

Best Practices for SAP BI using DB2 9 for z/OS

**Benefits of SAP Business Information
Warehouse on z/OS**

**Best practices and
troubleshooting**

**Scalability of the BI
Accelerator**



**Lydia Parziale
Wilfried Alle
Martin Cardoso
Lindy Crawford
Giovanni Grita
Theodor Kuebler
Georg Mayer**



International Technical Support Organization

Best Practices for SAP BI using DB2 9 for z/OS

March 2008

Archived

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

Third Edition (March 2008)

This edition applies to DB2 Version 9 for z/OS, Release 1, and SAP BI Version 7.0 .

© Copyright International Business Machines Corporation 2005, 2007, 2008. 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

Notices	ix
Trademarks	x
Preface	xi
The team that wrote this IBM Redbook	xii
Become a published author	xv
Comments welcome	xv
Chapter 1. Introduction	1
1.1 Product overview	2
1.2 The audience for this IBM Redbooks publication	3
1.3 Why run SAP BI on IBM System z	4
1.3.1 Value provided by System z, z/OS, and DB2	4
Chapter 2. Overview of SAP BI and related data	9
2.1 BI architecture	10
2.2 Which cubes, DSO, and PSA objects are in your BI system	12
2.2.1 Report SAP_BWTOOLPGM_DB2	13
2.3 BI data and DB2 partitioning	14
2.3.1 Why partitioning	15
2.3.2 Range partitioning of F- fact tables	17
2.3.3 Partitioning of PSA tables	19
2.3.4 Partitioning of DataStore objects	20
2.3.5 Usage of data partitioned secondary indexes (DPSI)	21
2.4 SAP BI snowflake schema	25
2.5 New features and functions of SAP BI 7.x	26
2.6 Terminology: BI objects and processes	27
Chapter 3. Top ten SAP BI recommendations	29
3.1 Base recommendations - SAP BI implementations for DB2 on z/OS	30
Chapter 4. Benefits of DB2 9 for SAP BI	35
4.1 Dynamic index ANDing for star schema queries	36
4.2 Database statistics for BI tables	39
4.2.1 Processing RUNSTATS for BI tables	40
4.2.2 New to DB2 9: histogram statistics	45
4.3 Parallelism enhancements	48
4.3.1 Global query optimization	48
4.3.2 Cross query block optimization	49

4.4	Extended sparse index and in-memory workfile	49
4.5	TRUNCATE TABLE statement	50
4.6	FETCH FIRST n ROWS ONLY in subselects	51
4.7	RENAME INDEX	53
4.8	RANK expression	53
4.9	Further DB2 9 enhancements and new features	53
Chapter 5. Best practices for PSA		55
5.1	Load PSA	56
5.1.1	Load PSA	58
5.1.2	Management of PSA	68
5.1.3	Load PSA DB2 compression	69
5.1.4	PSA partitioning	70
Chapter 6. Best practices for InfoCubes		73
6.1	Types of InfoCubes	76
6.1.1	Physical data stores (data targets)	76
6.1.2	VirtualProviders	76
6.2	Repartitioning of InfoCubes	78
6.2.1	Repartitioning implementation	80
6.2.2	Advantages of partitioning/repartitioning	88
6.3	Remodeling InfoProviders	89
6.3.1	Procedure for remodeling an InfoProvider	91
6.3.2	Benefits of remodeling InfoProviders	96
6.4	SAP compression for loads	96
6.4.1	Advantages	96
6.4.2	Data validation	97
6.4.3	Cumulative data versus non-cumulative key figures	98
6.4.4	Areas of impact	98
6.4.5	Implementing SAP compression	100
6.4.6	Speed up compression of non-cumulative cubes	100
6.4.7	Best practices for compressing large requests	100
Chapter 7. Best practices for DataStore		103
7.1	Differences between DataStore and ODS	104
7.2	Types of DataStore	104
7.2.1	Standard DataStore object	104
7.2.2	DataStore object for direct update	105
7.2.3	Write-optimized DataStore object	105
7.3	Secondary indexes on DataStore objects	105
7.3.1	Creating a secondary index on a DataStore object (DSO)	107
7.4	Partitioning and clustering	108
7.4.1	Choosing the partitioning key	110
7.4.2	Choosing and changing the clustering index	112

7.4.3 Data Partitioned Secondary Indexes (DPSIs)	112
7.4.4 Partitioning a DataStore object	114
7.5 DB2 compression	132
7.5.1 Materialized query tables	133
Chapter 8. Query and Load performance	135
8.1 Troubleshooting general system performance	136
8.2 Analyzing load performance	142
8.3 Analyzing query performance	157
8.3.1 Starting the analysis from ST03	158
8.3.2 Determining whether aggregates are required	161
8.3.3 Front-end processing time	164
8.3.4 Database performance	165
8.4 DB2 Visual Explain	174
8.5 SAP Query Analysis Tool	178
8.5.1 Display of SQL access plan	180
8.5.2 SAP Query Visualizer	186
8.5.3 Features of SAP Query Visualizer	190
8.5.4 Example of using the query analysis tool	192
Chapter 9. SAP BI Accelerator	195
9.1 The BIA architecture	196
9.2 Sizing BIA	197
9.3 System z requirements	198
9.4 BIA installation	200
9.5 BIA administration	201
9.6 Benefits of BIA	201
9.7 Overview	202
Chapter 10. Tips for SQL efficiency	203
10.1 SQL efficiency for DSO	204
10.2 SQL efficiency for InfoCube	211
10.3 Indexing of DSOs	219
10.3.1 Index design for post-DSO access	223
10.4 Indexing of InfoCubes	224
10.4.1 facttable indexing	224
10.4.2 Index design for dimension/master data tables	224
Chapter 11. Project Jupiter: large BI Accelerator scalability evaluation	227
11.1 Background	228
11.1.1 Project overview	228
11.2 Jupiter environment infrastructure	229
11.2.1 Hardware and software configuration	230
11.2.2 Network infrastructure	232

11.3	SAP BI database and InfoCube load	234
11.3.1	SAP BI database	235
11.3.2	BI database InfoCube load	235
11.4	Test case scenarios	238
11.4.1	BI Accelerator index creation	238
11.4.2	Multi-user query	246
11.5	Analysis and conclusions	250
11.5.1	Recommendations for configuration	252
11.5.2	Help getting started	252
Chapter 12. Concepts, activities, and terminology		255
12.1	Terminology: BI objects and processes	256
12.2	SAP BI information model	257
12.2.1	Other key elements in the information model	258
12.3	Dataflow in SAP BI	260
12.4	Information access	262
12.5	Hierarchies	263
12.6	Extended star schema (InfoCubes)	264
12.7	Data load into InfoCube	267
12.8	Aggregates	268
12.8.1	Aggregate example	269
12.8.2	Maintaining aggregates	270
12.9	Compression of requests	273
12.10	DataStore (DSO)	273
12.11	SAP BI naming conventions	276
12.12	The SAP BI functional components	277
12.13	SAP BI business content	279
Appendix A. Glossary for SAP BI		281
Appendix B. SAP BI installation and checks under DB2 and z/OS		315
	IT scenario	316
	Useful SAP documentation	317
	Checks under z/OS and DB2	318
	Installation of stored procedure SAPCL	318
Appendix C. SAP BW OSS Notes		321
	List of SAP BW OSS Notes	322
Appendix D. Project Jupiter: evaluation details		325
	Network configuration: static multipath	326
	SAP BI database	329
	BIA index creation	334

Related publications	347
IBM Redbooks	347
Other publications	347
Online resources	348
How to get IBM Redbooks	348
Help from IBM	349
Help from SAP	350
Index	351

Archived

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 about the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

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

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

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

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

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

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


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

COPYRIGHT LICENSE:

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

Trademarks

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

Redbooks (logo)  ®	DB2®	OS/390®
z/OS®	DRDA®	Parallel Sysplex®
zSeries®	DS4000™	POWER5™
z9™	FlashCopy®	Redbooks®
AIX®	FICON®	RMF™
BladeCenter®	General Parallel File System™	Storyboard™
Collation®	Geographically Dispersed	System p™
CICS®	Parallel Sysplex™	System p5™
Distributed Relational Database	GDPS®	System x™
Architecture™	GPFS™	System z™
DB2 Connect™	Informix®	System z9®
DB2 Universal Database™	IBM®	System Storage™

The following terms are trademarks of other companies:

BAPI, ABAP, SAP NetWeaver, mySAP, SAP R/3, SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

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

Snapshot, and the Network Appliance logo are trademarks or registered trademarks of Network Appliance, Inc. in the U.S. and other countries.

InfiniBand, and the InfiniBand design marks are trademarks and/or service marks of the InfiniBand Trade Association.

Java, JavaScript, JDBC, J2EE, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Access, ActiveX, Excel, Microsoft, PivotTable, SQL Server, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

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

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

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

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

Preface

This IBM® Redbooks® publication describes the benefits of DB2® 9 for z/OS® and SAP® Business Intelligence. It lists best practices and provides performance and tuning recommendations for SAP Business Intelligence.

In today's fast-moving business environment, there is a need for more and better business information. Data warehousing and business intelligence (BI) systems deliver that information. This imposes new requirements to manage, control, and use this data for business purposes.

SAP Business Intelligence is the central reporting tool for almost all SAP business solutions. It is based on building blocks called InfoObjects that contain data about customers, sales, and business information. InfoObjects include InfoSources, DataStore objects, and InfoCubes. The business intelligence solution from IBM and SAP can help you aggregate and leverage this information, giving you a system-wide view of your business data and delivering it across your enterprise to support sound business decisions.

However, this structure can lead to slow performance if the system is not set up and managed according to good database principles. This book describes best practices for this product on a System z™ platform, and provides performance and tuning recommendations for loading and querying data. It also addresses general system administration and troubleshooting.

The audience for this IBM Redbooks publication includes SAP and DB2 administrators. Knowledge of these products and of the z/OS environment is assumed.

This Redbooks publication is designed to aid the reader based on specific needs. After describing the product and summarizing its advantages on the zSeries® platform, we list best practices for the PSA, InfoCube, DSO, and DB2 components. We give tips on how to gain efficiency by using compression and indexing. We also include a troubleshooting chapter that describes what to do when best practices do not address a problem.

This edition of the book includes a new chapter (Chapter 11, "Project Jupiter: large BI Accelerator scalability evaluation" on page 227) and an appendix (Appendix D, "Project Jupiter: evaluation details" on page 325) that describes Project Jupiter, a joint venture between IBM and SAP to test the scalability and performance of a very large (25 terabyte) database.

The team that wrote this IBM Redbook

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

The third edition has added a chapter on BIA scaling, Chapter 11, “Project Jupiter: large BI Accelerator scalability evaluation” on page 227. The authors of this chapter were: Seewah Chan, Leticia M. Cruz, Veng Ly, Howard Poole.

These are the authors of the second edition of this book (all but chapter 11).



Lydia Parziale is a Project Leader for the ITSO team in Poughkeepsie, New York, with domestic and international experience in technology management including software development, project leadership, and strategic planning. Her areas of expertise include e-business development and database management technologies. Lydia is a Certified IT Specialist with an MBA in Technology Management and has been employed by IBM for 23 years in various technology areas.



Giovanni Grita is an Infrastructure IT Architect working in SAP Technical Sales in IBM Italy. He joined IBM in 1988, working as C++ Object Oriented developer, system programmer, branch office IT Specialist. He started working with SAP products in 1998 and has earned Basis Consultant certification. Giovanni is now involved in SAP capacity planning, sizing, and integration in IBM environments. He holds a degree in mathematics from “La Sapienza” University of Rome.



Georg Mayer is a Senior Development Engineer for the Joint SAP IBM Porting Team in Rot, Germany, focusing on SAP BI for DB2 on z/OS. He joined IBM Data Management in 2001 and has over 18 years of experience with ERP software, databases, OLTP, and performance tuning. For the last 10 years Georg has worked as a development expert for a variety of SAP database porting teams in Walldorf. Prior to working at IBM, he worked at Informix® and Atos. He holds a master's degree in Computer Science and Mathematics from the University of Stuttgart.



Theodor Kuebler is a Consultant and Trainer for SAP and Data Management products. He is founder of the Logos Informations Systeme (LIS) AG in Switzerland (www.lis-ag.ch), an IBM enabled Education Center for IBM Software (ECIS), and an IBM Business Partner, providing classroom and on-site courses in Data Management and Tivoli. He has 12 years of experience in implementation and tuning of SAP products on database management systems, such as DB2 UDB, Adabas-D, and Oracle on different platforms. He teaches SAP classes.



Martin Cardoso is an IT Architect leading the SAP Technical Team in Package Application Services in IBM Argentina. He joined IBM in 2002 from PwC Consulting, working as a SAP Basis Consultant and SAP Basis SME for the SSA region. He has more than 10 years of experience working with SAP technologies. His last Basis certification was SAP NetWeaver® Technical Consultant. His expertise includes storage and related software, database technologies, SAP NetWeaver technologies, SAP capacity planning, SAP performance and tuning, SAP heterogeneous migrations, and now he is specializing in designing SAP infrastructure architectures.



Lindy Crawford joined IBM in South Africa in 2005 and has six years of experience in SAP Basis Support. Her current role includes providing SAP Basis Support for local customers, which includes problem analysis and troubleshooting, installations, and upgrades. In addition, she is an IBM Certified Database Associate - DB2 Universal Database™ V8.1, IBM Certified Specialist - System p™, AIX® System Support and an IBM Certified Deployment Professional - TSM 5.3.



Wilfried Allé started as an IT Analyst and Programmer in an Austrian oil and gas company (OMV-AG) in 1972. He is an Advisory Programmer in PL/1 and Assembler, as well as in online programming (CICS®, VSAM, and IMS-DB). In 1987 he led his team to SAP's standard software R/2 on CICS and IMS-DB. He has trained hundreds of his colleagues in SAP fundamentals. Wilfried has also served as an IT lecturer in a high school setting. Currently, he works as a SAP Consultant for IBM Premier

Business Partner IT-Systeme und Consulting GmbH in Vienna, Austria, on a project basis.

Thanks to these authors from the Project Jupiter team for documenting the results of their testing in Chapter 11 and Appendix D:

Leticia Cruz works in the IBM Systems z Benchmark Center, Poughkeepsie, New York.

Veng Ly works in the IBM SAP/R3 Performance Test group in Poughkeepsie, New York.

Seewah Chan works in the IBM Systems Development SAP Performance group in Poughkeepsie, New York.

Howard Poole works in the IBM SAP/R3 Performance Test group in Poughkeepsie.

Rick Burns works for Wintercorp, which is based in Waltham.

Chris Williams works in the IBM Systems z Benchmark Center in Poughkeepsie, New York.

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

Richard M Conway
Mike Ebbers
Robert Haimowitz

International Technical Support Organization, Poughkeepsie Center

- ▶ Thanks to the authors of the first edition of this book, *Best Practices for SAP BI using DB2 9 for z/OS*, published in May 2005:
Brenda Beane, Dave Bellion, Nick Dyke, Andrew Hartman, Theodor Kuebler, Veng Ly, Mike Mardis, Bala Prabahar, Michael Thelosen
- ▶ Thanks to the authors of *SAP on DB2 for z/OS and OS/390: DB2 System Cloning*, SG24-6287:
Viviane Anavi-Chaput, Jan Baisden, Markus Dellerman, Don Geissler, Andrew Hartman, Mary Siart

Become a published author

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

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will 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 book or other Redbooks in one of the following ways:

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

ibm.com/redbooks

- ▶ Send your comments in an e-mail to:

redbooks@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Introduction

In any enterprise, access to up-to-date data is a vital part of daily operations. The business intelligence solution from IBM and SAP can help you aggregate and leverage this information to support sound business decisions. It gives you a system-wide view of your business data and delivers it across your enterprise.

We assume that you have some knowledge about this product. However, if you would like to read more about its concepts and terminology, see Appendix A, “Glossary for SAP BI” on page 281.

1.1 Product overview

The business warehouse solution from IBM and SAP can help your company collect, analyze, and distribute critical data. SAP Business Intelligence provides a full suite of end-user reporting and analysis tools, including the following:

- ▶ **Advanced analytics**
Statistical and mathematical functions with a full range of multidimensional analysis
- ▶ **Mobile business intelligence**
Accessible to a full range of wireless and mobile devices, providing anytime, anywhere access
- ▶ **Business content**
Report templates and metrics, tailored for different user roles in different vertical industries

This end-to-end implementation can give your business the ability to:

- ▶ Add commentaries to reports and key figures, essentially automating approval processes.
- ▶ Visually represent data analysis in a wide variety of formats including grids, graphs, charts, and maps through a unique toolset that allows for custom reporting environments.
- ▶ Analyze data from both internal and external sources, and link traditional business environments and e-business operations through data warehousing.
- ▶ Combine, view, and deliver structured information (such as database reports) and unstructured information (including e-mail), as well as internal and external data through your existing enterprise portal.

