table IT.
```

The semantic of this statement is pretty simple. After a successful call, the content of the internal ABAP table must be found in the database table. This is implemented with insert and update operations. If the inserts returns with an SQLCODE -803 (duplicate key) this row gets updated. The insert and update statements have always the structure shown in Example 5-9.

Example 5-9 DB2 statements to emulate the ABAP MODIFY statement

INSERT statement for MODIFY:

```
INSERT INTO T ( "F1",..., "Fn", "K1", ..., "Km" )
VALUES ( ?, ....., ? )
```

UPDATE statement for MODIFY:

```
UPDATE T SET F1 = ?, ..., Fn = ?
WHERE K1 = ?, ..., Km = ?
```

The row count for the multi-insert statement is calculated as in the multi-fetch case (see "SELECT statements" on page 118).

Example of an ABAP MODIFY statement

Let us assume that a table T has only two columns F1 of type varchar(1) and F2 of type varchar(2) and there exists a unique index on F1.

The table T has the contents listed in Table 5-1.

Table 5-1 Table T contents

F1	F2
1	a
2	b
3	c
4	y
5	z

The contents of the ABAP table are listed in Table 5-2.

Table 5-2 ABAP table IT contents

F1	F2
4	d
5	e
6	f
7	g
8	h

The MODIFY statement causes the following multi-row operations:

1. Multi-row insert with the whole ABAP table IT (row count = 5)
2. Single-row updates for all records that got an SQLCODE=-819 (duplicate key). These are rows 4 and 5.

After the modify, the table T contents are as shown in Table 5-3.

Table 5-3 Results after modify

F1	F2
1	a
2	b
3	c
4	d
5	e
6	f
7	g
8	h

INSERT statement

The ABAP statement shown in Example 5-10 causes a multi-row INSERT statement in DB2 V8.

Example 5-10 ABAP INSERT statement

```
data: IT like T occurs 0 with header line.  
...  
insert T from table IT accepting duplicate keys
```

The ABAP statement accepts duplicate keys and therefore we need the NOT ATOMIC attribute when the multi-row insert statement gets prepared. The row count for the multi-row insert statement is calculated as in the multi-row fetch case (see “SELECT statements” on page 118).

5.4 Query optimization

In this section we describe several enhancements related to query optimization.

- ▶ **RUNSTATS improvements:** These improvements fall into two categories:
 - **SAP and distribution statistics:** These statistics can be critical to the DB2 optimizer by choosing the most efficient method of servicing an SQL request in case of non-uniform distributions. The DSTATS tool has been widely available for some time now, and works with DB2 Version 5 and above. DB2 V8 effectively provides equivalent functions in the normal RUNSTATS utility.
 - **REPORT NO UPDATE NONE option:** The parameter combination of REPORT NO and UPDATE NONE has been added to the RUNSTATS utility, and the usage of these two options will invoke the invalidation of DSC statements involving the object(s) in question.
- ▶ **Host variables impact on SAP access paths:** We describe a new feature of DB2 V8 that eliminates the disadvantages of REOPT(VARS) and still takes the host variables into account at access path selection.
- ▶ **Index-only access path for varying-length columns:** We describe the new option to create indexes that do not pad varying-length columns of data type VARCHAR and VARGRAPHIC. This allows the DB2 optimizer to consider index-only access for indexes that contain VARCHAR or VARGRAPHIC columns, which accelerates query execution time due to saved I/O operations against the corresponding table space. As SAP creates all character columns of tables using data type VARCHAR or VARGRAPHIC, this new feature potentially speeds up query execution in all areas.
- ▶ **Multiple value IN lists and SAP:** We describe a DB2 V8 enhancement to the handling of INLIST predicates involving multiple values in the INLIST. The SAP application makes extensive use of INLIST predicates in dynamic SQL.

5.4.1 SAP and distribution statistics

In this section we first introduce the concept of SAP client, and then we look at the statistics distribution issues.

Effect of SAP Client concept on cardinality

One factor that is unique to the SAP application, is the concept of a “client”. In short, this allows a number of “virtual data” environment to exist within one SAP system, running in one instance of a DB2 DBMS. Each of these data environments are isolated from each other through the SAP programming model.

The concept of a client is effected through having a specific column on a large percentage of the tables in the system, which contains the 3-digit numeric client. This column is almost always called either MANDT or CLIENT. Generally, in a non-production system, multiple clients will exist and hence this column will have variable cardinality. However, in a production system, the situation is more complex, as the recommendation is to only run one client.

This results in the cardinality of the leading column always being low, and most commonly this column is also used as the first column in the indexes associated with the table. Most commonly, the cardinality of the column will be 1-4 in a production system, with one value dominating the frequency distribution, this value being the 3-digit client number chosen by the customer.

The result of this application constraint is that the DB2 optimizer is often left with less information than is desirable to make an informed access path choice. However, the good news is that the design of the SAP database layout makes it rare that this has a major impact. The design of the database, in terms of table structure, primary and secondary index structure, is closely tied with the SQL coded in the accompanying programs, hence a high degree of affinity.

However, in some cases, it is possible to determine that the DB2 optimizer is in fact choosing a less than ideal access path. As SAP exclusively used Dynamic SQL, this means that once the “wrong” decision is made, it remains the path for all identical statements until the statement is re-prepared. If the impact of this wrong decision is having an impact on performance of the overall system, it may be possible that better statistical data could allow DB2 to make a more informed decision. This is particularly the case for in-lists and screening predicates, and these types of SQL constructs are commonly found in an SAP system.

Another frequently occurring issue for the optimizer is when multiple columns in a table exhibit correlation, that is, the value in one column exactly or closely follows or determines the likely values in another column. If this correlation can be determined by analyzing the data content of a table, the optimizer can use this information in making intelligent choices of how to access data to service certain SQL requests. A good example of this phenomenon is where a ZIP code effectively determines the State, even though the State also exists as another discrete column.

Details of RUNSTATS improvements

When there is an asymmetrical distribution of data, not having distribution statistics on non-leading indexed columns and/or non-indexed columns can cause DB2 to make sub-optimal table join order and table join method decisions. In the case of an SAP system, as discussed in the previous section, the presence of a unique column in the form of SAP client (MANDT or CLIENT) as the first column makes this quite possible. This ultimately results in queries which perform inefficiently or do not complete.

A function to avoid this problem is provided by DSTATS (Distribution Statistics for DB2 for OS/390), a tool made available for use with Versions 5, 6, and 7. DSTATS is a standalone DB2 application program containing embedded dynamic and static SQL statements which builds the SYSCOLDIST catalog table entries previously not being collected by RUNSTATS. DSTATS is still available for download from:

<ftp://www.redbooks.ibm.com/redbooks/dstats/>

With DB2 V8, equivalent and standard function is provided by RUNSTATS. It greatly improves the accuracy of the filter factors determined by DB2. More accurate filter factor computations should lead to better optimization choices. Thus the query performance improves with better filter factor information in the DB2 catalog. This enhancement is implemented only in RUNSTATS, and not in inline statistics.

The RUNSTATS enhancement provides the following functionalities:

- ▶ Frequency value distributions for non-indexed columns or groups of columns
- ▶ Cardinality values for groups of non-indexed columns
- ▶ LEAST frequently occurring values, along with MOST for both index and non-indexed column distributions

Figure 5-5 shows the syntax of the additional parameters for the RUNSTATS utility.

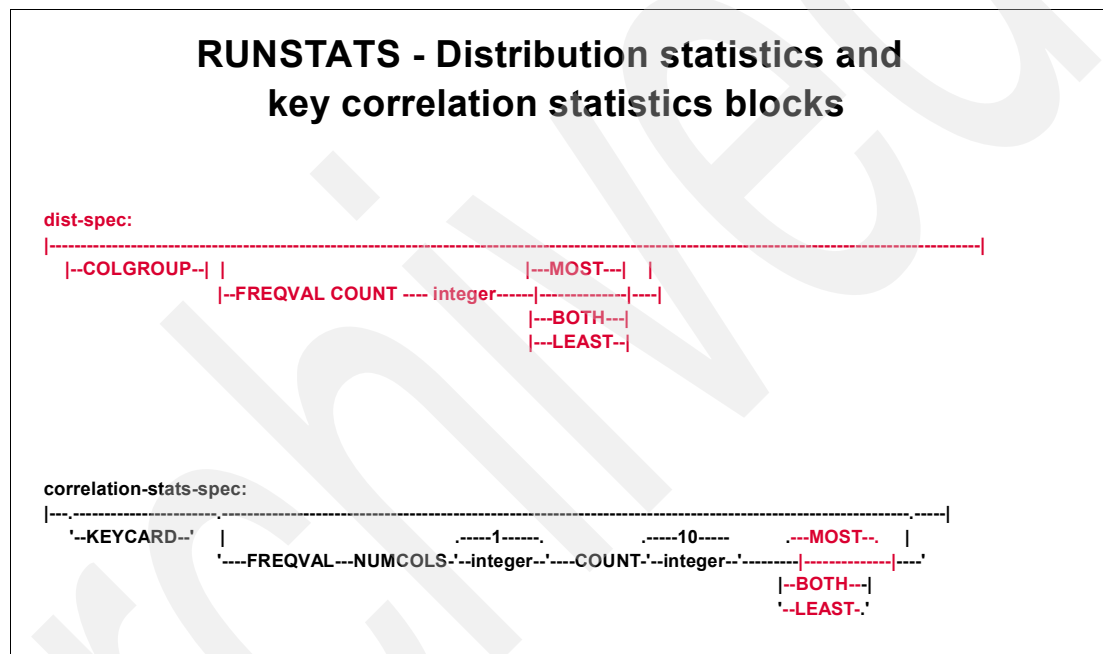


Figure 5-5 RUNSTATS Distribution Statistics

Collecting cardinality and distribution statistics

To enable the collection of cardinality and distribution statistics on non-indexed columns, a new dist-spec block is introduced. New keywords COLGROUP, MOST, LEAST, and BOTH are introduced in this block. In addition the existing keywords FREQVAL and COUNT are also used. Cardinality and distribution statistics are collected only on the columns explicitly specified. Cardinality and distribution statistics are not collected if you specify COLUMN ALL.

For an SAP system, usually prior to proceeding down this path, you will have one or more specific problems you are addressing, and probably have some application knowledge that determines which columns about which you are attempting to influence the optimizer. Because of the huge number of objects in an SAP system, it is impractical to collect distribution statistics on all columns for table spaces and indexes, as the total number of columns is prohibitively large.

Tip: When searching for more clues in addressing specific performance problems around specific SAP tables, it may be possible to run RUNSTATS, select a number of columns in the COLGROUP specification, a reasonable value such as 20 for COUNT, and additionally specify UPDATE NO. The statistics produced in report form may lead to a better understanding of the problem, and DB2's rationale for its choices.

COLGROUP

When the keyword COLGROUP is specified, the set of columns specified with the COLUMN keyword is treated as a group. The cardinality values are collected on the column group. When COLGROUP is not specified, the columns are treated as individual columns and cardinality values are collected on the columns specified in the list.

FREQVAL

This controls the collection of frequent value statistics. These are collected either on the column group or on individual columns depending on whether COLGROUP is specified or not. If FREQVAL is specified, then it must be followed by the keyword COUNT.

COUNT integer

COUNT indicates the number of frequent values to be collected. Specifying an integer value of 20 means to collect 20 frequent values for the specified columns. No default value is assumed for COUNT. Although the syntax might suggest that the default value is 10, you have to explicitly specify COUNT 10. This can be optionally followed by the keyword MOST (which is the default), LEAST, or BOTH.

MOST

The most frequent values are collected when the keyword MOST is specified.

LEAST

The least frequent values are collected when the keyword LEAST is specified.

BOTH

The most frequent values and the least frequent values are collected when the keyword BOTH is specified.

Example

In Example 5-11, by specifying the COLGROUP keyword, the cardinality of the column group is collected for the specified group of columns (RELID, SRTFD, SRTF2) for the SAP table PCL2.

Example 5-11 Updating column distribution statistics

```
RUNSTATS TABLESPACE A130X996.PCL2
        TABLE(SAPR3.PCL2)
        COLGROUP(RELID, SRTFD, SRTF2)
```

5.4.2 REPORT NO UPDATE NONE option

In certain situations, such as catalog statistics having been manually updated, or a new index created on a table, it is desired to allow SQL statements to use another access path. In general, most situations are the end product of investigations of performance problems, and quite often it will be desired to solve the problem as non-disruptively as possible. As SAP exclusively uses dynamic SQL for application related calls, this means the cached versions of the prepared statements will continue to take effect. Usage of the RUNSTATS utility in this

way elegantly allows all DSC statements referencing the object in question to be invalidated. This can be an improvement to other techniques such as ALTER to AUDIT NONE.

Example 5-12 illustrates an example of invoking the RUNSTATS utility specifically to invalidate DSC statements that use table MBEWH.

Example 5-12 Invalidating DSC STATS using RUNSTATS

```
RUNSTATS TABLESPACE A040X998.MBEWH
REPORT NO
UPDATE NONE
```

SAP exploits this new way to invalidate statements from dynamic statement cache in transaction code ST04. In the cached statements statistics part of ST04, there is a button available that invalidates a selected SQL statement (see Figure 5-12). When pressing the button, SAP calls the stored procedure DSNUTILS to invoke the RUNSTATS utility with options REPORT NO UPDATE NONE on the table space of the first table that is referenced in the selected statement.

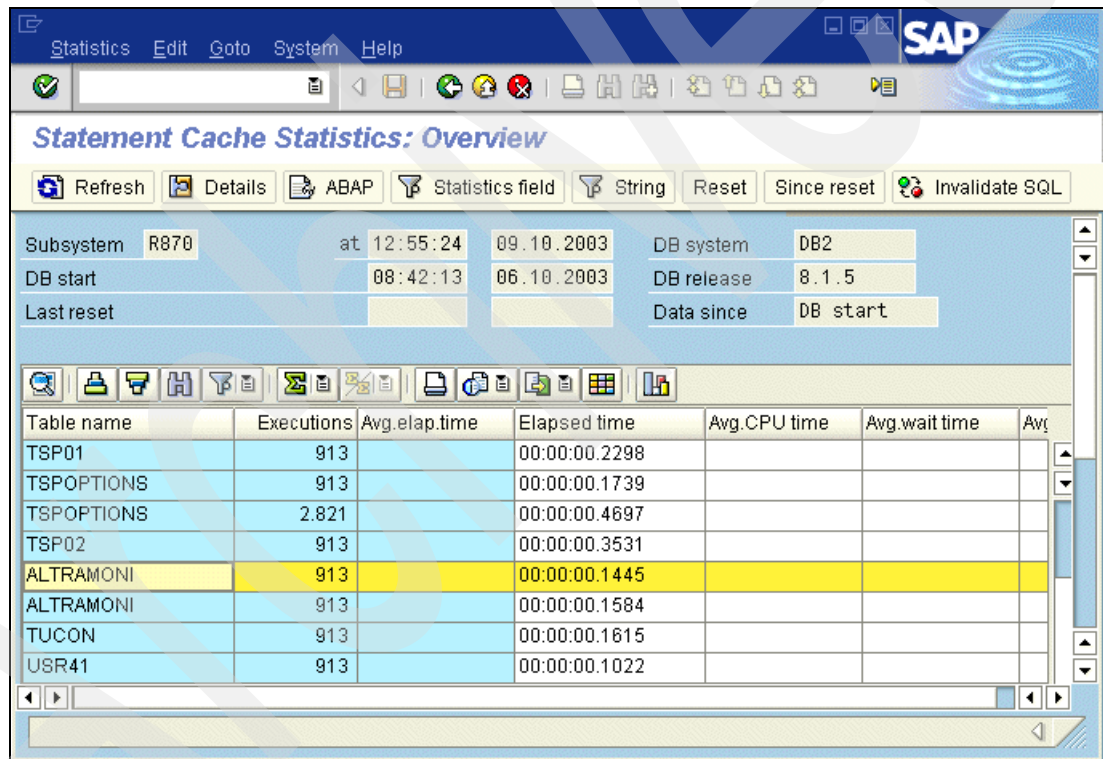


Figure 5-6 SAP transaction ST04: Invalidate SQL from DSC

5.4.3 Host variables impact on SAP access paths

Usage of host variables (parameter markers) in SAP is very pervasive. The reason for this is to maximize the benefits of the DB2 dynamic statement caching mechanism, by minimizing the number of “different” statements. However, this poses additional challenges to the optimizer, because it cannot use predicate values frequency distribution. In the absence of this data, DB2 prior to V8 uses defaults and it has proved to be wrong in cases where the values distribution is skewed.

REOPT(ONCE) is a new bind option that tries to combine the benefits of REOPT(VARS) and dynamic statement caching. For an SQL statement with input host variables, the access path chosen by the optimizer during prepare time (before the values of host variables are available) may not always be optimal. The bind option REOPT(VARS) solves this problem by (re)-preparing the statement at run time when the input variable values are available, so that the optimizer can re-optimize the access path using the host variable values. However, for frequently called SQL statements that take very little time to execute, re-optimization using different input host variable values at each execution time is expensive, and it may affect the overall performance of applications. Other unwanted ramifications of REOPT(VARS) are increased locking contention (DBD locks) and no monitoring support. Both of these disadvantages are caused by KEEP DYNAMIC NO, which must be in effect when using REOPT(VARS).

The idea of REOPT(ONCE) is to defer access path determination until the first execution. Statement preparation takes place only once using the first set of host variable values, no matter how many times the same statement is executed. The access path chosen based on the set of input variable values is stored in cache and used for all later executions. This solution is based on the assumption that the chosen set of host variable values at run time are better than the default ones chosen by optimizer at bind time.

SAP example of a problem situation and the solution

In our example problem, a calculation took about 40 hours which was considered totally unacceptable relative to the amount of data in the tables. The statement that caused the problem was a 3-table join. The tables are: AUFK, AFKO, and AFPO. The recommendation to solve this problem — using the ABAP hint ‘SUBSTITUTE VALUES’ — improved the response time to 2.5 hours. We now can get the same results if we are using the new bind option REOPT(ONCE), as shown in Example 5-13.

Example 5-13 Statement that caused the trouble

```

SELECT
  T_01 . "AUFNR"
FROM
  ( ( "AUFK" T_00 INNER JOIN "AFKO" T_01 ON T_01 . "MANDT" = :m AND
    T_00 . "AUFNR" = T_01 . "AUFNR" ) LEFT OUTER JOIN "AFPO" T_02 ON
    T_02 . "MANDT" = :m AND T_01 . "AUFNR" = T_02 . "AUFNR" )
WHERE
  T_00 . "MANDT" = :m AND T_01 . "MAUFNR" IN ( :h1 , :h2 , :h3 , :h4 , :h5 , :h6 ) FOR
  FETCH ONLY WITH UR

```

Example 5-14 shows the related trace.

Example 5-14 ST05 trace output of the calculation

```

Prepare with parameter markers:

18.151 AUFK    PREPARE  0    0 SELECT WHERE T_00 . "MANDT" = ? AND T_01 . "MAUFNR" IN ( ? , ? ,
18 AUFK      REOPEN   0    0 SELECT WHERE T_00 . "MANDT" = '000' AND T_01 . "MAUFNR" IN (
24.897.32 AUFK    FETCH    0    0
21 AUFK      REOPEN   0    0 SELECT WHERE T_00 . "MANDT" = '000' AND T_01 . "MAUFNR" IN (
24.900.13 AUFK    FETCH    0    0

```

The SAP trace in Example 5-14 shows the problem. The FETCH took about 25 seconds and no row has been returned because of a “not found” condition. This statement gets issued very often during the calculation. The prepare was done with parameter markers. The field MANDT is usually not selective at all.

In Example 5-15, the Explain output shows that, because DB2 is using defaults, the optimizer has chosen the wrong access path.

Example 5-15 ST05 Explain output of the highlighted statement — Wrong access path

Access path of statement
Explanation of query block number: 1 step: 1 Query block type is SELECT Performance appears to be good Index is used. Index scan by matching index. Method: . access new table. . data pages are read in advance . pure sequential prefetch is used . new table: . SAPR3.AUFK . table space locked in mode: N . Accesstype: by index. . Index: SAPR3.AUFK~0 (matching index) . Index columns (ordered): . MANDT . AUFNR . with 1 matching columns of 2 index columns.
Explanation of query block number: 1 step: 2 Query block type is SELECT Performance appears to be good Index is used. Index scan by matching index. Method: . scan composite table and new table in order of join . column, and join matching rows (merge scan join). . 1 columns are joined during merge scan join . new table: . SAPR3.AFKO . table space locked in mode: N . Accesstype: by an index, when the predicate contains the IN keyword. . Index: SAPR3.AFKO~5 (matching index) . Index columns (ordered): . MANDT . MAUFNR . PRODNET . with 2 matching columns of 3 index columns.
Explanation of query block number: 1 step: 3 Query block type is SELECT Performance appears to be good Index is used. Index scan by matching index. Method: . scan composite table and new table in order of join . column, and join matching rows (merge scan join). . 1 columns are joined during merge scan join . new table: . SAPR3.AFPO . table space locked in mode: N . Accesstype: by index. . Index: SAPR3.AFPO~0 (matching index) . Index columns (ordered): . MANDT . AUFNR . POSNR . with 1 matching columns of 3 index columns.

We now check what the access path should be.

The solution is an ABAP hint, which tell the SAP database interface to defer the prepare and to replace the parameter markers by real values. Now the DB2 optimizer can use the statistics to select the right access path, as shown in Example 5-16.

Example 5-16 ST05 trace output of the calculation, after implementing the ABAP hint

With values:

880 AUFK	PREPARE	0	0	SELECT WHERE T_00 . "MANDT" = '000' AND T_01 . "MAUFNR" IN ('
18 AUFK	REOPEN	0	0	SELECT WHERE T_00 . "MANDT" = '000' AND T_01 . "MAUFNR" IN ('
13.910 AUFK	FETCH	0	0	
4.600 AUFK	CLOSE	0	0	
628 AUFK	PREPARE	0	0	SELECT WHERE T_00 . "MANDT" = '000' AND T_01 . "MAUFNR" IN ('
18 AUFK	REOPEN	0	0	SELECT WHERE T_00 . "MANDT" = '000' AND T_01 . "MAUFNR" IN ('
12.257 AUFK	FETCH	0	0	

The highlighted statements are the same as in Example 5-14. The response time is now about 2000 times better than before. Example 5-17 shows the good access path.

Example 5-17 ST05 Explain output of the highlighted statement — Right access path

```

|Access path of statement
|-----
|Explanation of query block number: 1  step: 1
|Query block type is SELECT
|Performance appears to be good
|Index is used. Index scan by matching index.
|Method:
|.   access new table.
|.   unknown or no sequential prefetch is used
|.   new table:
|.           SAPR3.AFKO
|.           table space locked in mode: N
|.   Accesstype: by an index, when the predicate contains the IN keyword.
|.           Index: SAPR3.AFK0~5 (matching index)
|.           Index columns (ordered):
|.                   MANDT
|.                   MAUFNR
|.                   PRODNET
|.           with 2  matching columns of 3  index columns.
|
|Explanation of query block number: 1  step: 2
|Query block type is SELECT
|Performance appears to be excellent
|Index-only access. Index scan by matching index.
|Method:
|.   join each row of composite table, with matching rows
|.   of new table (nested loop Join).
|.   unknown or no sequential prefetch is used
|.   new table:
|.           SAPR3.AUFK
|.           table space locked in mode:  N
|.   Accesstype: by index.
|.           Index: SAPR3.AUFK~0 (matching index)
|.           Index columns (ordered):
|.                   MANDT
|.                   AUFNR
|.           with 2  matching columns of 2  index columns.
|.           Access to index alone is sufficient to satisfy
|.           the request. No further data must be accessed.

```

```

Explanation of query block number: 1  step: 3
Query block type is SELECT
Performance appears to be good
Index-only access. Index scan by matching Index.
Method:
.      join each row of composite table, with matching rows
.      of new table (nested loop Join).
.      unknown or no sequential prefetch is used
.      new table:
.          SAPR3.AFPO
.          table space locked in mode:  N
.      Accesstype: by index.
.          Index: SAPR3.AFPO~0 (matching index)
.          Index columns (ordered):
.              MANDT
.              AUFNR
.              POSNR
.          with 2  matching columns of 3  index columns.
.          Access to index alone is sufficient to satisfy
.          the request. No further data must be accessed.

```

The disadvantage of this solution is that the statement has to be prepared every time the values are changing. With REOPT(ONCE) you get the same excellent access path. But REOPT(ONCE) is a much more elegant solution because there is no need to change the ABAP program; also, there is no additional cost for the prepare, and the dynamic statement cache is not affected badly because all other statements are replaced.

5.4.4 Index-only access path for varying-length columns

When possible, an index-only access path regularly provides optimal performance. However, up to DB2 V7, if the query includes variable character columns, the index-only access path is often disabled. The reason is the way the variable length columns are stored in the index: they are extended to their maximum length by padding. That prevents returning the column value directly from the index because the padded value (instead of the actual one) would be passed back to the calling application. Another example of incorrect output that would result from an index-only access path is the LENGTH function. The actual length of a varchar column is not necessarily its maximum length, and the maximum value is what the index-only access path would return. Therefore, there are enough reasons to discourage use of the system parameter RETVLCFK that allows index-only access even for variable length columns.

In V7, DB2 has been enhanced to select index-only access if there is no danger to return wrong data. This will cover the case when a varchar field is involved in the WHERE clause (excluding LIKE predicates or any predicate that has the varchar column as input to a scalar function or in any Stage 2 predicate), and the column does not appear in the SELECT list, any scalar function, ORDER BY or GROUPBY clause, or in the HAVING clause.

Still, this leaves out many cases where an index-only access would significantly improve performance, but cannot be done. SAP extensively uses varchar columns: every character column, even with length 1, is defined as varchar. This is why the DB2 V8 index design enhancement that introduces non-padded indexes (where the column length is included in the index) is very beneficial to SAP.

Varying-length index keys

Prior to V8, the ZPARM RETVLCFK=YES enabled you to use VARCHAR columns of an index and still have index-only access. However, when columns in the SELECT list of the query are retrieved from the index when using index-only access, they are padded to the maximum length and the actual length of the variable length column is not provided. Rather, the length of the padded value is returned. Moreover, the LIKE predicate potentially yields the wrong result set. Example 5-18 demonstrates the problem. The query in the example returns the record that contains 'A' as value for SAMPLEC. The underscore character represents exactly one character though. Hence, this record must not qualify.

Example 5-18 DB2 V7: Varying-length index keys and LIKE predicate

Table SAMPLET contains column SAMPLEC with type VARCHAR(2)
Value: 'A'

Index SAMPLEI column pads VARCHAR column to maximum length
Value 'A '

Query with wrong result set:

```
SELECT *  
FROM   SAMPLET  
WHERE  SAMPLEC LIKE 'A_'
```

Therefore, these “full length” variable length columns are not applicable to SAP. SAP does not pad varying-length data and always stores VARCHAR or VARGRAPHIC columns using the actual string length.

DB2 V8 supports true varying-length key columns in an index. Varying-length columns will no longer need to be padded to their maximum lengths. This will usually reduce the storage requirements for this type of index since only actual data is stored. Furthermore, this allows for index-only access to index key columns of varying-length in all cases, since the length of the variable length column is now stored in the index, and can potentially improve performance. Varying length indexes are marked in the new SYSIBM.SYSINDEXES column, PADDED, with a value of 'N'.

Indexes can now be created or altered to contain true varying-length columns in the keys. The padding of both VARCHAR and VARGRAPHIC data to their maximum length can now be controlled.

The new keywords NOT PADDED or PADDED on a CREATE and ALTER INDEX statement specify how varying-length columns will be stored in the index.

- ▶ **NOT PADDED** specifies that varying-length columns will not be padded to their maximum length in the index. If there exists at least one varying-length column within the key, length information will be stored with the key. For indexes comprised only of fixed length columns, there is no length information added to the key. The default on the CREATE INDEX statement, NOT PADDED for new V8 NFM installations, can be controlled through the new PADIX ZPARM. A sample create of a non-padded index is shown in Example 5-19.

Example 5-19 Create a NOT PADDED index

```
CREATE UNIQUE INDEX DSN8710.XDEPT1  
ON DSN8710.DEPT (DEPTNO ASC)  
NOT PADDED  
USING STOGROUP DSN8G710  
PRIQTY 512  
SECQTY 64  
ERASE NO  
BUFFERPOOL BP3
```



```
CLOSE YES
PIECESIZE 1 M;
```

- **PADDED** specifies that varying-length columns within the index are always padded with the default pad character to their maximum length. All indexes prior to DB2 V8 NFM are padded by default. Example 5-20 shows the CREATE INDEX statement for a padded index.

Example 5-20 Create a PADDED index

```
CREATE UNIQUE INDEX DSN8710.XDEPT1
ON DSN8710.DEPT (DEPTNO ASC)
PADDED
USING STOGROUP DSN8G710
PRIQTY 512
SECQTY 64
ERASE NO
BUFFERPOOL BP3
CLOSE YES
PIECESIZE 1 M;
```

- An index can be created or altered to NOT PADDED, even though the index key may not contain any varying-length columns. The index will be marked as “not padded”, which will allow for varying-length columns to be added in the future without the index being set to pending state.

You can continue to use existing indexes that contain padded varying-length columns. However, with DB2 V8, you now have the ability to convert padded indexes to varying-length indexes and also to convert varying-length indexes back to padded indexes.

Indexes from a prior release will not automatically convert to NOT PADDED, even if an ALTER TABLE ALTER COLUMN SET DATATYPE statement is executed and the altered column is part of an the index. You have to use the ALTER INDEX statement to change a PADDED index to NOT PADDED.

Altering a PADDED index to a NOT PADDED index can be done as shown in Example 5-21.

Example 5-21 Alter an index to NOT PADDED