Many IBM Redbooks have been published about SAP systems on IBM System z. See “Related publications” on page 347. The following subjects have been covered in these publications:

- ▶ Reliability, availability, and serviceability of SAP systems on IBM System z
- ▶ Security and integrity
- ▶ Scalability
- ▶ Continuing compatibility
- ▶ Changing workload

These are beneficial subjects for business intelligence (BI) systems because SAP BI queries involve the batch processing of terabytes of data. The IBM mainframe solution is very strong in this area.

Furthermore, the SAP BI application is different from other SAP modules, especially regarding the use of a database. Thus, it is important to look into the performance of SAP BI with DB2 v9.1 for z/OS, as this Redbooks publication does.

1.2 The audience for this IBM Redbooks publication

The goals of this book are to give an integrated view of key SAP BI and DB2 9 for z/OS performance indicators and to offer a process for using them together.

The audience for this document includes both System z DB2 DBAs and SAP BI Basis Administrators responsible for administering SAP BI on System z DB2 production systems, and SAP BI functional architects responsible for designing the BI system's InfoCubes, Aggregates, and Data Storage Objects (DSO) objects.

Note: It is important that there is early cooperation between the functional designers and the technical implementation specialists. All those listed will benefit from reading this book.

The SAP BI solution is not the only DB2 9 solution within a company. Most companies have different early systems running on System z. For the IBM mainframe infrastructure, a support organization exists within the company. Most groups tend to work independently from each other.

With SAP BI on System z, as with other SAP solutions on the IBM mainframe, you must communicate with all groups that are responsible for the operation of the SAP solution. Ideally there should be a person in your organization who bridges the gap between the different support groups — the user acceptance and performance of your SAP solution will benefit.

The focus of this IBM Redbooks publication is on performance problems: how to avoid them, how to troubleshoot performance problems that cannot be avoided, and how to solve the problems once identified. Most performance problems can be avoided if foresight is used and design decisions take certain issues into consideration.

This book can also be used for decisions concerning SAP on System z, as discussed in Chapter 4, "Benefits of DB2 9 for SAP BI" on page 35. Key areas we cover are load performance, query performance, performance during administration processing, and storage reduction.

In general, database administrators (DBAs) are not responsible for some of the processes discussed in this book, for example, analyzing and defining SAP aggregates (which are summary tables optimized for specific queries). It is important, however, for DBAs to be aware of these activities, since aggregate analysis and definitions should generally be done before solving a problem, when the root problem is a lack of aggregates. Missing or improperly defined aggregates can cause increased CPU usage, excessive I/O activity, and increased DB2 memory usage.

Evaluation of query performance for SAP BI requires both SAP information (such as query performance indicators RSDDSTAT, or aggregate valuation and definition in RSA1) and also DB2 indicators (such as cached statement statistics, or using DB2 EXPLAIN PLAN to analyze the access path used by DB2).

Likewise, an SAP BI administrator may not be familiar with the technical issues of DB structure, SQL evaluation, and DB2 parallelism capabilities. But if, after evaluating the data model and aggregates, there are still performance problems, the SAP BI administrator can work with the DB2 DBA to evaluate the DB2-related changes to address performance problems. Not all problems can be solved with aggregates, and the BI administrator may need to turn over some problems to the DB2 DBA.

1.3 Why run SAP BI on IBM System z

The *z* in System *z* signifies zero downtime. System *z* hardware is designed to meet the world's most critical computing needs. Reliability characteristics such as dynamic CPU sparing, logical partitioning, error detection and correction, and hardware cryptography are built into the architecture.

If the nonspecialist IBM system offers 99.99% availability, the result is only 53 minutes of unplanned downtime per year. If the sysplex gives 99.999% availability, there is only 5 minutes per year of unplanned downtime.

1.3.1 Value provided by System z, z/OS, and DB2

In this section we give an overview of the value provided by System *z*, z/OS, and DB2.

One-stop shopping

IBM provides a one-stop shop for all three components — hardware, operating system, and database.

7 x 24 application availability

99.999% database reliability offers resilience to unplanned outages. Such database availability requires:

- ▶ A robust operating system. z/OS has more than one million lines of recovery code, and has been evolving since 1966.
- ▶ Reliable hardware with internal redundancy and error correction logic. z/990 has a mean time to failure (MTF) greater than 30 years.
- ▶ That a processor continues to operate after the loss of an engine.
- ▶ That there is no single point of failure (SPOF). This support is provided by Parallel Sysplex® with DB2 Data Sharing.
- ▶ Coupling Facilities, which fully support structure duplexing and automated takeover.
- ▶ Reliable DBMS. DB2 has been evolving for more than 20 years and has been stress-tested by more than 8,000 of the largest companies in the world.
- ▶ Dynamic CPU sparing. At a minimum, one of the z/Series processing units is designated as a spare. If a running CPU chip fails and the instruction retry is unsuccessful, the spare processing unit begins executing at precisely the instruction where the other failed.
- ▶ Activation of the spare is done completely by the hardware, enabling the system to be restored to full capacity in less than one second. Because most hardware failures requiring repair will not cause an outage at the time of failure or at the time of repair, the IBM System z achieves near-zero outage for unplanned hardware repairs.
- ▶ Minimal planned outages. This is also known as *continuous operations*:
 - Parallel Sysplex, coupled with DB2 Data Sharing and SAP Sysplex Failover, allows rolling maintenance and upgrades of hardware, z/OS, and DB2 without an outage.
 - DB2 Online Utilities eliminate database maintenance, which is the leading cause of planned outages in a UNIX® or Windows® environment.
 - All of this is further complemented by hot-pluggable features, including capacity on demand of System z and z/OS.

Capacity and scalability

Using a System z means that you have both vertical and horizontal capacity and scalability:

- ▶ Vertical growth is achieved through more or faster engines in a single symmetric multiprocessor (SMP) — the traditional growth path for UNIX and Windows.

- ▶ Horizontal growth is achieved by clustering SIMPs with a single copy of the database (also known as *parallel operation*).
- ▶ Parallel Sysplex provides horizontal growth: 1–32 engines x 32 CECs.
- ▶ DB2 Data Sharing is the only SAP-certified generally available (GA) clustered DBMS. DB2 Data Sharing has been certified since 1997.
- ▶ DB2 Data Sharing has been in use in customer SAP environments since 1997.
- ▶ 64-bit real System z storage is in use by many SAP customers. z/OS sustains a higher I/O rate than other platforms with 6–10 K I/Os/sec per I/O controller.
- ▶ DB2 v9.1 delivers 64-bit virtual storage, which allows more data in memory and the easing of many system constraints related to addressability.
- ▶ Improved database-intensive performance

Application servers running under z/OS provide data transfer using cross-memory services. Linux® application servers on System z can use IBM unique hipersockets for application server-to-database server communications.
- ▶ Sysplex with Coupling Facility.
- ▶ IO bandwidth, data in memory.

Large database manageability

System z provides large database manageability:

- ▶ Multi-terabyte databases with hardware compression.
- ▶ Online backup and reorganizations.
- ▶ Storage technology (FlashCopy®, Snapshot™).
- ▶ Self-tuning DASD — PAV, multiple allegiance.
- ▶ Leverage and protect investment.
- ▶ Multiple LPARs and priority schemes:
 - DB2 hardware data compression is unique, and affects both performance and cost.
 - UNIX and Windows solutions cannot afford the overhead of software compression.
 - On average, UNIX and Windows require 50% more disk than DB2 9 for z/OS.
 - Customers typically have 4 to 10 copies of their production database, which quickly grows into multi-terabyte size.
 - z/OS and Linux application servers.

- MCOB - System z first - reduces complexity.
- Proven automation and end-to-end systems management.
- DR ready - GDPS® - cold, warm, *and* hot.
- Non-disruptive CBU on/off.
- z/OS and DB2 skills and tools.
- Online backup and reorganizations (DB2 Online Reorg solutions are unique) eliminate the major cause of planned outages.
- Storage technology (FlashCopy, Snapshot). Integration of the disk controller database copy with the DBMS is unique with DB2 and ESS providing seamless, low overhead database backup.
- Self-tuning DASD - Parallel Access™ Volumes, multiple allegiance. Disk controller Parallel Access Volumes (PAVs) with z/OS eliminate disk *hot spot* tuning typical of a UNIX or Windows database server environment. PAV operates across LPARS for multiple allegiance.
- ▶ Leverage and protect investment. Some System z/DB2 capabilities that provide total cost of ownership (TCO) and service level advantages are:
 - Multiple LPARs and priority schemes.
 - z/OS and Linux application servers.
 - MCOB - System z first - reduce complexity.
 - Proven automation and end-to-end systems management.
 - Disaster Recovery (DR) ready using a Geographically Dispersed Parallel Sysplex™ (GDPS), which is a S390 implementation that enables high availability across sites up to 40 km apart — cold, warm, *and* hot.
 - Capacity backup (CBU), primary site 12 engines, DR site 6 engines, in event of failover dynamically turn on additional engines.
 - Peer-to-peer remote copy (PPRC).
 - Asynchronous remote copy (XRC).

Note: For more information about product concepts, see Appendix A, “Glossary for SAP BI” on page 281.

For more information about the benefits of IBM DB2 9 for z/OS and System z in your SAP BI environment, see Chapter 4, “Benefits of DB2 9 for SAP BI” on page 35.

Archived



Overview of SAP BI and related data

This chapter provides an overview of the SAP BI architecture and the most important SAP BI data objects. We introduce and explain the different kinds of tables for master data, cubes, PSA, and Data Storage Objects (DSOs) and introduce the BI snowflake schema. We also provide an abstract on the usage of table partitioning and DPSIs.

We also give an overview about some important new features and functions of SAP BI and provide a brief explanation of the most important BI terminology.

2.1 BI architecture

Figure 2-1 shows the most relevant objects used in SAP BI. From the database point of view these objects are master data, persistent staging area (PSA), ODS (operational data store, recently renamed to DSO for data store objects), and cubes.

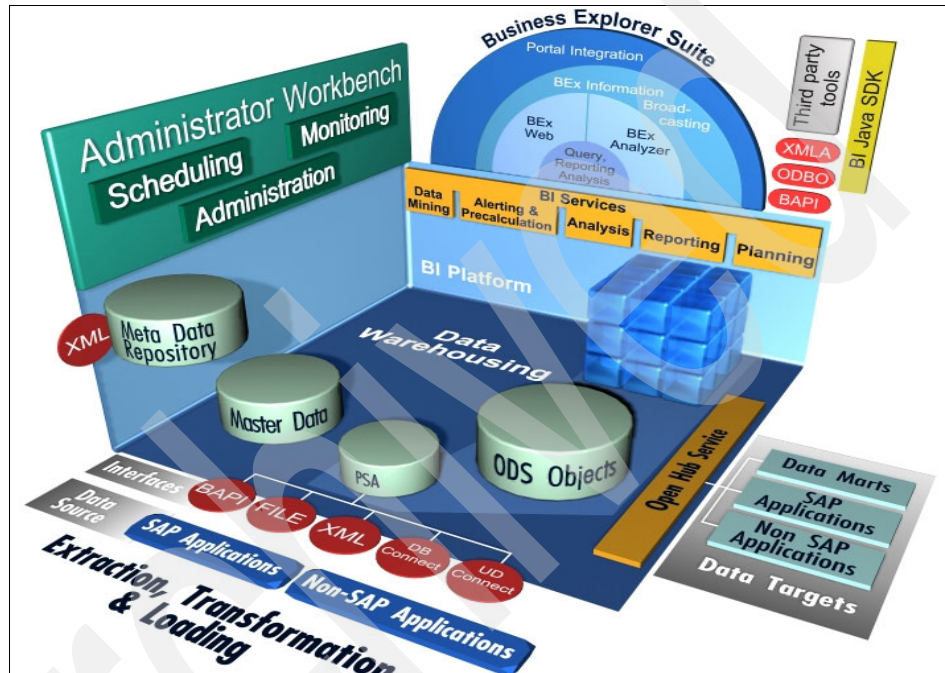


Figure 2-1 SAP BI architecture

We talk about these objects and their related database tables later.

A *PSA* is a transparent database table in which request data is stored in the form of the transfer structure. A PSA is created per DataSource and source system. It represents an initial store in BW, in which the requested data is saved unchanged for the source system.

An *InfoCube* (or simply *cube*) describes a self-contained dataset (from the reporting view), for example, for a business-oriented area. This dataset can be evaluated with the BEx query. Furthermore, an InfoCube is a quantity of relational tables that are created according to the star schema (a large E fact table in the center, with several dimension tables surrounding it).

A *DSO*, or data store object, is a data administration layer that saves cleaned data in flat, transparent tables. This layer can make data available in real time and enables operational reporting.

Figure 2-2 shows the key components of SAP BI shown within the three pillars Access, Business Logic, and Presentation. This book only covers the first pillar, Access, to the different kinds of BI data.

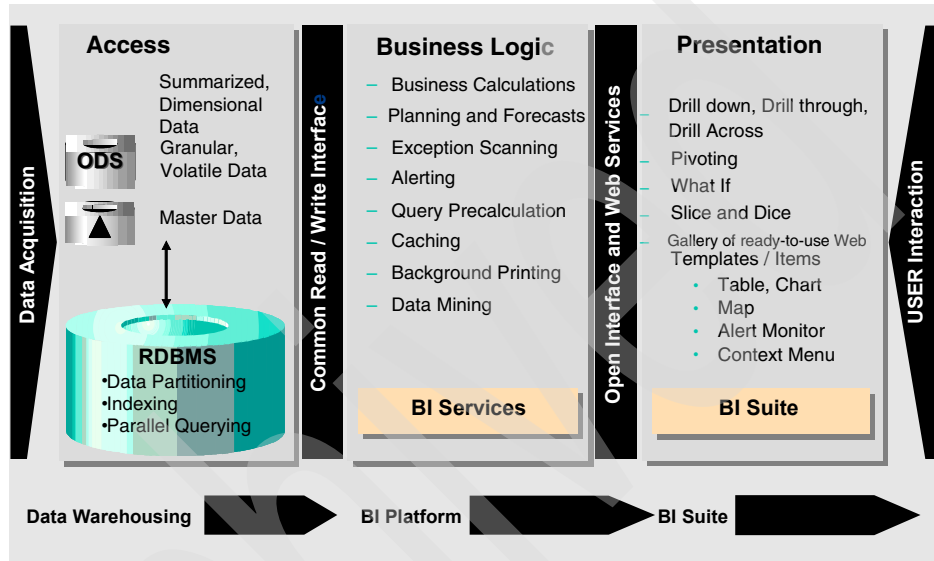


Figure 2-2 SAP BI key components

Terminology changes for SAP NetWeaver 2004

As of SAP NetWeaver 2004, some existing BI terms have been changed to better reflect the enhanced functions of the tools and objects. See Table 2-1.

Table 2-1 Terminology changes for SAP NetWeaver 2004s

New term	Old term
DateStore Object (DSO)	ODS Object
DateStore object for direct update	Transactional ODS object
Real-time InfoCube	Transactional InfoCube
Virtual Provider	RemoteCube, SAP RemoteCube, InfoCube with services
Data Warehousing Workbench	Administrator Workbench
Extraction Monitor	Monitor

New term	Old term
Analysis authorizations	Reporting authorizations
Analytic Engine	OLAP Engine
BI Query Runtime Statistics	OLAP Statistics

2.2 Which cubes, DSO, and PSA objects are in your BI system

This section describes how to determine which InfoCubes, data storage objects, and persistent staging areas are in your SAP BI system.

2.2.1 Report SAP_BWTOOLPGM_DB2

If you want a quick overview over the data in your BW system and to search for answers to, for example, the following questions, you can run report SAP_BWTOOLPGM_DB2 by transaction SE38 to get answers. Look at SAP note 859583 to get more detailed information about this report. Figure 2-3 and Figure 2-4 on page 14 show an example for the left and right halves of the output list of the report listing cubes in the SAP BI system.

- ▶ How big are the DSO/PSA objects or cubes and aggregates?
- ▶ Which of my BW objects (PSA/DSO/ fact tables) are partitioned and how many partitions do they have?
- ▶ Which are your widest fact tables?
- ▶ Which non-cumulative cubes are there?
- ▶ In which PSA table is space wasted?
- ▶ What do the dimension cardinals of my cubes look like?