```
ALTER INDEX DSN8710.XDEPT1 NOT PADDED;
```

After an index has been altered from PADDED to NOT PADDED or vice versa, the index will be marked as being in pending state, if there exists at least one varying-length column in the index. The pending state is Rebuild Pending (RBDP), except for non-partitioned secondary indexes that are put in page set Rebuild Pending state (PSRBD). A REBUILD of the index makes the index available again.

To take advantage of the performance improvement, you should convert the indexes of SAP systems, which have been created prior to V8, to NOT PADDED. New indexes should be created as NOT PADDED. The only downside to this definition is due to the extra 2 bytes added for each varchar column in a key, some extra CPU time in processing, and extra padding and unpadding at sort time.

Example of index-only access

To illustrate index-only access with VARCHAR columns, let us consider a query on table VBDATA, which is central to SAP applications as it is employed for the SAP asynchronous update protocol. The table definition is displayed in Figure 5-7. The primary key index VBDATA~0 of table VBDATA contains the columns VBKEY, VKMODCNT and VBBLKNO. From SAP's point of view, column VBKEY is of type character with length 32. As SAP maps its character data type to the DB2 data type VARCHAR, index VBDATA~0 is an index on varying length columns.

The screenshot shows the SAP Dictionary interface for table VBDATA. The table is active and its short description is 'Update data'. The 'Fields' tab is selected, showing a list of fields with their attributes.

Field	Key	Initi...	Data element	Data T...	Length	Deci...	Short Description
<u>VBKEY</u>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	VBKEY_D	CHAR	32	0	Key for update and enqueue/dequeu
<u>VBMODCNT</u>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	VBMODCNT	INT4	10	0	Update module ID
<u>VBBLKNO</u>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	VBBLKNO	INT4	10	0	No. of update data block
<u>VBLEN</u>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	VBLEN	INT2	5	0	Length of update data
<u>VBDATA</u>	<input type="checkbox"/>	<input type="checkbox"/>	VBDATA_D	LRAW	3800	0	Update data

Figure 5-7 Index-only access: Definition of table VBDATA

The SQL statement that is shown in Figure 5-8 queries the data records that qualify according to a predicate on column VBKEY. All the columns that are requested are part of index VBDATA~0. The figure displays the access path for this statement as presented by the SAP built-in SQL Explain functionality. It demonstrates that the DB2 optimizer selects index-only access for the statement in face of the VARCHAR column VBKEY, which is part of the result set.

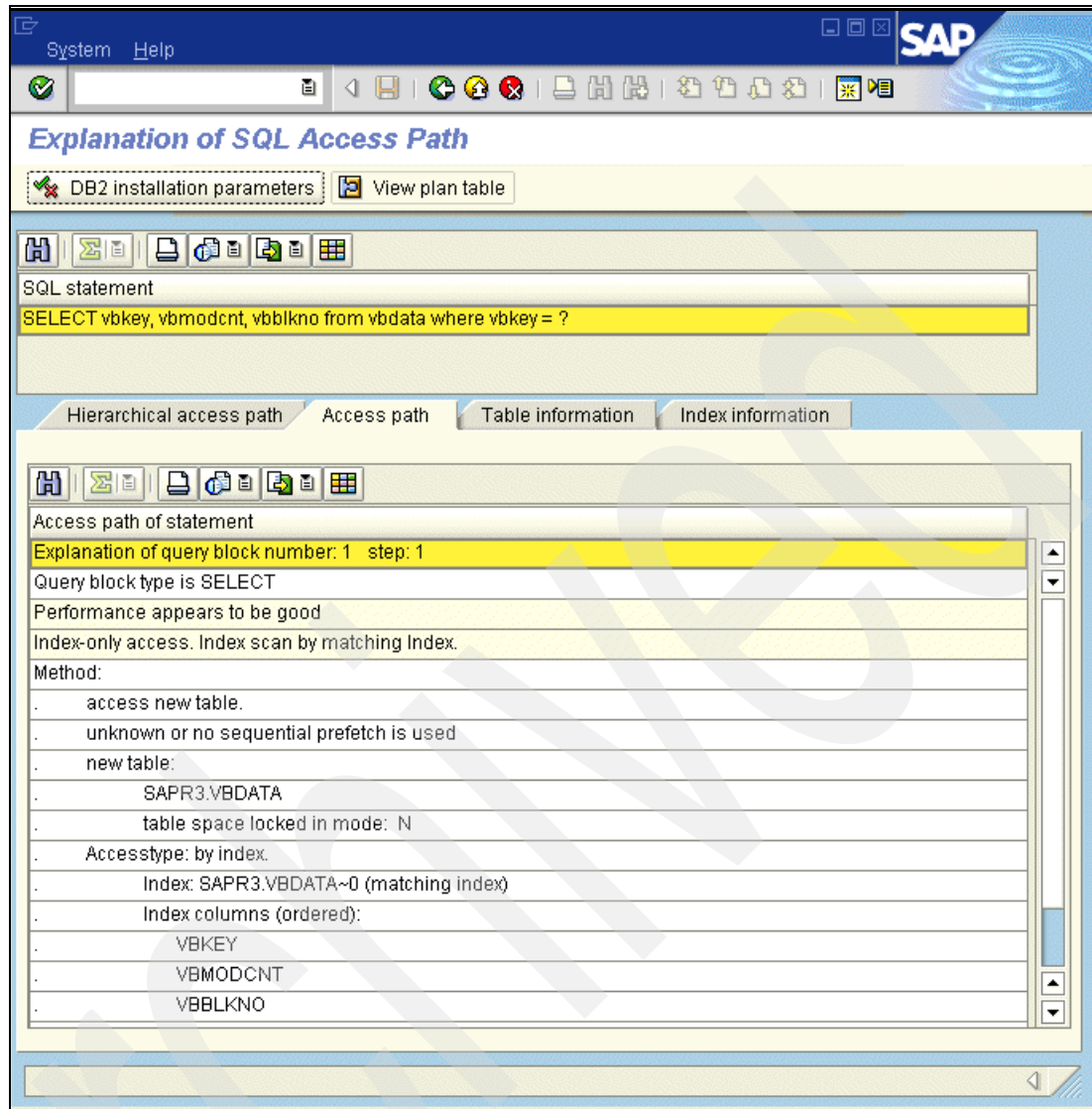


Figure 5-8 Index-only access path with VARCHAR column

5.4.5 Multiple value IN lists and SAP

The ABAP Open SQL construct FOR ALL ENTRIES results in an SQL statement with a potentially large number of keys contained in some internal table. When the internal table contains a single column, this result in SQL statements with large numbers of values in the INLISTs. Rather than sending these large INLISTs to the database for execution, the SAP dbsl divides large statements into multiple smaller SQL statements according to the values of the `rsdb/min_in_blocking_factor` and `rsdb/max_in_blocking_factor` parameters. These parameters normally are set to 6 and 35.

The DB2 memory commitments associated with the caching of SQL statements depend in part on the total number of unique SQL statements generated by the SAP application. The division of large INLISTs in this manner typically results in one or two SQL statements for any given OpenSQL statement containing a FOR ALL ENTRIES construct.

INLISTs derived from the OpenSQL IN predicate generally contain only a few values and are passed through the dbsl without modification.

Optimization challenge associated with IN predicates

Traditionally the preparation of an SQL statement containing INLIST predicates and parameter markers results in an access path that assumes a uniform distribution. The optimizer estimates the filtering quality of the predicated using a filter factor of derived from the number of entries in the INLIST divided by the column cardinality. Even when distribution statistics are available SQL statements prepared with parameter markers are not able to take distribution into account.

Data distribution effects

There are situations within SAP in which INLIST predicates operate against columns of tables having non-uniform distribution characteristics. Common examples of this include:

- ▶ Status values for which an SQL statement is checking for some set of exceptional, low probability, conditions
- ▶ Hierarchy relationships within a single table where most records have no relevant parent record and therefore contain blanks
- ▶ Hierarchy relationships within multiple tables where different object types in one table have different numbers of related records in other tables

It is possible, with careful DB2 statistics adjustment, to improve these access paths and performance in general. This is neither a general nor good solution because such changes are difficult to maintain, and there is always the possibility of degrading the performance of some other query.

Size of the IN list

Prior to V8, as the number of INLIST items increases, the DB2 optimizer often tended to select a table scan over a matching index scan. This often resulted in poor performance as this is often not the optimal access path.

It is possible, with careful DB2 statistics adjustment, to improve these access paths and performance in general. This is neither a general nor good solution because such changes are difficulty to maintain and there is always the possibility of degrading the performance of some other query.

DB2 V8 enhancements

With DB2 V8 there are several enhancements targeted at improving optimization and access path selection for queries involving IN predicates. Several of these have been retrofitted back to DB2 Version 7 and are available in PTF form.

Select INLIST improvements (V7 APAR PQ68662)

The problem here is poor access path selection on a single table query with IN predicate on index columns. The optimizer struggles between I-scan and R-scan and as the number of entries in the INLIST increases the query will tend to select the R-scan. This enhancement improves the index scan cost model to more accurately reflect the chance that the needed index or data page may already be cached in the buffer pool when the last INLIST item is processed.

RUNSTATS FREQVAL enhancements and REOPT(ONCE)

The optimization challenge associated with data distribution issues was addressed in part with the REOPT(VARS) capability introduced in DB2 Version 5. When an SQL statement is executed with REOPT(VARS) the SQL statement is re-optimized at each execution using the actual host variables. This re-optimization allows the optimizer to select an access path based on the distribution statistics stored in the DB2 catalog.

This is especially good for SQL statements of significant duration where the cost of a full prepare is offset by a significantly improved access path. It seems a rather expensive solution to the problem of poor access path selection on simple SQL statements.

► **ME21 Case Study:**

Transaction ME21 has been reported to be slow. Figure 5-9 shows a STAT record for an execution of this transaction where performance is poor.

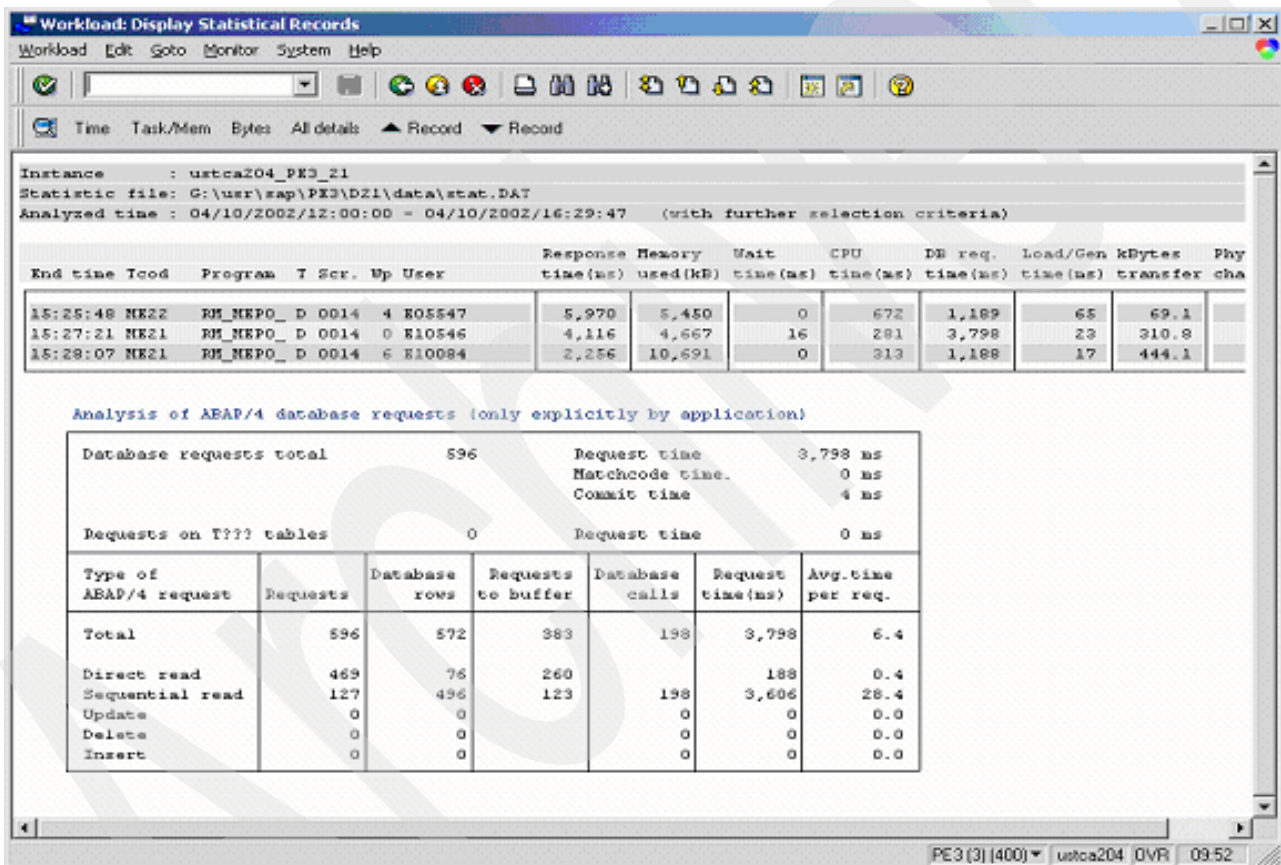


Figure 5-9 STAT Record for ME21 transaction

The STAT record shows that there is one dialog step that seems to have inefficient SQL — nearly 30 ms per select, and almost 10 ms per row (3606 ms for select / 496 rows = 7 ms per row). This is rather slow. Since “database rows” is much greater than “requests”, we use the per-row times to evaluate performance. Figure 5-10 shows an ST05 trace pointing to a select on table KSSK as being the problem.

Duration	ObjectName	Op.	Rec	RC	Statement
24	KSSK	REOPEN	0	0	SELECT WHERE "MANDT" = '400' AND "OBJEK" = '0000000360' AND "MA
1,948	KSSK	FETCH	0	0	
22	KSSK	REOPEN	0	0	SELECT WHERE "MANDT" = '400' AND "CLINT" IN (0000000360 , 00000
2,897,905	KSSK	FETCH	1	0	
25	AUSP	REOPEN	0	0	SELECT WHERE "MANDT" = '400' AND "OBJEK" IN ('01P0' , '01P0' ,
1,994	AUSP	FETCH	0	0	
18	KSSK	REOPEN	0	0	SELECT WHERE "MANDT" = '400' AND "CLINT" = 0000000360 AND "MAFI
2,344	KSSK	FETCH	0	0	
19	NRIV	REOPEN	0	0	SELECT WHERE "CLIENT" = '400' AND "OBJECT" = 'EINRBELEG' AND "S
2,636	NRIV	FETCH	1	0	

Figure 5-10 Poor performance from KSSK select with INLIST

When we examine the SQL statement in more detail, we can see that the SQL statement shown in Figure 5-11 originates from an ABAP program using the “for all entries” construct. The actual list when the program was executed only had a single value resulting. The kernel parameter `rsdb/min_in_blocking_factor` is set to 6, which results in the INLIST being padded to 6 values.

```

SQL Statement

SELECT
  "OBJEK" , "CLINT" , "STATU" , "AENNR" , "DATUV" , "ADZHL" , "LKENZ"
FROM
  "KSSK"
WHERE
  "MANDT" = ? AND "CLINT" IN ( ? , ? , ? , ? , ? , ? ) AND "MAFID" = ? FOR
  FETCH ONLY WITH UR

Variables

A0(CH,3) = 400
A1(NU,10) = 0000000360
A2(NU,10) = 0000000360
A3(NU,10) = 0000000360
A4(NU,10) = 0000000360
A5(NU,10) = 0000000360
A6(NU,10) = 0000000360
A7(CH,1) = 0
  
```

Figure 5-11 SQL Statement and Host Variables

It does not look good — Figure 5-12 shows the execution plan, no index is used, and the entire table is scanned.

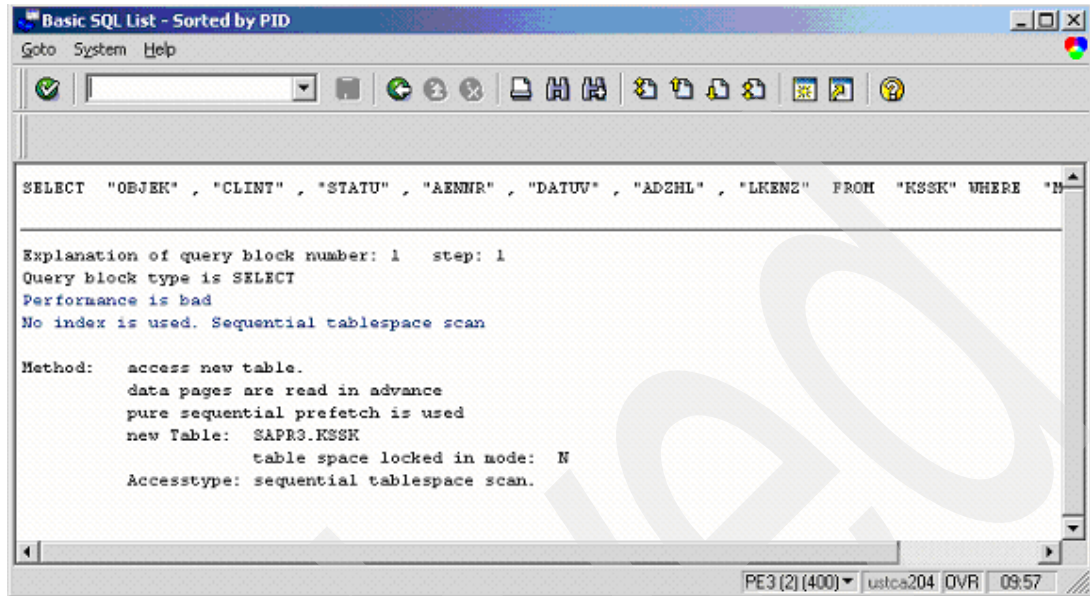


Figure 5-12 Explain showing table scan

After using SE16 to verify that there was only one row satisfying the predicate in the table, we use the DB2 catalog browser or SE11 to check for indexes that are suitable for this query. The KSSK~N1 index consists of MANDT, CLINT and MAFID; this is an exact match and yet it is not used in favor of a table scan.

Looking at the index cardinalities in Figure 5-13, we can see that the FULLKEYCARDF value for this index is 15. The DB2 optimizer has no information indicating that this index will provide much value as a filter. However, we know the result of our select returned one row, so we suspect the CLINT column has a skewed distribution.

CREATOR	TENAME	NAME	COLCOUNT	NLEAF	NLEVELS	FIRSKEYCARDF	FULLKEYCARDF
SAPR3	KSSK	KSSK~N1	3	352	3	2	15
SAPR3	KSSK	KSSK~0	6	5,438	3	2	152,577
SAPR3	KSSK	KSSK~3	2	351	3	2	2

Figure 5-13 Index cardinalities of KSSK Indexes

From the COLCARDF value on the CLINT column of 9 values, we can see that there are very few unique values. For sites running SAP 4.5B and later, one can modify the ABAP source to use the hint (USE VALUES FOR OPTIMIZATION - REOPT(VARS)) which will cause the statement to re-optimized at execution, when the variables are passed to DB2. With this hint, the DB2 column distribution statistics can be used by the optimizer; thus DB2 will know that the program is searching for a value that seldom occurs, and will choose the KSSK~N1 index. RUNSTATS with FREQVAL must be run on KSSK for USE VALUES FOR OPTIMIZATION to be effective. This will solve the problem, but at some significant cost, as the re-optimization of the SQL statement for each execution will consume additional resources and in this case would require a repair to an SAP program.

Further analysis shows that, of the 9 values for CLINT, 8 values represent singleton rows, while the ninth value is a string of blanks and is in all remaining rows. The access pattern is such that the SAP transaction always issues the SQL statement with non-blank values in the INLIST and expects to retrieve 0 or 1 rows. REOPT(VARS) is overkill for a select returning a single row.

► **ME21 Case Study DB2 V8 solution:**

The REOPT(ONCE) capability is directly applicable to this case. Given the nature of the SQL access, a single re-optimization with values, would be sufficient to cause the proper index to be used. This is a better solution than REOPT(VARS), because it will reduce the cost of each successive execution of the statement. We can expect that at some future date, REOPT(ONCE) will become the default for SAP installations.

Predicate pushdown for IN list predicates (V7 APAR PQ73454)

Performance problems have been observed with complex vendor generated queries involving IN list predicates and materialized views or table expressions. This performance problem was due to the large workfile that resulted from materialized view or table expression. If we allow qualified IN list predicates to be pushed down into the materialized view or table expression, the materialized result set becomes much smaller. This fix is activated by setting specification of a new DSNZPARM INLISTP to a value of 3 or more.

The following example demonstrates the effect of this change.

```
SELECT *
      FROM (
            SELECT * FROM T1
            UNION
            SELECT * FROM T2
            ) X(C1,C2)
WHERE X.C1 IN ('AAA','CDC','QAZ');
```

The transformed query will be equivalent to the following coding:

```
SELECT *
      FROM (
            SELECT * FROM T1 WHERE T1.C1 IN ('AAA','CDC','QAZ')
            UNION
            SELECT * FROM T2 WHERE T2.C1 IN ('AAA','CDC','QAZ')
            ) X(C1,C2)
WHERE X.C1 IN ('AAA','CDC','QAZ');
```

This transformation could result in an order of magnitude performance improvement in some cases, due to:

- Filtering at the early level to reduce the intermediate workfile size
- Possible exploitation of indexes on T1 and/or T2

Overall query performance should improve after setting the INLISTP parameter.

Correlated subquery transformation enhancement (V7 APAR PQ73749)

Performance problems have been observed with complex vendor generated queries involving IN list predicates in correlated subqueries. These IN list predicates are constructed in such a way that the elements are drawn in from a previously constructed table, that is, the size of the list for a constructed IN list predicate depends on the size of such a table. Allowing a correlated IN list predicate to be generated by the transitive closure process and by pulling the generated IN list predicate to the parent query block both will improve performance. The addition of the IN list predicate to the parent query block allows filtering to occur earlier and results in a reduction in the number of subquery executions. Additional performance improvements may be achieved if the IN list predicate on the parent query block allows a better access path as a consequence of indexability of the IN list predicate.

The following conditions must all be met in order for an IN list predicate to be generated from transitive closure and bubbled up from the subquery to its parent query block.

- ▶ Boolean term predicates, COL1 IN (Lit1, Lit2,...) and COL1 = COL2, both appear in the query.
- ▶ This subquery is on the RHS of a Boolean term predicate of type EXISTS, IN, or ANY.
- ▶ COL2 is correlated (to the parent query block or some ancestor of the parent query block).
- ▶ Parent query block in Inner Join, a single base table, or a view. This feature is disabled if the parent query block is an Outer Join.
- ▶ The subquery does not contain COUNT or COUNT_BIG.

An IN list predicate of more than one literal satisfying all of the above conditions will be generated in the parent query block and will not appear in the subquery. Other predicates that used to participate in transitive closure (including singleton IN list) will still be generated in the subquery as before; if these predicates can be bubbled up, then they will appear in both the parent query block and in the subquery.

Let us look at Example 5-22 in order to show the behavior prior to this enhancement.

Example 5-22 IN list query before transformation

```
SELECT *
  FROM A
 WHERE (EXISTS (SELECT 1
                FROM B,C
                WHERE B1 = 5
                   AND B1 = C1
                   AND C1 = A.A1
                   AND B2 IN (10, 20)
                   AND B2 = A.A2
                )
        ) ;
```

The SQL statement in Example 5-22 will generate the following three predicates within the EXISTS subquery:

```
C1 = 5
A.A1 = 5
A.A1 = B1
```

This enhancement, combined with an appropriate setting for the INLISTP DSNZPARM value, will cause a qualified IN list and equal predicates to be candidates for cross query block transitive closure process. This process, when conditions allow, will result in an IN list predicate being deduced by transitive closure and generated in the parent query block rather than in the subquery.

The predicates generated, at the parent query level, by this process are:

```
A.A1 = 5  
A.A2 IN (10,20)
```

This is equivalent to the transformation of the query shown in Example 5-23.

Example 5-23 Transformed IN list query

```
SELECT *  
  FROM A  
 WHERE  
   A.A1 = 5  
 AND A.A2 IN (10,20)  
 AND (EXISTS (SELECT 1  
              FROM B,C  
              WHERE B1 = 5  
                 AND B1 = C1  
                 AND C1 = A.A1  
                 AND B2 IN (10, 20)  
                 AND B2 = A.A2  
              )  
      ) ;
```

INLISTP installation parameter

It is evident that strategic placement of IN list predicates in an appropriate query block, whether pushing down or pulling out, is an extremely effective way to achieve the performance level for some queries. These optimizations are focused on popular vendor generated queries, so such performance improvement are expected to benefit a broad set of customers. The effectiveness of these optimizations have already been tested at the IBM Teraplex site, as well as at a customer site.

The reasons for proposing this system parameter INLISTP with respect to IN list predicate optimization are as follows:

- ▶ Allowing customer to tune the INLISTP value to their workload
- ▶ Prevention of SQLCODE -101
- ▶ Prevention of potential degradation in the access path for some queries

In DB2 V8, the default value for INLISTP will be stepped up to 50. The parameter INLISTP will remain as a hidden keyword installation parameter.

Notice that for the predicate pushdown optimization, the INLISTP value of 1 will have the same effect as the value of 0. This is because an IN list predicate of a single element is transformed into an equal predicate internally, and a qualified equal predicate is already being pushed down into a materialized view or a table expression. However, for the cross query block transitive closure optimization, the INLISTP value of would have a different effect than the value of 0. This is because no predicate has ever been considered for a cross query block transitive closure. For a positive INLISTP value, equal predicates, in addition to IN list predicates of more than 1 element, will also be considered as candidates to be pulled out to the parent query block position.

5.5 Support for backward index scan

In this section we discuss the capability of DB2 V8 to perform a backward index scan. SAP applications take advantage of this as they increasingly retrieve the most recently added record. The backward index scan accomplishes this in a well-performing way.

With the enhancements introduced to support dynamic scrollable cursors, DB2 also provides the capability for backward index scans. This allows DB2 to avoid a sort and allows you to define fewer indexes. With this enhancement it is no longer necessary to create an ascending and descending index on the same table columns.

For example, if you create an ascending index (the default) on the ACCT_NUM, STATUS_DATE and STATUS_TIME columns of the ACCT_STAT table, DB2 can use this index for backward index scanning for the following SQL statement:

```
SELECT STATUS_DATE, STATUS
FROM ACCT_STAT
WHERE ACCT_NUM = :HV
ORDER BY STATUS_DATE DESC, STATUS_TIME DESC
```

DB2 can use the same index for forward index scan for the following SQL statement:

```
SELECT STATUS_DATE, STATUS
FROM ACCT_STAT
WHERE ACCT_NUM = :HV
ORDER BY STATUS_DATE ASC, STATUS_TIME ASC
```

This is true also for static scrollable cursors and non-scrollable cursors. In V7 you have to create two indexes for the situation described above, to avoid a sort for both queries.

To be able to use the backward index scan, you have to create the index on the same columns as the ORDER BY and the ordering must be exactly opposite of what is requested in the ORDER BY.

Exploitation by SAP applications

Particularly in the SAP financial applications, there is an increasing number of situations where the most recent value needs to be returned for a given entity. This value was established in a different transaction scope.

If you consider a given stock out of a stock portfolio and a table that holds the stock values over a certain period of time, the most recent value is requested. The table is very large and there already exists an index on the relevant columns, but in ascending order. This fits other statements. Due to the size of the table, it is not desirable to create an additional index.

Example 5-24 shows a scenario of a table and an ascending index defined on it where the most recent value of some account is requested.

Example 5-24 Fast retrieval of most recent value

Table ACCOUNTS contains the following columns:

```
MANDT
ACNUM_INT
BAL_TYPE
BAL_YEAR
BAL_MONTH
TURNOVER_CLASS
BALANCE
```

Index ACCOUNTS~0 is defined in ascending order

```
MANDT
ACNUM_INT
BAL_TYPE
BAL_YEAR
BAL_MONTH
```

The following query retrieves the most recent value of a given account:

```
SELECT BALANCE
FROM ACCOUNTS
```

```
WHERE MANDT      = :HV1
  AND ACNUM_INT = :HV2
  AND BAL_TYPE  = :HV3
ORDER BY BAL_YEAR DESC, BAL_MONTH DESC
FETCH FIRST 1 ROW only
```

In V8 the DB2 optimizer is enabled to select index ACCOUNTS~0 for the access path of the query from Example 5-24. The index is traversed in the backward direction to directly retrieve the most recent value for the specified account.

5.6 Faster DSC short prepares

When global caching is active, DB2 maintains “skeleton” copies of each prepared statement in the EDM pool. This storage is not an issue for the short prepare focus. However, whenever a thread issues a prepare, this indicates that thread is going to need its own copy of the prepared statement, in order to execute the statement. When a thread finds a previously executed identical statement in the global cache, DB2 acquires storage and makes a copy of the statement for that thread. Prior to DB2 Version 8, this storage comes from a thread-based storage pool. So each thread that gets anything from the cache will have one of these pools.

In a system with many long-running threads, such as an SAP system, these pools can get quite large and fragmented if left uncontrolled. The current design (prior to DB2 V8) “contracts” the pool quite frequently, almost at every commit, so that the value specified by ZPARM MAXKEEPD is adhered to. The contraction logic freed most of the unused space back to the OS, keeping the size to a minimum. This meant that when thread performed a prepare after a commit, DB2 most probably had to go back to the OS to GETMAIN a new piece of storage for the pool, so it had somewhere to put the statement. This effectively meant DB2 had to perform a GETMAIN and FREEMAIN pair for almost every prepare.

Measurements showed that this was the dominant cost of the short prepare path (although the FREEMAIN part really happens at commit). APAR PQ37895, closed not long ago, was aimed to reduce the number of contractions, while still keeping storage under control. This was somewhat successful, but DB2 still does many GETMAINS and FREEMAINS.

The new approach aims to virtually eliminate the cost of the OS level GETMAINS and FREEMAINS by going to a centralized storage approach. With this approach, DB2 will use some number of storage pools that are owned by the system, not a particular thread. To avoid latch contention, the new implementation uses a fixed number of pools. When a new piece of storage is needed, a hash function is used, to “randomly” assign a pool. The hash uses the statement identifier as input. Each of the pools is a “bestfit” pool and it extends its size as needed, in 1 MB chunks. With this approach, we rely on the best fit logic to keep the pools at a minimum size. With the thread-based storage model, further reductions in contractions will lead to some storage increase.

Example

Assume we have three threads. Sequentially, that is, at discrete different times, each one prepares and executes a statement that is 10K in size, then it commits, freeing the storage back to the pool.

With the thread-based approach, at the end of the sequence, each thread will have 10 KB allocated for a total of 30 KB. With the centralized approach, the storage would get reused at each thread’s statement prepare, so there would be only ever be 10 KB allocated to the pool. If the three threads’ statement executions in fact do not occur at different times, there are still benefits. Assume, for example, that all three executed concurrently. The threads will have

used 30 KB in the centralized pool as well. But assume that after they all commit, one thread subsequently prepares and executes a statement that is between 10 KB and 30 KB in size. This could be satisfied from the existing centralized pool. But with the previously used thread-based pool, DB2 would need to GETMAIN a new 10 - 30 KB chunk to its unique pool.

The centralized approach may use as much storage as the thread-based approach in the worst case, but most likely the timing will be such that it will use much less. The major benefit is then the removal of the majority of OS calls to GETMAIN and FREEMAIN storage, and this benefit occurs regardless of the level of storage usage improvement.

Another benefit occurs in the case of short-running threads, which in fact occur frequently in an SAP system. When a thread that connects, prepares, and executes only a few statements, then ends, the centralized approach is clearly better. In this case the thread does not need to create its storage pool or GETMAIN any storage. The central pool is highly likely to already contain sufficient free storage with the normal ebb and flow of other activity to satisfy this small number of executed statements.

The more centralized management of the available storage means that we can use the available storage to the best advantage among all threads. Idle threads will also not be holding storage that they are not using, leading to more efficient usage of storage resources, and reduced induced overhead in a busy running SAP system.

5.7 Data sharing enhancements

In this section we describe data sharing enhancements of DB2 V8 that are of particular interest to your SAP system performance.

This section contains the following:

- ▶ SAP and data sharing
- ▶ CF lock propagation reduction
- ▶ CF request batching
- ▶ Improved LPL recovery

5.7.1 SAP and data sharing

SAP fully embraces the DB2 data sharing capabilities. It supports all variations of data sharing topologies. For scalability reasons, SAP instances may run DB2 in data sharing mode. To ensure continuous availability, SAP systems exploiting data sharing are also very common. The SAP application server is capable to fail over to a secondary DB2 member if it detects problems with the connection to the member to which it is currently connected. Moreover, the SAP critical asynchronous update protocol tables can be configured such that an affinity between SAP application server and DB2 member is established for the access to these tables. The latest SAP application server allows you to define up to 10 DB2 members per application server and obeys to customer-defined rules that control the relative precedence of each member.

The ABAP infrastructure also takes data sharing into consideration. For example, ABAP batch scheduling groups have recently been introduced that effectively allow you to define an affinity between a batch job and a subset of DB2 members. Moreover, some SAP applications from the banking sector are designed in a data sharing-aware way that aims at minimizing global lock contention

The SAP support for data sharing is also central to automated SAP solutions that are based on IBM Tivoli® System Automation. The IBM Redbook, *SAP on DB2 UDB for OS/390 and z/OS: High Availability Solution Using System Automation*, SG24-6836, elaborates on the solution that is based on Tivoli System Automation for z/OS. The Redpaper, *mySAP Business Suite Managed by IBM Tivoli System Automation for Linux*, REDP-3717, discusses the solution with Tivoli System Automation for Linux.

The panel in Figure 5-14 is available from the SAP transactions DB2 or ST04. It indicates the data sharing topology of an SAP system and shows to which DB2 member individual SAP work processes are currently connected. This information is also summarized per SAP application server and per DB2 member.

DB2 member/SAP app.server/Work process	Dialog	Update	Update2	Background	Enqueue	Spool
SGF1(SSID:SGF1, z/OS:SAPD)	0	0	0	0	0	0
SGF2(SSID:SGF2, z/OS:SAPF)	4	1	0	3	1	1
ihls08_M7E_75	4	1	0	3	1	1

Figure 5-14 Data sharing topology

5.7.2 CF lock propagation reduction

DB2 V8 remaps parent IX L-locks from XES-X to XES-S locks. Data sharing locking performance benefits, because parent IX and IS L-locks are now both mapped to XES-S locks and are therefore compatible and can now be granted locally by the XES (cross-system extended services) component of z/OS. DB2 does no longer need to wait for global lock contention processing to determine that a new parent IX or IS lock is compatible with existing parent IX or IS locks.

This enhancement reduces data sharing overhead by reducing global lock contention processing. It is not uncommon for parent L-Locks to cause global contention. On page set open (an initial open or open after a pseudo close) DB2 normally tries to open the page set in RW. To do this, DB2 must ask for an X or IX page set L-Lock. If any other DB2 member already has the data set open, global lock contention occurs prior to DB2 V8.

The purpose of this enhancement is to avoid the cost of global contention processing whenever possible. It also improves availability due to a reduction in retained locks following a subsystem failure. Figure 5-15 shows the subsystem activity panel of the SAP integrated database performance monitor (SAP transaction code ST04). It includes the global and false contention rate. In the following section we explain this enhancement in more detail.

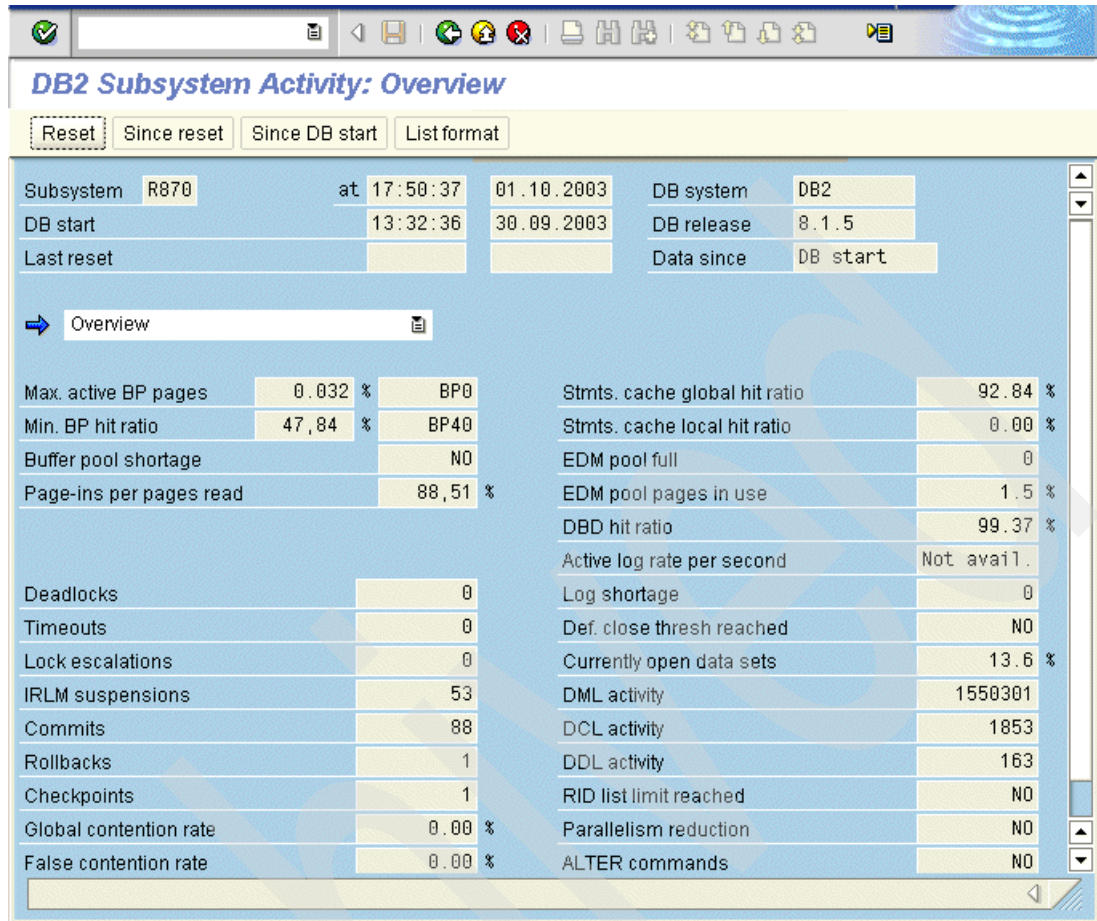


Figure 5-15 ST04 database performance monitor: subsystem activity

Data sharing locking revisited

DB2 data sharing uses two types of locks, physical and logical:

► Physical locks (P-locks):

Briefly, physical locks are used to track the “level of interest” that DB2 data sharing members have in a particular page set or partition.

There are two kinds of P-locks, page set and page:

– Page set physical locks:

Page set P-locks are used to track inter-DB2 read-write interest, thereby determining when a page set has to become GBP (group buffer pool)-dependent.

When a DB2 member requires access to a page set or partition, a page set P-lock is taken. This lock is always propagated to the lock table on the coupling facility and is owned by the member. No matter how many times the resource is accessed through the member, there is always only one page set P-lock for that resource for a particular member. This lock has different modes depending on the level of interest (read or write) that the member has in the resource.

The first member to acquire a page set P-lock on a resource takes the most restrictive mode of lock possible. This is an S page set P-lock for read interest and an X page set P-lock for write interest. An X page set P-lock indicates that the member is the only member with interest (read or write) in the resource. Once another member becomes interested in the resource, the page set P-lock mode can be negotiated, that is, it can

be made less restrictive if the existing page set P-lock is incompatible with the new page set P-lock request. The negotiation always allows the new page set P-lock request to be granted, except when there is a retained X page set P-lock. A retained P-lock cannot be negotiated. Retained locks are locks that must be kept to protect possibly uncommitted data left by a failed DB2 member. Page set P-lock negotiation signifies the start of GBP dependence for the resource.

Although it may seem strange that a lock mode can be negotiated, remember that page set P-locks do not serialize access to a resource, they are used to track which members have interest in a resource and for determining when a resource must become GBP-dependent.

Page set P-locks are released when a page set or partition data set is closed. The mode of page set P-locks is downgraded from R/W to R/O when the page set or partition is not updated within an installation-specified time period or number of checkpoints. When page set P-locks are released or downgraded, GBP dependency is reevaluated.

– **Page physical locks:**

Page P-locks are used to ensure the physical consistency of a page across members of a data sharing group in much the same manner as latches do in a non-data sharing environment. A page P-lock protects the page while the structure is being modified. Page P-locks are used not only when row locking is in effect, but are also used in other ways, for example, when changes are being made to GBP-dependent space map pages or row level locking is being used. Page physical locks are also used to read and update index pages.

► **Logical locks (L-locks):**

Logical locks are also referred to as transaction locks. L-locks are used to serialize access to data to ensure data consistency.

L-locks are owned by a transaction, and the lock duration is controlled by the transaction. For example, the lock is generally held from the time the application issues an update until the time it issues a commit. The locks are controlled locally per member by each member's IRLM.

P-locks and L-locks work independently of each other, although the same processes are used to manage and maintain both. The lock information for all these locks is stored in the same places (the IRLM, XES, and the coupling facility).

Explicit hierarchical locking

Conceptually, all locks taken in a data sharing environment are global locks, that is, they are effective group-wide, even though all locks do not have to be propagated to the lock structure in the coupling facility.

DB2 data sharing has introduced the concept of explicit hierarchical locking, to reduce the number of locks that must be propagated to the coupling facility.

Within IRLM, a hierarchy exists between certain types of L-locks, where a parent L-lock is the lock on a page set; and a child L-lock is the lock held on either the table, data page, or row within that page set.

By using explicit hierarchical locking, DB2 is able to reduce the number of locks that must be propagated to the lock structure in the coupling facility. The number of locks that are propagated to the lock structure for a page set or partition is determined by the number of DB2 members interested in the page set and whether their interest is read or write. Wherever possible, locks are granted locally and not propagated to the coupling facility.

If a lock has already been propagated to XES protecting a particular resource for this member, a subsequent lock requests for the same lock does not have to be sent to XES by the same member for the same resource. They can be serviced locally. In addition, a parent L-lock is propagated only if it is more restrictive than the current state that XES knows about for this resource from this member.

SAP specifies release commit on the bind parameter. In SAP environments, parent L-locks are therefore released when the transaction commits. Child L-locks are propagated to the lock table in the coupling facility only when there is inter-DB2 read-write interest for the page set.

Child locks (page and row locks) are propagated to XES and the coupling facility based on inter-DB2 interest on the parent (table space or partition) lock. If all the table space locks are IS, then no child locks are propagated. However, if there is an IX lock on the table space or partition, which indicates read/write interest, then all the child locks must be propagated.

Lock contention

Figure 5-16 presents a logical overview of how the IRLMs in a data sharing group cooperate to maintain data integrity for a page set where both DB2 members have interest in the page set.

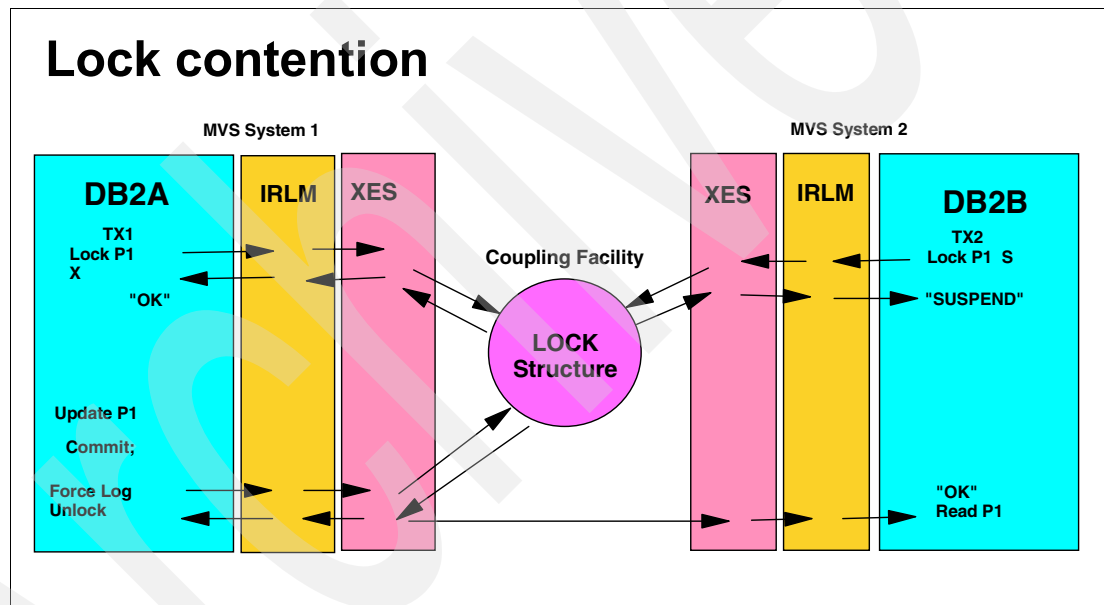


Figure 5-16 Global locking

Consider transaction TX1 on DB2A, which needs an X lock on page P1. IRLM passes this lock request to XES and the lock is granted. Now, transaction TX2 on DB2B needs an S lock on page P1. IRLM passes this lock request through XES to the coupling facility. As transaction TX1 already has an X lock for page P1, transaction TX2 must be suspended.

Transaction TX1 now updates page P1 and commits. The IRLM releases the X lock and passes an unlock request through XES to the coupling facility. The S lock is now granted and transaction TX2 can be un-suspended to continue its work.

Now, let us have a closer look at the various reasons why a transaction may be suspended. Lock information is held in three different components:

- ▶ IRLM
- ▶ XES
- ▶ Lock structure on the coupling facility

The types of lock granularity supported by each component differ. IRLM contains the most detailed lock information; whereas XES and the lock table on the coupling facility recognize only two types of locks — S and X. Each IRLM lock maps to a particular XES lock. IS and S map to XES-S locks; while U, IX, SIX and X map to XES-X locks.

Lock contention occurs when a task is requesting a lock for a resource and the lock may already be held by another task. In data sharing, the other task could be running in the same DB2 subsystem or running in another DB2 member. For this discussion we are only concerned about global contention, when the contention is across DB2 members.

In data sharing, three types of global contention can occur. These are listed in order of increasing time needed for their resolution:

► **False contention:**

False contention occurs when the hashing algorithm for the lock table provides the same hash value for two different resources. The different resources then share that one lock table entry.

False contention can occur only when the lock table entry is managed by a global lock manager or when the lock request causes global management to be initiated, that is, there is inter-DB2 R/W interest in the page set. The XES requesting the lock needs to know the owning resource name to resolve this apparent contention. That information already resides in the XES that is the global lock manager for the lock table entry. If the global lock manager is not the requesting XES, communication between XES components is needed to resolve the false contention. At SAP installations, false contention should be below 1.5%

In our example, false contention would occur if transaction TX2 were to request a lock for a different resource, say page P2, and the lock request hashed to the same lock table entry in the coupling facility.

Transaction TX2 must be suspended while the XES who is the global lock manager for the lock table entry, determines that the lock can be granted.

► **XES contention:**

The z/OS XES component is aware of only two lock modes, share and exclusive. IRLM locking supports many additional lock modes. When the z/OS XES component detects a contention because of incompatible lock modes for the same resource, that contention is not necessarily a real contention by IRLM standards. For example, the IRLM finds the IX-mode to be compatible with the IS-mode. For the XES component, however, these are not IX-mode and IS-mode, but X-mode and S-mode, which are incompatible. To see if a real contention exists, XES must give control to the IRLM contention exit associated with the global lock manager. The IRLM contention exit must determine if the contention is real or not, that is, if the locks are incompatible. If the contention is not real, it is called “XES contention” and the requested lock can be granted.

In our example, XES contention would occur if transaction TX1 held an IX lock on page P1 and transaction TX2 was requesting an IX lock on page P1. Both of these lock requests are passed to XES as X locks. XES sees these lock requests as not compatible however IRLM knows they are compatible.

Transaction TX2 must be suspended while the XES who is the global lock manager for the lock table entry, must defer to IRLM to decide if the lock request can be granted.

► **Real contention:**

Real contention is caused by normal IRLM lock incompatibility between two members. For example, two transactions may try to update the same resource at the same time. DB2 PM reports real contentions as IRLM contentions.

This is the example we have just explained. Transaction TX2 is requesting a lock which is not compatible with a lock already held by transaction TX1. Transaction TX2 must be suspended while XES defers to IRLM who cannot grant the lock.

Resolving contention

Contentions require additional XES and XCF services if the requesting member is not global lock manager, that is, the owner of the lock registered in the lock table entry.

Information about locks that IRLM passes to XES is stored in XES. When contention occurs (false, XES, or real contention), one of the XES instances is assigned to be the global lock manager to resolve the contention. This resolution involves all of the other XESs in the group that have locks which have been assigned to the lock table entry, passing their lock information to the global lock manager XES. This global lock manager XES can then drive resolution of the contention.

When any contention occurs, execution of the requester's SQL statement is suspended until the contention is resolved. If the contention is real, the requester remains suspended until the incompatible lock is released.

Therefore, any contention can adversely impact performance. The SQL is suspended while the contention is resolved and extra CPU is consumed resolving the contention.

The enhancement

In DB2 V8, parent IX L-locks is remapped to XES-S locks, rather than XES-X locks. This allows parent IX L-locks to be granted locally by XES when only IS or IX L-locks are held on the object.

To ensure that parent IX L-locks remain incompatible with parent S L-locks, S table and table space locks are remapped to XES-X locks. This means that additional global contention processing is done to verify that a page set S L-lock is compatible with another page set S L-lock, but gross S- or X-locks are requested extremely rarely for SAP applications. The major reason is lock escalation and the goal is to avoid lock escalation at all. Table 5-4 summarizes the new mapping of IRLM locks to XES locks.

Table 5-4 Mapping of IRLM locks to XES locks

IRLM lock	XES lock with DB2 V8	XES lock prior to DB2 V8
IS	S	S
S	X	S
U	X	X
IX	S	X
SIX	X	X
X	X	X

The majority of cases involve intent gross locks only:

- ▶ **IS-IS:** A member wants to execute some read-only SQL against a page set and there are a number of other members who currently have some read-only SQL active against the same page set.
- ▶ **IS-IX:** A member wants to execute some update SQL against a page set and there are a number of other members who currently have some read-only SQL active against the same page set.

- ▶ **IX-IX:** A member wants to execute some update SQL against a page set and there are a number of other members who currently have some update SQL active against the same page set.

Hence, global contention processing is reduced by this enhancement. Parent lock contention with parent S L-locks is less frequent than checking for contention with parent IS and IX L-locks.

Child-L lock propagation with DB2 V8

Another impact of this change is that child L-locks are no longer propagated to the lock structure of the coupling facility based on the parent L-lock. Instead, child L-locks are propagated based on the held state of the page set P-lock. If the page set P-lock is negotiated from X to SIX or IX, then child L-locks are propagated.

It can happen that some child L-locks are acquired before the page set P-lock is obtained. In this case already acquired child L-locks are automatically propagated. This situation can occur, because DB2 always acquires locks before accessing the data. DB2 acquires the page set L-lock before opening the page set to read the data. This can also happen during DB2 restart.

An implication of this change is that child L-locks are propagated for longer than they are needed, however this should not be a concern. There will be a short period from the point in time when there is no inter-system read/write interest until the page set becomes non-GBP-dependent, that is, before the page set P-lock reverts to X. During this time, child L-locks are propagated unnecessarily.

Another consequence of this enhancement is that, since child L-lock propagation is no longer dependent upon the parent L-lock, parent L-locks are no longer held in retained state following a system failure. This means for example that a page set IX L-lock is no longer held as a retained X-lock after a system failure. This can provide an important availability benefit in a data sharing environment. Because there is no retained X-lock on the page set anymore, most of the data in the page set remains available to applications running on other members. Only the rows with retained X-lock will be unavailable for SAP systems, which use row-level locking.

Benefits of less lock propagation to the coupling facility

Data sharing locking performance benefits largely, because this enhancement allows IX and IS parent L-locks, which are by far the most often used gross locks at SAP installations, to be granted locally. Global lock contention processing does not need to be invoked to determine that the new IX or IS lock is compatible with existing IX or IS locks.

Prior to DB2 V8, the best data sharing performance is achieved by using the bind option `RELEASE(DEALLOCATE)` to reduce XES messaging for page set L-locks. This is no longer true with DB2 V8. With respect to data sharing, `RELEASE(COMMIT)` now yields equivalent performance as `RELEASE(DEALLOCATE)`. This means that SAP applications, which uniformly employ `RELEASE(COMMIT)`, directly take advantage of this enhancement.

Additionally, the ability to grant IX and IS locks locally implies less thrashing on changing inter-system interest levels for parent locks, which reduces both IRLM SRB time and XCF messaging. When DB2 decides to propagate its locks to the coupling facility for a given page set, DB2 needs to collect and propagate all the locks that it currently owns for that page set to the coupling facility. This can cause some overhead, particularly when a page set is not used often enough for constant lock propagation. Page set P-locks are long duration locks and tend to be more static than L-locks. Therefore, it is more likely that lock propagation continues longer, which avoids situations where DB2 members have to propagate all locally held locks for a given page set to the coupling facility.

Since page set IX L-locks are not held as a retained X-lock after a system failure anymore, availability in data sharing environments is further improved.

Coexistence and enablement

Since the new locking protocol cannot coexist with the old one, the new protocol only takes effect after the first group-wide shutdown when the data sharing group is in New Function Mode. No other changes are required to take advantage of this enhancement

If you recover the catalog and directory to a point in time prior to New Function Enable Mode, a group-wide shutdown is required. On the next restart, whether it be on V7 or V8, the new locking protocol is disabled.

5.7.3 CF request batching

Prior to V8, DB2 allows multiple pages to be registered to the coupling facility with a single command. z/OS 1.4 and CF level 12 introduce two new “batch” processes to:

- ▶ Write And Register Multiple (WARM) pages of a GBP with a single command.
- ▶ Read multiple pages from a GBP for castout processing with a single CF read request. The actual command is called Read For Castout Multiple (RFCOM).

DB2 V 8 exploits these new CF commands to reduce the amount of traffic to and from the coupling facility for write operations to GBP and for read operations from GBP for castout processing, thus reducing the data sharing overhead for many workloads. As mentioned, SAP applications try to minimize the data sharing overhead in different areas. CF request batching potentially reduces data sharing overhead where these efforts do not apply. Hence, the most benefit is expected for workloads which update large numbers of pages belonging to GBP-dependent objects.

The following subsections explain how CF request batching benefits write operations to GBPs, GBP castout processing and index page splits. Besides these advantages, CF request batching improves DB2 commit processing performance in case any remaining changed pages must be synchronously written to the GBP during commit processing. For GPBCAHE(ALL) page sets, DB2 is now able to more efficiently write prefetched pages into the group buffer pool as it reads them from disk (Figure 5-17).

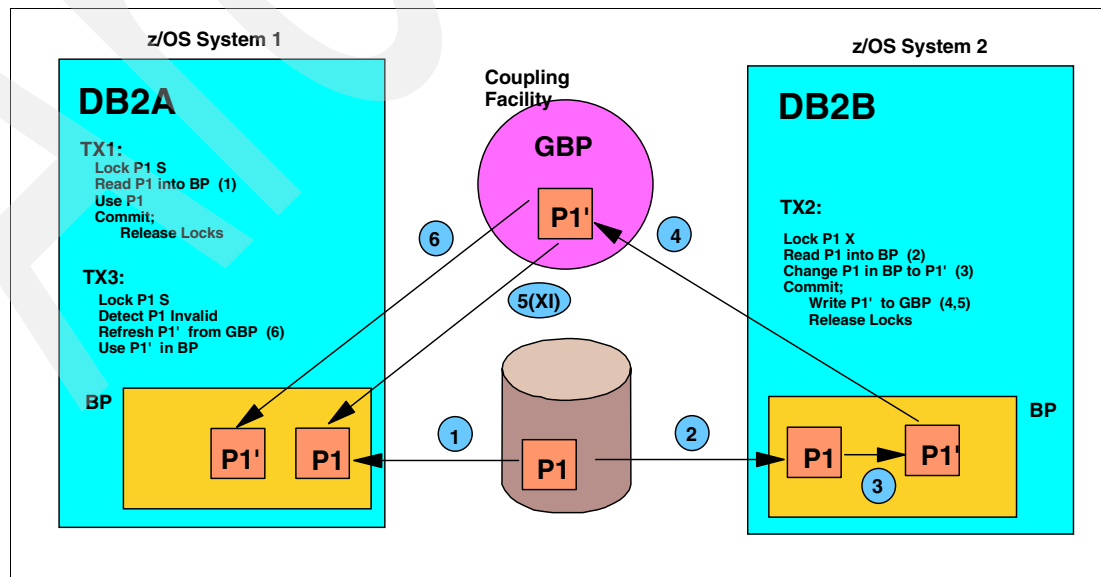


Figure 5-17 Data sharing: Inter-DB2 buffer pool coherency

Inter-DB2 buffer pool coherency

SAP applications — as any other application — can access data from any DB2 subsystem in a data sharing group. Many subsystems can potentially read and write the same data. To prevent inconsistencies, DB2 uses special data sharing locking and caching mechanisms to ensure data consistency. Figure 5-17 on page 151 provides a brief overview of how shared data is updated and how DB2 protects the consistency of data.

Suppose that an application issues an UPDATE statement from DB2A and that the data does not reside in the member's buffer pool or in the group buffer pool. In this instance, DB2A must retrieve the data from disk and get the appropriate locks to prevent another DB2 from updating the same record at the same time.

Because no other DB2 subsystem shares the table at this time, DB2 does not need to use data sharing integrity mechanisms to process DB2A's update.

Next, suppose another application, running on DB2B, needs to update that same data page. DB2 knows that inter-DB2 interest exists, so DB2A writes the changed data page to the GBP. DB2B then retrieves the data page from the GBP.

After DB2B updates the data, it moves a copy of the data page into the GBP, and the data page is invalidated in DB2A's buffer pool. Cross-invalidation occurs from the GBP.

When DB2A needs to read the data, the data page in its own buffer pool is invalid. Therefore, it reads the latest copy from the GBP.

Prior to V8, DB2 already allows you to register multiple pages to the CF in a single command. If DB2 V8 runs on z/OS 1.4 or higher and the CF level is at least 12, DB2 is enabled to write multiple pages to a GBP in a single operation. This reduces the amount of traffic to the CF and hence improves performance. In the above example, if DB2B not only updates a single page but a large number of pages, DB2 now uses CF request batching to write the changed pages to GBP in fewer operations.

Castout processing

To reduce restart time after an unplanned outage, DB2 periodically writes changed pages from the GBP to disk. This process is called *castout*.

There is no physical connection between GBPs and disk, so the castout process involves reading the pages from a GBP into a group member's private buffer, which is not part of the member's buffer pool storage, and writing the page from the private buffer to disk.

Castout is triggered when:

- ▶ A GBP checkpoint is taken.
- ▶ The GBP castout threshold is reached.
- ▶ The class castout threshold is reached.
- ▶ GBP dependency is removed for a page set.

Within a group buffer pool, there are a number of castout classes; the number of classes is an internal value set by DB2. Page sets using the GBP are mapped to a specific castout class. DB2 preferably has only one data set assigned to a particular castout class, although it is possible to have more than one data set mapped into the same castout class, depending on how many data sets are using the group buffer pool concurrently.

Castout classes are used to limit the number of changed pages a data set can have in a GBP at any one time, thereby limiting the amount of I/O to the data set at castout time. Large amounts of I/O could cause disk contention. This limitation is achieved through the use of the castout class threshold. The default of the castout class threshold parameter is 10, which means that castout is initiated for a particular class when 10% of the GBP contains pages for that class or, if only one data set is assigned to that class, when 10 percent of the GBP contains pages for that data set. The castout class threshold applies to all castout classes. You can change the castout class threshold by using the ALTER GROUPBUFFERPOOL command.

Data sets have a GBP castout owner assigned to them. The group buffer pool castout owner is the first member to express write interest in the data set. After castout ownership is assigned, subsequent updating DB2 subsystems become backup owners. One of the backup owners becomes the castout owner when the original castout owner no longer has read-write interest in the page set or partition. At castout time the castout owner is responsible for performing the actual castout process for all changed pages for the data set.

Castout processing predominantly involves reading multiple pages from a GBP. Therefore, CF request batching is well-suited to yield performance benefits that should make castout processing less disruptive.

Batched index page splits

In previous versions of DB2, page splits of indexes with inter-DB2 read/write interest require up to five synchronous log writes and up to five separate write operations to GBP. The five separate writes are to ensure that when a leaf page split occurs, the index pages involved are written out in the correct order. This prevents other members from seeing a reference to an index page that doesn't yet exist in the group buffer pool.

This can cause significant overhead for GBP-dependent indexes, resulting in additional synchronous GBP writes, and even more important, in high wait times for synchronous log I/O.

In V8, DB2 accumulates the index page split updates and processes them as a single entity. This reduces log writes and coupling facility traffic. The write operations of the accumulated updates to the coupling facility employ CF request batching.

SAP applications that heavily insert data in tables with GBP-dependent indexes see the most benefit of this enhancement. However, applications that exploit the SAP capabilities to avoid GBP-dependency at all are not affected.

5.7.4 Improved LPL recovery

Prior to V8, pages that DB2 puts into the logical page list (LPL) need to be recovered manually. Issuing the START DATABASE command with the SPACENAME option accomplishes this. It drains the entire page set or partition though even if only a single page is in LPL. The complete page set or partition is unavailable for the duration of the LPL recovery process.

DB2 V8 recovers LPL pages without draining page sets or partitions. It only locks the LPL pages during the recovery process, leaving the remaining pages in the page set or partition accessible to applications. This significantly improves system performance and enhances data availability. Instead of draining, DB2 makes a write claim on the affected page set or partition so that the intact pages can still be accessed.

Also, DB2 V8 adds automatic recovery of LPL pages, which does not drain page sets or partitions either. When pages are added to the LPL, DB2 issues message DSNB250E, which is enhanced to indicate the reason why the pages are added to the LPL. DB2 then attempts automatic recovery, except in cases where the recovery cannot succeed. These cases are:

- ▶ Disk I/O error
- ▶ During DB2 restart
- ▶ GBP structure failure
- ▶ GBP loss of connectivity

If automatic LPL recovery completes successfully, DB2 deletes the pages from the LPL and issues message DSN1021I, which indicates completion.

The improved LPL recovery contributes to enhanced availability characteristics of SAP systems that run on DB2 V8. The recovery process is automated where possible, to make it faster and less disruptive.



Tools and administration

In this chapter we describe the following features:

- ▶ Automatic Space Management
- ▶ DSC statement ID in Explain
- ▶ Long-running non-committing readers
- ▶ Lock escalation alert
- ▶ SAP and DB2 Control Center
- ▶ SAP transaction-based DB2 accounting and workload management

6.1 Automatic Space Management

In this section we describe the new space management features introduced in DB2 V8. For an SAP implementation, these features are extremely valuable for continuous availability, as they provide adaptability when data growth patterns are not predictable, or do not follow those expected.

The new Automatic Space Management feature introduced in DB2 V8 is considered highly valuable for all SAP implementations. It will potentially eliminate one of the main causes of failures where growth has not been completely anticipated. As discussed in this section, administrative tasks still remain, but reducing or eliminating a cause of failure is a major step forward. We cover the following topics:

- ▶ Why the new features are important in an SAP system
- ▶ How these features work

6.1.1 Challenges of data management of an SAP system

After the initial SAP installation activities, the database will have a well defined layout, with a large number of DB2 objects, and a set of data inserted throughout these tables. The actual layout will be quite similar for most customers, but will vary according to the versions of the mySAP component and DB2 for z/OS.

What happens to this system after the initial installation will vary, depending on a number of factors, which include:

- ▶ The role of the system, for example, development, testing, or production
- ▶ The mySAP component, and the choice of available functionality used by the customer. Usually most customers use less than the complete set of possible features
- ▶ The amount of customer specific data loaded into the system
- ▶ Activity taking place in the system that results in significant changes to the database content

Differences between customers

One of the greatest challenges in administering SAP systems is that every customer varies in the usage of the supplied package. In some cases these differences are great, such as the difference between a customer who installs SAP Enterprise and implements an HR/Payroll system, and another customer who implements a Sales and Distribution system. Even customers implementing the same or a very similar set of SAP modules will make choices in the implementation process that result in a different subset of database objects being actively utilized, both in terms of data volumes and activity on this data.

Usually the supplied mySAP component has rules for the generation of the database layout that cater for most “average” customers, but in reality there is no such thing as an absolutely average customer. Invariably, everyone is at least slightly different from the average.

Impact on database administration (DBA) function

If a database object, either in the form of a table space, tables pace part, or index reaches the maximum number of extents permitted, usually the following message is issued on the system console:

```
DSNP007I csect – EXTEND FAILED FOR  
data-set-name. RC=00D70014  
CONNECTION-ID=xxxxxxx,
```

CORRELATION-ID=yyyyyyyyyyyyy,
LUW-ID=logical-unit-of-work-id =token

The unit of work issuing the SQL statement resulting in a space request will abend, and subsequently repeating any work trying to add data to this object will similarly fail until the issue is resolved.

This situation can potentially be avoided through the correct “pre-sizing” of the objects to be used in an SAP system; however, it is not always possible anticipate future space requirements. If these failures do occur, they are generally either during SAP system implementation, or after the system “goes live”.

Issues arising during SAP system implementation

Typically, at some point in the process of implementing a mySAP component, there is a phase, or a number of phases, where a customer will perform activities normally referred to as “data loads”. This is where data from other systems is placed into the new SAP system, using a number of means available to perform the transfer. The various techniques for loading data vary in their time to implement, and runtime efficiency, but eventually all lead to rows being inserted and/or modified in a table or series of tables. Due to the complex and highly integrated nature of SAP, it is sometime difficult to know ahead of time what database objects will carry the new data.

In the situation where many different types of data loads are to be performed over a period of time, and these are occurring into a newly installed system, it can be very frustrating to iteratively deal with out-of-space conditions. Where the out-of-space conditions occur due to the maximum number of extents being reached (255), this would have been avoided had suitable values been chosen for PRIQTY and SECQTY of the database object. It may be possible to anticipate the size of the major tables which will have data inserted into them.

However, in an SAP system, it is quite often the case that it is not the well known large tables in the system that cause the most angst during data loads, but in fact, it is usually the small to medium size tables. This can happen due to the size estimates provided in the SAP system “out of the box” not being adequate to handle the data expected in every customer. One solution would of course be to size every table to cater for all possible customers, but this would result in an inordinately large database, most of which would be free space. An equal or greater amount of work would subsequently be required to resize the objects to their actual required space, and subsequently REORG to remove the excess space.

The provision of the new Automatic Space Management function fundamentally addresses this issue by allowing a more adaptive scheme to take place when table spaces and indexes have more data placed in them than anticipated. It should be noted that this solution addresses the problem of failures associated with reaching the maximum permitted number of extents, particularly where this results in a data set size at the time of failure that is less than the theoretical maximum of the value DSSIZE for table spaces or PIECESIZE for indexes.

As such, here are a few issues that are not within the scope of this enhancement:

- ▶ There may be a situation in which the database object will have sufficient data inserted into it that it would also receive a failure when it reaches the DSSIZE or PIECESIZE limit. This is probably very rare in the case of an SAP system. When loading data into an empty system, in most cases the major tables that will have data loaded into them are well known to those programmers and teams performing the loads. In these cases, as part of their normal practice, the teams should have alerted those responsible for the DBA function to those tables in question, and the anticipated number of records involved. In many cases these tables may be partitioned to address size limitations.

- ▶ The database object in question may, after the data loading activity, still have a large number of extents, while not having reached the maximum.

There may be some performance and administrative benefits that result in the subsequent resizing of the object, through altering the PRIQTY and SECQTY and subsequently REORG on the table space or index. This is considered to be much less of an issue than actually suffering space related failures, as they can be addressed in a timely manner while allowing the implementation process to continue.

- ▶ No space is available on the disk where DFSMS routines require the data set extents to be placed.

If a growing data set is extending, and the VSAM Access Method Services routines are not able to find sufficient space on the disk volumes allowed, it is possible that an extend failure will occur before 255 extents are reached. Also, if the remaining disk space is in small, fragmented pieces, it is possible that 255 extents will be reached earlier due to numerous small extents being taken to fulfill space requirements.

Often the actual process of a company beginning to use their SAP system happens during a specific changeover time, where the existing systems have data moved into the new system prior to its usage by the company's users. Quite often this action is taken over a short period of time, due to the unavailability of any system to the users while the changeover occurs. As such, these "go-live" times involve the actual running of the data load jobs in a very short time. Any significant unforeseen problems during this time can delay the start of usage of the new SAP system, and may result in a significant business issue, such as lost sales opportunities, etc.

Issues arising after SAP goes live

After an SAP system begins its normal operations, usually after the aforementioned go-live actions, it is usual that a consolidation time takes place, where the system is now being used by the company's staff. While the system is being actively used, the technical staff are able to start analyzing the data about the actual usage and performance of the system. Additionally, the DBA staff have the first chance to see how the user's activity impacts the content of the DBMS, and how this compares to what was anticipated.

It is normal immediately after the go-live to find a significant number of database objects needing REORG, either because the underlying data set has reached a non-trivial number of extents, or statistics indicate that data within the space is disorganized enough to be causing performance impacts.

After this "cleanup phase", ongoing administrative tasks involve monitoring, among other things, the number of extents and relative organization of data within objects. As DB2 captured statistics indicate objects crossing customer determined thresholds (extents and disorganization indicators), action is normally taken to REORG the objects in question. The function of DB2 Real Time Statistics (RTS) introduced in DB2 for OS/390 and z/OS Version 7 aids this task considerably by providing the data upon which these decisions are made in real time.

SAP has an "extent monitor" built into the administrative transaction DB02. This provides user customizable thresholds, above which the table space or index will appear on the SAP extent monitor transaction. The administrator is then able to take appropriate action as objects grow, and REORG them at a time appropriate to the system availability considerations. By setting the thresholds to appropriate levels, this allows the database administrator a certain "buffer" when planning future administrative actions, minimizing the possibility of unexpected outages due to space related issues.

SAP also provides a job which issues DB2 catalog queries to determine which tables and indexes are sufficiently disorganized to the point where performance is probably less than ideal. Using a similar mechanism to that described for data set extents, the administrator can then schedule a task to build the JCL to reorganize these objects, and run the jobs when appropriate to the customer's workload.

6.1.2 Automatic space management functions

Automatic Space Management is enabled with the new DSNZPARM of MGEXTSZ, which can be set to either YES or NO, and the default is NO. That is, after moving to DB2 V8, the new functionality is disabled initially, and space management occurs as it previously did in DB2 Version 7.

When enabled by setting MGEXTSZ, in certain situations, DB2 will automatically adapt the size of each subsequent secondary allocation requested. It does this in such a way that it follows a sliding scale that guarantees the object will reach its theoretical maximum data set size (DSSIZE) prior to reaching the maximum number of extents allowed for a VSAM data set of 255.

New defaults TSQTY and IXQTY

APAR PQ53067 for DB2 V6 and V7 introduced the new DSNZPARM parameters TSQTY and IXQTY, allowing the administrator to override the default allocation values for primary and secondary quantities for the cases where these are not specified in the object creation DDL.

- ▶ The actual value applied by DB2 for default PRIQTY will be determined by the applicable ZPARMs, TSQTY or IXQTY, which were introduced with APAR PQ53067. The ZPARMs TSQTY and IXQTY will now have global scope.
- ▶ TSQTY will apply to non-LOB table spaces. For LOB table spaces, a 10x multiplier will be applied to TSQTY to provide the default value for PRIQTY.
- ▶ IXQTY will apply to indexes. ZPARMs TSQTY and IXQTY will continue to have a default value of 0 (zero), but this value will indicate a new default value of 720 KB (1 cylinder) is to be applied.
- ▶ If TSQTY is set to 0, then 1 cylinder will be the default PRIQTY space allocation for non-LOB table spaces and 10 cylinders will be the default PRIQTY space allocation for LOB table spaces.
- ▶ If IXQTY is set to 0 then 1 cylinder will be the default PRIQTY space allocation for indexes.

The user can provide override values for TSQTY and IXQTY ZPARMs to avoid wasting excessive disk space. For example on a development subsystem, TSQTY and IXQTY may be set to 48 KB for track allocation. The use of the default for PRIQTY will be recorded in the associated PQTY column as -1 in the SYSIBM.SYSTABLEPART or SYSIBM.SYSINDEXPART catalog table.

Implementation details

The main objective of Automatic Space Management is to avoid the situation where a DB2 managed page set reaches the VSAM maximum extent limit of 255 before it can reach the maximum data set size, which may be less than 1 GB, 2 GB, 4 GB, 8 GB, 16 GB, 32 GB, or 64 GB. The actual secondary allocation quantity applied will not be reflected in the Catalog, and will not exceed DSSIZE or PIECESIZE. For an index, the user specified PIECESIZE, which limits data set size, and can start as low as 256 KB. It allows for a managed DB2 page set to reach the maximum page set size before running out of extents. This will help to reduce the number of out-of-space conditions, improve user productivity, and additionally avoid the performance penalty associated with small extent sizes.

Two sliding scales can be used, one for 32 GB and 64 GB data sets, and one for the rest (less than 1 GB, 1 GB, 2 GB, 4 GB, 8 GB, 16 GB). Maximum data set size can be determined based on DSSIZE, LARGE and PIECESIZE specification for the object. Both sliding scales will allocate an increasing secondary quantity size up to 127 extents and a constant number thereafter. The constant is 559 cylinders for the 32 GB and 64 GB data sets, and 127 cylinders for the rest.

The effect of using MGEXTSZ=YES is illustrated in Figure 6-1, where after the number of extents has exceeded the SECQTY value for the table space or index (in cylinders), the size of each subsequent extent is gradually increased, until the object reaches 127 extents. A fixed secondary extent, depending on the maximum possible data set size, is then allocated for extent numbers 128-255.

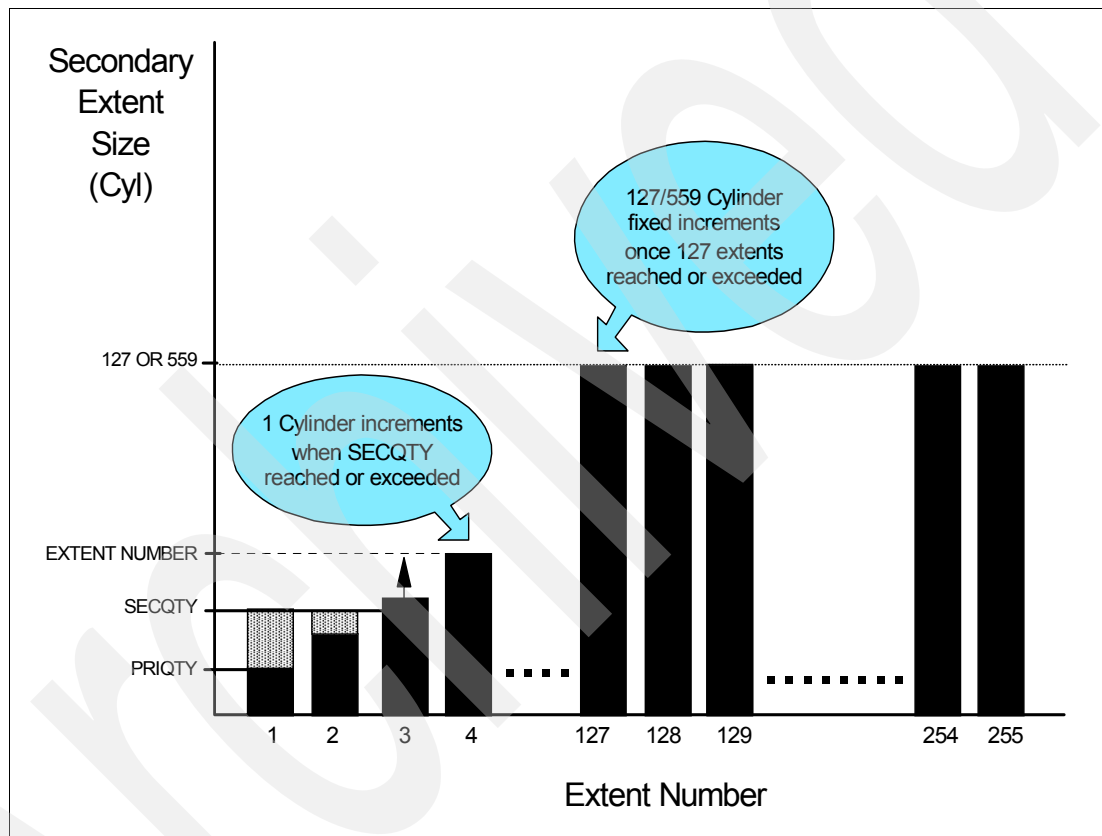


Figure 6-1 Increasing secondary allocation with sliding scale

This approach of sliding the secondary quantity minimizes the potential for wasted space by increasing the extents size slowly at first, and it also avoids very large secondary allocations from extents 128-255, which will most likely cause fragmentation where multiple extents have to be used to satisfy a data set extension. The solution will address new data sets that will be allocated, and also existing data sets requiring additional extents. Therefore, in the case of an already installed SAP system, after upgrading to DB2 V8, and enabling the new ZPARM, the full benefits of this enhancement can be realized immediately.

When MGEXTSZ is enabled, if the SECQTY value specified by the user is greater the 0, the actual secondary allocation quantity will be the maximum of the calculated quantity size using sliding scale methodology, and the SECQTY value specified by the user. When a page set spills onto a secondary data set, the actual secondary allocation quantity will be determined as described and applied, and the progression will continue. Prior to DB2 V8, the PRIQTY would have been used.

Tip: If SECQTY is specified by the user as 0 to indicate “do not extend”, this will always be honored. So for certain circumstances, especially DSNDB07 workfiles, where users deliberately set SECQTY=0 to prevent data set extension, this will remain the case even with the MGEXTSZ parameter enabled.

6.1.3 Scenarios and benefits of Automatic Space Management

One of the major benefits of automatic space Management is preventing disruptions to all users, based on the unexpected activity against one or more database tables in an SAP system. Most often this will occur in situations such as data loads prior to the productive usage of the SAP system, or implementation of additional functions within the SAP application. It is possible, but much more unlikely, that out-of-space conditions will occur during normal operations. In this case, the growth would be expected to have some steady rate. Normal monitoring of data set extents across all objects would normally catch this growth, and corrective action taken prior to objects reaching 255 extents.

Because the MGEXTSZ ZPARM can be enabled and disabled, it is possible that it could be set to YES only during known periods of growth. This would be particularly useful before the first execution of loading data into the SAP system for a new functional area. In this way it would prevent what commonly occurs in this situation, where the loads run until one object reaches 255 extents. The load process abends, possibly requiring a non-trivial time to back out, depending on how it is written in regard to commit frequency. The database object then needs to be resized through ALTER commands and subsequent REORG utility execution. The data loads then continue, and often this is just long enough until the next space related error happens, and the above steps are repeated. Typically in an SAP system, this can occur with indexes on tables, as there are many indexes, and frequently their default allocation results in growth to 255 extents very quickly.

In this situation, where the application knowledgeable personnel are unable to comprehensively specify which database objects are expected to receive what volume of data, this new DB2 functionality will remove this frustration. It should be stressed again that subsequent action will most likely need to take place after the data load phase completion, with table spaces and indexes that are in a non-trivial number of extents and/or disorganized being REORGed. But this can happen in a much more controlled and less time critical manner.

Finally, while it may be possible to deactivate the new MGEXTSZ parameter at times where no exceptional activity is expected, it is highly recommended that the parameter is active (that is set to YES) at all times. The benefit of reducing user impact due to out-of-space conditions greatly outweighs the slight possibility of using additional space unnecessarily. The algorithm used ensures that while objects are in small numbers of extents, the potential wasted space is small. And with good practices in place to detect objects growing into larger numbers of extents, and subsequently addressing them, the best of all worlds is achieved.

Tip: While the activation of the new MGEXTSZ DSNZPARM has to be considered in terms of the benefits and drawbacks discussed in this section, we feel that all SAP implementations would benefit from “turning it on, and leaving it on” !

An illustration of the difference possible with the new function is shown in Figure 6-2, where an index has been defined with PRIQTY = 25 cyls, and SECQTY = 10 cyls. Notice that while this is displayed as a small index allocation, in many customer situations the allocation may be even smaller than this. With the Automatic Space Management function, the result will still be that the data set can grow to 4 GB, but will come closer to the 255 extent limit in doing so, while never reaching it.

This in contrasted with what would happen if MGEXTSZ was set to NO. In this case, the data set size would reach 2565 cylinders, at which point subsequent activity requiring more space would fail, as the object is at the maximum VSAM limit of 255 extents. These failures occur despite the fact the DSSIZE parameter of the object indicates it should be able to grow to 4 GB or ~22,000 cylinders.

In the example in Figure 6-2, the secondary extent allocations will all be 10 cylinders until the 10th extent is allocated. At this point, the 11th extent would be allocated at 11 cylinders, the 12th at 12 cylinders, and so on. When the 127th extent is reached, the subsequent allocation from that point onwards would be 127 cylinders, until the 4 GB data set size limit is reached.

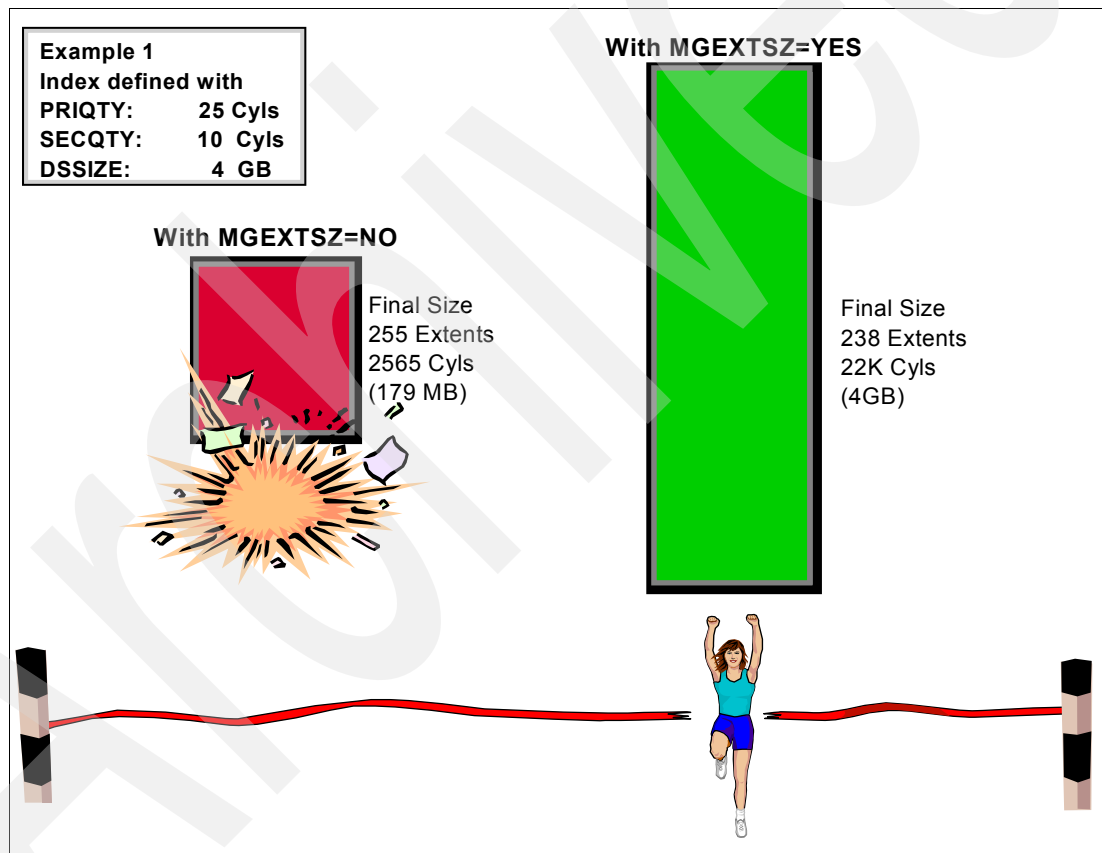


Figure 6-2 Example of small index scenario

6.2 DSC statement ID in Explain

In this section we describe a DB2 V8 enhancement to the Explain function necessary to support the REOPT(ONCE), REOPT(ALWAYS), and RUNSTATS enhancements.

Prior to DB2 V8, the EXPLAIN statement has always shown the access path that a query would use, if a PREPARE or BIND were executed against the database in the current state. The ability of the RUNSTATS utility to invalidate cached statements meant that, under certain circumstances, SQL statements currently executing in the DB2 subsystem could be using a different access path than that indicated by an Explain. While not ideal, this was acceptable because support personnel were aware of utility executions and the ST04 dynamic statement cache provided information related to RUNSTATS invalidation and current use within the SAP environment.

The introduction of REOPT(VARS) further complicates things in that an EXPLAIN against an SQL statement in the DSC will now report an access path that may or may not reflect the path being used. Again, this is hardly ideal, but was acceptable because the parameter values could be captured from an ST05 SQL trace and an EXPLAIN executed with literals replacing parameter markers in the SQL statement.

DB2 V8 introduces REOPT(ONCE), which causes the selection of the access path for an SQL statement after the first set of execution host variables are known. The path will then be fixed for all subsequent executions. The DB2 V7 function of Explain is no longer sufficient because it is now impossible to determine, with any accuracy, the access path being used by an executing SQL statement or any statement that is already extant in the Dynamic Statement Cache.

6.2.1 Description of the enhancement

DB2 V8 enhances the DB2 Explain to allow an Explain to be executed against any SQL statement based on that statements STMTID or STMTTOKEN. This will allow the determination of the actual access path being taken for any executing SQL statement currently resident in the Dynamic Statement Cache. This is a significant improvement over pre-DB2 V8 capabilities and will remove the uncertainties associated with the use of EXPLAIN in an SAP environment. Figure 6-3 shows the syntax of the DB2 V8 EXPLAIN STMTCACHE statement.

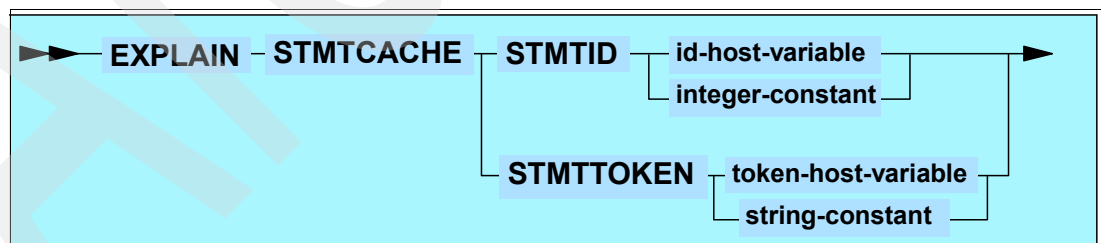


Figure 6-3 DB2 V8 EXPLAIN STMTCACHE syntax diagram

6.2.2 New parameters and plan table changes

The DB2 V8 EXPLAIN statement includes two new keyword parameters for the STMTCACHE option and some changes to the structure and contents of the PLAN table.

STMTCACHE STMTID

The STMTID parameter takes either a host variable or an integer constant. This statement identifier uniquely identifies a statement cached in the Dynamic Statement Cache. The unique value for a particular SQL statement can be obtained from IFCID 316 or 124 and is also available in some diagnostic IFCID trace records such as 172, 196, and 337 (Example 6-1).

Example 6-1 STMTID parameter

```
SID = 124;  
EXEC SQL EXPLAIN STMTCACHE STMTID :SID;
```

STMTCACHE STMTOKEN

The STMTOKEN parameter takes either a host variable or a string constant. This statement token string is associated with the cached statement by the application program that originally prepares and inserts the statement into the cache. The application does this by using the RRSF SET_ID function or by using the sqlseti API from a remotely-connected program (Example 6-2).

Example 6-2 STMTOKEN parameter