List of all cubes, sorted after the cardinality of E-facttable
cardinalities are read from sysibm systables

a "*" in the compression info shows that not all partitions have the compression flag set or are physical compressed
"Y" means, that the table must be reorganized first, only the flag is set

Compression rate

Basiccube	Ncols Inventory	part_field	E-facttable	partitions		DPSI		compression		cardinalites	card E
				#E	#F	#E	#F	E%	F%	card F	
ZSD_C01	25	0CALMONTH	/BIC/EZSD_C01	133	1	5		Y			32.120.000
ZSD_C01_U	25	0CALMONTH	/BIC/EZSD_C01_U	133	1	2	5	67*	Y		32.120.000
ZSD_C01_2	25	0CALMONTH	/BIC/EZSD_C01_2	25				Y	Y		5.840.000
ZSD_C01_1	25	0CALMONTH	/BIC/EZSD_C01_1	25				Y	Y		5.840.000
ZSD_C01_3	25	0CALMONTH	/BIC/EZSD_C01_3	25				Y	Y		5.840.000
JOPART3	9	0CALMONTH	/BIC/EJOPART3	25				Y	Y	20	4.121.865
JOBEST12	6	0CALDAY	/BIC/EJOBEST12					Y	Y		4.023.576
ZSD_C02	25	0CALMONTH	/BIC/EZSD_C02	133				Y		33.086.858	2.728.000
JOCUBE	9	0CALMONTH	/BIC/EJOCUBE	25				Y	Y		969.919
OLDPART	9	0CALMONTH	/BIC/EOLDPART	25							960.668
JOBEST1	6	0CALMONTH	/BIC/EJOBEST1					Y	Y	100.000	85.805
JOBEST4	6	0FISCPER	/BIC/EJOBEST4					Y	Y		28.890
TEST1	9	0CALMONTH	/BIC/ETEST1	25				Y*			3.000
MUNI_DECU	17	0CALMONTH	/BIC/EMUNI_DECU	13	2		6				417
TEST3	9	0CALMONTH	/BIC/ETEST3	25	1		4	Y	Y	1-	12
GMPART	9	0CALMONTH	/BIC/EGMPART	13	2		3	Y	Y	1-	10
JOCUBE2	9	0CALMONTH	/BIC/EJOCUBE2	25	2		4	Y	Y		10
FIRSTCUBE	12		/BIC/EFIRSTCUBE		5		6	Y	Y	421	2
VENGCUBE1	9	0CALMONTH	/BIC/EVENCUBE1	25							1
TESTPART	9	0CALMONTH	/BIC/ETESTPART	25	1						
JOIT	6		/BIC/EJOIT		2						
0BWTC_C09	13		/BIO/EOBWTC_C09								
0BWTC_C07	8		/BIO/EOBWTC_C07								
0BWTCFC1	17		/BIO/EOBWTCFC1								
0BWTCFC2	12		/BIO/EOBWTCFC2								

Cumulative cubes

F-facttable has 6 different DPSI indices

Figure 2-3 Output list of report SAP_BW_TOOLPGM_DB2 for cubes

Tablespaces																
E_KB_alloc	F_KB_alloc	F_KB_used	E-fact	F-fact	BPool E	BPool F	Cube	Dimension cardinalities					T=Table	V=View		
9.769.164	288		EZSDXC0	FZSDXC0	BP11	BP8	ZSD_C01	T1-100001	T2-100001	T3-80920	T4-100	T5-100				
6.870.440	48	22	EZSDXC0	FZSDXC0	BP8	BP8		1003	T3-80921	T4-100	T5-101					
1.656.984	336		EZSDXC0	FZSDXC0	BP8	BP8		1002	T3-80922	T4-100	T5-100					
1.677.488	336		EZSDXC0	FZSDXC0	BP8	BP8	ZSD_C01_1	T1-100003	T2-100001	T3-80923	T4-100	T5-100				
1.677.528	336		EZSDXC0	FZSDXC0	BP8	BP8	ZSD_C01_3	T1-*	T2-*	T3-*	T4-*	T5-*	Tp-*	Tt-*		
577.840	409.396	141	EJOPART	FJOPART	BP8	BP8	JOPART3	T1-6	T2-*	T3-*	Tp-3	Tt-13	Tu-2			
410.696	92.496	13	EJOBEST	FJOBEST	BP8	BP8	JOBEST12	T1-*	T2-*	Tp-*	Tt-*					
540.232	9.680.360	9.512.821	EZSDXC0	FZSDXC0	BP8	BP2	ZSD_C02	T1-100001	T2-100001	T3-80920	T4-100	T5-100				
51.980	240		EJOCUBE	FJOCUBE	BP8	BP8	JOCUBE	T1-6	T2-*	T3-*	Tp-*	Tt-8	Tu-1			
268.364	82.416		EOLDPAR	FOLDPAR	BP8	BP8	OLDPART	T1-6	T2-22	T3-10001	Tp-1	Tt-13	Tu-2			
54.256	54.704	7.647	EJOBEST	FJOBEST	BP8	BP8	JOBEST1	T1-*	T2-*	Tp-*	Tt-*					
32.244	30.960		EJOBEST	FJOBEST	BP8	BP8	JOBEST4	T1-*	Tp-*	Tt-*	Tu-2					
20.924	240		ETEST1	FTEST1	BP8	BP8	TEST1	T1-6	T2-22	T3-2583	Tp-1	Tt-13	Tu-2			
340			EMUNIXD	FMUNIXD	BP8	BP8	MUNI_DECU	T1-33	T2-26	T3-9	T4-2	T5-9	Tp-1	Tt-19	Tu-5	
16			ETEST3	FTEST3	BP8	BP8	TEST3	T1-*	T2-*	T3-*	Tp-*	Tt-*	Tu-*			
212	4		ECMPART	FCMPART	BP8	BP8	CMPART	T1-*	T2-*	T3-*	Tp-*	Tt-*	Tu-*			
220	1.104		E					6	T2-10	T3-11	Tp-1	Tt-8	Tu-2			
292	1.480		E					33	T2-26	T3-9	T4-2	T5-9	Tp-5	Tt-19	Tu-5	
196	240		EVENGCU	FVENGCU	BP8	BP8	VENGCU1	T1-2	T2-2	T3-3	Tp-1	Tt-2	Tu-2			
144					BP8	BP8	TESTPART	T1-1	T2-1	T3-1	Tp-1	Tt-1	Tu-1			
96					BP8	BP8	JOIT	T1-148	T2-2200	Tp-2	Tt-13	Tu-2				
144					BP2	BP2	0BWTC_C09	V1-1	T2-1	Tp-1	Tt-1					
192	240		0BWTCT	F0BWTCT	BP2	BP2	0BWTC_C07	V1-1	T2-1	T3-1	Tp-1	Tt-1				
48	48	75	0BWTCT	F0BWTCT	BP2	BP2	0BWTCTC1	T1-1	T2-1	T3-1	T4-1	T5-1	T6-1	T7-1	Tp-1	Tt-
240	288		0BWTCT	F0BWTCT	BP2	BP2	0BWTCTC2	T1-1	T2-1	T4-1	T7-1	Tp-1	Tt-1	Tu-1		
240	288		0BWTCT	F0BWTCT	BP2	BP2	0BWTCTC2	T1-1	V2-1	V3-1	T4-1	Tp-1	Tt-1			
288	64.392	2.144	EJOSDXC	FJOSDXC	BP8	BP8	JOSD_C01	T1-2	T2-1001	T3-2	T4-2	T5-1	Tp-2	Tt-13	Tu-2	
144	192		0BWTCT	F0BWTCT	BP2	BP2	0BWTCT_C03	T1-1	V4-1	Tp-1	Tt-1					
192	240		0BWTCT	F0BWTCT	BP2	BP2	0BWTCT_C06	V1-1	T2-1	T3-1	Tp-1	Tt-1				
48	48	35	0BWTCT	F0BWTCT	BP2	BP2	0BWTCT_C04	T1-1	V2-1	V3-1	Tp-1	Tt-1				

Dim 2 has the cardinality 100.001

* means, that table Dim1 is missing statistics

negative values means missing statistics

V (View) shows that Dim3 is a line-item

Figure 2-4 Output list of report SAP_BW_TOOLPGM_DB2 for cubes (right)

2.3 BI data and DB2 partitioning

Database administrators often need to know which tables are going to be the largest ones because these are candidates for DB2 data partitioning. While we discuss the partitioning of PSA, DSO, and cubes and all of the related specifics later in this book, candidates for partitioning along with their attributes that make them good candidates for partitioning are:

- ▶ Persistent staging area (PSA) - large tables, large row size, flat table structure
- ▶ Data Warehouse and Data Store Objects (DSO - listed in Figure 2-5 on page 15 as ODS) - large tables, large row size, flat table structure
- ▶ Multi-dimensional models—fact tables: large tables

Note that in Figure 2-5 the Administrator Workbench is now called the Data Warehousing Workbench.

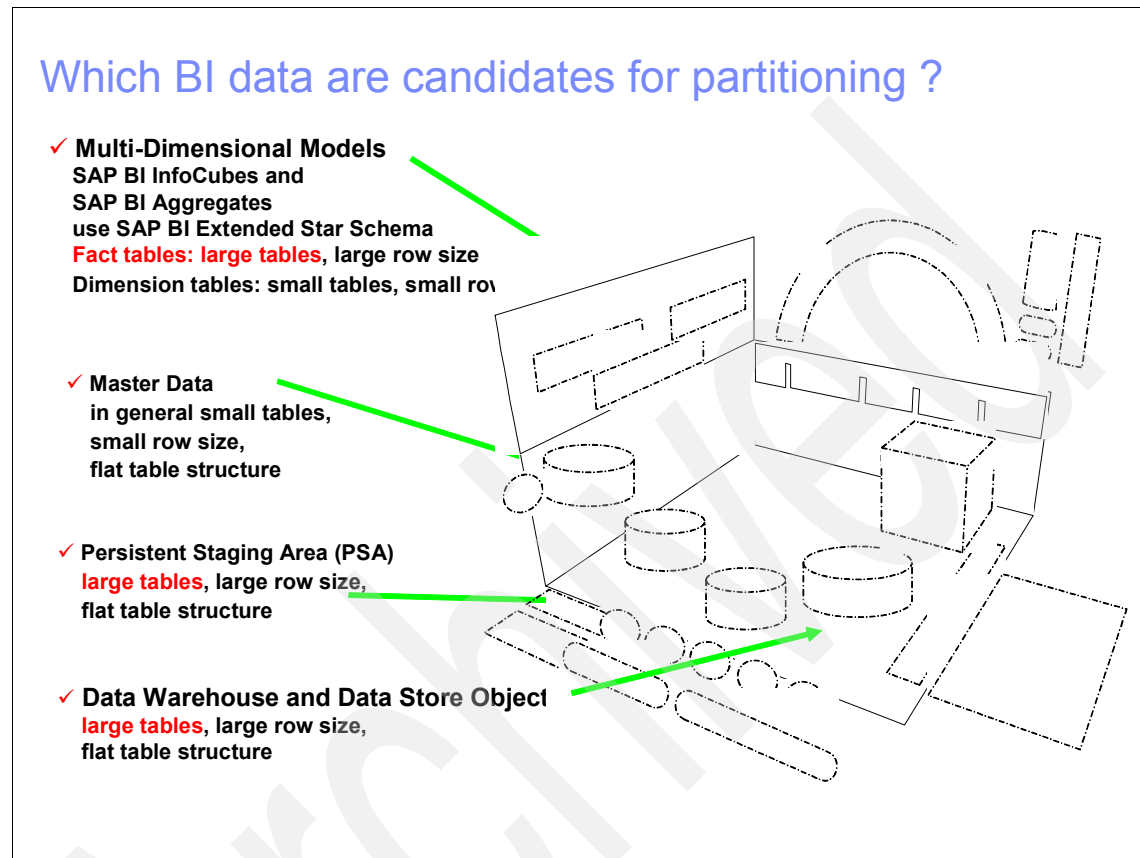


Figure 2-5 BI data and DB2 partitioning

2.3.1 Why partitioning

The reasons for partitioning tables within SAP BI are:

- ▶ Better performance
 - Faster deletion of a whole partition by using LOAD NULLFILE (note #894045) RSDAMIN parameter:
`DB2_OVERWRITE_PART_BY_NULLFILE = X`
 - For INSERT/DELETE by using partitioned indexes (DPSI)
Smaller index tree of every partition instead of one huge tree

- When calculating database statistics (RUNSTATS)
RUNSTATS on partition level
- At query processing by using partition pruning
BI generates an additional predicate on the E- fact table partitioning column if there is a restriction:
and F.“SID_0CALMONTH“ between 200602 and 200604
- Support of parallel SQL operations and parallel utilities (REORG, REBUILD INDEX, RUNSTATS, LOAD, and so on)
- ▶ Larger total table size
 - Depends from DSSIZE (<=64 GB), page size (4/8/16/32 KB) and maximum number of partitions (V7: 254, V8: 4096)
 - Up to 128 TB with DSSIZE=64GB and 32 K page size

2.3.2 Range partitioning of F- fact tables

Range partitioning for E- fact tables can be turned on at the cube maintenance panel. Figure 2-6 discusses range partitioning further.

- Select Extras -> Partitioning
choose between calmonth and fiscal period
- To avoid the E-facttable partitioning for small aggregates use RSADMIN parameter **DB2_AGGR_NOT_PARTITIONED** (note #703480)

for all aggregates of a given cube
DB2_AGGR_NOT_PARTITIONED_1 = MYCUBE

for certain aggregates
DB2_AGGR_NOT_PARTITIONED_2 = 100171
DB2_AGGR_NOT_PARTITIONED_3 = 100190
- **How to partition non empty E-fact tables later on ?**
 - BW 3.5: ABAP Report Z_SAP_BW_CONV_TO_PART_CUBE_DB2 (note #752725)
 - BW 7.0: Repartitioning toolbox for a cube
 - Complete Repartitioning (change of partitioning column possible)
 - Extension of a Rangepartitioned Cube

Figure 2-6 E- fact table partitioning

If you specify a partitioning schema for the E- fact table when you define an InfoCube, the E- fact tables of all aggregates defined for this InfoCube are also partitioned according to this schema. However, that partitioning often does not make sense, especially in the case of small aggregates.

Partitioning of F-Fact table

- Partitioned by request (DIMID of Requestnumber)
- BW 3.x (new) since SP30 / SP24 / SP16 for (3.0 / 3.1 / 3.5)
- by setting RSADMIN parameter **DB2_PARTITIONED_F_FACT** and **DB2_AGGR_PARTITIONED_F_FACT** as desired. (note #917568) for all cubes (recommended):
DB2_PARTITIONED_F_FACT = X
for all aggregates:
DB2_AGGR_PARTITIONED_F_FACT = X

Or only for certain cubes or aggregates:
DB2_PARTITIONED_F_FACT_1 = CUBE2
DB2_AGGR_PARTITIONED_F_FACT_2 = 100123
DB2_AGGR_PARTITIONED_F_FACT_3 = CUBE2
- No manual repartitioning necessary
 - Will be done at compression time, when the F-facttable becomes empty after compression of all requests

Figure 2-7 E- fact table partitioning

The performance can improve in some areas by partitioning the F- fact table. In particular, requests can be deleted significantly faster. In addition, you can restrict the update of database statistics to partitions whose contents have actually changed. The performance of queries also increases because access to partitions can be restricted with relevant data and parallel processing can be used efficiently.

If an F- fact table is partitioned, it has a dedicated partition for each request (that is, for each package dimension ID).

If a package dimension ID that does not yet have a partition is created during the update into the InfoCube, a new partition is created by the ADD PARTITION SQL statement. The limit of this new partition is the package dimension ID.

After a request is deleted, a check is carried out on all partitions that exist before the partition with the request to be deleted to see whether they are empty. If this is the case, these partitions are moved to the end of the E- fact table by the

ROTATE SQL statement. This defines the partition limits in such a way that they are one size greater in each case than the limit of the preceding partition. These partitions are used during the next update into the InfoCube.

As a result of this procedure, the F- fact table generally has more partitions than requests. To ensure that the number of partitions does not become too large, a check is carried out when the contents of the F- fact table are completely deleted to see whether this fact table has 25 or more partitions. If this is the case, the table is deleted and created again with one partition only. You can overwrite the threshold value of 25 using the DB2_FFACT_DROP_DEL_NUM_PARTS parameter in Table RSADMIN.

2.3.3 Partitioning of PSA tables

This section is about partitioned PSA tables. Partitioning in principal subdivides big data clusters into more and smaller ones. The benefits of partitioned PSA tables are faster loading and also faster deleting data as permanently needed in volatile tables. Not to expand the number of partitions more than necessary, there exists the rotating principle of partitioned tables, shown in Figure 2-8.