```
EXEC SQL EXPLAIN STMTCACHE STMTOKEN 'SELECTTEMP'
```

Plan table changes

Each row inserted into the plan table, statement table, or function table by the Explain statement will contain the statement token in the newly defined STMTOKEN column. The QUERYIN column takes the value of the unique statement identifier from the statement cache. The PROGRAM column has the value DSND CACH for all rows inserted resulting from an EXPLAIN of a statement from the cache.

6.3 Long-running non-committing readers

In an SAP system, it sometimes happens that a long-running read-only batch program never commits. The corresponding DB2 thread does not do any updates to the database, but it still holds some S-locks (share locks). If the batch program issues an SQL statement that needs to be prepared, S-locks on the catalog pages for prepares are required. For each table space, the batch program uses an intent lock. If the batch program reads with the isolation level CS or RS, S-locks on data rows are required. These locks reduce overall concurrency and, more importantly, the active claim (due to non-committing) blocks the online Reorg switch phase (this may take hours) from establishing its drain.

It is important to be able to identify such programs and inform the DB2 administrator so that he can react accordingly and allow DB2 online Reorgs to run smoothly. Very often, these long-running non-committing batch programs are the problem. DB2 V7 is already able to identify long-running units of recovery (such as non-committing, updating batch programs). With this enhancement, DB2 V8 is also now able to identify long-running non-committing readers. Now all information is available to help the DB2 administrator to decide when to run a DB2 Reorg.

If the long-running ZPARM LRDRTHLD option is enabled, DB2 issues IFCID 313 records which allow you to identify long-running non-committing readers. SAP shows this information in the transaction DB2. The parameter is set by the LONG-RUNNING READER THRESHOLD option in the DSNTIPE install panel. Example 6-4 shows an example of the transaction DB2.

SSID	Date	Time	Application Server	WWP ID	WWP type	WWP #	Alert type	Log records	Check
\$GF2	06.10.2003	16:20:25	IHLS08	17579	DIA	0	LONG READER	0	

Figure 6-4 SAP transaction DB2 showing long-running not-committing readers

6.4 Lock escalation alert

In this section we introduce the new IFCID 337 record that reports lock escalations. We also describe why this new record is useful in SAP environments and how it can be exploited in database administration.

6.4.1 DB2 lock management for SAP applications

Some SAP applications tend to accumulate a large number of locks, because they perform massive updates or deletes and do not commit often. To accommodate for this, SAP gives specific recommendations on different DB2 parameters that control DB2 locking behavior, for example, NUMLKUS and IRLMRWT. They are summarized in *SAP on IBM DB2 UDB for OS/390 and z/OS: Database Administration Guide — SAP Web Application Server*.

One of the tuning recommendations is to allow lock escalating by setting LOCKMAX to 1000000. The DB2 ZPARAM NUMLKTS specifies the maximum number of locks that a single DB2 thread can hold on a table space. At table space level, this parameter can be overridden by the LOCKMAX clause of CREATE TABLESPACE or ALTER TABLESPACE. This eases some difficulties that are caused by a large number of locks being held, such as increased CPU consumption or hitting the limit on the number of locks that a single DB2 thread can hold (ZPARAM NUMLKUS), which terminates the offending DB2 transaction.

However, a lock escalation prevents access to the table space or partition for which the row level locks were escalated until the gross-lock holding transaction commits or rolls back. This limits overall concurrency and might cause deadlocks. Therefore, it should occur only sporadically.

If you can identify the applications that cause lock escalation, you can implement remedial actions. These actions allow you to either solve the underlying problem or to fine-tune the settings to prevent lock escalations in the future. They include:

- ▶ Increasing the commit frequency
- ▶ Determining an optimal value of LOCKMAX that takes the particular workload into consideration

6.4.2 IFCID 337 reports lock escalations

DB2 V8 introduces IFCID 337, which alerts you when lock escalations occur. IFCID 337 is part of Statistics trace class 3 and Performance trace class 6. Before V8, database administrators had to rely on DB2 message **DSNI031I**, which reports lock escalations in the z/OS log. When analyzing performance problems, it is better to capture all DB2 performance related information in a single place, though. By using a DB2 performance monitoring tool that supports IFCID 337 as part of its exception processing apparatus, you can now also monitor lock escalations through online performance monitors.

The information in IFCID 337 includes the following details:

- ▶ Object on which lock escalation was triggered
- ▶ Lock state after escalation
- ▶ Number of lower-level locks that were released by escalation
- ▶ DSC statement ID of the statement that triggered lock escalation
- ▶ DB2 client information (correlation ID, workstation name, transaction name, client user ID)

The statement ID from the dynamic statement cache is crucial in analyzing lock escalations, because it allows you to identify both the escalating SQL statement and the ABAP code location that originates the statement. Using the statement ID, IFCID 317 can be used to get hold of the complete statement text. The IFCID 316 record on cached statements contains field QW0316US that hosts a client provided identification string. SAP assigns to this field the ABAP report name and the position within the report where the SQL statement of the IFCID 316 record was initially prepared. Hence, IFCID 337 enables to pinpoint the ABAP source code position of the statement that caused lock escalation. This in turn allows you to assess whether the commit frequency of the report can be enhanced without impairing data integrity. It might also help in finding an optimal LOCKMAX value for the involved table spaces or table space partitions.

6.5 SAP and DB2 Control Center

DB2 Universal Database (UDB) Control Center is a database administration tool that you can use to administer your DB2 Universal Database environment, which includes DB2 UDB for z/OS. DB2 UDB Control Center is an integrated part of DB2 UDB for Linux, UNIX, and Windows. It ships a set of stored procedures that must be installed at each DB2 UDB for z/OS subsystems that you want to work with using DB2 UDB Control Center. DB2 UDB Control Center and its stored procedures are included in the DB2 Management Clients package, a no-charge feature of DB2 UDB for z/OS. In this section we introduce new features of DB2 UDB Control Center that were recently added and that are beneficial to SAP applications. The new features are mainly stored procedures that are invoked from within the SAP CCMS functionality. They apply to general database administration tasks though and could be integrated in any application.

Besides the stored procedures that are described in more detail in this section, DB2 Control Center also introduced the cloning wizard. The Control Center cloning wizard assists in performing system clones of entire DB2 subsystems, for example, to create a quality assurance system out of a production system. The cloning wizard is described in the redbook *SAP on DB2 for OS/390 and z/OS: Multiple Components in One Database (MCOD)*, SG24-6914.

DB2 V8 also includes new and enhanced stored procedures that aim at facilitating database administration. The stored procedure DSNUTILU executes DB2 utilities and accepts input parameters in Unicode. DSNACCOR is enhanced to provide better recommendations on the objects that require maintenance.

Attention: DB2 V8 deprecates the support for DB2-managed stored procedures.

In this section we describe the following topics:

- ▶ Stored procedures and batch programs for database administration
- ▶ Stored procedures for data set manipulation
- ▶ Stored procedures for submitting JCL jobs and UNIX commands

6.5.1 Stored procedures and batch programs for database administration

Control Center has been enhanced to cope with the requirements that are posed by SAP systems, such as the ability to handle a large number of database objects.

The following new Control Center stored procedures and batch programs deal with DB2 database administration. They generally support DB2 V6, V7, and V8.

Parallel utility execution stored procedure (DSNACCMO)

The stored procedure DSNACCMO allows you to invoke a utility on multiple objects in parallel. It is able to determine the optimal parallel degree that performs the given task in the shortest overall utility execution time. The maximum parallel degree can be restricted by the means of a stored procedure parameter. DSNACCMO fits SAP very well, because a common task at SAP installations is to process a large number of objects. It facilitates this exercise tremendously and automatically optimizes its execution. DSNACCMO is well-suited to be exploited by the SAP-integrated database administration tool (SAP transaction code DB13).

Parallel utility batch program (DSNACCMB)

The batch program DSNACCMB is the batch interface of the stored procedure DSNACCMO. It provides similar functions to the stored procedure DSNACCMO.

Object maintenance batch program (DSNACCOB)

The batch program DSNACCOB automates object maintenance by running utilities like COPY, MODIFY RECOVERY, REORG, RUNSTATS MODIFY STATISTICS, and STOSPACE on objects for which maintenance actions are recommended. Similar to the stored procedure DSNACCOR, which is shipped as part of DB2, DSNACCOB gives policy-based recommendations that are deduced from DB2's real-time statistics. In contrast to DSNACCOR, it also exploits the recommendations and runs utilities on the recommended objects. APAR PQ75973 provides DSNACCOB for DB2 V7 and DB2 V8.

DB2 command submission stored procedure (DSNACCMD)

The stored procedure DSNACCMD executes DB2 commands on the connected DB2 subsystem. This stored procedure appeals to SAP, because it lessens the dependency on the external program `rfcosco1`. SAP systems use the `rfcosco1` executable to issue DB2 commands by means of IFI. If `rfcosco1` is not operational, DB2 commands cannot be submitted from the SAP system.

Other stored procedures

SAP applications can exploit the stored procedure DSNACCSS to query the SSID of the DB2 subsystem to which it is currently connected. They may use DSNACCSI to find out the host name of the connected DB2 subsystem. This information is relevant to SAP instances that exploit DB2 data sharing.

6.5.2 Stored procedures for data set manipulation

Control Center also provides stored procedures that enable applications to remotely manipulate z/OS data sets. SAP applications accomplish data set manipulation as part of the CCMS functionality and during system installation or upgrade. The stored procedures that manipulate data sets are described in the following sections. They generally support DB2 V6, V7, and V8.

Stored procedure to create or write to a data set (DSNACCDS)

The stored procedure DSNACCDS creates a data set member or a data set of type PS (physical sequential data set), PDS (partitioned data set) and PDSE (partitioned data set extended) and writes data to it. It either appends or replaces an existing data set or data set member. Moreover, it creates a new GDS (generation data set) for an existing GDG (generation data group). The amount of space that it allocates is minimal. Due to a JCL limitation, DSNACCDS only creates or writes to data sets with LRECL set to 80.

Stored procedure to rename a data set (DSNACCDR)

The stored procedure DSNACCDR renames a data set of type PS, PDS or PDSE, or a data set member.

Stored procedure to delete a data set (DSNACCDD)

The stored procedure DSNACCDD deletes a data set member, or a data set of type PS, PDS or PDSE, or a GDS.

Stored procedure to check if data set exists (DSNACCDE)

The stored procedure DSNACCDE checks if a non-VSAM data set exists. It accomplishes this by querying the ICF (Integrated Catalog Facility) catalog of z/OS. Also, it checks if a data set member exists.

Stored procedure to query data set properties (DSNACCDL)

The stored procedure DSNACCDL lists data sets, data set members, VSAM clusters, generation data sets or GDGs. It provides attributes on these objects, such as the primary extent size. This stored procedure allows SAP to stop using IDCAMS LISTCAT jobs to find out data set sizes and properties. This information is displayed in the SAP tables and indexes space monitor (SAP transaction code DB02). SAP systems today rely on submitting IDCAMS LISTCAT jobs through FTP. They parse the output of these jobs to determine the relevant information, which is a cumbersome approach that is also complex to set up. DSNACCDL yields the following data set attributes:

- ▶ Creation date
- ▶ Type of data set
- ▶ Volume name
- ▶ Primary extent size
- ▶ Secondary extent size
- ▶ Unit (track, block, cylinder)
- ▶ Extents in use
- ▶ Actual disk usage in bytes

For VSAM data sets, it additionally provides the high-allocated RBA and the high-used RBA.

6.5.3 Stored procedures for submitting JCL jobs and UNIX commands

The Control Center stored procedures introduced so far are well-suited to replace database administration tasks that SAP currently realizes by generating JCL jobs and submitting them by means of FTP. They are optimized, easier to handle, and can be directly called by simple SQL calls. However, it is advantageous to completely abolish FTP-based JCL submission, which is cumbersome. Moreover, SAP today allows you to issue UNIX commands in UNIX System Services by submitting a JCL job that in turn invokes the UNIX command. This fairly indirect and error-prone way of issuing UNIX commands should also be replaced with a more elegant solution. Each level of indirection that is removed enhances quality and results in faster processing.

Control Center offers stored procedures that allow you to submit JCL jobs and to issue UNIX commands. They support DB2 V8 only. This enables SAP to dispose of FTP. Figure 6-5 visualizes the general concept of these stored procedures.

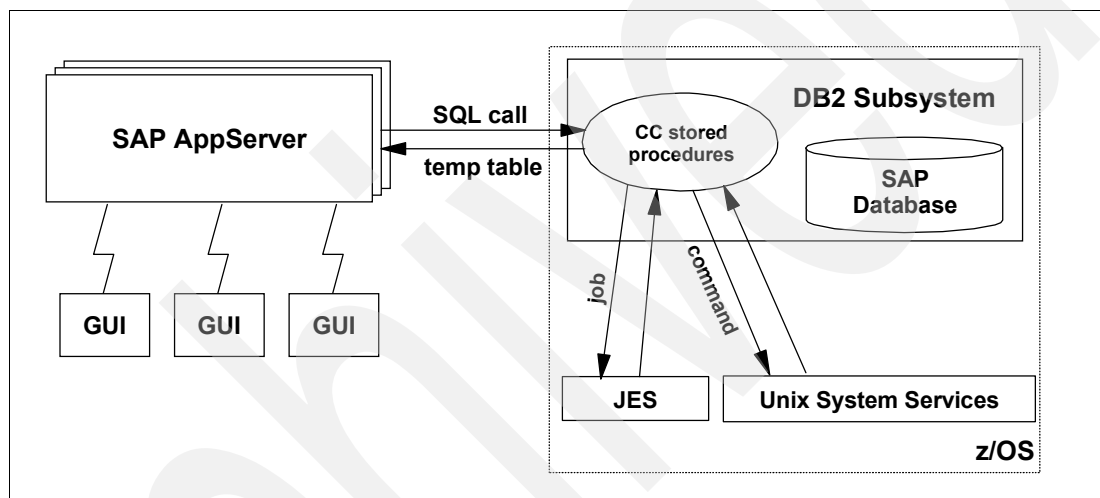


Figure 6-5 Submitting JCL and UNIX commands through Control Center stored procedure

Stored procedure to submit a JCL job (DSNACCJS)

The stored procedure DSNACCJS submits a JCL job for batch execution. It returns the ID that JES assigns to the job. The text of the job is passed to DSNACCJS through a global temporary table.

Stored procedure to fetch output of a JCL job (DSNACCJF)

The stored procedure DSNACCJF fetches the spool output files of a specified JCL job. It inserts the lines of the output listings into a result set table.

Stored procedure to cancel a JCL job (DSNACCJP)

The stored procedure DSNACCJP purges or cancels a specified job. It returns related console messages.

Stored procedure to retrieve JCL job status (DSNACCJQ)

The stored procedure DSNACCJQ allows you to inquire the status of a JCL job. Depending on the requested mode, it returns related console messages or a status value. The returned status values indicate if a job is currently in the input queue, is active or is in the output queue.

Stored procedure to issue a UNIX command (DSNACCUC)

Finally, the stored procedure DSNACCUC issues UNIX commands in UNIX System Services. The commands are executed in the UNIX environment of the user that DSNACCUC runs under. It returns the output in a result set table.

6.6 SAP transaction-based DB2 accounting and workload management

In this section we describe the DB2 V8 enhancements that allow SAP to provide DB2 accounting and workload management on the granularity of individual SAP transactions and SAP end users. As other enterprise packaged applications, SAP anonymizes the identity of its individual programs and end users to DB2. DB2 performs all requests from SAP under the same user ID. Hence, the work that is performed within DB2 is hard to be differentiated.

Before DB2 V8, SAP already made use of the DB2 client identifiers like workstation name and transaction name to identify the static properties of an SAP system. These identifiers are set when SAP establishes connections; this is accomplished when SAP creates work processes, and the connections are not changed during the lifetime of SAP work processes and DB2 threads. Among the characteristics that SAP passes to DB2 are application server name and work process type such as dialog, batch or spool. These characteristics are exploited to analyze performance problems — most of the DB2 client identifiers are part of the IFC correlation header — and to ensure basic workload management (WLM) support. For example, WLM can be exploited to assign different priorities to dialog work and to batch work.

DB2 V8 provides significant enhancements in the area of accounting and support for workload management. Briefly, these enhancements allow SAP to support DB2 accounting and workload management at the granularity of individual SAP transactions, reports, batch jobs and end users. This enables IT departments to charge back the costs that a specific department generated in DB2 to this department. Moreover, by means of WLM a higher priority can be assigned to an important SAP transaction or batch job.

This section contains the following topics:

- ▶ Rollup accounting data for DDF and RRSAF
- ▶ Writing accounting records with KEEP DYNAMIC(YES)
- ▶ WLM support for DB2 client identifiers end user ID and workstation name
- ▶ Lock waiter inherits WLM priority by lock holder
- ▶ New DDF terminology
- ▶ New DDF-related default values
- ▶ RRSAF function SET_CLIENT_ID
- ▶ Automatic dropping of declared temporary tables
- ▶ Opportunity for SAP

6.6.1 Rollup accounting data for DDF and RRSAF

When using inactive DDF connections, DB2 cuts an accounting record on every COMMIT or ROLLBACK (6.6.3, “Other DDF and RRSAF enhancements” on page 173 describes the new term *inactive connection*). In high volume OLTP environments, this may generate a large number of accounting records. This can become a problem that compromises the ability to do charge-back or performance monitoring and tuning. RRS Attach may encounter similar problems if a workload generates many accounting records.

In V8, the collection of DB2 accounting data is enhanced to optionally accumulate accounting data for DDF and RRSAF threads. This reduces the amount of accounting records that are externalized, which improves overall system throughput.

A new installation option is added to activate the new behavior. The DB2 installation tracing panel DSNTIPN contains a new field “DDF/RRSAF ACCUM” (see Figure 6-6). This sets the new DSNZPARM parameter ACCUMACC.

```

          DSNTIPN          INSTALL DB2 - TRACING PARAMETERS
====>

Enter data below:

1  AUDIT TRACE           ====> NO  Audit classes to start. NO,YES,list
2  TRACE AUTO START     ====> NO  Global classes to start. YES,NO,list
3  TRACE SIZE           ====> 64K Trace table size in bytes. 4K-396K
4  SMF ACCOUNTING       ====> 1   Accounting classes to start. NO,YES,list
5  SMF STATISTICS       ====> YES  Statistics classes to start. NO,YES,list
6  STATISTICS TIME      ====> 30  Time interval in minutes. 1-1440
7  STATISTICS SYNC      ====> NO  Synchronization within the hour. NO,0-59
8  DATASET STATS TIME   ====> 5   Time interval in minutes. 1-1440
9  MONITOR TRACE        ====> NO  Monitor classes to start. NO,YES,list
10 MONITOR SIZE         ====> 8K  Default monitor buffer size. 8K-1M
11 UNICODE IFCIDS       ====> NO  Include UNICODE data when writing IFCIDS
12 DDF/RRSAF ACCUM     ====> 10  Rollup acctg for DDF/RRSAF. NO,2-64K
13 AGGREGATION FIELDS ====> 0   Rollup acctg aggregation fields