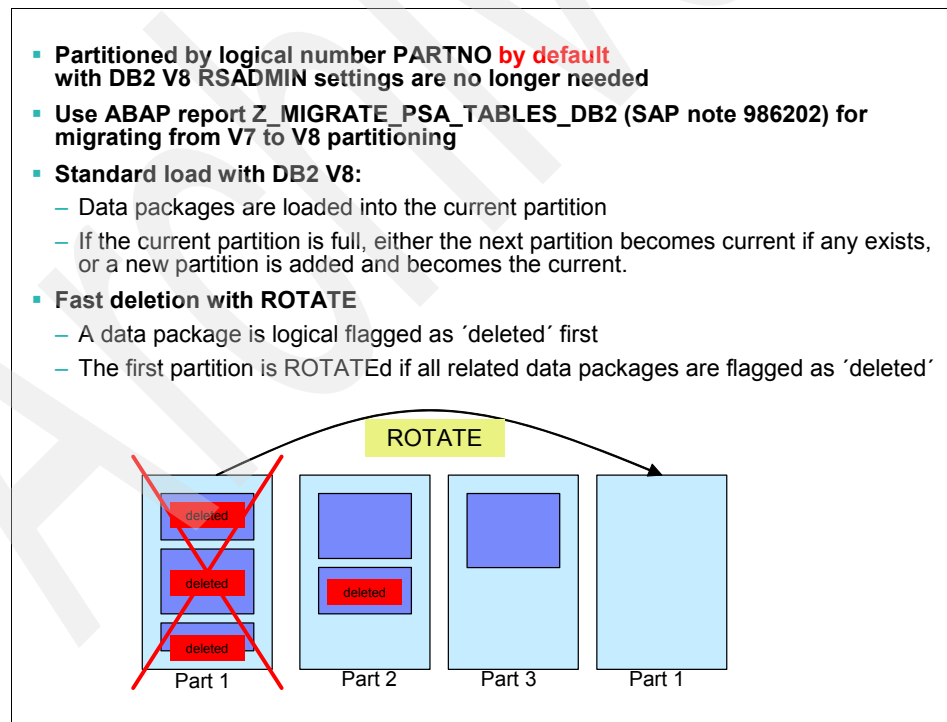


Figure 2-8 Partitioning of PSA tables

Important: Never partition a PSA table manually.

If you want to enforce the partitioning for all existing nonpartitioned PSA tables, you have to create a new version for this PSA. There is one easy way to do this automatically for all nonpartitioned PSA tables. Run the Postmigration Report, RS_BW_POST_MIGRATION, and choose only the option L_PSAVER (PSA version). Make sure that you remove all the other options and choose *only* the option L_PSAVER. Then execute this report. It will then create a new partitioned PSA version for every nonpartitioned PSA table.

2.3.4 Partitioning of DataStore objects

This section is about partitioned DSO tables. Partitioning in principal subdivides big data clusters into more and smaller ones. DSOs provide granular data and therefore might grow to big tables. The benefits of partitioned DSO tables are faster loading data and also faster querying data because of the possibility of parallelism.

Figure 2-9 highlights some points of partitioning DSOs.

Partitioning depends of the DSO type:

- **1. Standard DSO.**
 - Rel >= 3.5: **Changelog table** /BIC/B<10-digit-number> of DSO is PSA-like partitioned, by default
 - Rel > 7.0: **active DSO table** /BIC/A<ODS-Objekt>00 is range-partitioned, same manual configuring and handling as for cubes, but
 - Partition-field calmonth or fiscper is alphanumeric, instead of numeric as for cubes
- **2. Write-optimized DSO**
 - The **active DSO table** /BIC/A<ODS-Objekt>00 is PSA-like partitioned by default
- **3. direct write (former transactional) DSO**
 - No partitioning possible

Figure 2-9 Partitioning of DataStore objects

2.3.5 Usage of data partitioned secondary indexes (DPSI)

Data partitioned secondary indexes (DPSIs) provide the capability to partition an index according to the table-controlled partitioning specification of the underlying table. Each partition of a DPSI is a separate B-tree.

DPSI ensures full partition independence at the physical level. For example, there is no REORG BUILD2 phase for these indexes. Deleting a single partition (via LOAD REPLACE with empty input file) becomes really fast because there is no need to remove entries from secondary indexes one by one. Recovering can be done on a per-partition basis, which reduces the amount of data that needs to be recovered as well as enables a higher degree of parallelism.

The only downside of DPSI is potential access path deficiencies. If a query includes only predicates for the DPSI columns, each of the DPSI partitions (B-trees) will need to be traversed. With NPSIs (non partitioned index) there is only a single traversal. On the other hand, if the query includes a column that allows partition pruning, DPSI can be more efficient than a NPSI. The DB2 optimizer has been enhanced to take the new indexing structure into account and selects the optimal access path.

Figure 2-10 explains the difference between DPSI and NPSI for a table with two partitions.

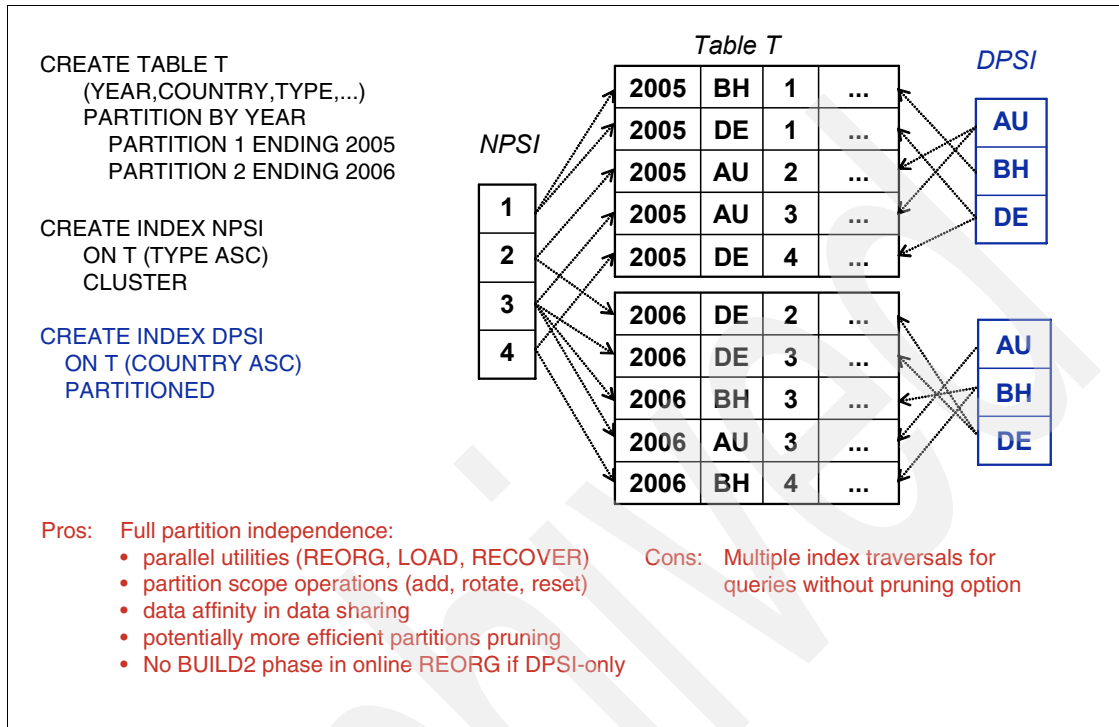


Figure 2-10 DPSIs versus NPSIs

Figure 2-11 shows the various indexes that are created for fact tables on the DB2 database. The P means packet-dimension, T stands for time dimension, and U stands for unit dimension. D1 and D2 are for dimension 1 and dimension 2, respectively, and kf is for keyfigure. So you see, there are some differences between the indexes for E and F- fact tables.

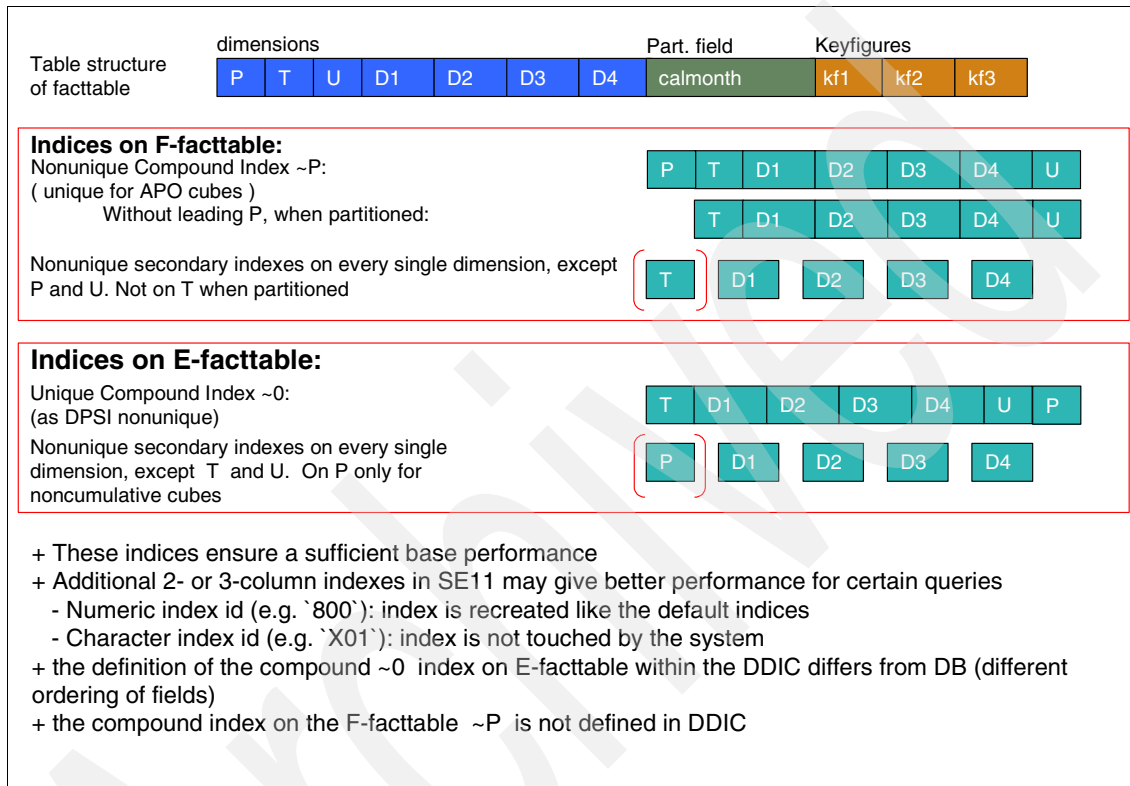


Figure 2-11 Indices for fact tables

Finally, Figure 2-12 explains which indexes of the partitioned tables are partitioned indexes.

F-Facttables

- Indices of partitioned F-Facttables are always partitioned

E-Facttables

- Default: no partitioned indices
- by setting RSADMIN parameter DB2_PART_E_FACTIDX and DB2_AGGR_PART_E_FACTIDX as desired (note #917568).
 - + for all cubes and aggregates (recommended):
DB2_PART_E_FACTIDX = X
DB2_AGGR_PART_E_FACTIDX = X
 - + Or only for certain cubes or aggregates:
DB2_PART_E_FACTIDX_1 = CUBE2
DB2_AGGR_PART_E_FACTIDX_2 = 100153
DB2_AGGR_PART_E_FACTIDX_3 = CUBE2

DSO and PSA tables

- Indices are never partitioned
- Will be supported later by SAP (DPSI as unique index)

Figure 2-12 Usage of DPSIs in SAP BI

2.4 SAP BI snowflake schema

In Figure 2-13 the BI snowflake schema for a cube is shown. Here we have the E- fact table in the middle, surrounded by the dimension tables, which themselves are joined to different kinds of SID tables. SID is an abbreviation for surrogate identifier. SIDs are used to create small unique identifiers for master data and attributes. This figure also shows the SAP BI naming convention and pattern for the different kinds of tables.

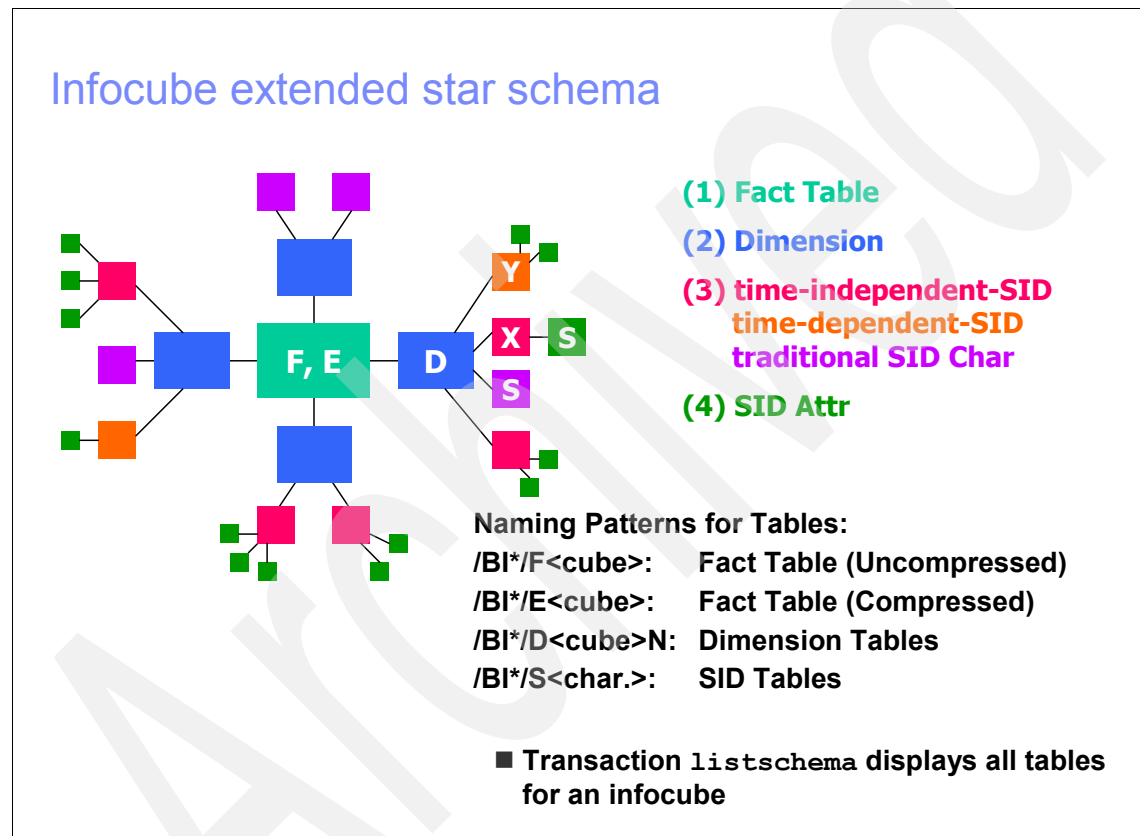


Figure 2-13 BI snowflake schema

Figure 2-14 shows how the snowflake tables are linked together by dimension identifiers (DIMIDs) and SIDs. This example shows some rows of snowflake tables from cube IUSALES. These lists are taken by using transaction SE16 for the respective table. The red arrows show the linkage of DIMIDs and SIDs from the E- fact table to the dimension table, from the dimension table to the X-table, and finally to the S-table of attribute IUPOP.

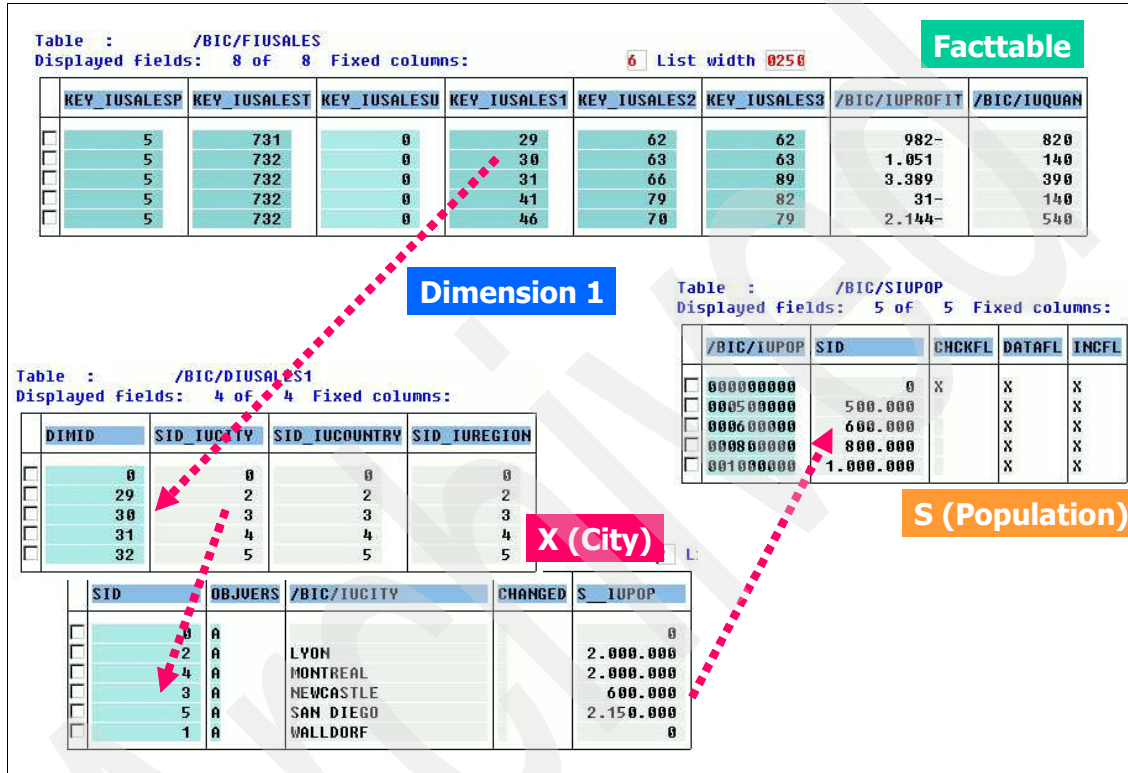


Figure 2-14 cube IUSALES with tables and data

2.5 New features and functions of SAP BI 7.x

Within this book we talk about the following new functions and features of SAP BI 7.x:

- ▶ SAP BI Accelerator

SAP BIA is discussed in Chapter 9, “SAP BI Accelerator” on page 195.

- ▶ SAP Analyze Tool
This new query analyze tool is explained in 8.5, “SAP Query Analysis Tool” on page 178.
- ▶ Repartitioning of InfoCubes and DSO objects
This is discussed in 6.2, “Repartitioning of InfoCubes” on page 78, and 7.4, “Partitioning and clustering” on page 108, respectively.
- ▶ Remodelling of cubes
See 6.3, “Remodeling InfoProviders” on page 89.

2.6 Terminology: BI objects and processes

Here we review some terminology.

- ▶ InfoProvider
In SAP BI, objects that can be analyzed are called InfoProviders. There are two classes of InfoProviders: those that contain physical data (such as InfoObjects, InfoCubes, and DSO objects) and those that do not (such as InfoSets, RemoteCubes, and MultiProviders).
- ▶ MultiProvider
A MultiProvider provides access to data from several InfoProviders and makes it available for reporting and analysis.
- ▶ Aggregate
A materialized, summarized view of the data in an InfoCube, this data can be read directly from the database during query execution without having to be built at run time.
- ▶ Roll-up
This is the process of loading new data packets from an InfoCube into an existing aggregate.
- ▶ Condense
This refers to using a program to compress the contents of an InfoCube- fact table.
- ▶ SQL efficiency
This refers to a method to simplify the analysis of a query against an InfoCube, as described in Chapter 6, “Best practices for InfoCubes” on page 73.

- ▶ **Realignment**

This refers to the process of synchronizing (realigning) all related fields after a change is made to one data field.
- ▶ **Tablespace partitioning**

This refers to the process of reducing the physical size of a large tablespace into multiple smaller physical parts, according to a common key field — frequently time-based. Partitioned tablespaces allow parallel access across multiple partitions. Partitioned tablespaces also allow the accessing of individual partitions without affecting other partitions.
- ▶ **DB2 compression**

This process utilizes compression routines, supplied by IBM, to compress DB2 data at the hardware level.
- ▶ **SAP compression**

This refers to the consolidation and summarization of data from an InfoCube's F- fact table into its E- fact table. Compression results in the Package ID being stripped off and all records summarized into the E- fact table.
- ▶ **Data granularity**

This refers to the difference between the frequency that data is updated in an InfoCube's F- fact table compared to the frequency in the E- fact table. It ranges from low to high.
- ▶ **Low granularity**

After compression, a high number of records exist in the F- fact table in relation to the E- fact table. Take an example where an InfoCube holds a weekly view of the data, and sales orders were entered, accessed, and updated many times during the week. There would be many records in the F- fact table, which would be compressed into a single record in the E- fact table.
- ▶ **High granularity**

After compression, the same number of records is found in the F- fact table as in the E- fact table. In our example, if sales orders were entered only once a week for each customer, there would be a one-to-one relationship between the data in the F- fact table and the compressed data in the E- fact table.



Top ten SAP BI recommendations

We have worked with many customers who have implemented SAP BI on System z, and we have noticed that there are recognizable patterns apparent when solving their performance problems. As a result, we have updated the top ten list of recommended actions that will greatly reduce the occurrence of performance problems and allow technical staff to move more quickly to the troubleshooting phase.

3.1 Base recommendations - SAP BI implementations for DB2 on z/OS

There is a base set of recommendations for SAP BI implementations on DB2 9 on z/OS. When followed, customers have demonstrated greater maintenance and query stability, resulting in better performance of the SAP BI on System z installation.

These recommendations are:

1. Review your data model with respect to query performance.

We strongly recommend being careful in your BI modeling. If your modeling has any weakness, this can lead to performance trouble later on and there is no guarantee that these performance issues can be solved by tuning the DB2 database. A good way is to follow the recommendations from the SAP BI performance workshop.

Some examples of a weak design are:

- Dimensions with too many characteristics that are queried, say, more than 16.
- Unbalanced hierarchies.
- Multi-provider over a large number of cubes, where many queries are accessed for many cubes. See SAP note 629541 for details.

2. Utilize SAP options to improve performance.

Before tuning the DB2 database, first apply and check the following SAP BI query tuning methods:

- Define aggregates as needed and use overnight or *spare* system capability to fill them.
- Use the OLAP cache.
- Use the pre-calculation of Web templates.
- Evaluate the usage of BIA.

3. Set DB2 ZPARMS as per SAP note 390016.

The ZPARM settings provided in SAP note 390016 have been recommended by IBM as optimal for SAP BI. Customers should check this SAP note for updates as part of their regular DB2 and SAP maintenance schedules. After migration to DB2 9, adapt the ZPARMS as described in SAP note 1032273. The most important change for BI is the new parameter MXDTCACH, which replaces the former ZPARM SJMXPOOL.

4. Ensure that the user selects the option to run RUNSTATS after data loads.

A cost-based optimizer relies on current and correct statistics to ensure optimal SQL access path selection. In an SAP BI environment, statistics do not require re-collection unless the object's data content has altered based upon a load, SAP compression, rollup, deletion, attribute change run, or data archive. Therefore it is only these activities that require the updating of DB2 statistics, and mostly the RUNSTATS are triggered automatically by the SAP BI functions itself.

With a load, new packets are loaded into the F- fact table. Without knowledge of the newly loaded data, the optimizer may not choose an optimal access path for subsequent reporting queries, aggregate rollups, or SAP compressions that specify a search criteria for the individual packet number. Therefore it is important to ensure the RUNSTATS is triggered as part of the load process by selecting **RSA1 → Manage InfoCube → Performance → Create Statistics (Btch)**.

For process chains see SAP Note 778437.

5. Keep current with SAP support packs and DB2 maintenance level.

Each new SAP support pack delivers fixes and also performance enhancements based upon prior customer requirements or identified areas for improvement by joint DB2 and SAP development.

As with the SAP software, each maintenance release of DB2 provides additional fixes and enhancements based upon prior customer problems and requirements. To minimize exposure to known problems, we recommend that you stay current with the SAP certified maintenance level and apply additional PTFs identified in SAP note 81737.

6. Utilize SAP compression (E- fact table).

When F- fact table packets are consolidated into the E- fact table via a process known as InfoCube compression, queries on the E- fact table filter by meaningful dimension/master data characteristics rather than the arbitrary packet number. This can result in better exploitation of available indexes on the E- fact table since the F- fact table must carry an index led by packet number to improve compression and rollup performance, but not query performance.

Given the volatility of the F- fact table, RUNSTATS is required more frequently than on the E- fact table. Therefore keeping the F- fact table a moderate size by compressing packets to E can improve overall RUNSTATS performance and cost for the entire InfoCube.

And finally the most important reason for SAP compression is for non-cumulative key figures whereby an additional row is inserted or updated in the E- fact table holding for each combination of dimension IDs the most current key figure values. Rather than requiring to sum all delta values from

the F- fact table, the E- fact table reference point can be accessed directly for current data, thus improving query performance.

There is one exception when SAP compression can be omitted. This is the case when you are using the SAP BIA and when the compression ratio is less than 1.5.

7. Use fast deletion of table partitions.

Apply SAP note 894045 to activate the fast deletion of table partitions for partitioned PSA and F- fact tables.

8. Partition E-Fact and F- fact tables.

The E- fact table (result of SAP compression) can be partitioned via SAP cube maintenance, which provides greater opportunity for improved query performance and elapsed time, and also potential reduction in operational costs.

Partitioning the E- fact table increases the opportunity for DB2 parallelism, which can significantly reduce the query elapsed time. Similarly, queries with time-based filtering will be able to eliminate unnecessary partitions from being accessed regardless of the filtering dimensions or indexes chosen for fact table access. These two benefits also result in greater access path stability for queries against the InfoCube.

The F- fact table will be automatically partitioned by request by setting a RSADMIN parameter. In addition to improving query performance, this significantly speeds up the deletion of a request and the RUNSTATS. Details are in SAP note 860830.

An additional benefit of partitioning is the ability for partition level operations such as COPY utility, REORG, or RUNSTATS to be invoked by the DBA. With time-based partitions, historical partitions do not require further REORG/RUNSTATS because their data will not change over time. An increase in E- fact table size should not result in a significant increase in operational maintenance costs.

For these reasons we recommend partitioning the E- fact tables and using InfoCube compression.

9. Use DB2 HW compression.

DB2 hardware compression is one of the strengths that differentiates DB2 on System z from other platforms. Compression rates of 70% and greater are possible for large SAP BI objects, such as fact tables, data store objects, and large aggregates. The four main benefits of compression include:

- Reduced disk space
- Improved elapsed time for sequential processing (for example, queries, copy utility)

- Improved bufferpool hit ratio (more rows per page)
- Reduced log data volume where there is insert-intensive processing

With the compression dictionary moving above the 2 GB bar in V8, which may result in virtual storage relief for DBM1, there is more motivation for customers to exploit System z strong compression capabilities.

10. Use the SAP transaction/programs to remove unused data.

Master data tables can be preloaded with SAP default data and loaded with data by a user that never gets utilized. This unused data can disrupt the cost-based optimizer's estimates of how many rows will be returned, or can result in this data being unnecessarily retrieved for every query involving that table.

We recommend purging unwanted data using the following methods:

- Trancode RSRV for dimension data.
- The program to delete master data is RSDMD_DEL_BACKGROUND or functional module RSDMD_DEL_MASTER_DATA.
- The program RSCDS_DEL_OLD_REQUESTS removes obsolete rows from F- fact tables (SAP note 609164).

Archived

Benefits of DB2 9 for SAP BI

This chapter describes and discusses the benefits and new features of DB2 9 for z/OS that are most relevant to SAP BI. Here we discuss the new dynamic index ANDing, the new histogram statistics, the parallelism enhancements, and some further BI relevant DB2 9 features. We do not discuss XML, backup and recovery enhancements, OSC, LOB improvements, Online REORG enhancements, Universal tablespaces, Clone table support, MERGE statement, or Index Compression. All of these are very important DB2 9 for z/OS enhancements and new features and have no extra SAP BI specific usage. They are a benefit to every SAP system, whether it is NetWeaver or SAP BI.

4.1 Dynamic index ANDing for star schema queries

The new dynamic index ANDing is a new enhanced star join access method introduced in DB2 9 for z/OS as a main requirement from SAP BI. The goal of the new access method is to improve and stabilize data warehouse query performance and to lead to more predictable query performance.

This is done by introducing self-automating and self-defending access paths and uses parallelism. This means that the access path used at run time is dynamic and can be changed within the query processing by the DB2 engine. One further goal of dynamic index ANDing is to simplify the index design on fact tables. The main idea behind dynamic index ANDing access is to apply the most filtering dimensions or snowflake branches before the E- fact table and to do this in an efficient way.

Dynamic index ANDing has the following characteristics:

- ▶ Better exploitation of SAP BI single column fact table indexes
 - No need for additional multi-column indexes for leveraging star join access
- ▶ Consistent parallelism
 - Independent filtering dimension access in parallel
 - fact table access (and post fact table) in parallel
- ▶ Adaptive query execution based upon runtime filtering
 - Less filtering dimensions can be discarded for pre- fact table access.
 - Self-automating and self-defending access path.
 - Runtime fallback to workfile for RID processing. This avoids RID pool failures.
- ▶ RID pool overflow to workfile
- ▶ Less dependent on perfect statistics
 - Although optimizer costing is still performed
 - Better tolerance of less than perfect access path choice
- ▶ Provides more predictable query performance

Figure 4-1 shows an SAP BI snowflake with the E- fact table F in the middle and the corresponding dimension branches consisting of dimension tables (Dn), SID-tables (Sn), and X-tables (Xn). The arrows visualize the filtering for any of the four dimension branches.

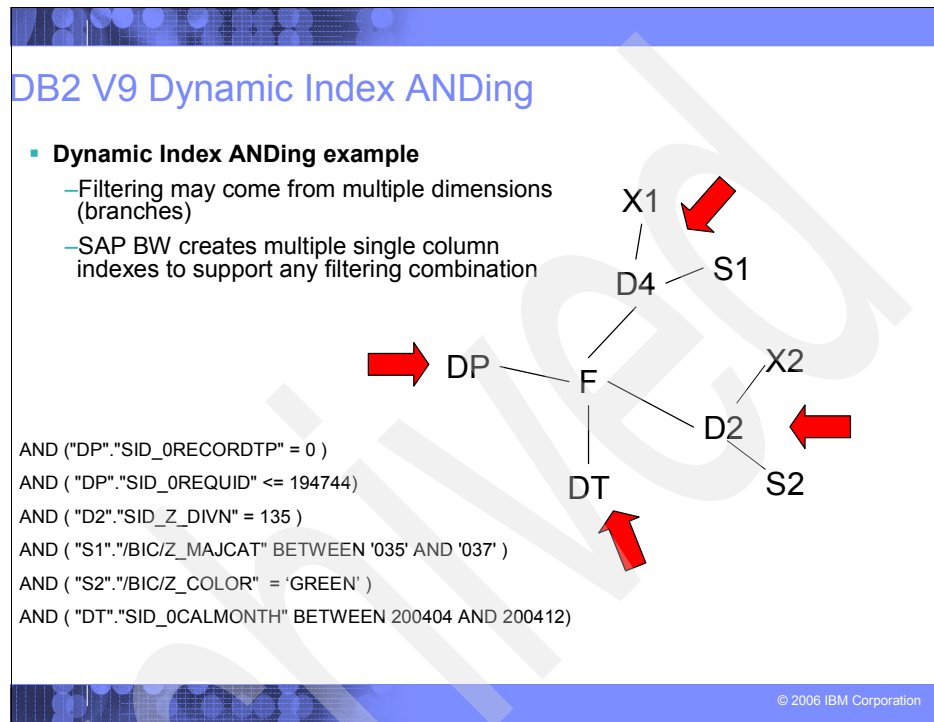


Figure 4-1 Filtering on dimension branches of F- fact table F

Figure 4-2 and Figure 4-3 on page 39 show the principle of processing dynamic index ANDing for pre- fact table access. Unlike traditional star join group, the pair-wise join represents each dimension table join with fact table separately. The result of each pair-wise join is a set of RIDs of F- fact table. The RIDs are ANDing together produce a final RID list for accessing the E- fact table pages.