PRESS:  ENTER to continue  RETURN to exit  HELP for more information

```

Figure 6-6 DB2 installation panel DSNTIPN

The new field specifies whether DB2 accounting data should be accumulated by a concatenated client identifier for DDF and RRSAF. If a value between 2 and 65535 is specified, then DB2 writes an accounting record every 'n' occurrences of the concatenated client identifier, where 'n' is the number specified for this parameter. The default value is 10. A concatenated client identifier is identified by the concatenation of the following three fields:

- ▶ End user ID (IFCID field QWHCEUID, 16 bytes)
- ▶ Transaction name (IFCID field QWHCEUTX, 32 bytes)
- ▶ Workstation name (IFCID field QWHCEUWN, 18 bytes)

To ensure maximum rollup flexibility, the new DB2 ZPARM ACCUMUID allows you to specify a subset of these fields. On panel DSNTIPN this parameter is called “AGGREGATION FIELDS” (see Figure 6-1). The fields that do not belong to the specified subset are ignored during the rollup process. For example, if ACCUMUID is set to rollup accounting data with respect to end user ID and transaction name, and two accounting records differ in workstation name, they end up in the same accounting bucket.

If data accumulation is activated, then, when a database transaction commits or rolls back, instead of immediately writing an accounting record at that time, DB2 adds the accounting values to that bucket in virtual storage that accumulates the accounting data of the transaction’s concatenated client identifier. If there is no bucket yet for this concatenated client identifier, then a new bucket is created. Separate rollup buckets are created for DDF AND RRSAF.

Note: The correlation header of an accounting record rollup block (IFI record QWHC) reports the last thread that added accounting data to this rollup block. This information may be valuable in some cases, for example if all accounting records of a rollup block were generated for a single batch job.

There are certain cases where detailed accounting data for the DDF and RRSF threads is desired, such as detailed performance monitoring. Both ACCUMACC and ACCUMUID can be dynamically altered to activate or deactivate accounting data accumulation on the fly.

DB2 externalizes the accumulated accounting data of a concatenated client identifier on any of the following events:

- ▶ The number of occurrences for this concatenated client identifier reaches the threshold as specified by ACCUMACC.
- ▶ An internal threshold on the number of accounting buckets is reached.
- ▶ An internal threshold on the timestamp of the last update to the bucket is reached.
- ▶ Accounting interval is set to commit for RRSF threads.

The DB2 client identifiers that are the basis of accounting data rollup can be altered at transaction boundaries. CLI clients employ the routine `sqlreseti` to establish values for these identifiers. Like CLI, the DB2 Java Universal Driver implements the DRDA protocol. Therefore, DB2 treats CLI clients and Java Universal Driver clients uniformly. The Universal Driver extends the Connection interface from the JDBC standard to also provide the following methods, which set client identifier values:

- ▶ `DB2Connection.setDB2ClientUser(String user)`
- ▶ `DB2Connection.setDB2ClientWorkstation(String name)`
- ▶ `DB2Connection.setDB2ClientApplicationInformation(String info)`

RRSAF allows you to set client identifiers by employing the functions SIGNON, AUTH_SIGNON and CONTEXT_SIGNON at connection boundaries and SET_CLIENT_ID at transaction boundaries.

6.6.2 Writing accounting records with KEEP_DYNAMIC(YES)

If the DB2 ZPARM CMTSTAT is set to INACTIVE (new default for DB2 V8), DB2 writes an accounting record when a transaction completes and the thread qualifies to become inactive. Using KEEP_DYNAMIC(YES) as bind option keeps DDF threads always active. As a consequence accounting records were not cut at transaction boundaries. Likewise, DDF did not reestablish WLM enclaves at transaction boundaries. As SAP exploits dynamic statement caching, it is desirable that the bind option KEEP_DYNAMIC(YES) does not interfere with the collection of accounting data with transaction scope.

DB2 V8 tolerates KEEP_DYNAMIC(YES) by cutting accounting records at transaction boundaries. although KEEP_DYNAMIC(YES) still prevents DDF threads from becoming inactive. That is, a DDF thread pretends to be eligible for inactivation if the only reason why it cannot become inactive is the presence of cached dynamic SQL statements due to KEEP_DYNAMIC(YES). DDF then writes accounting records and completes the WLM enclave as if KEEP_DYNAMIC(YES) is not specified. When a new request arrives from the client system, a new enclave is created and a new accounting interval is started.

This new behavior is supported for DRDA clients. DB2 for z/OS clients that use the DB2 private protocol are not affected by this change. As in V7, the presence of held cursors or

A related enhancement is that DB2 V8 exploits the full length of 143 bytes of the WLM qualifier accounting information (AI) for its accounting string.

Lock waiter inherits WLM priority by lock holder

Assume that a transaction executes with a low WLM priority and makes updates to the database. This means that it must acquire X-locks on the pages that it modifies and that these X-locks are held until the transaction reaches a commit point. If such a transaction acquires an X-lock on a row that another transaction is interested in, this transaction is forced to wait until the first transaction commits. If the lock waiter performs very important work and is thus assigned a high WLM priority, it would be desirable if it is not slowed down by other transactions that execute in a low priority service class. It would be better if the waiting transaction could temporarily assign its own priority to the transaction that holds the lock until this transaction frees the locked resource.

The WLM component of z/OS 1.4 provides a set of APIs that can be used to accomplish this. This service is called WLM enqueue management. In z/OS 1.4, this support is limited to transactions running on a single system.

DB2 V8 exploits WLM enqueue management. When a transaction has spent roughly half of the lock timeout value waiting for a lock, then the WLM priority of the transaction, which holds the lock, is increased to the priority of the lock waiter if the latter priority is higher. If the lock holding transaction completes, it resumes its original service class. In case multiple transactions hold a common lock, this procedure is applied to all of these transactions.

New default values for DDF and RRSF related ZPARMs

There are new default values for some installation parameters in DB2 V8 that are related to DDF and RRSF. Table 6-1 lists the new default values.

Table 6-1 New default values for DDF and RRSF related ZPARM

DB2 ZPARM	Description	New default	Old default
MAXDBAT	Max. number of concurrent DBATs	200	64
CONDBAT	Max. number of DDF connections	10000	64
CMTSTAT	Inactivate thread after commit or rollback	INACTIVE	ACTIVE
IDTHTOIN	Idle time of active thread before it is canceled	120	0
CTHREAD	Max. number of concurrent allied threads	200	64
IDBACK	Max. number of batch connections	50	20

Add DDF accounting string to RRSF

As part of the V7 maintenance stream, DB2 introduced the RRSF function SET_CLIENT_ID. SET_CLIENT_ID sets client identifiers that are passed to DB2 when the next SQL request is processed. It enables an inexpensive way of producing accounting data with transaction scope. In V7 it allows RRSF applications to set the fields accounting token, end user ID, transaction name, and workstation name.

Tip: APAR PQ67681 makes SET_CLIENT_ID available in DB2 V7.

DB2 V8 adds the accounting string to the set of client identifiers that can be exploited using SET_CLIENT_ID and the RRSF signon routines. It is placed in the DDF accounting trace records in IFI field QMDASQLI. This ensures consistent behavior across local and remote clients. It is optional to set the accounting string. If it is not specified, no accounting string is associated with the connection.

6.6.4 Automatic dropping of declared temporary tables

If an application creates a declared global temporary table to temporarily store data, it is up to the application to ensure that this table is dropped when it is not needed anymore. Otherwise it survives until the end of the application process.

Furthermore, stored procedures employ declared temporary tables to pass result sets to the calling application. Therefore, the declared temporary tables cannot be dropped by the stored procedure. On the other hand, the calling application does not need to know the name of the declared temporary table to fetch the passed result sets. If a declared temporary table persists when a transaction completes, DDF is not able to write an accounting record.

DB2 V8 provides relief in this area. It allows you to automatically drop declared temporary tables on commit. The DDL statement DECLARE GLOBAL TEMPORARY TABLE now supports the new clause ON COMMIT DROP TABLE, which is mutually exclusive with ON COMMIT DELETE ROWS and ON COMMIT PRESERVE ROWS. If it is employed to declare a temporary table, the table is implicitly dropped at commit and does not prevent accounting records to be written.

6.6.5 Opportunity for SAP

SAP employs the bind option KEEP DYNAMIC(YES) to fully benefit from dynamic statement caching. As the work dispatcher within the SAP application server pools the work requests from end users and distributes them across the invariable number of work processes, SAP does not benefit from connection pooling at the database level. Therefore, it always uses DB2 threads.

Before DB2 V8, this behavior of SAP makes DB2 generate only a single accounting record when an SAP work process finally terminates the connection to DB2, which allows the DB2 thread to terminate. This accounting record contains information for thousands of separate transactions and is therefore useless. Also, the priority of the WLM enclave that hosts the DB2 thread is fixed for the lifetime of the thread. Accounting intervals need to contain single transactions to provide better granularity, which allows a more accurate monitoring of transaction activity.

By running with inactive thread support, which is the default behavior in DB2 V8, SAP is enabled to exploit accounting data and workload management capabilities at the granularity of individual database transactions. Since SAP still relies on dynamic statement caching, the DB2 threads that serve SAP systems remain always active. Hence, there is still a one-to-one relationship between SAP work process and DB2 thread. DB2 V8 is capable of writing accounting records at commit or rollback if KEEP DYNAMIC(YES) SQL sections are present; transaction level accounting data and SAP are no longer mutually exclusive.

To prevent potentially large volumes of accounting records, DB2 offers the accumulation of accounting data. As long as there are no held cursors or declared temporary tables active, accounting intervals are completed when transactions complete. To control the life span of

declared temporary tables, DB2 equips application and stored procedure developers with implicit dropping of declared temporary tables at transaction end. Held cursors interconnect related database transactions. Hence, it should not pose a problem if such transactions share an accounting interval.

SAP can pass fairly granular information that is the basis of the generated accounting data and the basis of workload management to DB2. The following SAP identifiers appear to be valuable:

- ▶ The name of the end user that called an SAP transaction code or ABAP program
- ▶ The name of the ABAP program that issues the SQL requests of a database transaction
- ▶ The SAP transaction code that the end user called
- ▶ The name of the SAP batch job that issues the SQL requests of a database transaction

If SAP passes these identifiers to DB2 at transaction boundaries, then accounting and workload management can be accomplished based on SAP identifiers.

System point in time back-up and recovery

DB2 V8 provides enhanced backup and recover capabilities at the DB2 subsystem or data sharing group level. The purpose is to provide an easier and less disruptive way to make fast volume level backups of an entire DB2 subsystem or data sharing group with minimal disruption, and recover a subsystem or data sharing group to any point in time, regardless of whether you have uncommitted units of work. SAP exploits these new recovery functions.

Two new utilities provide the vehicle for system level point in time recovery (PITR):

- ▶ The *BACKUP SYSTEM* utility provides fast volume-level copies of DB2 databases and logs.
- ▶ The *RESTORE SYSTEM* utility recovers a DB2 system to an arbitrary point in time. *RESTORE SYSTEM* automatically handles any creates, drops, and LOG NO events that might have occurred between the backup and the recovery point in time.

As a further enhancement to taking system-level backups, the SET LOG SUSPEND command now quiesces 32 KB page writes (for page sets that are not defined with 32 KB CI size) and data set extensions avoiding integrity exposures.

The *BACKUP SYSTEM* and *RESTORE SYSTEM* utilities rely on new DFSMSHsm™ services, and SMS constructs in z/OS V1R5 that automatically keep track of which volumes need to be copied.

7.1 The need for PITR

In this section we discuss some of the fundamental requirements of an SAP system that drive backup and recovery requirements that are challenging to address, and sometimes quite different from other DB2 applications.

- ▶ We discuss the concept of the entire database as a unit of consistency.
- ▶ We outline some scenarios that can occur in an SAP production system that drive different recovery requirements.
- ▶ We discuss what many customers have put into place prior to DB2 V8, and how the new features impact existing and new SAP customers on this platform.

7.1.1 SAP application fundamentals

SAP is designed to run on a variety of DBMS and operating system platforms. This requires the application itself to perform tasks over and above the lowest common denominator of the platforms supported. In some cases these tasks overlap with features available in some DBMS platforms.

Backup and recovery implications

One of the fundamental designs of the SAP application is the concept of Referential Integrity (RI) being handled exclusively by the application software, as opposed to being handled by the DBMS. RI is, broadly speaking, the logical relationship of the content of tables in the system, such as checking if a record actually exists in another table with a foreign key relationship. The ultimate result of this design in DB2 terms is that the entire DB2 subsystem, including all objects in it, are considered as one single entity in terms of recovery purposes. This is particularly the case for what is called prior point in time recovery, or PPIT, as all database objects must simultaneously be recovered to the exact point of time chosen in the past to guarantee consistency.

Disruptions to users, online or batch, during any process fosters the perception that the system does not perform well, and the requirement for near 7x24 availability makes it desirable to have a fast, non-disruptive backup methodology. Currently, if the data integrity of an SAP system is compromised, for whatever reason, the time and effort required to recover the entire subsystem or data sharing group to a point in time prior to the corruption is prohibitive. This demands a fast, reliable process to recover a subsystem or data sharing group to an arbitrary point in time.

Operational reality in an SAP environment

In general terms, backup and recovery requirements can be explained in the two general categories described in Figure 7-1. The stark reality of an SAP implementation is that the most likely recovery scenario will be driven by causes not related to DB2 for z/OS or the underlying technical platform. Because of the high degree of integration of the SAP application, procedural or operational errors may result in the desire to “back out” the actions in question. Often there is no corresponding function available to do this automatically, and it quickly can become a problem that is passed to the DBA and SAP system administrators to resolve. In many of these cases, however, the solution is difficult, as DB2 has not reported any error, and, as it has simply performed the workload imposed upon it, sees the database and its contents as being entirely consistent and error free.

In many cases other activities occurred at the same time as the action that causes the error, and it is desired that these activities are preserved in the system. It is in this case that the highly integrated nature of SAP, and the presence of effective RI rules not being known to the DBMS, becomes the second side of a two-edged sword.

<u>Technical failure</u>	<u>Application failure</u>
<ul style="list-style-type: none"> ● Typically failure caused by hardware or system software fault. ● Causes Include: <ul style="list-style-type: none"> ■ Media failure ■ System or subsystem crash ■ Software bug 	<ul style="list-style-type: none"> ● All DATA is consistent from DB2 and “SAP Basis” viewpoint. ● Causes include: <ul style="list-style-type: none"> ■ Transport induced error ■ Introduction and execution of bad code ■ Incorrect usage of application component ■ Deleting data through SAP actions causing DB2 drops

Figure 7-1 Breakdown of major causes of recovery actions

To clarify the failure type definitions, it is useful to look at which recovery scenarios will be used most commonly for the situation. The information in Figure 7-2 indicates that there are two major courses of action:

- ▶ For a *technical failure*, the most common action is to recover to current. This action may be achieved through a number of different methods. The recovery to a prior time as a results of technical failure generally happens to achieve a faster recovery. It may also happen if recovery to current for objects is not possible, due to missing Image Copies, for instance.
- ▶ For an *application failure*, recovery methods available to the DBA and SAP Basis staff are generally limited to recovery of everything to a prior point in time. The most effective solution to the problem is to run a “Compensating Unit of Work”, if possible. In the case of transport induced errors, it may be possible to resolve the problem through importing transports of the correct versions of the object in question, again if possible. In the absence of “programmatic” solutions to the problem, recovery to a prior point in time may be considered.
- ▶ If recovery to a prior point in time is requested for an application failure, it is important to consider that all work subsequent to the chosen point in time will be regressed due to the recovery.

Recovering selected objects instead of the entire database is something that can only be done in extreme circumstances, and requires a full knowledge of the requirements and consequences. This will typically happen in conjunction with appropriate assistance from SAP support staff.

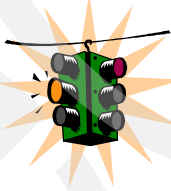
	Application Failure	Technical Failure
Recover all database objects to current point in time	N/A By definition, problem still exists	YES Most common action following this failure.
Recover all database objects to a previous point in time	YES Most common action following this failure.	Possible If nature of technical failure requires this action for timeliness
Recover selected objects to prior time	By recovering objects to different points in time, this violates the DB as one unit of recovery.	

Figure 7-2 Recovery options for different failure types

7.1.2 Past, present, and future for backup and recovery

Prior to DB2 V8, it is necessary to back up an SAP system with a combination of the DB2 COPY utility and some form of volume dump methodology. The DB2 COPY is taken primarily as a means of correcting errors with one or a limited set of database objects, and most commonly this recovery takes place to 'CURRENT'. Managing thousands of image copies is complex, but required to allow the ability to recover individual objects, as opposed to the entire database.

The volume dump methodology prior to DB2 V8 required use of the SET LOG SUSPEND command to stop all write activity while the copy relationship was being established, causing a disruption in service. After the copy was complete, the data had to be dumped off to tape so that it could be available at some point in the future in case recovery was required.

Recovering a system meant restoring the volume backups from tape to the DB2 source data volumes if a volume dump strategy was employed, identifying which objects had been changed with some kind of log scanning process, creating recovery JCL for all objects identified as requiring recovery, and recovering each object to the same point in time. At best this process was time consuming, labor intensive, and very prone to error.

DB2 V8 uses DFSMSHsm functionality to simplify and improve the performance and reliability of the backup and recovery process. The BACKUP SYSTEM and RESTORE SYSTEM utilities encapsulate all tasks previously required to perform each function into one utility statement each. The new Copy Poolconstruct and Copy Poolbackup storage group types in DFSMSHsm V1R5 make use of fast replication support. A full system Copy Poolbackup can also be used for cloning and disaster recovery.

7.2 DFSMShsm fast replication support

The DB2 V8 BACKUP SYSTEM and RESTORE SYSTEM utilities work with DFSMShsm to take advantage of the new fast replication support in z/OS V1R5. DFSMShsm introduces the new Copy Pool construct and SMS Copy Pool backup storage group type in support of this new functionality. DFSMShsm manages the fast replication backups, and recovery can be performed at the volume or Copy Pool level but not at the data set level. The following requirements must be met to take advantage of this new functionality:

- ▶ z/OS V1R5 or above
- ▶ SMS managed DB2 data sets
- ▶ Disk control units that support ESS FlashCopy
- ▶ Copy Pools for data and logs
- ▶ Backup storage groups for each source storage group in a Copy Pool

In this section we describe the new fast replication support with DFSMShsm V1R5 that works with DB2 V8 to provide for a simple, fast, and reliable mechanism to create system level backups and provide for system level point in time recovery. We describe the new DFSMShsm Copy Pool and SMS Copy Pool target backup storage group type and the commands used to make this functionality possible.

7.2.1 The Copy Pool construct

A Copy Pool is a set of SMS pool storage groups that can be processed by fast replication operations as one unit with one command. For DB2, only two types of Copy Pools are valid:

- ▶ A database Copy Pool, containing all database objects, including DB2 catalog and directory, and all user data
- ▶ A log Copy Pool, containing all active logs and BSDS data sets.

The ISMF has been enhanced to support these SMS enhancements.

A Copy Pool can contain up to 256 pool storage groups to be processed for fast replication operations, and each pool storage group must be associated with a new type of storage group called the Copy Pool Backup Storage Group. A pool storage group can have only one Copy Pool backup storage group associated, and many pool storage groups can be associated with the same Copy Pool backup storage group. So in a Copy Pool backup storage group, we can have different versions of different pool storage groups all together. The BSDS keeps control of the copies we have. Each Copy Pool has a VERSIONS attribute that specifies how many versions should be maintained on disk, with a default of 2 and a maximum of 85. In the BSDS, only up to 50 versions are allowed.

Volumes to be copied are evaluated at processing time rather than at definition time so that changes to the Copy Pool after definition are reflected in future processing. The Copy Pool backup storage group must contain enough volumes for a unique one to one relationship with the volumes in the pool storage group. The Copy Pools must follow a strict naming convention of the form DSN\$locn-name\$cp-type, where:

- ▶ DSN is the unique DB2 product identifier.
- ▶ \$ is a required delimiter.
- ▶ *locn-name* is the DB2 location name.
- ▶ \$ is a required delimiter.
- ▶ *cp-type* is the Copy Pool type; DB for database, LG for logs.

For example, DB2 DB1P would have Copy Pools named DSN\$DB1P\$DB and DSN\$DB1P\$LG.

The BACKUP SYSTEM utility requires that at least a database Copy Pool exists to take a data-only backup. If you plan to take full system backups (database, logs, BSDS) you must define the log Copy Pool as well. It is recommended that you create separate ICF catalogs for each Copy Pool. The RESTORE SYSTEM utility only uses the database Copy Pool.

An individual storage group may be contained in as many as 50 Copy Pools. DFSMSHsm can keep up to 85 backup versions for each Copy Pool. Keeping at least two versions is recommended because, before a new copy is created, the oldest one is invalidated and the target volumes are overwritten. If that backup fails, there is no way to recover the backup that was invalidated. It is recommended that if n backups are required, $n+1$ should be kept.

7.2.2 Copy Pool backup storage group type

The new Copy Pool backup storage group is used to contain backup target volumes for DFSMSHsm fast replication operations. An eligible target volume must have the same track form as the source volume, be the exact size of the source volume, reside in the same LSS as the source volume, not be a primary or secondary volume in an XRC or PPRC volume pair, and not be in another FlashCopy relationship. The current requirement is that all backup storage group volumes be on the same LSS as the source volumes; this requirement might be lifted, depending on the 2105 model F20 and 800 controller, and the z/OS 1.5 maintenance level.

The Copy Pool backup storage group cannot be accessed by ACS routines as SMS will prevent new data set allocations to this type of storage group. There must be a sufficient number of volumes in the backup target storage group to satisfy the number of backup versions specified for a source storage group. For example, if a system has 10 source volumes and the VERSIONS attribute has been specified as 2, the backup storage group must have at least 20 volumes to satisfy 2 backup versions of 10 volumes each.

SMS provides a new storage group attribute to associate a source storage group to a backup target storage group. Notice that SMS does not verify that extend and overflow storage groups that are associated with main source pool storage groups have been included in a Copy Pool definition. They must be included in the storage group list for appropriate Copy Pools and they also must be associated to back up target storage groups.

Figure 7-3 illustrates the Copy Pool structure and the relationship between source and backup target storage groups.

CopyPool

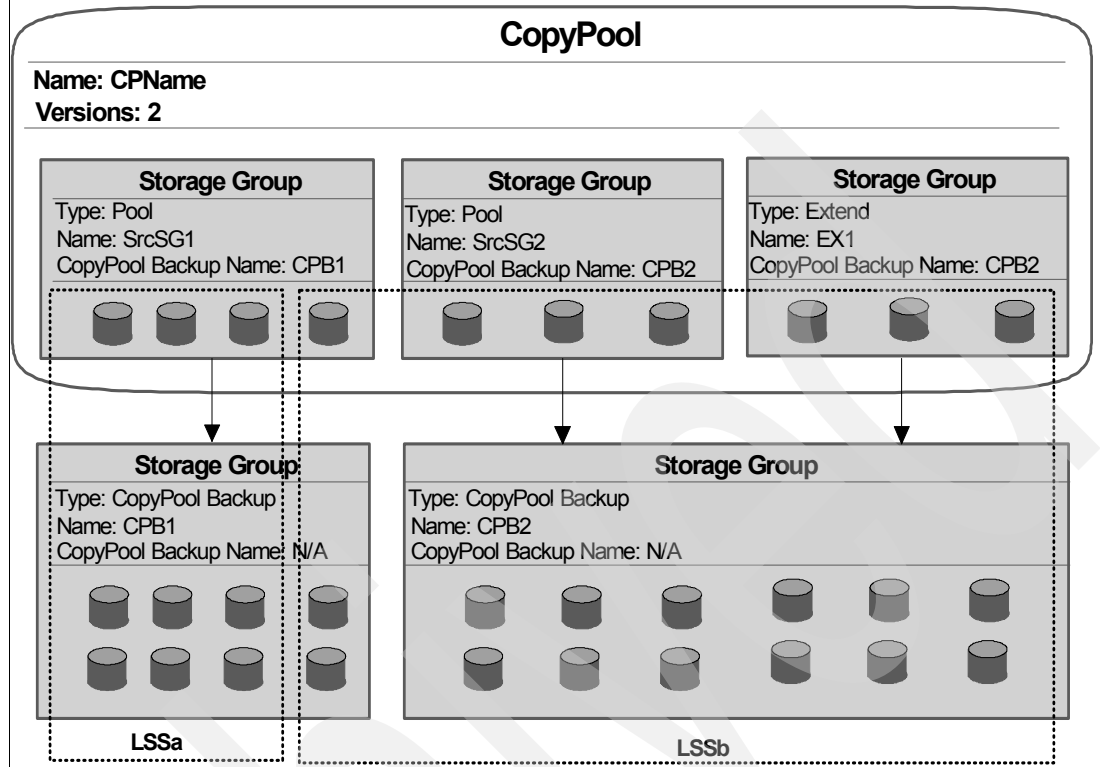


Figure 7-3 Copy Pool structure

In the illustration, the Copy Pool contains three source storage groups, two with source data (SrcSG1, SrcSG2), and an extend storage group (EX1). Two Copy Pool backup target storage groups (CPB1, CPB2) are associated with the source storage groups.

7.2.3 DFSMSHsm commands

The following DFSMSHsm commands are used to support the new fast replication operations include in z/OS V1R5.

FRBACKUP

This command is used to create a fast replication backup version for each volume in a specified Copy Pool. DB2 will take care of issuing this command and any necessary parameters under the covers when running the BACKUP SYSTEM utility. FRBACKUP PREPARE should be used to validate the fast replication environment and reduce the elapsed time of the actual Copy Pool backup.

With FRBACKUP, DFSMSHsm invokes the DFSMSdss™ COPY FULL function. The operation is considered successful after the fast replication relationship for each source volume is made with a target volume. If any relationship fails, the entire function fails, but processing continues to identify any other relationship failures. The errors are then reported, already established relationships are withdrawn, and the version is marked invalid.

An optional TOKEN parameter, which can be up to 40 bytes long, can be specified to identify the version. Use the WITHDRAW keyword to withdraw outstanding copy relationships. DFSMSHsm can process up to 64 concurrent invocations of DFSMSdss with 15 being the default. Example 7-1 shows the FRBACKUP syntax.

Example 7-1 FRBACKUP syntax

```
>>-FRBACKUP COPYPOOL(cpname)-----><
| -EXECUTE-----|
| | -TOKEN(token)-| | -NOVTOCENQ-|
| |
| -PREPARE-----|
| -WITHDRAW-----|
| -VERIFY(Y|N)-|
```

FRRECOV

This command is used to recover a target volume or pool of volumes back to the source volume or pool of volumes from the managed backup versions. DB2 will take care of issuing this command and any necessary parameters under the covers when running the RESTORE SYSTEM utility.

Use FRRECOV TOVOLUME(volser) FROMCOPYPOOL(cpname) to re-attempt a failed volume recovery. The FROMCOPYPOOL is an optional keyword that must be used if the volume being recovered resides within a storage group that is shared by multiple Copy Pools.

Use the FRRECOV COPYPOOL(cpname) VERIFY(Y) to prevent starting a recovery before an in progress backup is completed.

If a version other than the current version is to be recovered, then use either the GENERATION, VERSION, DATE or TOKEN keywords. There is no option to perform a recovery at the storage group level. Like FRBACKUP, DFSMSHsm can process up to 64 concurrent invocations of DFSMSDss with 15 being the default. Example 7-2 shows the FRRECOV syntax.

Example 7-2 FRRECOV syntax

```
>>-FRRECOV----->
| -TOVOLUME(volser)-----|
| | -FROMCOPY(cpname)-|
| |
| --COPYPOOL(cpname)-----|
| -VERIFY(Y|N)-|
>-----><
| -GENERATION(genum)-----|
| -VERSION(vernum)-----|
| -DATE(date)-----|
| -TOKEN(token)-----|
```

FRDELETE

This command is used to delete one or more unneeded backup versions. Normal processing will replace old backups with new versions. If the number of Copy Pool versions is decreased, the next time FRBACKUP COPYPOOL is issued, the unneeded versions will be deleted and the volumes will be released for reuse. You can use the VERSIONS or TOKEN keywords to identify a specific backup version to be deleted. Use the FRDELETE command to free volumes when a Copy Pool is no longer needed. Example 7-3 shows the FRDELETE syntax.

Example 7-3 FRDELETE syntax

```
>>-FRDELETE COPYPOOL(cpname)-----><
| -VERSIONS(...)-|
```

Additional commands

Use the LIST COPYPOOL command to ensure that you do not invalidate the last valid backup version which would prevent any fast replication recovery.

Use the QUERY COPYPOOL command to inquire about the status of a copy relationship. This command reports the percent of completion of a copy for each volume queried, or that there is no copy relationship.

7.2.4 Preparing for DFSMSHsm fast replication

Your storage systems administrator needs to define *database* and *log* Copy Pools and associated source and backup storage groups in order to use fast replication operations. The database Copy Pool should contain all volumes that contain the DB2 catalog and directory, and all user data. The “log” Copy Pool should contain active logs and the BSDS.

Use FRBACKUP PREPARE to validate the fast replication environment. DFSMSHsm will allocate backup versions with an empty TOKEN value and the required number of volumes for each of the versions specified in the Copy Pool definition. This TOKEN value is used by DB2 in the recovery process. In our example above, 2 versions of 10 backup target volumes each, or 20 backup volumes, would be allocated. This will ensure that there are sufficient target volumes and will move the target volume selection process outside of the fast replication window. If you don't use the PREPARE keyword prior to taking backups, only copy pool backup volumes for the backup in progress will be allocated.

The following steps should be used to define the necessary Copy Pool, source storage group, and backup storage group structures for fast replication support:

- ▶ Start to define your Copy Pool definitions by selecting option P, Specify Storage Groups for Copies, from the ISMF Primary Option Menu as shown in Figure 7-4.

```
ISMF PRIMARY OPTION MENU - z/OS DFSMS V1 R5
Enter Selection or Command ==> P

Select one of the following options and press Enter:
0 ISMF Profile           - Specify ISMF User Profile
1 Data Set              - Perform Functions Against Data Sets
2 Volume                - Perform Functions Against Volumes
3 Management Class     - Specify Data Set Backup and Migration Criteria
4 Data Class            - Specify Data Set Allocation Parameters
5 Storage Class         - Specify Data Set Performance and Availability
6 Storage Group         - Specify Volume Names and Free Space Thresholds
7 Automatic Class Selection - Specify ACS Routines and Test Criteria
8 Control Data Set     - Specify System Names and Default Criteria
9 Aggregate Group      - Specify Data Set Recovery Parameters
10 Library Management  - Specify Library and Drive Configurations
11 Enhanced ACS Management - Perform Enhanced Test/Configuration Management
C Data Collection      - Process Data Collection Function
L List                 - Perform Functions Against Saved ISMF Lists
P Copy Pool           - Specify Pool Storage Groups for Copies
R Removable Media Manager - Perform Functions Against Removable Media
X Exit                - Terminate ISMF
Use HELP Command for Help; Use END Command or X to Exit.
```

Figure 7-4 Select option 'P', Copy Pool

- ▶ Enter the database Copy Pool name using the required form of DSN\$locn-name\$DB and select option 3, Define a Copy Pool, as shown in Figure 7-5.

```

COPY POOL APPLICATION SELECTION
Command ==>>

To perform Copy Pool Operations, Specify:
CDS Name . . . . 'SMSCTL.SCDS'
                                     (1 to 44 character data set name or 'Active' )
Copy Pool Name  DSN$P870$DB          (For Copy Pool List, fully
                                     or partially specified or * for all)

Select one of the following options :
3 1. List   - Generate a list of Copy Pools
    2. Display - Display a Copy Pool
    3. Define  - Define a Copy Pool
    4. Alter   - Alter a Copy Pool

If List Option is chosen,
Enter "/" to select option      Respecify View Criteria
                                Respecify Sort Criteria

```

Figure 7-5 Define the “database” Copy Pool

- ▶ Enter the storage group name and number of backup versions for DFSMSHsm to manage. Notice that DFSMSHsm is asked to manage up to 15 Copy Pool backup versions on disk, as shown in Figure 7-6.

```

COPY POOL DEFINE                               Page 1 of 3
Command ==>>

SCDS Name . . . : SMSCTL.SCDS
Copy Pool Name : DSN$P870$DB

To DEFINE Copy Pool, Specify:
Description ==> COPY POOL FOR P870
              ==>
Number of Recoverable DASD Fast
Replicate Backup Versions . . . . 15      (1 to 85 or blank)
Storage Group Names: (specify 1 to 256 names)
==> P87VCAT
==>
==>

```

Figure 7-6 Define the source storage group to the “database” Copy Pool

- ▶ If you wish to take full system backups, enter the log Copy Pool name using the required form of DSN\$locn-name\$LG and select option 3, as shown in Figure 7-7.


```

COPY POOL APPLICATION SELECTION
Command ==>

To perform Copy Pool Operations, Specify:
  CDS Name . . . . 'SMSCTL.SCDS'
                                     (1 to 44 character data set name or 'Active' )
  Copy Pool Name   DSNP870$LG      (For Copy Pool List, fully
                                     or partially specified or * for all)

Select one of the following options :
  3 1. List      - Generate a list of Copy Pools
    2. Display   - Display a Copy Pool
    3. Define    - Define a Copy Pool
    4. Alter     - Alter a Copy Pool

If List Option is chosen,
  Enter "/" to select option      Respecify View Criteria
                                   Respecify Sort Criteria

```

Figure 7-7 Define the “log” Copy Pool

- ▶ Enter the storage group name and number of backup versions for DFSMSHsm to manage, as shown in Figure 7-8.

```

COPY POOL DEFINE                               Page 1 of 3
Command ==>

SCDS Name . . . : SMSCTL.SCDS
Copy Pool Name : DSNP870$LG

To DEFINE Copy Pool, Specify:
  Description ==> COPY POOL FOR P870 BSDS + LOG DATASETS
                ==>
  Number of Recoverable DASD Fast
  Replicate Backup Versions . . . .15      (1 to 85 or blank)
  Storage Group Names: (specify 1 to 256 names)
  ==> DSNP870
  ==>
  ==>

```

Figure 7-8 Define the source storage group to the “log” Copy Pool

- ▶ Connect the source storage groups with their associated backup storage groups using option 6, Specify Volume Names and Free Space Thresholds, as shown in Figure 7-9.

```

ISMF PRIMARY OPTION MENU - z/OS DFSMS V1 R5
Enter Selection or Command ====> 6

Select one of the following options and press Enter:
0 ISMF Profile           - Specify ISMF User Profile
1 Data Set               - Perform Functions Against Data Sets
2 Volume                 - Perform Functions Against Volumes
3 Management Class      - Specify Data Set Backup and Migration Criteria
4 Data Class             - Specify Data Set Allocation Parameters
5 Storage Class          - Specify Data Set Performance and Availability
6 Storage Group          - Specify Volume Names and Free Space Thresholds
7 Automatic Class Selection - Specify ACS Routines and Test Criteria
8 Control Data Set       - Specify System Names and Default Criteria
9 Aggregate Group        - Specify Data Set Recovery Parameters
10 Library Management    - Specify Library and Drive Configurations
11 Enhanced ACS Management - Perform Enhanced Test/Configuration Management
C Data Collection        - Process Data Collection Function
L List                   - Perform Functions Against Saved ISMF Lists
P Copy Pool              - Specify Pool Storage Groups for Copies
R Removable Media Manager - Perform Functions Against Removable Media
X Exit                   - Terminate ISMF
Use HELP Command for Help; Use END Command or X to Exit.

```

Figure 7-9 Select the Storage Group

- Enter the source storage group name and select option 3, Alter a Storage Group, to associate the storage groups, as shown in Figure 7-10.

```

STORAGE GROUP APPLICATION SELECTION
Command ====>

To perform Storage Group Operations, Specify:
CDS Name . . . . . 'SMSCTL.SCDS'
                                     (1 to 44 character data set name or 'Active' )
Storage Group Name   P87VCAT         (For Storage Group List, fully or
                                     partially specified or * for all)
Storage Group Type   (VIO, POOL, DUMMY, COPY POOL BACKUP,
                                     OBJECT, OBJECT BACKUP, or TAPE)

Select one of the following options :
3 1. List   - Generate a list of Storage Groups
   2. Define - Define a Storage Group
   3. Alter  - Alter a Storage Group
   4. Volume - Display, Define, Alter or Delete Volume Information

If List Option is chosen,
Enter "/" to select option   Respecify View Criteria
                             Respecify Sort Criteria

```

Figure 7-10 Alter by Storage Group Name

- Enter a description, the backup Copy Pool name in the *Copy Pool Backup SG Name* field, and 'Y' in the *SMS Alter Storage Group Status* field, as shown in Figure 7-11.

```

POOL STORAGE GROUP ALTER
Command ==>

SCDS Name . . . . . : SMSCTL.SCDs
Storage Group Name : P87VCAT
To ALTER Storage Group, Specify:
Description ==> FOR P870 CONNECT P87VCATP TO P87VCAT
==>

Auto Migrate . . N (Y, N, I or P)   Migrate Sys/Sys Group Name . .
Auto Backup . . N (Y or N)         Backup Sys/Sys Group Name . .
Auto Dump . . . N (Y or N)        Dump Sys/Sys Group Name . . .
Overflow . . . . N (Y or N)       Extend SG Name . . . . .
                                     Copy Pool Backup SG Name . . . P87VCATP
Dump Class . . .                   (1 to 8 characters)
Dump Class . . .                   Dump Class . . .
Dump Class . . .                   Dump Class . . .
Allocation/migration Threshold: High . . 85 (1-99)      Low . . (0-99)
Guaranteed Backup Frequency . . . . . (1 to 9999 or NOLIMIT)

ALTER      SMS Storage Group Status . . . Y (Y or N)

```

Figure 7-11 Associating source and target storage groups

Associate source storage group P87VCAT with Copy Pool backup P87VCATP and set SMS Storage Group Status to 'Y'. Do the same with the log storage groups.

Be sure to validate the backup environment each time it is changed — for example, when volumes in a source or backup storage group change, when the number of versions to maintain changes, or when the storage groups defined to a Copy Pool have changed. Be aware of how and when your system configuration has changed before you use a Copy Pool (with RESTORE SYSTEM or outside of DB2) to restore a system.

7.3 The PITR DB2 functions

In this section we describe how the BACKUP SYSTEM and RESTORE SYSTEM utilities work together with SMS to provide for a simple, fast, and reliable system level point in time recovery, and additionally, how to restore a system to a full system backup. We also include a short description of other changes to support this functionality.

7.3.1 Suspend database writes

Prior to DB2 V8, it was possible to have a recovery point in the middle of a 32 KB page write or data set extend while taking a SPLIT MIRROR type system backup. DB2 Version 8 has enhanced the SET LOG SUSPEND/SET LOG RESUME commands to quiesce 32 KB page writes and data set extends during SPLIT MIRROR processing. When the SET LOG SUSPEND command is issued, DB2 waits for any in-progress 32 KB database writes and data set extends to complete, and quiesces any additional writes or data set extends as long as the command is in effect. The SET LOG RESUME command resumes these activities. This helps guard against the possibility of partial page writes taking place during SPLIT MIRROR backup processing.

7.3.2 CI size up to 32 KB

DB2 table spaces and index spaces are defined as VSAM linear data sets. Up to DB2 V7, every page set has been allocated in control intervals of 4 KB, even though VSAM allows CI sizes multiple of 4 up to 32 KB for linear data sets, and DB2 has chained the CIs up to the required page size.

DB2 V8 introduces support for CI sizes of 8, 16, and 32 KB, activated by the default of the new DSVCI ZPARM in panel DSNTIP7. This is valid for user defined and DB2 defined table spaces. Index spaces only use 4 KB pages. If you decide to activate the new CI sizes (once you are in New Function Mode), all new table space page sets will be allocated by DB2 with a CI corresponding to the page size. The page sets already existing at the time of migrating will be later converted by the execution of Loads or Reorgs. The DB2 install procedure will also prepare the correct JCL for the (user) defined DB2 catalog table spaces, and will convert them to the new page size during the ENFM phase.

The new CI sizes reduce integrity exposures, relieve some restrictions on concurrent copy and the use of striping, and provide the potential for reducing elapsed time for table space scans.

7.3.3 Forward log recovery

In the event the integrity of an SAP instance is compromised, often it is not enough to restore a system to a prior backup. The amount of data that would be lost could represent many hours of work, especially if the time between backups is very long and the corruption occurred close to the time the next backup was scheduled. In versions prior to DB2 V8, it is too disruptive to the business, in terms of outage, staff, and probability of error, to restore an SAP DB2 instance to a point in time other than that at which a backup was taken.

DB2 V8 provides a fast, easy, minimally disruptive way to create volume-level backups and a fast, reliable way to recover to an arbitrary point in time with the new BACKUP SYSTEM and RESTORE SYSTEM utilities. As mentioned in the previous section, DB2 uses technology that is new with DFSMSHsm V1R5 to accomplish this. The new DFSMSHsm Copy Pool and SMS Copy Pool backup target storage group provide for this functionality. In order to take advantage of this functionality, the following conditions must be met:

- ▶ All DB2 data sets must be SMS managed.
- ▶ You are running z/OS V1R5.
- ▶ Your disk control units support ESS FlashCopy.
- ▶ You have defined a Copy Pool for your “database” data. You will also need a Copy Pool for your “log” data if you plan to take full system backups. Be sure to follow the DB2 naming conventions for each type of Copy Pool.
- ▶ You have defined a Copy Pool backup storage group for each source storage group in the Copy Pools.

Once everything is in place, creating backups or restoring a system to an arbitrary point in time can be accomplished with one utility statement. When using RESTORE SYSTEM with the LOGONLY keyword, it is assumed that you have already restored the “database” volumes by another volume copy means.

The BACKUP SYSTEM utility

The BACKUP SYSTEM utility invokes new fast replication services in DFSMSHsm V1R5 to take volume level copies:

- ▶ **BACKUP SYSTEM DATA ONLY**, copies only the “database” portion of a system (DB2 catalog, directory, and user data)
- ▶ **BACKUP SYSTEM FULL** (default), copies the entire DB2 subsystem and includes the “database” and “log” (active logs and BSDS) portions of a subsystem or data sharing group

These copies are taken without DB2 having to take any quiesce points and can be used to restore a subsystem or data sharing group to a prior point in time, even when there is uncommitted data.

Database only backups can be taken by using the **DATA ONLY** keywords. This tells the utility to copy only the “database” Copy Pool. Full system backups are the default and can be explicitly specified with the **FULL** keyword. Both types of backups can be used for point in time recovery with the **RESTORE SYSTEM** utility because it only uses the “database” Copy Pool to restore the data prior to applying logs. A full system backup can also be used to restore a system to the time the backup was taken, for disaster recovery, or for cloning purposes. In a full system backup, the “database” Copy Pool is copied first and the “log” Copy Pool is copied second, so that normal DB2 restart recovery processing can be used to restore data consistency when restoring to a full backup.

During backup, DB2 records a recovery based “log” point (RBLP) in the header page of DBD01. The RBLP is identified as the most recent system checkpoint prior to a backup log point, and the point at which DB2 starts scanning logs during a **RESTORE SYSTEM** recovery operation. DB2 updates its BSDS with backup version information and can keep track of up to 50 backup versions. In the case of data sharing, the submitting member records the backup version in its BSDS and also in the SCA. Figure 7-12 illustrates what happens during backup.

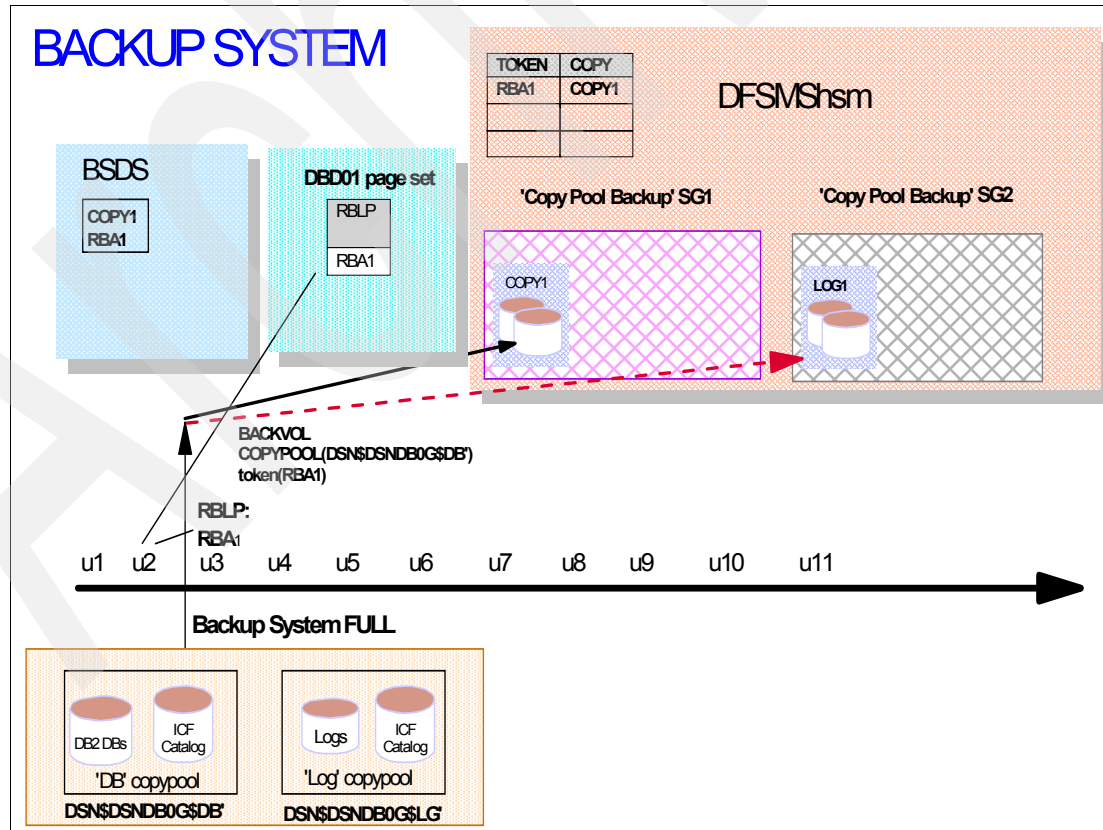


Figure 7-12 *BACKUP SYSTEM* utility execution for DSNDB0G

One BACKUP SYSTEM utility has been run, and a full system backup has been taken for DSNDB0G. The information is recorded in the DB2 BSDS, the header page of DBD01, and DFSMSHsm. DB2 only records up to 50 backup versions in the BSDS. However, DFSMSHsm is able to manage up to 85 backup versions on disk, but you must have enough backup target storage group volumes to satisfy the number of versions you want to keep. If you wish to keep more than the maximum 85 versions, you must dump them off to tape before they are invalidated by subsequent backup attempts.

Figure 7-13 illustrates what happens during the BACKUP SYSTEM FULL process. The BSDS is updated with COPY1 version information at RBA1, the RBLP in DBD01 is updated with the most recent system checkpoint RBA or LRSN prior to backup RBA1 at u2. DFSMSHsm records COPY1 and RBA1 and keeps track of the DB and LG Copy Pool copies.

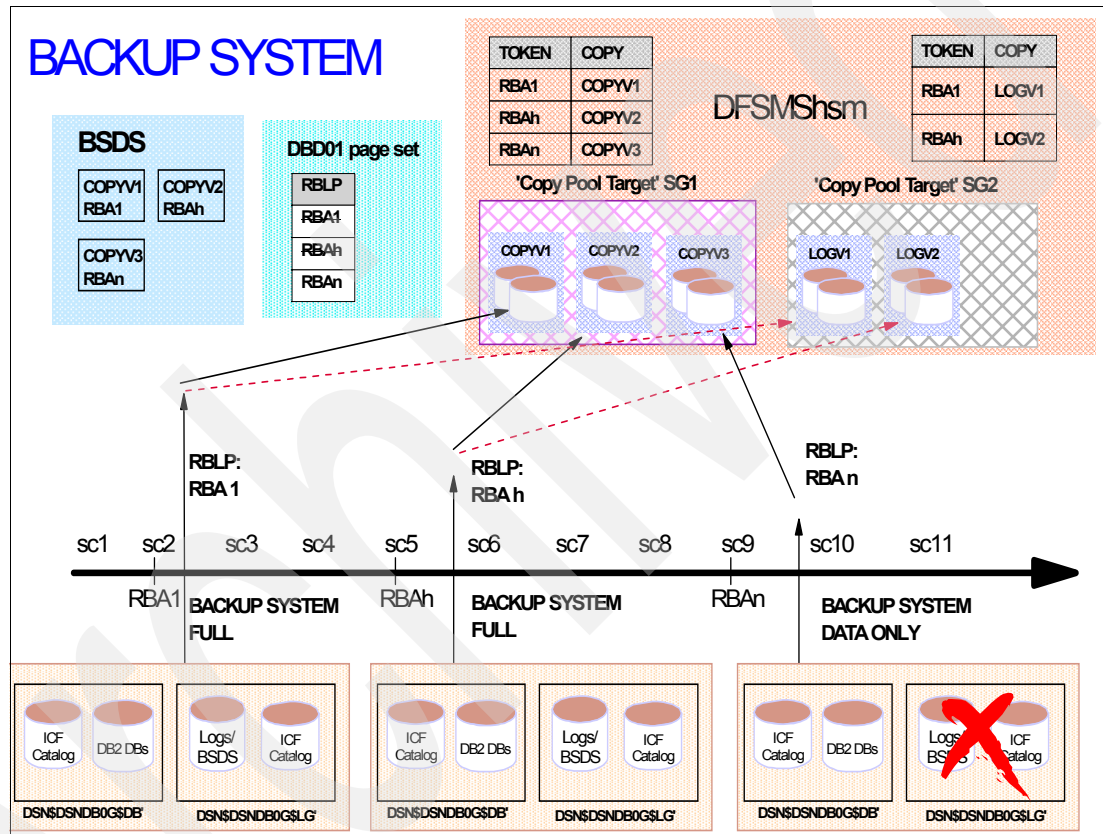


Figure 7-13 Two BACKUP SYSTEM FULL, and one DATA ONLY are taken

Figure 7-13 also illustrates what happens as more BACKUP SYSTEM copies are taken.

Three BACKUP SYSTEM backups have been taken for DSNDB0G. Two full system backups and one data only backup. The second backup is also a full system backup and the same sequence of events occurs, but the third backup is a data only backup. Notice that the same information is recorded in the BSDS and the header page of DBD01. DFSMSHsm records the same copy version information but only takes a copy of the DB Copy Pool.

In general, the BACKUP SYSTEM utility performs the following steps:

- Takes a new exclusive lock to ensure that no other backup utility can execute. If data sharing, it takes a global lock.

- ▶ Suspends 32 KB page writes for objects created prior to NFM and not migrated afterwards. You can avoid the write suspension by REORGing these objects. If data sharing, all members are notified.
- ▶ Suspends data set creation (create table space, index, etc.), deletion (drop table space, index, etc.), renames (online REORG fast switch), and extensions. If data sharing, all members are notified.
- ▶ Suspends system checkpoints. If data sharing, all members are notified.
- ▶ Prevents data sets from pseudo close. If data sharing, all members are notified.
- ▶ Records the RBLP RBA or LRSN in the header page of DBD01 and writes the page to disk. If data sharing, the system checkpoint prior to the lowest LRSN of all active members is used.
- ▶ Invokes DFSMSHsm to take a FlashCopy of the “database” Copy Pool.
- ▶ Updates the BSDS with the system backup information. If data sharing, only the BSDS for the submitting member is updated.
- ▶ Invokes DFSMSHsm to take a FlashCopy of the “log” Copy Pool if it is a full system backup.
- ▶ Resumes all suspend activities above. If data sharing, notifies all members.
- ▶ Releases the exclusive lock. If data sharing, notifies all members.
- ▶ Issues an informational message indicating the backup is complete.

The syntax of the utility statement for a full system backup of “database” and “log” Copy Pools is:

```
BACKUP SYSTEM FULL (FULL is the default and can be omitted)
```

The syntax of the utility statement for a backup of only the “database” Copy Pool is:

```
BACKUP SYSTEM DATA ONLY
```

An example of the output you would see from a full system backup is shown in Figure 7-14. Notice that the “database” Copy Pool is copied first and the “log” Copy Pool is copied second.

```

DSNUGUTC - BACKUP SYSTEM
DSNU1600I DSNUVBBD - BACKUP SYSTEM UTILITY FOR DATA STARTING,
              COPYPOOL = DSN$P870$DB
              TOKEN = X'D7F8F7F0BA140298CDE3D14200120CDAC090'.
DSNU1614I DSNUVBBD - BACKUP SYSTEM UTILITY FOR DATA COMPLETED SUCCESSFULLY,
              COPYPOOL = DSN$P870$DB
              TOKEN = X'D7F8F7F0BA140298CDE3D14200120CDAC090'
              ELAPSED TIME = 00:00:03.
DSNU1600I DSNUVBBD - BACKUP SYSTEM UTILITY FOR LOGS STARTING,
              COPYPOOL = DSN$P870$LG
              TOKEN = X'D7F8F7F0BA140298CDE3D14200120CDAC090'.
DSNU1614I DSNUVBBD - BACKUP SYSTEM UTILITY FOR LOGS COMPLETED SUCCESSFULLY,
              COPYPOOL = DSN$P870$LG
              TOKEN = X'D7F8F7F0BA140298CDE3D14200120CDAC090'
              ELAPSED TIME = 00:00:05.
DSNU1602I DSNUVBBD - BACKUP SYSTEM UTILITY COMPLETED, ELAPSED TIME = 00:00:08
DSNU010I DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0

```

Figure 7-14 *BACKUP SYSTEM* utility output

Both types of backups can be used for system recovery to an arbitrary point in time.

You need SYSCTRL or SYSADM authorization to run this utility, and it is compatible with all other utilities, but only one BACKUP SYSTEM utility can be running at a time. The utility can be displayed or terminated only if the commands are entered on the subsystem on which the utility was submitted. The utility can be terminated, but termination checking takes place only before the “database” Copy Pool is copied, and, if DATA ONLY is not specified, before the “log” Copy Pool is processed. The utility can be restarted, but it starts the backup at the beginning. Once the number of existing backups on disk matches the number specified in the Copy Pool VERSIONS attribute and a new backup is started, DFSMSHsm marks the oldest maintained backup invalid before it starts copying the Copy Pool(s) for the new backup. If the utility is not successful and is restarted, the version marked invalid will continue to be written to until a successful backup completes.

If BACKUP SYSTEM is not supported by your installation, volume level backups can be taken by using the SET LOG SUSPEND/SET LOG RESUME commands in conjunction with another type of volume backup solution. If you are using z/OS V1R5, Copy Pools can be used to simplify the task.

The RESTORE SYSTEM utility

Use the RESTORE SYSTEM utility only when you want to recover a subsystem or data sharing group to an arbitrary point in time. The utility restores only the “database” Copy Pool of a data only or full system backup, and then applies logs until it reaches a point in the log equal to the log truncation point specified in a point in time conditional restart control record (SYSPITR CRCR) created with DSNJU003. You cannot explicitly name the backup to use for recovery. That is implicitly determined by the log truncation point used to create the SYSPITR CRCR.

The RESTORE SYSTEM utility uses the RBLP stored in the header page of DBD01 and updated by the BACKUP SYSTEM utility as the log scan starting point. The log apply phase uses Fast Log Apply to recover objects in parallel. DB2 handles table space and index space creates, drops and extends, and marks objects that have had LOG NO events as RECP (table spaces and indices with the COPY YES attribute) or RBDP (indices with COPY NO attribute). An informational message will be issued to let the user know if there are any objects that need additional recovery.

If you want to restore a system to the point at which a full backup was taken, do not use the RESTORE SYSTEM utility. Use HSM FRRECOV COPYPOOL(cpname) GEN(gen) to restore both “database” and “log” Copy Pools. Then start DB2, which will use normal restart recovery processing to back out in-flight URs.

When the utility is used with the LOGONLY keyword, DB2 skips the call to DFSMSHsm, assumes the data was restored by another method, and executes only the log apply phase.

You must run the RESTORE SYSTEM utility with install SYSADM authority and NFM must be enabled.

The syntax of the RESTORE SYSTEM utility is simply:

```
RESTORE SYSTEM
```

Figure 7-15 shows an example of the output you should see from a successful execution of the RESTORE SYSTEM utility.


```

DSNU000I  DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = RESTORE
DSNU1044I DSNUGTIS - PROCESSING SYSIN AS EBCDIC
DSNU050I  DSNUGUTC - RESTORE SYSTEM
DSNU1606I DSNUVBRD - RESTORE SYSTEM UTILITY STARTING,
                COPYPOOL = DSN$P870$DB
                TOKEN = X'D7F8F7F0BA140298CDE3D14200120CDAC090'.
DSNU1627I DSNUVBRD - RESTORE SYSTEM PRE-LOG APPLY COMPLETED SUCCESSFULLY,
                COPYPOOL = DSN$P870$DB
                TOKEN = X'D7F8F7F0BA140298CDE3D14200120CDAC090'
                ELAPSED TIME = 00:00:04.
DSNU1604I -P870 DSNUVARL - RESTORE SYSTEM PHASE LOG APPLY STARTED AT LOG POINT =
X'00120CDAC090'.
DSNU1628I DSNUVBRD - RESTORE SYSTEM PHASE LOG APPLY COMPLETED, ELAPSED TIME = 00:04:54.
DSNU010I  DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0

```

Figure 7-15 RESTORE SYSTEM utility output

Figure 7-16 illustrates what occurs during a system level point in time recovery with the RESTORE SYSTEM utility. DB2 recognizes the log truncation point in the SYSPITR CRCR, checks the BSDS to determine which backup to use, calls DFSMSHsm to restore the correct “database” Copy Pool version, gets the RBLP from DBD01, then scans the logs and applies log records until reaching the log truncation point.

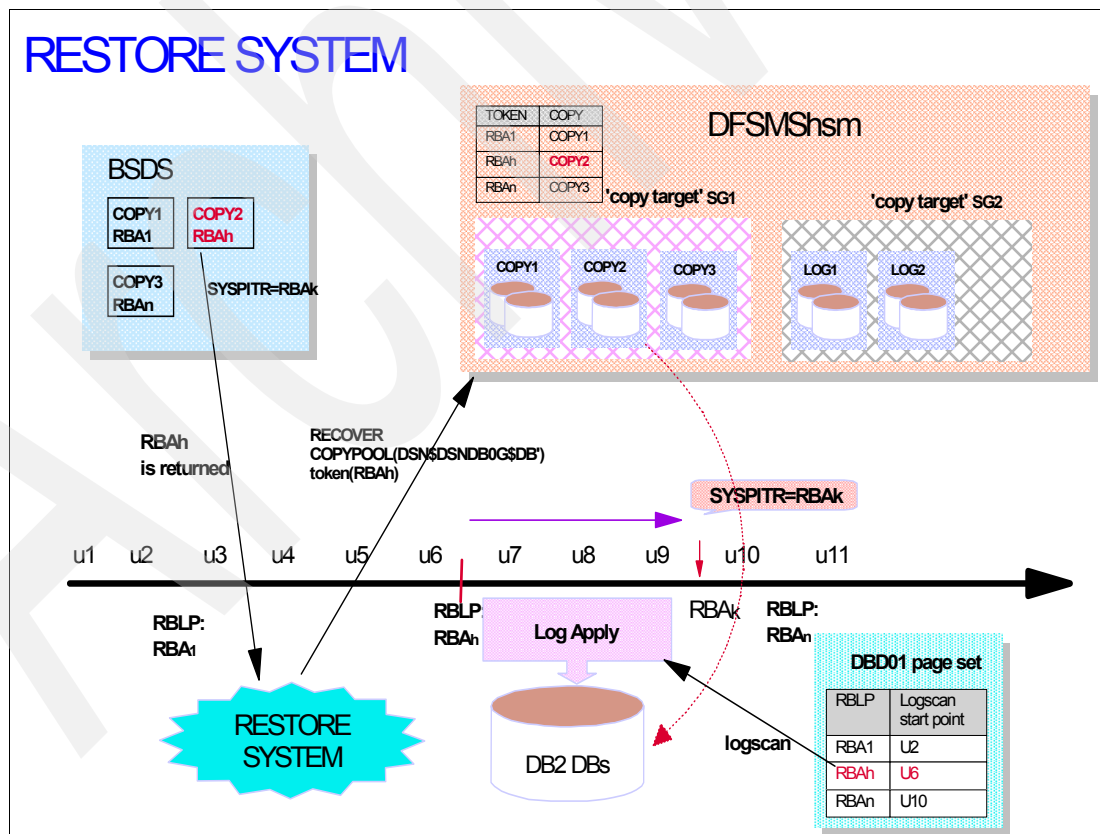


Figure 7-16 Recovering to an arbitrary point in time using the RESTORE SYSTEM utility.

In general, to restore a system to a prior point in time with the RESTORE SYSTEM utility, use the following steps:

- ▶ Stop DB2. If data sharing, stop all members.
- ▶ Use DSNJU003 to create a SYSPITR CRCR specifying the point to which you wish to recover the system. If data sharing, create a SYSPITR CRCR for each member.
- ▶ If data sharing, delete all coupling facility structures.
- ▶ Start DB2. If data sharing, start all members of the data sharing group.
- ▶ DB2 will enter into system recover-pending mode, access(maint), and will bypass recovery except for indoubt URs.
- ▶ Execute the RESTORE SYSTEM utility. The utility must be completed on the original submitting member.
 - Restores the most recent “database” Copy Pool version that was taken prior to the log truncation point.
 - Performs log apply function.
- ▶ If a method other than the BACKUP SYSTEM utility was used to copy the system, restore the data manually and use RESTORE SYSTEM LOGONLY.
 - This option can be used with z/OS V1R3 and above.
 - Backs up data with another volume dump solution and uses SET LOG SUSPEND/RESUME.
 - Performs log apply function only.
- ▶ Recycle DB2 to reset system recovery-pending status. If data sharing, stop all members.
- ▶ Display and terminate any active utilities.
- ▶ Display restricted objects and recover objects in RECP status and rebuild objects in RBDP status.

The SET LOG SUSPEND/RESUME commands can be used with other volume backup solutions if BACKUP SYSTEM isn't available. DFSMSHsm Copy Pools can be used to simplify the backup process if you are on z/OS V1R5. The SET LOG SUSPEND command is more disruptive than the BACKUP SYSTEM utility because it halts all update activity on a subsystem. Update activity is resumed only when the SET LOG RESUME command is issued. If data sharing is active, both commands must be entered on all active members of the data sharing group. The SET LOG SUSPEND command updates the RBLP in DBD01 so that recovery with RESTORE SYSTEM LOGONLY is possible.

Other changes to support system level point in time recovery

Stand alone utilities have been changed to reflect the new functions.

DSNJU003 Change Log Inventory

A new option has been added to DSNJU003, Change Log Inventory, to allow you to create a new type of conditional restart control record to truncate logs for a system point in time recovery in preparation for running the RESTORE SYSTEM utility. The syntax of the option is:

```
CRESTART CREATE SYSPITR=log-point
```

The log-point is the RBA or LRSN (for data sharing) to which you want to recover the system. The SYSPITR option can not be specified with any other option and is only allowed after NFM is enabled. When DB2 is started with a SYSPITR CRCR, the system starts in system recovery-pending mode and only the RESTORE SYSTEM utility can be run. Most DB2 commands will work, with the exception of START DATABASE and TERM UTIL. A DIS UTIL command will display only the status of the RESTORE SYSTEM utility. Data is unavailable until the utility completes successfully and DB2 is recycled. DB2 must be recycled after recovery is complete to reset recovery-pending status.

If data sharing is active, each member that has been active after the log truncation point must have a SYSPITR CRCR created with the same log truncation point and must be started prior to recovery. The RESTORE SYSTEM utility cannot be run until all members have been started.

DSNJU004 Print Log Map

DSNJU004 Print Log Map has been enhanced to show SYSPITR CRCR information if it is present and system backup information. Figure 7-17 illustrates the changes.

```

23:47:48 OCTOBER 08, 2003
**** ACTIVE CRCR RECORD ****
CRCR IDENTIFIER 0001
  USE COUNT 0
  RECORD STATUS
    CRCR ACTIVE
    CRCR NOT USED
  PROCESSING STATUS
    FORWARD = YES
    BACKOUT = YES
  SYSPITR SYSTEM LEVEL RECOVERY MODE RESTART
  STARTRBA                NOT SPECIFIED
  ENDRBA                  NOT SPECIFIED
  ENDLRSN                 BA245C6867D9 <-----SYSPITR
  EARLIEST REQUESTED RBA  000000000000
  FIRST LOG RECORD RBA    000000000000
  ORIGINAL CHECKPOINT RBA 000000000000
  NEW CHECKPOINT RBA (CHKPTRBA) 00001022BE2E
  END CHECKPOINT RBA      00001023AF66
  CRCR CREATED            23:47:18 OCTOBER 08, 2003
  TIME OF CHECKPOINT      23:41:28 OCTOBER 08, 2003
  RESTART PROGRESS        STARTED  ENDED
                          =====  =====
  CURRENT STATUS REBUILD  NO        NO
  FORWARD RECOVERY PHASE  NO        NO
  BACKOUT RECOVERY PHASE  NO        NO