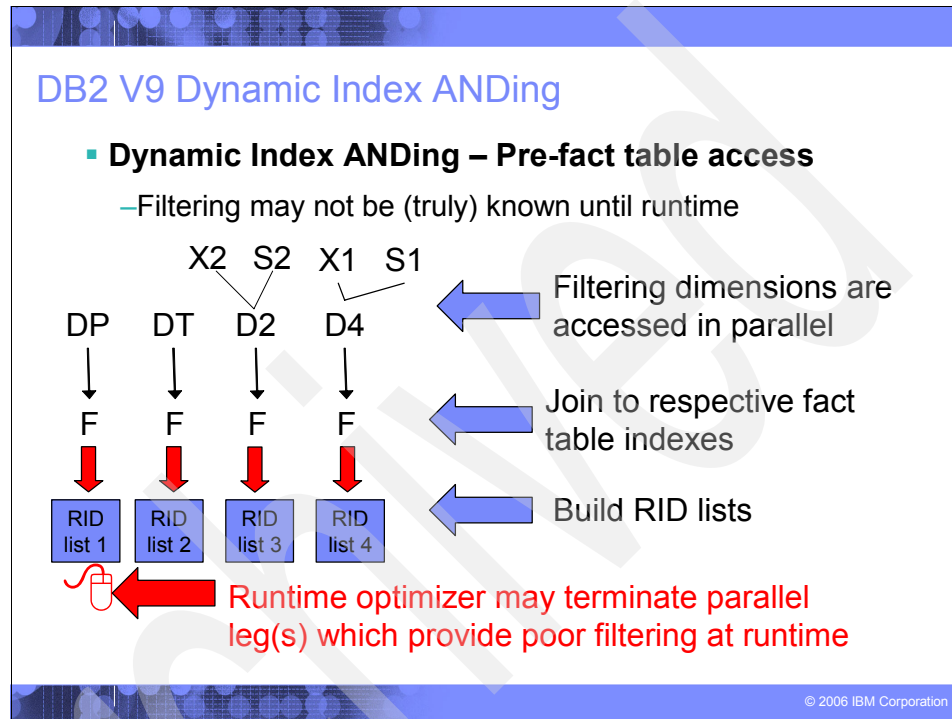
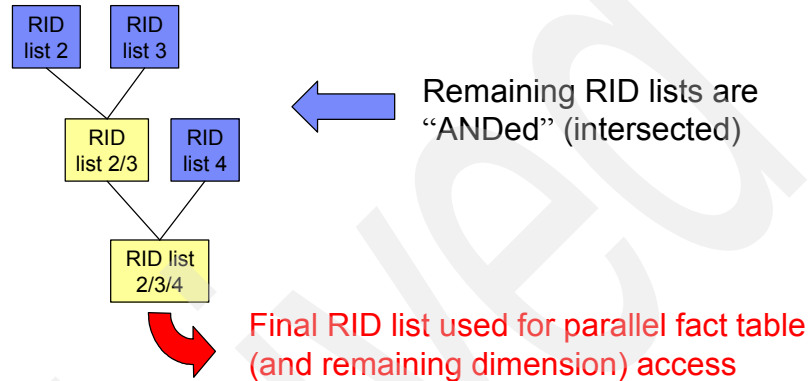


Figure 4-2 Pre- fact table access

DB2 V9 Dynamic Index ANDing

▪ Dynamic Index ANDing – Fact table access

–Filtering may not be (truly) known until runtime



© 2006 IBM Corporation

Figure 4-3 Building the final RID list before E- fact table access

The final fact table RIDs are used to retrieve data from fact table and join back to dimension table as necessary for obtaining data from dimension tables.

To summarize, the pair-wise join has the following characteristics:

1. Joins each dimension table with a fact table through an index independently
2. Performs RID Sort and RID Merge (ANDing) of the RID lists obtained from step 1 to form the final fact table RID list
3. Fetches data from the fact table using the final RID list
4. Joins back the dimension table only if it has select out column from dimension tables

4.2 Database statistics for BI tables

Processing RUNSTATS for BI tables and histogram statistics is a newly introduced feature that can help you manage performance of your environment.

4.2.1 Processing RUNSTATS for BI tables

Up from BW 3.0 Support Package 30, 3.1C Support Package 24, 3.5 Support Package 16, and BI 7.0 Support Package 6, SAP has optimized the ABAP™ coding for processing RUNSTATS on BI tables. The first newly introduced main feature is RUNSTATS on partition level. Call RUNSTATS only if necessary. And up from BW 3.5 Support Package 20 and BI 7.0 Support Package 11, SAP supports the new DB2 9 feature, histogram statistics.

On SAP BI systems on DB2 for z/OS all database statistics for BI tables are refreshed by the function module RSDU_ANALYZE_TABLE_DB2. Figure 4-4 explains the most relevant parameter switches.

New parameters for function RSDU_ANALYZE_TABLE_DB2 (note 915398) allow easy call of RUNSTATS within SE37	
I_SIMULATE = X	Simulation mode, shows only the generated RUNSTATS command, without execution. Is empty, if the statistics are sufficient
I_FORCE_RUNSTATS = X	Ignore the current statistics and recalculate them. If I_SIMULATE=X, show the generated RUNSTATS command
I_PARTITION = 3	Work only on partition 3 of the given table
I_SHOW_RUNSTATS_PROTOCOL = X	Shows the RUNSTATS command and the output in a easy readable format

Figure 4-4 Parameters for RSDU_ANALYZE_TABLE_DB2

If you are interested in how SAP BI calculates DB2 statistics, which means which RUNSTATS commands are fired and which statistics are possibly faked by directly changing statistic data in DB2 system tables, execute function module RSDU_ANALYZE_TABLE_DB2 over SAP transaction SE37 for a given table and mark the parameters I_SHOW_RUNSTATS_PROTOCOL and I_FORCE_RUNSTATS, as shown in Figure 4-5.

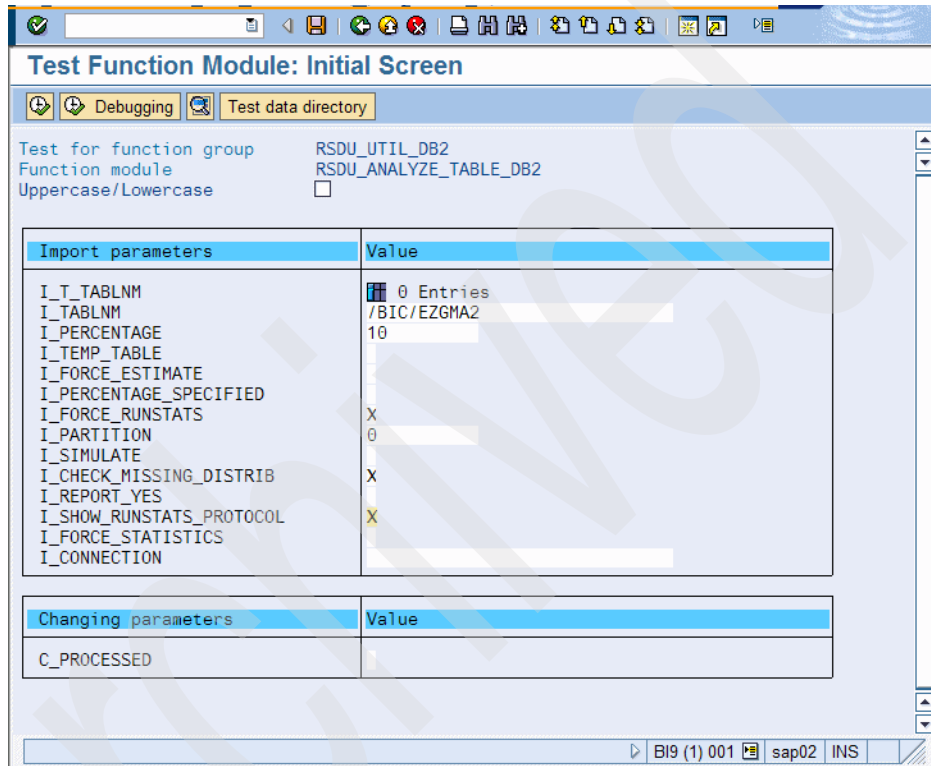


Figure 4-5 Executing RUNSTATS by function RSDU_ANALYZE_TABLE_DB2

RUNSTATS will then be called over the stored procedure DSNUTILS. So you have to wait a couple of minutes for large tables. When RUNSTATS is processed, the function comes back with the window shown in Figure 4-6.

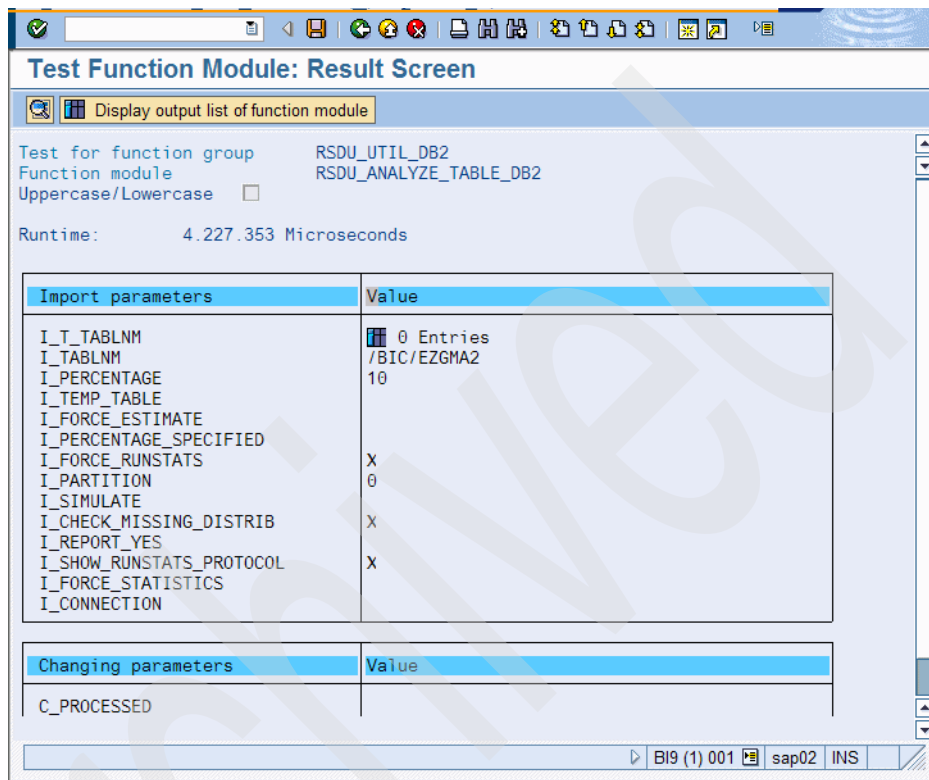


Figure 4-6 Testmode for function RSDU_ANALYZE_TABLE_DB2

Click **Display output list of function module** to see the output of the function (Example 4-1).

Example 4-1 Output of function module RSDU_ANALYZE_TABLE

```

22.03.2007                               SV Parameter                               1
-----
RUNSTATS TABLESPACE "FA40XK1C"."EZGMA2X"
TABLE(BI9DB."/BIC/EZGMA2") SAMPLE 010
COLUMN( KEY_ZGMA21,KEY_ZGMA22,KEY_ZGMA23,KEY_ZGMA24,KEY_ZGMA25,KEY_ZGMA2P,KEY_ZGMA2T,KEY_ZGMA2U )
COLGROUP ("KEY_ZGMA2P") FREQVAL COUNT 0 HISTOGRAM NUMQUANTILES 50
COLGROUP ("KEY_ZGMA2U") FREQVAL COUNT 0 HISTOGRAM NUMQUANTILES 50
INDEX(
BI9DB."/BIC/EZGMA2~0" KEYCARD FREQVAL NUMCOLS 1 COUNT 0 HISTOGRAM NUMCOLS 4 NUMQUANTILES 50
,BI9DB."/BIC/EZGMA2~040" KEYCARD FREQVAL NUMCOLS 1 COUNT 0 HISTOGRAM NUMCOLS 1 NUMQUANTILES 50
,BI9DB."/BIC/EZGMA2~050" KEYCARD FREQVAL NUMCOLS 1 COUNT 0 HISTOGRAM NUMCOLS 1 NUMQUANTILES 50
,BI9DB."/BIC/EZGMA2~060" KEYCARD FREQVAL NUMCOLS 1 COUNT 0 HISTOGRAM NUMCOLS 1 NUMQUANTILES 50
,BI9DB."/BIC/EZGMA2~070" KEYCARD FREQVAL NUMCOLS 1 COUNT 0 HISTOGRAM NUMCOLS 1 NUMQUANTILES 50
,BI9DB."/BIC/EZGMA2~080" KEYCARD FREQVAL NUMCOLS 1 COUNT 0 HISTOGRAM NUMCOLS 1 NUMQUANTILES 50

```

```

)
SHRLEVEL CHANGE REPORT NO
cause for calling RUNSTATS
parameter force_runstats set
old_card / rows_changed / percent_changed = 0 / 0 / 9.99900E+03
Runtime für RUNSTATS: 3,817003 secs
20.070.322.203.600,9062580 - 20.070.322.203.604,7232610
IDSNU0001 081 16:36:00.33 DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = BI924116
DSNU10441 081 16:36:00.39 DSNUGTIS - PROCESSING SYSIN AS EBCDIC
ODSNU0501 081 16:36:00.40 DSNUGUTC - RUNSTATS TABLESPACE "FA40XK1C"."EZGMA2X" TABLE(BI9DB./BIC/EZGMA2") SAMPLE
10 COLUMN(KEY_ZGMA21, KEY_ZGMA22, KEY_ZGMA23, KEY_ZGMA24, KEY_ZGMA25, KEY_ZGMA2P, KEY_ZGMA2T, KEY_ZGMA2U) COLGROUP(
"KEY_ZGMA2P") FREQVAL COUNT 0 HISTOGRAM NUMQUANTILES 50 COLGROUP("KEY_ZGMA2U") FREQVAL COUNT 0 HISTOGRAM
NUMQUANTILES 50 INDEX(BI9DB./BIC/EZGMA2~0" KEYCARD FREQVAL NUMCOLS 1 COUNT 0 HISTOGRAM NUMCOLS 4 NUMQUANTILES 50,
BI9DB./BIC/EZGMA2~040" KEYCARD FREQVAL NUMCOLS 1 COUNT 0 HISTOGRAM NUMCOLS 1 NUMQUANTILES 50,
BI9DB./BIC/EZGMA2~050" KEYCARD FREQVAL NUMCOLS 1 COUNT 0 HISTOGRAM NUMCOLS 1 NUMQUANTILES 50,
BI9DB./BIC/EZGMA2~060" KEYCARD FREQVAL NUMCOLS 1 COUNT 0 HISTOGRAM NUMCOLS 1 NUMQUANTILES 50,
BI9DB./BIC/EZGMA2~070" KEYCARD FREQVAL NUMCOLS 1 COUNT 0 HISTOGRAM NUMCOLS 1 NUMQUANTILES 50,
BI9DB./BIC/EZGMA2~080" KEYCARD FREQVAL NUMCOLS 1 COUNT 0 HISTOGRAM NUMCOLS 1 NUMQUANTILES 50) SHRLEVEL CHANGE
REPORT NO
DSNU0421 081 16:36:01.01 DSNUGLSR - SORT PHASE STATISTICS -
NUMBER OF RECORDS=0
ELAPSED TIME=00:00:00
DSNU6101 -DB2T 081 16:36:01.67 DSNUSUTP - SYSTABLEPART CATALOG UPDATE FOR FA40XK1C.EZGMA2X SUCCESSFUL
DSNU6101 -DB2T 081 16:36:01.67 DSNUSUPT - SYSTABSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:01.68 DSNUSUPC - SYSCOLSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:01.68 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:01.68 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:01.68 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:01.68 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6041 -DB2T 081 16:36:01.69 DSNUSEF2 - TABLESPACE IS EMPTY
DSNU6101 -DB2T 081 16:36:01.69 DSNUSUTB - SYSTABLES CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:01.70 DSNUSUCO - SYSCOLUMNS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:01.70 DSNUSUTS - SYSTABLESPACE CATALOG UPDATE FOR FA40XK1C.EZGMA2X SUCCESSFUL
DSNU6231 -DB2T 081 16:36:01.70 DSNUSEF2 - SYSIBM.SYSCOLDIST CATALOG NOT UPDATED WITH AGGREGATE STATISTICS FOR
BI9DB./BIC/EZGMA2.KEY_ZGMA2P BECAUSE SOME PARTITIONS HAVE NO VALID STATISTICS
DSNU6231 -DB2T 081 16:36:01.70 DSNUSEF2 - SYSIBM.SYSCOLDIST CATALOG NOT UPDATED WITH AGGREGATE STATISTICS FOR
BI9DB./BIC/EZGMA2.KEY_ZGMA2U BECAUSE SOME PARTITIONS HAVE NO VALID STATISTICS
DSNU6101 -DB2T 081 16:36:02.55 DSNUSUIP - SYSINDEXPART CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~0 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:02.55 DSNUSUPI - SYSINDEXSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~0 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:02.55 DSNUSUPC - SYSCOLSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:02.56 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:02.56 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:02.81 DSNUSUIP - SYSINDEXPART CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~040 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:02.81 DSNUSUPI - SYSINDEXSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~040 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:02.81 DSNUSUPC - SYSCOLSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:02.81 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:02.82 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:03.10 DSNUSUIP - SYSINDEXPART CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~050 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:03.10 DSNUSUPI - SYSINDEXSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~050 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:03.10 DSNUSUPC - SYSCOLSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:03.10 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:03.10 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:03.10 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:03.34 DSNUSUIP - SYSINDEXPART CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~060 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:03.35 DSNUSUPI - SYSINDEXSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~060 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:03.35 DSNUSUPC - SYSCOLSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:03.35 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:03.35 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:03.35 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:03.60 DSNUSUIP - SYSINDEXPART CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~070 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:03.60 DSNUSUPI - SYSINDEXSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~070 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:03.60 DSNUSUPC - SYSCOLSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:03.60 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:03.60 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:03.60 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:03.85 DSNUSUIP - SYSINDEXPART CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~080 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:03.85 DSNUSUPI - SYSINDEXSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~080 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:03.85 DSNUSUPC - SYSCOLSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU6101 -DB2T 081 16:36:03.85 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL

```

```

DSNU610I -DB2T 081 16:36:03.85 DSNUSUPD - SYSCOLDISTSTATS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2 SUCCESSFUL
DSNU604I -DB2T 081 16:36:03.85 DSNUSEOF - INDEXSPACE IS EMPTY
DSNU610I -DB2T 081 16:36:03.85 DSNUSUCO - SYSCOLUMNS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~0 SUCCESSFUL
DSNU610I -DB2T 081 16:36:03.85 DSNUSUIX - SYSINDEXES CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~0 SUCCESSFUL
DSNU604I -DB2T 081 16:36:03.87 DSNUSEOF - INDEXSPACE IS EMPTY
DSNU610I -DB2T 081 16:36:03.87 DSNUSUCO - SYSCOLUMNS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~040 SUCCESSFUL
DSNU610I -DB2T 081 16:36:03.87 DSNUSUIX - SYSINDEXES CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~040 SUCCESSFUL
DSNU604I -DB2T 081 16:36:03.87 DSNUSEOF - INDEXSPACE IS EMPTY
DSNU610I -DB2T 081 16:36:03.87 DSNUSUCO - SYSCOLUMNS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~050 SUCCESSFUL
DSNU610I -DB2T 081 16:36:03.87 DSNUSUIX - SYSINDEXES CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~050 SUCCESSFUL
DSNU604I -DB2T 081 16:36:03.87 DSNUSEOF - INDEXSPACE IS EMPTY
DSNU610I -DB2T 081 16:36:03.87 DSNUSUCO - SYSCOLUMNS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~060 SUCCESSFUL
DSNU610I -DB2T 081 16:36:03.88 DSNUSUIX - SYSINDEXES CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~060 SUCCESSFUL
DSNU604I -DB2T 081 16:36:03.88 DSNUSEOF - INDEXSPACE IS EMPTY
DSNU610I -DB2T 081 16:36:03.88 DSNUSUCO - SYSCOLUMNS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~070 SUCCESSFUL
DSNU610I -DB2T 081 16:36:03.88 DSNUSUIX - SYSINDEXES CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~070 SUCCESSFUL
DSNU604I -DB2T 081 16:36:03.88 DSNUSEOF - INDEXSPACE IS EMPTY
DSNU610I -DB2T 081 16:36:03.88 DSNUSUCO - SYSCOLUMNS CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~080 SUCCESSFUL
DSNU610I -DB2T 081 16:36:03.88 DSNUSUIX - SYSINDEXES CATALOG UPDATE FOR BI9DB./BIC/EZGMA2~080 SUCCESSFUL
DSNU620I -DB2T 081 16:36:03.89 DSNUSEOF - RUNSTATS CATALOG TIMESTAMP = 2007-03-22-16.36.00.422387
DSNU010I 081 16:36:04.01 DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
> connection: DEFAULT DB2 User: SAPR3

```

Since some specific RUNSTATS options are used for BI tables, statistics should not be created for SAP BI systems using external tools. Use the “RUNSTATS on objects needing new statistics” option in the planning calendar (DB13) to update the DB2 statistics. We recommend that you schedule this to run daily or at least once a week, preferably at night. Since the DB2 statistics for BI programs are already updated automatically for most BI tables, the DB13 statistics update should be very quick.

Some additional comments:

- ▶ RUNSTATS on the partition level. For performance reasons, SAP BI uses only the PART option of RUNSTATS if the partitioned table has more than one million rows.
- ▶ *Call RUNSTATS only if necessary* means that RUNSTATS is only called if the old database statistics need to be refreshed. For this, the real time statistics (RTS) data are compared against the old statistics. If more than 10% of the data has been changed or if the table is small (fewer than 1000 rows), RUNSTATS is called again.

More details are described in SAP note 915398. Additionally, this note describes the input parameter of the above-used function module RSDU_ANALYZE_TABLE_DB2.

Figure 4-7 explains how to influence the DB2 optimizer to use the appropriate index by explicitly faking the DB2 statistics. You will find the details in SAP note 688133. Make sure that you test your RSADMIN entries by running function module RSDU_ANALYZE_TABLE_DB2 for the given table and check the output list where the generated and executed INSERT or UPDATE statements are shown. Figure 4-7 further highlights SAP note 688133.

How to influence DB2 statistics (note 688133)

- **specific RSADMIN parameters for influencing the DB2 statistics**
- **Use this possibility only if no other solution helps**
- **Collect additional frequencies for an given index**
 - DB2_ "<Indexname>" = FREQVAL NUMCOLS n COUNT c
 - Example: **DB2_"/BIC/DMYDIMIT~0"** = **FREQVAL NUMCOLS 5 COUNT 30**
- **Change the clusterratio of an index**
-> to force the optimizer to use a known index set the clusterratio to 100
 - DB2_ "<Indexname>" = CLUSTERRATIO x
 - Example: **DB2_"/BIC/FMYCUBE~030"** = **CLUSTERRATIO 100**
 - DB2 catalog is updated accordingly after the RUNSTATS run

Figure 4-7 Faking the DB2 statistics with RSADMIN parameters

4.2.2 New to DB2 9: histogram statistics

In this section we discuss histogram statistics.

Overview and advantages

Data distribution statistics are very important for query optimization. DB2 chooses the best access path based on costing, and the very basic foundation of costing is predicate selectivity estimation, which heavily relies on data distribution statistics.

Up to DB2 v8, the only distribution statistics supported in DB2 for z/OS were frequency statistics collected on single values, which could be for either single-column or multi-column. For multi-column, it is a concatenated multi-column value. Such frequency statistics are most commonly collected on the most biased values (that is, the most frequent or the least frequent).

Often the single-value based frequency statistics can hardly help DB2 with the predicate selectivity other than uniform interpolation on the rest of the (uncollected) value range. This would be a wild guess and may lead to an undesirable access path.

Histogram describes the data distribution over the entire value range. The predicate selectivity gets a more accurate calculation if the searching range matches the boundary of any one quantile or any group of consecutive quantiles. Even if there is no perfect match, the predicate selectivity interpolation now is done within one or two particular quantiles. With the interpolation done in a much smaller granularity, the predicate selectivity is expected to be evaluated with more accuracy.

Histograms in detail

Histogram statistics are introduced to DB2 9 for enhancing predicate selectivity estimation and to enhance the DB2 access path selection in general.

Histogram statistics is a way of summarizing data distribution on an interval scale (either discrete or continuous). It divides up the range of possible values in a dataset into quantiles, for which a set of statistics parameters is collected.

There are several types of histogram statistics being researched. DB2 for z/OS uses only Equal-depth histogram statistics.

Histogram statistics enable DB2 to improve access path selection by estimating predicate selectivity from value-distribution statistics that are collected over the entire range of values in a data set.

DB2 chooses the best access path for a query based on predicate selectivity estimation, which in turn relies heavily on data distribution statistics. Histogram statistics summarize data distribution on an interval scale by dividing the entire range of possible values within a data set into a number of intervals.

DB2 creates equal-depth histogram statistics, meaning that it divides the whole range of values into intervals that each contain about the same percentage of the total number rows. The following columns in a histogram statistics table define an interval:

QUANTILENO	An ordinary sequence number that identifies the interval
LOWVALUE	A value that serves as the lower bound for the interval
HIGHVALUE	The value that serves as the upper bound for the interval

Note the following characteristics of histogram statistics intervals:

- ▶ Each interval includes approximately the same number, or percentage, of the rows. A highly frequent single value might occupy an interval by itself.

- ▶ A single value is never broken into more than one interval, meaning that the maximum number of intervals is equal to the number of distinct values on the column. The maximum number of intervals cannot exceed 100, which is the maximum number that DB2 supports.
- ▶ Adjacent intervals sometimes skip values that do not appear in the table, especially when doing so avoids a large range of skipped values within an interval. For example, if the value 30 above has 1% frequency, placing it in the seventh interval would balance the percentage of rows in the sixth and seventh intervals. However, doing so would introduce a large skipped range to the seventh interval.
- ▶ HIGHVALUE and LOWVALUE can be inclusive or exclusive, but an interval generally represents a non-overlapped value range.
- ▶ NULL values, if any exist, occupy a single interval.
- ▶ Because DB2 cannot break any single value into two different intervals, the maximum number of intervals is limited to the number of distinct values in the column, and cannot exceed the DB2 maximum of 100 intervals.

Example of equal-depth histogram statistics

Equal-depth Histogram cuts the whole value range so that each quantile has about the same number of rows. The important parameters to describe equal-depth histogram statistics includes the total number of quantiles (N) and for each quantile, the pair of LOWVALUE/HIGHVALUE, the number of distinctive values (CARD) and the frequency (or the number of rows). To give an example, on a table IBM_EMPLOYEE column YRS_OF_EXPERIENCE, it may have (N=7) quantiles, as shown in Table 4-1.

Table 4-1 Histogram statistics example

SEQ.NR	LOWVALUE	HIGHVALUE	CARD	FREQUENCY
1	0	3	4	14%
2	4	15	8	14%
3	18	24	7	12%
4	25	25	7	12%
5	26	26	1	15%
6	27	30	4	16%
7	35	40	6	14%

Several things worth noting are:

- ▶ There may be skipped values between adjacent quantiles. If there are, then it means that those skipped values are of zero appearance in the table.
- ▶ Each quantile has approximately the same number (or percentage) of rows. The highly frequent single value may alone by itself occupy a quantile. The single-value will not be broken into more than one quantiles. Therefore, the number of quantiles maximum is equivalent to the number of distinct values in that column. However, DB2 has a maximum limit of 100 quantiles for storage purposes.
- ▶ One particular value may be included into a quantile just because it is *closer* to that quantile. The purpose is to avoid the skipped range as much as possible. For example, value 30 from above has frequency=1%, including it in the 6th quantile would make the fifth and sixth quantile with frequency=15% even. But doing that would introduce a large skipped range [28, 34] to the sixth quantile and we try to avoid that.
- ▶ A special case of (3) is that HIGHVALUE or LOWVALUE, when it is too far away from the rest of the value range, the single value of HIGHVALUE or LOWVALUE will occupy a quantile by itself.
- ▶ The NULL value will occupy a quantile by itself.
- ▶ Due to aggregation, quantiles may not always have closed range, that is, the high value or low value can be inclusive or exclusive. But overall, the quantiles represent the non-overlapped value range.

You will find more details about histogram statistics in the description of the corresponding system catalog tables (SYSCOLDIST, SYSCOLDISTSTATS, SYSKEYTGTDIST, and so on) and in the documentation for RUNSTATS.

4.3 Parallelism enhancements

This section discusses parallelism enhancements such as global query optimization, which enables DB2 to optimize a query as a whole rather than as an independent part and cross query block optimization, which is optimization across, rather than within, query blocks.

4.3.1 Global query optimization

With DB2 V8 the DB2 optimizer chooses the lowest cost *sequential* plan and then determines how to *parallelize* the access path. With Version 9 the way of using parallelism has been extended. The V9 optimizer considers multiple sequential plans for parallelism and then executes the one with the lowest costs.

Global query optimization addresses query performance problems caused when DB2 V8 breaks a query into multiple parts and optimizes each of those parts independently. While each of the individual parts may be optimized to run efficiently, when these parts are combined the overall result may be inefficient. For example, consider the following query:

```
SELECT * FROM T1
WHERE EXISTS (SELECT 1 FROM T2, T3 WHERE T2.C2 = T3.C2
AND T2.C1 = T1.C1);
```

DB2 V8 will break this query into two parts (the correlated subquery and the outer query), and each of these parts will be optimized independently. The access path for the subquery does not take into account the different ways in which the table in the outer query may be accessed and vice versa.

DB2 V8 may choose to do a table scan of T1, resulting in much random I/O when accessing T2, while a nonmatching index scan of T1 would avoid the random I/O on T2. In addition, DB2 will not consider reordering these two parts. The correlated subquery will always be performed after accessing T1 to get the correlation value. If T1 is a large table, and T2 is a small table, it may be much more efficient to access T2 first and then T1 (especially if there is no index on T2.C1 but here is an index on T1.C1).

In summary, Global Query Optimization will allow DB2 to optimize a query as a whole rather than as independent parts. This is accomplished by allowing DB2 to:

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

4.3.2 Cross query block optimization

This new feature addresses query performance problems caused when DB2 breaks a query into multiple parts and optimizes each of those parts independently. The V9 optimizer treats subqueries as virtual tables, which allows it to do a cross query block optimization.

4.4 Extended sparse index and in-memory workfile

DB2 9 introduces the new installation parameter MXDTCACH to specify the maximum memory allocated for data caching. The recommended minimal setting of MXDTCACH for SAP BI systems is 128 MB. The recommended settings of new V9 parameters after migration from a V8 SAP system are documented in SAP note 1032273.

The unit of a value specified to MXDTCACH is in MB (1 MB is 1048576 bytes). For example, if a value 300 is specified, the total amount of up to 300 MB memory can be allocated for data caching per thread.

- ▶ The DB2 default value of 20 MB is far too low.
- ▶ When the value of 0 (zero) is specified, the data caching method will not be used during query execution. Only sparse index (key+rid) can be applied.
- ▶ The possible range of the values is between 0 and 512.
- ▶ When data records are cached, the required memory amount is:
(number of rows)*((maximum length of the keys) + (total maximum length of all the relevant columns))

If the required memory exceeds the limit that MXDTCACH specifies or the memory allocation fails, the query will continue without using data caching. Sparse index will be used instead.

- ▶ When the thread terminates, the allocated memory for data caching will be freed from the local pool above the 2 G bar.

The new installation parameter MXDTCACH can be set on the DB2 installation panel.

Note: The existing installation parameter SJMXPOOL will be replaced by MXDTCACH. Accordingly, the data caching space for star join will be moved to the local pool above the 2 G bar.

4.5 TRUNCATE TABLE statement

The TRUNCATE TABLE statement deletes all data rows for either base tables or declared global temporary tables without activating delete triggers defined on the table (they are ignored). The base table can belong to a simple tablespace, a segmented tablespace, a partitioned tablespace, or a universal tablespace. If the table contains LOB or XML columns, its corresponding tablespaces and indexes are also emptied.

DB2 9 transforms the new TRUNCATE TABLE operation into a mass delete operation to take advantage of the current, optimized design and to provide greater flexibilities for users to deactivate existing delete triggers and harden the results of truncate operation without issuing a normal commit.

Under specific table attributes and statement options, the TRUNCATE TABLE statement may provide an optimal performance path to empty the table while in comparison with the current mass delete operation (that is, DELETE without

WHERE clause). The performance improvement can be gained only on a table with triggers defined. Otherwise, it will perform as today. Moreover, it provides a statement commit option (IMMEDIATE option) for the statement that allows the truncate operation to become permanent (that is, cannot undo) and immediately makes deallocated space available for new, subsequent inserts in the same unit of work.

SAP BI will use the TRUNCATE TABLE statement of DB2 9, but because BI tables usually have no triggers, the performance is the same as the mass delete operation.

4.6 FETCH FIRST n ROWS ONLY in subselects

DB2 8 for z/OS disallows the FETCH FIRST n ROWS ONLY clause in a subselect. That is, one could write:

```
SELECT * FROM T ORDER BY c1 FETCH FIRST n ROWS ONLY
```

(specifying the clauses as part of select-statement), but one could not write:

```
INSERT INTO any_table  
  (SELECT * FROM T ORDER BY c1 FETCH FIRST n ROWS ONLY)
```

(specifying the clauses within the subselect).

SAP BI needs this feature to implement a more sophisticated algorithm of the so-called BI *Selective Deletion* for InfoCube and DSO data. Here *selective* means that the user can choose an arbitrary selection criteria for the data to be deleted. In this general case the data to be deleted can be distributed over all table partitions of a partitioned table, so that the deletion of whole partitions cannot be used.

When deleting a huge amount of data rows from a fact or DSO table, it is better to separate this deletion in many small portions of delete transactions. Otherwise, you have one probably very long-running delete transaction, with the danger of a day-long or even longer rollback in the case the delete transaction has to be canceled for any purpose. This is a nightmare for every database administrator.