                          BACKUP SYSTEM UTILITY HISTORY
                          SUBSYSTEM ID DJ1G
                          23:47:49 OCTOBER 08, 2003
                          START STCK                DATA COMPLETE
DATA/LOG
COMPLETE
  DATA          LOG          RBLP          LRSN          DATE
  LTIME         LOCATION NAME
  -----
BA2458B70E2AC5AE 0000000000000000  BA245635C2B2  BA245635C2B2
2003/10/08

```

Figure 7-17 DSNJU004 output

SET LOG SUSPEND/RESUME

The SET LOG SUSPEND and SET LOG RESUME commands have been enhanced to quiesce 32 KB page writes and data set extends.

DSN1LOGP

DSN1LOGP has been enhanced to print a system event type log record, subtype=8, which is generated when a system goes into system recover-pending mode.

DSN1PRNT

DSN1PRNT has been enhanced to print the RBLP stored in the header page of DBD01.

7.4 Scenario 1: Restoring to a prior backup

This section describes the steps necessary to restore a DB2 subsystem or data sharing group to a point at which a volume level backup was taken.

The following two examples describe how to restore a system to a prior backup. Case 1 uses the BACKUP SYSTEM utility to take a volume level back. Case 2 uses an alternative method to take a full system backup. When restoring a system to the point in time at which a backup was taken, you must back up and restore both “database” and “log” Copy Pools or volumes.

7.4.1 Case 1: Restore a system using BACKUP SYSTEM

- ▶ Start DB2. In data sharing environments, start all dormant members.
- ▶ Execute BACKUP SYSTEM FULL utility to backup both “data” and “log” Copy Pools.
- ▶ Stop DB2. If data sharing, stop all members of the data sharing group.
- ▶ Restore all “data” and “log” volumes from the desired backup.
 - FRRECOV COPYPOOL(DSN\$locn\$DB) VERIFY(Y) to restore the database Copy Pool
 - FRRECOV COPYPOOL(DSN\$locn\$LG) VERIFY(Y) to restore the log Copy Pool.
- ▶ If data sharing, delete coupling facility structures.
- ▶ Restart DB2. If data sharing, restart all dormant members.
- ▶ If data sharing, execute GRECP/LPL recovery. This recovers changed data that was stored in the coupling facility at the time of the backup.

7.4.2 Case 2: Restore a system using SET LOG SUSPEND

- ▶ Start DB2. If data sharing, start all dormant members.
- ▶ Take a volume level backup of “data” and “log” data sets. If using z/OS V1R5, use Copy Pools to simplify the process:
 - Execute SET LOG SUSPEND. If data sharing, execute on all active members.
 - Take backups of “data” and “log” volumes. If at z/OS V1R5 use DFSMS Copy Pool constructs to simplify the backup process.
 - Execute SET LOG RESUME. If data sharing, execute on all active members.
- ▶ Stop DB2. If data sharing, stop all active members.
- ▶ Restore all “data” and “log” volumes from the desired backup, as noted above.
- ▶ If data sharing, delete all coupling facility structures.
- ▶ Restart DB2. If data sharing, restart all dormant members.
- ▶ If data sharing, execute GRECP/LPL recovery. This recovers changed data that was stored in the coupling facility at the time of the backup.

7.5 Scenario 2: Restore to a prior arbitrary point in time

This section describes the steps necessary to restore a DB2 subsystem and data sharing group to an arbitrary point in time, given in two examples.

7.5.1 Case 1: Arbitrary PITR using the BACKUP and RESTORE SYSTEM

In the following exercise we created and populated a table with “good” and “bad” data, took a data-only backup, then restored the system to a log point established somewhere between the “good” and “bad” DML. We also wanted to see if DB2 could handle recreating objects during forward log recovery, so we created an additional table space and table that should be recreated in the log apply phase.

- ▶ Start DB2. If data sharing, start all dormant members.
- ▶ Execute DDL to create a database, table space, and two tables each with one index.
- ▶ Execute BACKUP SYSTEM DATAONLY.
- ▶ Execute DML to insert rows into one table, then update some of the rows.
- ▶ Use the LOAD utility with the LOG NO attribute to load the second table.
- ▶ Create another table space, table, and index in an existing database.
- ▶ Use SET LOG SUSPEND/SET LOG RESUME to establish log truncation point *logpoint1*, the point to which you want to recover. If non-data sharing use the RBA, if data sharing, use the lowest LRSN among active members.
- ▶ Execute DML to insert rows into one of the tables, and to update and/or delete some rows.
- ▶ Stop DB2. If data sharing, stop all active members.
- ▶ Use DSNJU003 to create a SYSPITR CRCR record (CRESTART CREATE SYSPITR=*logpoint1*). This is the log truncation point established above. If data sharing, create a SYSPITR record for each member active member.
- ▶ If data sharing, delete all coupling facility structures.
- ▶ Restart DB2. DB2 will start in system recovery-pending mode. If data sharing, restart all members.
- ▶ Execute the RESTORE SYSTEM utility. If data sharing, the utility can only be executed on one member. If the utility terminates and must be restarted it can only be restarted on the member on which it was originally executed.
- ▶ After the utility ends successfully, stop DB2. If data sharing, stop all active members. This will reset system recovery-pending status.
- ▶ Restart DB2. If data sharing, start all members.
- ▶ Execute the display command to check for active utilities or restricted objects. Terminate any active utilities. Recover any objects in RECP or RBDP status. from the second table.
 - -DIS UTIL(*) and terminate any active utilities.
 - -DIS DB(DSNDB01) SP(*)
 - -DIS DB(DSNDB06) SP(*) LIMIT(*)
 - -DIS DB(*) SP(*) LIMIT(*) RESTRICT
- ▶ Validate that recovery was successful.

7.5.2 Case 2: Arbitrary PITR

The methodology for this type of point in time recovery is essentially the same as above. The only difference is in the way the system is backed up and the data is restored to the volumes. This methodology could be used if one of the requirements for BACKUP SYSTEM and RESTORE SYSTEM is not in place.

Following the same steps as above:

- ▶ Start DB2. If data sharing, start all dormant members.
- ▶ Execute DDL to create a database, table space, and two tables each with one index.
- ▶ Take a system backup:
 - Execute SET LOG SUSPEND to stop update activity.
 - Take backups of “data” volumes using existing volume copy or split mirror solutions. If on z/OS V1R5 you can use Copy Pools to simplify the process.
 - Execute SET LOG RESUME to resume update activity.
- ▶ Execute DML to insert rows into one table, then update some of the rows.
- ▶ Use the LOAD utility with the LOG NO attribute to load the second table.
- ▶ Create another table space, table and index in an existing database.
- ▶ Use SET LOG SUSPEND/SET LOG RESUME to establish log truncation point *logpoint1*, the point to which you want to recover. If non-data sharing use the RBA, if data sharing, use the lowest LRSN among active members.
- ▶ Execute DML to insert rows into one of the tables, and to update and/or delete some rows.
- ▶ Stop DB2. If data sharing, stop all active members.
- ▶ Use DSNJU003 to create a SYSPITR CRCR record (CRESTART CREATE SYSPITR=*logpoint1*). This is the log truncation point established above. If data sharing, create a SYSPITR record for each member active member.
- ▶ Restore only the “data” volumes using an existing volume copy process, or if on z/OS V1R5 you can use Copy Pools to simplify the process.
- ▶ If data sharing, delete all coupling facility structures.
- ▶ Restart DB2. DB2 will start in system recovery-pending mode. If data sharing, restart all members.
- ▶ Execute the RESTORE SYSTEM utility with the LOGONLY keyword. If data sharing, the utility only needs to be executed on one member. If the utility terminates and must be restarted it can only be restarted on the member on which it was originally executed.
- ▶ After the utility ends successfully, stop DB2. If data sharing, stop all active members. This will reset system recovery-pending status.
- ▶ Restart DB2. If data sharing, restart all members.
- ▶ Execute the display command to check for active utilities or restricted objects. Terminate any active utilities. Recover any objects in RECP or RBDP status.
 - -DIS UTIL(*) and terminate any active utilities.
 - -DIS DB(DSNDB01) SP(*)
 - -DIS DB(DSNDB06) SP(*) LIMIT(*)
 - -DIS DB(*) SP(*) LIMIT(*) RESTRICT
- ▶ Validate that recovery was successful.



DB2 V8 changes to installation defaults

In this appendix we provide a summary of the changes in the default values introduced by DB2 V8 for installation parameters.

A.1 New installation default values

In this appendix we describe the changes that have taken place to the default values of several buffer pool settings in DB2 V8. The new values are the result of tuning work done by the DB2 Development Performance Department in SVL, and considerations related to a more general applicability of those values.

A.1.1 New buffer pool threshold values

For the ALTER BUFFERPOOL statement the initial default for the deferred write thresholds are:

- ▶ DWQT — bufferpool deferred write threshold as percentage of the total virtual pool size.
The new value is 30%. Decreased from the old value of 50%.
- ▶ VDWQT — bufferpool vertical deferred write threshold as percentage of the total virtual pool size
The new value is 5%. Decreased from the old value of 10%.

For the ALTER GROUPBUFFERPOOL statement the initial default for the deferred write thresholds are:

- ▶ CLASST — threshold for the class castout to disk as percentage of the size of the data entries in the group pool.
The new value is 30%. Decreased from the old value of 50%.
- ▶ GBPOOLT — threshold for the class castout as percentage of the size percentage of the total virtual pool size
The new value is 5%. Decreased from the old value of 10%.
- ▶ GBPCHKPT — default for the GBP checkpoint interval in minutes.
The new value is 4. Decreased from the old value of 8.

A.1.2 Changes to default parameter values

Table A-1 summarizes the old and new ZPARM default values.

Table A-1 Changed ZPARM default settings

Macro	Parameter	New value	Old value	Panel	Description
DSN6ARV	BLKSIZE	24576	28672	DSNTIPA	Archive log block size
DSN6FAC	CMTSTAT	INACTIVE	ACTIVE	DSNTIPR	DDF threads
DSN6FAC	IDHTOIN	120	0	DSNTIPB	Idle thread timeout
DSN6FAC	TCPKALV	120 sec.	ENABLE	DSNTIP5	TCP/IP keepalive
DSN6SPRC	SPRMCTH	10	50	hidden	Commit points until thread storage pool contraction
DSN6SPRC	SPRMINT	120	180	DSNTIPC	DDF idle thread interval timeout
DSN6SPRC	SPRMSTHR	1048576	2097152	hidden	Threshold size for thread storage pool contractions
DSN6SPRC	SPRMTIS	135168	24576	hidden	Initial CT long duration
DSN6SPRC	SPRMTXS	65536	32768	hidden	Allocation pool extension factor

Macro	Parameter	New value	Old value	Panel	Description
DSN6SPRC	TRSTHOLD	3	1 row	DSNTIPC	Trigger workfile usage threshold
DSN6SPRM	AUTHCACH	3072	1024	DSNTIPP	Cache for plan authorization
DSN6SPRM	CACHEDYN	YES	NO	DSNTIP8	Cache for dynamic SQL
DSN6SPRM	DSMAX	10000	3000	DSNTIPR	Maximum open data sets
DSN6SPRM	EDMDBDC	102400 KB	5000 KB	DSNTIPC	EDM DBD cache
DSN6SPRM	EDMPOOL	32768 KB	7312 KB	DSNTIPC	EDM pool size
DSN6SPRM	EDMSTMTC	102400 KB	5000 KB	DSNTIPC	EDM statement cache
DSN6SYSP	ACCUMACC	10	NO	DSNTIPN	New with V8 Accounting accumulation for DDF
DSN6SYSP	CHKFR	500000	50000	DSNTIPL	Checkpoint log records frequency
DSN6SYSP	CONDBAT	10000	64	DSNTIPE	Max number of remote connected users
DSN6SYSP	CTHREAD	200	70	DSNTIPE	Max number of users
DSN6SYSP	IDBACK	50	20	DSNTIPE	Max number of batch connections
DSN6SYSP	IDFORE	50	40	DSNTIPE	Max number of TSO connections
DSN6SYSP	LOBVALA	10240 KB	2048 KB	DSNTIP7	Storage value by user
DSN6SYSP	LOGAPSTG	100 MB	0	DSNTIPP	Storage for log apply
DSN6SYSP	MAXDBAT	200	64	DSNTIPE	Max number of remote active users
DSN6SYSP	NUMTABLE	20	10	DSNTIPD	Number of tables, used in CLISTs
DSN6SYSP	SMFSTST	YES(1,3,4,5,6)	YES(1,3,4,5)	DSNTIPN	SMF classes for STATISTICS
DSN6SYSP	SYPLOGD	10	NO	DSNTIPN	BACKODUR multiplier
DSN6SYSP	EXTSEQ	YES	NO	DSNTIPR	Extended security

Archived

Abbreviations and acronyms

ACS	access control system	DBID	database identifier
AIX@	Advanced Interactive eXecutive from IBM	DBMS	database management system
APAR	authorized program analysis report	DBRM	database request module
AQR	automatic query re-write	DCL	data control language
AR	access register	DDCS	distributed database connection services
ARM	automatic restart manager	DDF	distributed data facility
ART	access register translation	DDIC	data dictionary
ASCII	American Standard Code for Information Interchange	DDL	data definition language
ATS	Advanced Technical Support	DLL	dynamic link library
BLOB	binary large object	DLL	dynamic load library
BSDS	bootstrap data set	DML	data manipulation language
CCA	client configuration assistant	DNS	domain name server
CCMS	Computer Center Management System	DPSI	data partitioned secondary index
CCSID	coded character set identifier	DRDA	distributed relational database architecture
CD	compact disk	DSC	dynamic statement cache, local or global
CEC	central electronics complex	DTT	declared temporary tables
CF	coupling facility	EA	extended addressability
CFCC	coupling facility control code	EBCDIC	extended binary coded decimal interchange code
CFRM	coupling facility resource management	ECS	enhanced catalog sharing
CLI	call level interface	ECSA	extended common storage area
CLOB	character large object	EDM	environment descriptor manager
CLP	command line processor	ENFM	enabling new function mode
CM	compatibility mode	ERP	enterprise resource planning
CPU	central processing unit	ESA	Enterprise Systems Architecture
CRLF	carriage return and line feed	ESS	Enterprise Storage Server
CSA	common storage area	ETR	external throughput rate, an elapsed time measure, focuses on system capacity
CTT	created temporary table	FTP	File Transfer Program
DAD	document access definition	GB	gigabyte (1,073,741,824 bytes)
DASD	direct access storage device	GBP	group buffer pool
DAT	dynamic address translation	GRS	global resource serialization
DB	database	GUI	graphical user interface
DB2 PM	DB2 performance monitor	HA	Host adapter
DBA	database administrator	HFS	Hierarchical File System
DBAT	database access thread	HLQ	high-level qualifier
DBCLOB	double byte character large object	HPJ	high performance Java
DBD	database descriptor	HSC	homogeneous system copy
DBET	database exception tables states		

I/O	input/output	MB	megabyte (1,048,576 bytes)
IBM	International Business Machines Corporation	MCOD	Multiple Components in One Database
ICF	integrated catalog facility	MQT	materialized query table
ICF	integrated coupling facility	MSM	Multidimensional Storage Manager
ICLI	Integrated Call Level Interface	NFM	new function mode
ICMF	internal coupling migration facility	NPI	non-partitioning index
ID	identifier	NVS	Non-Volatile Storage
IFCID	instrumentation facility component identifier	OBID	object identifier
IFI	instrumentation facility interface	ODB	object descriptor in DBD
IMG	Implementation Guide	ODBC	Open Data Base Connectivity
IMIG	incremental migration	OLAP	online analytical processing
IPLA	IBM Program Licence Agreement	OLTP	online transaction processing
IRLM	internal resource lock manager	OS	operating system
IRWW	IBM Relational Warehouse Workload	OS/390	Operating System/390®
ISICC	IBM SAP International Competence Center	PAV	Parallel Access Volume
ISOBID	indexspace object identifier	PDS	partitioned data set
ISPF	interactive system productivity facility	PIB	parallel index build
ISV	independent software vendor	PITR	point in time recovery
IT	information technology	PPT	prior point in time
ITR	internal throughput rate, a processor time measure, focuses on processor capacity	PSID	pageset identifier
ITSO	International Technical Support Organization	PSP	preventive service planning
IVP	installation verification process	PTF	program temporary fix
IX	index	PUNC	possibly uncommitted
JCL	job control language	QBIC®	query by image content
JDBC	Java Database Connectivity	QMF™	Query Management Facility
JFS	journaled file systems	RACF	Resource Access Control Facility
JIT	Just in time (Java compiler)	RACF	Resource Access Control Facility
JNI	Java Native Interface	RBA	relative byte address
JVM	Java Virtual Machine	RBLP	recovery based log point
KB	kilobyte (1,024 bytes)	RECFM	record format
LCU	logical control unit	RID	record identifier
LOB	large object	ROT	rule of thumb
LPAR	logical partition	RR	repeatable read
LPL	logical page list	RRS	resource recovery services
LRECL	logical record length	RRSAF	resource recovery services attach facility
LRSN	log record sequence number	RS	read stability
LSS	logical subsystem	RSM	Real Storage Manager
LVM	logical volume manager	RTS	real time statistics
		RVA	RAMAC® Virtual Array
		SAP	Systems, Applications, Products in Data Processing
		SAP AG	SAP Aktiengesell

SAP APO	SAP Advanced Planner And Optimizer
SAP BW	SAP Business Information Warehouse
SAP CRM	SAP Customer Relationship Management
SAP SEM	SAP Strategic Enterprise Management
SAP WAS	SAP Web Application Server
SDK	software developers kit
SDM	system data mover
SG	storage group
SMIT	System Management Interface Tool
SMS	Storage Management Subsystem
SPE	small program enhancement
SQL	Structured Query Language
SVL	IBM Silicon Valley Laboratory
TB	table
TCB	task control block
TMS	transport management system
TS	tablespace
UDB	Universal Database
UR	unit of recovery
USS	UNIX system services
USS	UNIX System Services
VSAM	Virtual Storage Access Method
WARM	Write and Register Multiple
WAS	WebSphere Application Service
WLM	Workload Manager
WSAD	WebSphere Studio Application Developer
XML	eXtensible Markup Language

Archived

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 210. Notice that some of the documents referenced here may be available in softcopy only.

- ▶ *SAP on DB2 for z/OS and OS/390: DB2 System Cloning*, SG24-6287
- ▶ *SAP R/3 on DB2 for OS/390: Database Availability Considerations*, SG24-5690
- ▶ *SAP on DB2 for OS/390 and z/OS: High Availability Solution Using System Automation*, SG24-6836
- ▶ *SAP in DB2 for OS/390 and z/OS: Multiple Components in One Database (MCOB)*, SG24-6914
- ▶ *mySAP Business Suite Managed by IBM Tivoli System Automation for Linux*, REDP-3717
- ▶ *DB2 for z/OS and OS/390 Version 7 Selected Performance Topics*, SG24-6884
- ▶ *DB2 for z/OS and OS/390 Version 7 Performance Topics*, SG24-6129
- ▶ *DB2 UDB Server for OS/390 Version 6 Technical Update*, SG24-6108
- ▶ *DB2 UDB for OS/390 Version 6 Performance Topics*, SG24-5351
- ▶ *DB2 for OS/390 Version 5 Performance Topics*, SG24-2213
- ▶ *DB2 UDB for z/OS Version 8 Technical Preview*, SG24-6871
- ▶ *DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know , ... and More*, SG24-6079

Other publications

These publications are also relevant as further information sources:

- ▶ *SAP on IBM DB2 UDB for OS/390 and z/OS: Database Administration Guide - SAP Web Application Server Release 6.20*
- ▶ *SAP Web Application Server 6.20 on UNIX: IBM DB2 Universal Database for OS/390 and z/OS - Installation Guide*
- ▶ *SAP on IBM DB2 UDB for z/OS: Database Administration Guide - SAP NetWeaver Version 6.40*

The above manuals, and all other SAP manuals, are available from the SAP service Web site, accessible only if you are a registered user:

service.sap.com/instguides

- ▶ *Evolution of Star Join Optimization - DB2 UDB for z/OS and OS/390*, July 2002, whitepaper published by Yewich, Lawson and Associates, Inc., authors: Terry Purcell, Gene Fuh, Yun Wang, Yoitchi Tsuji, Eva Hu.

Online resources

These Web sites and URLs are also relevant as further information sources:

- ▶ DB2 Connect Web site:
<http://ibm.com/software/data/db2/db2connect/>
- ▶ DB2 for z/OS Version 8:
<http://www.ibm.com/software/data/db2/os390/db2zosv8.html>

You must be registered as an SAP Service Marketplace user to access the following resources. The registration requires an SAP installation or customer number. To register, go to:

- <http://service.sap.com>
- ▶ SAP Service Marketplace quick link MCOD:
<http://service.sap.com/mcod>
- ▶ SAP Service Marketplace quick link INSTGUIDES:
<http://service.sap.com/instguides>
- ▶ SAP Service Marketplace quick link PATCHES:
<http://service.sap.com/patches>
- ▶ SAP Service Marketplace quick link QUICKSIZER:
<http://service.sap.com/quicksizer>

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads:

ibm.com/support

IBM Global Services:

ibm.com/services

Index

Numerics

00C900AE 74
0C9008E 110
16 terabyte tables 7
225 tables in a join 97
225 tables per query or view 7
64 bit virtual storage 14
64-bit platform 31
64-bit virtual storage 14, 51

A

ABAP 31
ABAP character data types 25
ABAP statement information passed through 35
access path 126
access path selection adjustment 9
accounting and workload management 170
accounting data for DDF and RRSAF 170
accounting data rollup 64
Accounting information 35
accounting management 18
ACCUMACC 203
adding index columns 44
adding new partitions 68
adding partitions 49
Advisory Reorg 48
Advisory Reorg Pending 42
ALTER column primary key 86
ALTER INDEX 44
Alter Index to redistribute partitions 8
ALTER statements and versioning 46
ALTER TABLE ADD PARTITION 49, 68
ALTER TABLE ALTER COLUMN 41
ALTER TABLE ALTER PARTITION ENDING AT 50
ALTER TABLE ALTER PARTITION ROTATE FIRST TO
LAST 50, 68–69
ALTER TABLE SET DATATYPE
 impacts 42
application failure 179
arbitrary PITR using the BACKUP and RESTORE SYS-
TEM 199
arbitrary PITR without using BACKUP and RESTORE
200
Architecture 14
AREO* 42, 48, 83
ASCII characters 28
asynchronous INSERT preformatting 10
AUTHCACH 203
Automatic Restart Manager 11
Automatic Space Management
 benefits 161
automatic space management 18

B

BACKUP 19
backup and recovery 178
BACKUP SYSTEM 177, 190
backward index scan 140
BLKSIZE 202
BOTH 124
buffer pool coherency 152
buffer pool control blocks 52
buffer pool in data space 7
buffer pool size
 determining 55
Buffer pools 52
buffers and EDM pools in data spaces 7
Business Information Warehouse 16, 30
BW 91

C

CACHE DYNAMIC SQL 4
CACHEDYN 4, 203
CARDINALITY 114
cartesian join 93
castout buffers 52
castout processing 152
CCSID UNICODE 26
CF lock propagation 17
CF lock propagation reduction 144
CF request batching 151
Change Log Inventory 196
checkpoint frequency in minutes 12
CHKFR 203
CI size 19
CI size up to 32 KB 190
CLASST 202
cluster tables 17
clustering 15, 45, 70
clustering and partitioning separation 71
clusters 17
CMTSTAT 202
code maintenance 39
codepoint 28
COLGROUP 123–124
column type change 16
column type changes 75
common table expression 98
common table expression and recursive SQL 16
compatibility mode 19
compression 28
compression dictionaries 52, 57
CONDBAT 203
CONSTOR 60
COPY 180
copy pool 180–181
correlated subquery transformation 139

COUNT 123
COUNT integer 124
create index dynamic statement invalidation 15
CTHREAD 203
CURRENT MAINTAINED TABLE TYPES FOR OPTIMI-
ZATION 105
CURRENT REFRESH AGE 105
CURRENT_VERSION 45, 47

D

DATA CAPTURE 49
data consistency 178
DATA INITIALLY DEFERRED 103
data management with SAP 156
Data Partitioned Secondary Indexes 68–69
data set control blocks 52, 58
data sharing 143
 lock contention 147
data type changes 41
 cached dynamic statements 43
 dynamic SQL 43
 index availability 42
 indexes 42
 table spaces 42
 views and check constraints 43
database administration with SAP 156
database copy pool 182
Database Exception Tables 48
DB2 Connect 29
 features needed by SAP 33
 ICLI compatible functions 33
 multi-row operations 33
DB2 Connect and DRDA 14
DB2 Control Center 18, 166
DB2 data types 25
DB2 Java Universal Driver 64
DB2 Restart Light 11
DB2 V8 overview 13
db2radm 38
dbdb2pwd 36
DBET 48
DBM1 5, 52
DDF and RRSAP
 new default values 174
DDF/RRSAP ACCUM 171
DDL 40
DDL ALTER 40
declared temporary tables 175
Defer defining data sets 8
deferred indexes 16, 74
DFSMShsm 181, 183
DISTINCT 88
distribution statistics 17, 123
DPSI 69
DRDA Ping Service 34
DSC short prepares 142
DSC statement ID in Explain 18
DSMAX 5, 203
DSN1LOGP 198
DSN1PRNT 198

DSN6ARV 202
DSN6FAC 202
DSN6SPRC 202
DSN6SPRM 203
DSN6SYSP 203
DSNACCDD 168
DSNACCDE 168
DSNACCDDL 168
DSNACCDD 168
DSNACCDS 168
DSNACCDS 168
DSNACCJF 169
DSNACCJP 169
DSNACCJQ 169
DSNACCJS 169
DSNACCMB 167
DSNACCMD 167
DSNACCMO 167
DSNACCOB 167
DSNACCOR 166
DSNACCUC 170
DSNB536I 54
DSNB610I 54
DSNI031I 166
DSNJU003 196
DSNJU004 197
DSNTIP4 4
DSNTIPE 4
DSNUTILU 166
DWQT 202
dynamic statement cache 15, 72
dynamic statement caching 4

E

EDM pool 52
EDM Pool in data space 8
EDM pool new default size 56
EDMDBDC 203
EDMPOOL 203
EDMSTMTC 203
emergency changes 39
enable new function mode 19
Enterprise Portal 30
Exchange Infrastructure 30
Explain 18
EXTSEQ 203

F

fact table 93
fast cached SQL statement invalidation 17
fast replication 180, 185
fast retrieval of most recent value 18
FETCH 17
fetch 117
filter factors 123
FlashCopy 181
forward log recovery 190
FRBACKUP 183
FRDELETE 184
FREQVAL 123–124, 135

FRRECOV 184
FULLKEYCARDF 137

G

GBPCHKPT 202
GBPOOLT 202

H

handling of TCP/IP error situations 34
host variables 125
host variables impact on access paths 18
HTTP 29

I

ICLI 29
IDBACK 203
IDFORE 203
IDTHTOIN 202
IFCID 0022 105
IFCID 0217 59
IFCID 0225 59
IFCID 313 164
IFCID 317 166
IFCID 337 166
IN lists 133
inactive DBAT 173
Index
 changing the clustering attribute 45
index changes 44
index only access for varchar 18
index screening in RID list processing 5
index-only access 132
index-only access path 129
inline statistics 123
INLISTP 140
in-memory workfiles 95
INSERT 17
insert 117
installation parameters 201
IRLM locks 52
IRLM PC=YES 59
IXQTY 159

J

Java 15, 31
Java and DB2 data type mapping 62
Java dictionary 63
JDBC driver 61

K

KEEPDYNAMIC(YES) 4, 172

L

larger buffer pool sizes 53
larger buffer pools 52
LEAST 124

length limits 23
LIST COPYPOOL 185
LISTDEF 48
LOB data 58
LOB ROWID transparency 16
LOBVALA 203
local SQL cache issues and short prepare 18
lock escalation alert 165
lock escalation reporting 18
locking 17, 109
LOCKMAX 165
LOGAPSTG 203
long names 16
longer table and column names 16
long-running non-committing readers 18, 164
long-running UR warning 12
LPL recovery 153
LRDRTHLD 164

M

MAINTAINED BY SYSTEM 103
MAINTAINED BY USER 103
maintenance changes 39
Master Data Management 30
materialized LOB values 52
materialized query tables 17, 100
MAX KEPT DYN STMTS 4
MAXDBAT 203
MAXKEEPD 18, 142
MGEXTSZ 159
Microsoft.NET 29
minimize impact of creating deferred indexes 16
Mobile Infrastructure 30
MODIFY 47
more log data sets 19
more partitions 15
more tables in join 16, 19
MOST 124
MQT 100
 creating 101
 populating 103
 query rewrite 104
 registering 102
 special registers 105
multiple DISTINCT clauses in SQL statements 16
multiple IN values 18
multi-row 117
 INSERT 121
 MODIFY 119
 SELECT 118
multi-row INSERT and FETCH 17
MXQBCE 97
mySAP CRM 31
mySAP ERP 31
mySAP PLM 31
mySAP SCM 31
mySAP SRM 31
mySAP™ Customer Relationship Management 30

N

native ASCII data storage 4
NetWeaver 29
Network statistics 34
new function mode 19
NOT PADDED 130
NPGTHRSH 9
NPI 49
NUMLKTS 165
NUMTABLE 203

O

OA04555 173
OBD 46
OLDEST_VERSION 45, 47
online partitioning changes 15
online schema changes 39
online schema evolution 39
online system parameters 11
optimization 121
Oracle 28
ORDER BY clause 4
outer join performance enhancements 6

P

PADDED 44, 130–131
padding 129
PADIX 44, 130
partition key update 116
partitioned secondary indexes 15
partitioning 15, 68
partitions
 adding new 68
 rotating 68
performance 17
PITR 177
 DB2 functions 189
 requirements 178
 scenarios 198
point in time back-up and recovery 177
PQ37895 142
PQ53067 159
PQ61458 96
PQ67681 175
PQ68662 134
PQ73454 138
PQ73749 139
PQ75973 167
Print Log Map 197
prior point in time 179
project changes 39
PSRBD 131

Q

QUERY COPYPOOL 185
query optimization 121

R

RBDP 48, 76
rebalancing partitions 50
Rebuild Pending 43
recursive SQL 99
Redbooks Web site 210
 Contact us xx
reduced lock contention 17
Referential Integrity 178
REFRESH DEFERRED 103
REOPT(ONCE) 126, 135
REOPT(VARS) 126, 163
REORG 15, 42
REORG utility enhancements 15
REORP 69
REPORT NO UPDATE NONE 125
RESTORE 19
 restore a system using BACKUP SYSTEM 198
 restore a system using SET LOG SUSPEND 198
RESTORE SYSTEM 177, 194
restore to a prior arbitrary point in time 199
restoring to a prior backup 198
RETVLCFK=YES 130
RID pool 52, 57
RIDLIST 52
ROTATE 48
rotating partitions 50, 68
ROWID 87
RRSAF 175
RUNSTATS 17, 43, 122
RUNSTATS improvements 17

S

SAP access path 125
SAP and DB2 partnership 1
SAP and Java 60
SAP and zSeries synergy 3
SAP Business Information Warehouse 91
SAP Data Dictionary 70
SAP dbsl 41
SAP Enterprise 70
SAP related enhancements in DB2 V5 3
SAP related enhancements in DB2 V6 5
SAP related enhancements in DB2 V7 9
SAP schema changes 48
SAP system implementation 157
SAP xApps 30
scalar fullselect 16
schema changes and application changes 49
Schema evolution 14
SCOPE PENDING 69
separation of partitioning and clustering 15
SET DATATYPE 41
SET LOG SUSPEND 180
SET LOG SUSPEND/RESUME 9, 197
SJMXPPOOL 95
SMFSTST 203
snowflake 92
sort pool 52, 58

sparse index 93
sparse index for star join 16
SPRMCTH 202
SPRMINT 202
SPRMSTHR 202
SPRMTIS 202
SPRMTXS 202
SQL statement 2 MB long 16
SQL statements 2 MB long 16
SQLCODE -913 110
SQLConnectW 33
star join 92, 97
STARJOIN 95
statement ID in Explain 163
STMTCACHE STMTID 164
STMTCACHE STMTOKEN 164
SYSIBM.SYSCOPY 46
SYSIBM.SYSINDEXES 46–47, 130
SYSIBM.SYSINDEXPART 46, 159
SYSIBM.SYSOBDS 46
SYSIBM.SYSTABLEPART 46, 159
SYSIBM.SYSTABLES 46
SYSIBM.SYSTABLESPACE 46
SYSLOGD 203
system level point in time recovery 19
SYSTEMPAGES YES 47

T

table size 68
table UDF cardinality 19
TCPKPALV 202
technical failure 179
Tracing feature 35
TRSTHOLD 203
TSQTY 159
two-phase commit 33

U

uncorrelated subquery 6
Unicode 14
 introduction 22
 performance and storage 26
 SAP view 22–23
UNION and UNION ALL operators 10
UNIQUE 69
Universal Driver for SQLJ and JDBC 14
unmatched column join for VARCHAR 6
UPDATE NONE 17
UTF-16 22–23
UTF-32 22–23
UTF-8 22

V

VARCHAR 42
varying length index keys 44
varying-length index keys 130
VDWQT 202
version information 46

version limits 46
versioning 45
views 43
VOLATILE 17
volatile tables 17, 110
volume dump 180

W

Walldorf 2
Web Application Server 30
WebAS xv, 68, 110
WebSphere 29
workfiles 95
WSAD 207

X

XML 29

Z

ZPARM default values 202
zSeries 31

Archived



DB2 UDB for z/OS V8: Through the Looking Glass and What SAP Found There



Browse through the DB2 for z/OS functions that benefit SAP solutions

Understand the major advantages in usability and performance

Get ready for point in time backup and recovery

DB2 UDB for z/OS Version 8 (DB2 V8) introduces a large number of new features that provide for unmatched scalability, availability, performance, and manageability to reliably handle any information needed in SAP solutions. The objective of this IBM Redbook is to identify the new features of DB2 V8 that are particularly beneficial to SAP and to describe how SAP applications can take advantage of these new features. While some of these features can be exploited transparently by the currently available versions of DB2, for some others the next technological release of SAP will be required. The redbook introduces the new features and provides real-life examples from SAP to demonstrate the usefulness of the new features for SAP and to show how SAP can exploit them.

The considerations within this redbook apply to the whole spectrum of business solutions within the mySAP Business Suite, such as mySAP ERP and mySAP Supply Chain Management. These solutions share the common technical application platform SAP NetWeaver that includes the SAP Web Application Server and SAP Business Information Warehouse.

While this redbook specifically investigates the new features of DB2 V8 in the context of SAP, the majority of the considerations also apply to other enterprise packaged applications. The reason for this is that from a database management perspective, these applications have many similarities, such as a strong usage of dynamic SQL and a large number of database objects. For a more general description of the new features of DB2 V8, see DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ... and More, SG24-6079.

**INTERNATIONAL
TECHNICAL
SUPPORT
ORGANIZATION**

**BUILDING TECHNICAL
INFORMATION BASED ON
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:
ibm.com/redbooks**

SG24-7088-00

ISBN 0738498505