With the DB2 9 feature of using FETCH FIRST n ROWS ONLY within a subselect, the portionwise deletion is implemented in the following way.

The large delete operation is split up into multiple smaller units of work. In Example 4-2, assume that the primary key of table *<deltable>* consists of the columns kf1, kf2, ..., kfn. *<delete_condition>* is a placeholder for the WHERE

condition, generated by SAP BI upon the deletion criteria specified by the BI user.

Example 4-2 New implementation of very large data deletions

```
repeat
  DELETE FROM <deltable>
    WHERE (kf1, kf2, ..., kfn) IN
      ( SELECT kf1, kf2, ..., kfn FROM <deltable>
        WHERE <delete_condition>
          FETCH FIRST <paketsize> ROWS ONLY );

  COMMIT WORK;
until nrows_deleted < paketsize.
```

To split up this delete into a loop of delete operations with each one deleting *<packetsize>* rows only, the statement **FETCH FIRST n ROWS ONLY** could be used. After each execution of the DELETE statement, a commit is performed. The loop continues until all rows specified are deleted.

So the deletion of the data is performed in a loop with a COMMIT after every deletion of one small packet of rows (packetsize can be configured). This new SAP BI implementation for data deletion will be available for the next SAP BI release coming after BI 7.0. SAP plans to make this new implementation for selective deletion also available for the BI 7.00 release, running with DB2 9 by setting a special RSADMIN parameter. This will be documented in a future SAP note.

Example 4-3 shows the delete statement that is generated to delete a maximum of 100,000 rows from the E- fact table of cube IUSALES.

Example 4-3 Using FETCH FIRST n ROWS ONLY for E- fact table

```
DELETE FROM "/BIC/EIUSALES"
WHERE ("KEY_IUSALES", "KEY_IUSALES", "KEY_IUSALES", "KEY_IUSALES1",
"KEY_IUSALES2", "KEY_IUSALES3")
IN ( SELECT "KEY_IUSALES",
"KEY_IUSALES", "KEY_IUSALES", "KEY_IUSALES1", "KEY_IUSALES2", "KEY_IUSALES3"
FROM "/BIC/EIUSALES"
WHERE <delete_condition>
FETCH FIRST 100000 ROWS ONLY )
```

4.7 RENAME INDEX

DB2 8 for z/OS only supports RENAME TABLE. RENAME INDEX was a feature request from SAP BI, because the renaming of tables and their corresponding indexes is used within SAP BI by functions like ACR (aggregate change run). DB2 9 now supports RENAME INDEX and therefore speeds up the ACR.

4.8 RANK expression

Online analytical processing (OLAP) specifications provide the ability to return ranking and row numbering information as a scalar value in the result of a query.

DB2 9 supports the new expressions ROW CHANGE, RANK, DENSE_RANK, and ROW_NUMBER, which are sometimes referred to as OLAP-specification or window functions.

RANK() returns the rank of each row within the partition of a result set. The rank of a row is one plus the number of ranks that come before the row in question. SAP BI uses the RANK expression for TOP-N queries.

Example 4-4 displays the ranking of employees that have a total salary of more than \$50,000, in order by name.

Example 4-4 Example for RANK

```
SELECT EMPNO, NAME, SALARY+BONUS AS TOTAL_SALARY,  
       RANK() OVER(ORDER BY SALARY+BONUS DESC) AS RANK_SALARY  
FROM EMPLOYEES  
WHERE SALARY+BONUS > 50000  
ORDER BY NAME;
```

SAP BI uses the RANK expression for TOP-N queries.

4.9 Further DB2 9 enhancements and new features

There are additional important DB2 9 enhancements and new features having no extra SAP BI specific usage but that are a general benefit for every SAP system.

Some of these additional enhancements include backup and recovery, Online REORG, Universal tablespaces, Clone table support, MERGE statement, Index Compression, OSC, LOB improvements, and XML.

There is a separate IBM Redbooks publication that discusses these new features and enhancements, entitled *Enhancing SAP by using DB2 v9.1 for z/OS*. You will find the reference for this in “Related publications” on page 347.

Best practices for PSA

One of the main areas of responsibility for the SAP BI system administrator is the loading of new data into the BI system. This is normally a scheduled overnight process that is managed by one or more process chains. The data usually must be available for reporting in the BI system by a certain time each morning. To ensure consistent service levels, the data load process must run efficiently.

In this chapter we discuss the best practices for optimizing the BI data load performance for PSA on DB2 for z/OS.

There are no changes in respect of DB2 9. However, with BI 7.0 (formerly called SAP Netweaver 2004s BI) SAP introduced a new concept for loading data into InfoProviders. BI 7.0 already creates a PSA table when a data source is created. The InfoPackage is then only used to load the data from the source system into PSA. The load of the data from PSA into InfoProviders is done in an additional step, the so-called data transfer process (DTP). So the DTP can be seen like an additional kind of InfoPackage. With this newly introduced DTP the user has more flexibility and transparency of the data staging process. A further goal was to obtain better performance when loading data in parallel. The new DTP uses something called *transformations* to change the data as needed.

The loading into PSA has not changed with BI 7.0 and there is no relevant change in respect to the load performance against the former BI release.

The most relevant change is that you always get a PSA with the new DTP.

5.1 Load PSA

In this section we discuss the best practices for SAP BI on DB2 9 for z/OS for loading data into the PSA as part of the regular data loads into the SAP BI system. We discuss techniques that can be employed to ensure that the PSA is managed efficiently. We also discuss techniques that will optimize data loading into the PSA as part of the data load into the SAP BI data targets.

The source systems store logically dependent data in the form of *DataSources*. Using the DataSources, the data is extracted and transferred to the SAP BI system. In an SAP BI system, the PSA is used as the input storage, which stores the data in unchanged form. This means that if the data in the source system is not consistent or contains errors, the data in the PSA is not error free.

The basic role of the PSA is to provide a quality check of the input data coming from the source system. Another role of the PSA is to separate the loading process into SAP BI from the follow-on processes like data analysis and queries. Based on this separation, the loading performance can be tuned independently from other SAP BI activities.

The usage of the PSA for loading into the SAP BI system is optional, but SAP strongly recommends it from the SAP BI point of view.

For the SAP BI 7.0 Data Update, SAP BI provides the following processing options:

- ▶ Only PSAData is just written to the PSA and is not transferred or updated to additional data targets (for example, InfoCube, DSO).

Defining the maximum number of processes in the source system influences the target system processes as well and improves the load performance.

- ▶ PSA and then data targets

Each data package is written to the PSA. The same process writes the data package to the data target after the PSA is updated successfully. Updating the data is provided serially, packet by packet.

- ▶ PSA and data targets in parallel

If the data package is successfully loaded into the PSA, a second process writes the data into the data targets, while the other data package, if any, is being loaded to PSA.

Figure 5-1 illustrates the data update types.

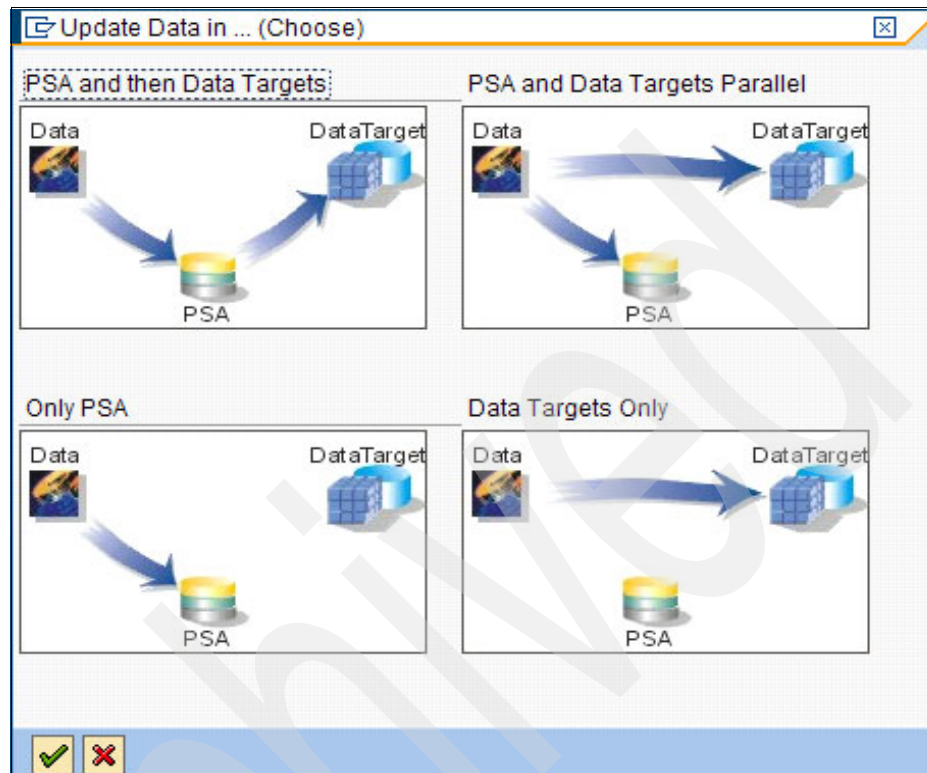


Figure 5-1 BI update types

Independent of the DB2 functions and tuning options, the data update types impact the SAP BI load performance and run time.

Note: Depending on the operation management processes and the business requirements, the following points should be considered:

- ▶ Control of the processes (best in PSA and then data targets)
- ▶ Data security (best in Only PSA or PSA and then data targets)
- ▶ Load performance (best in Only PSA or in PSA and data targets parallel)

With SAP BI 7.0, the data load from the PSA into the InfoProvider is done by a DTP.

5.1.1 Load PSA

To load data into the SAP BI system, we have to define the source, which provides and extracts the needed data. This process includes the specification of the *source system* and the DataSource.

Figure 5-2 shows a definition of the source system based on a logical SAP system, which represents the client 900. After definition of the DataSource, data can be extracted from this source system. The data will be extracted based on the SAP-provided ALE communicating from the business client of the source system.

The screenshot shows the 'Create SAP Source System' dialog box. It is titled 'Create SAP Source System' and has a close button in the top right corner. The dialog is divided into two main sections: 'Connection of BW to source system' and 'Connection of source system to BW'.
The 'Connection of BW to source system' section contains:
- 'Available destination': BI9CLNT100
- 'or create new destination:' section with three empty input fields:
 - Target computer (server)
 - System ID (system name)
 - System number
- 'Background user in source system': ALEREMOTE
- 'Password for source system user': masked with asterisks
- 'Repeat password': masked with asterisks
The 'Connection of source system to BW' section contains:
- 'Background user in BW': BWREMOTE
- 'Password for BW user': masked with asterisks
- 'Repeat password': masked with asterisks
At the bottom of the dialog, there are two icons: a green checkmark and a red X.

Figure 5-2 Create a source system

Figure 5-3 shows the available options.

Select Source System Type

- Automatic. Create SAP System
- Manually Create SAP System
- SAP Business Information Warehouse
- File System (Manual Metadata, Data Using File Interface)
- Database System (Data and Metadata Using SAP DB Connect)
- SAP UDC System (Data and Metadata from UDC and Portal Server)
- Webservice System (Metadata Manually, Data with Webservice Push)
- External System (Data and Metadata Transfer Using Staging BAPIs)

✓ ✗

Figure 5-3 Select the source system type

After the source system, the DataSource and the InfoPackage (see Figure 5-4) are defined. The Infopackage may be scheduled.

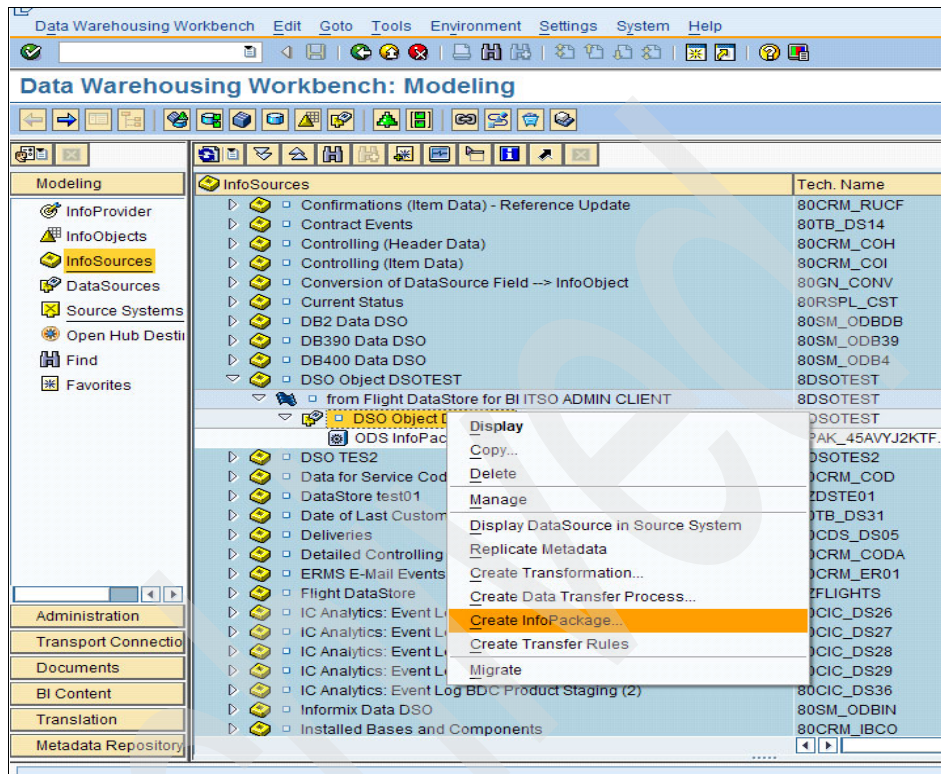


Figure 5-4 Create InfoPackage

In BI 7.x releases the scheduling of the InfoPackage is started by opening the context menu by right-clicking **DSO Infopackage**, as depicted in Figure 5-5.

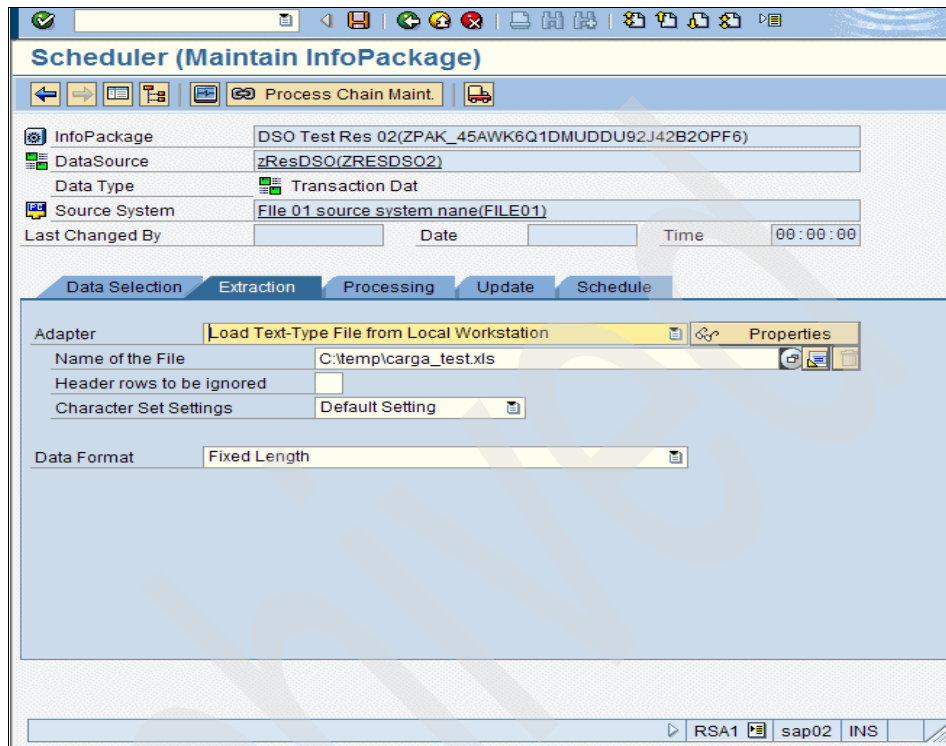


Figure 5-5 The location of external data

Note the available tabs (registers) in Figure 5-6, such as:

- ▶ Data Selection (defines loading transaction data from the source system)
- ▶ Extraction (defines the flat file location; see Figure 5-5 on page 61)
- ▶ The Processing options, as shown in Figure 5-7 on page 63 and described in “Processing options” on page 63
- ▶ The associated data targets such as DSO and cubes
- ▶ The Update Mode, such as full or delta update
- ▶ The Scheduling options

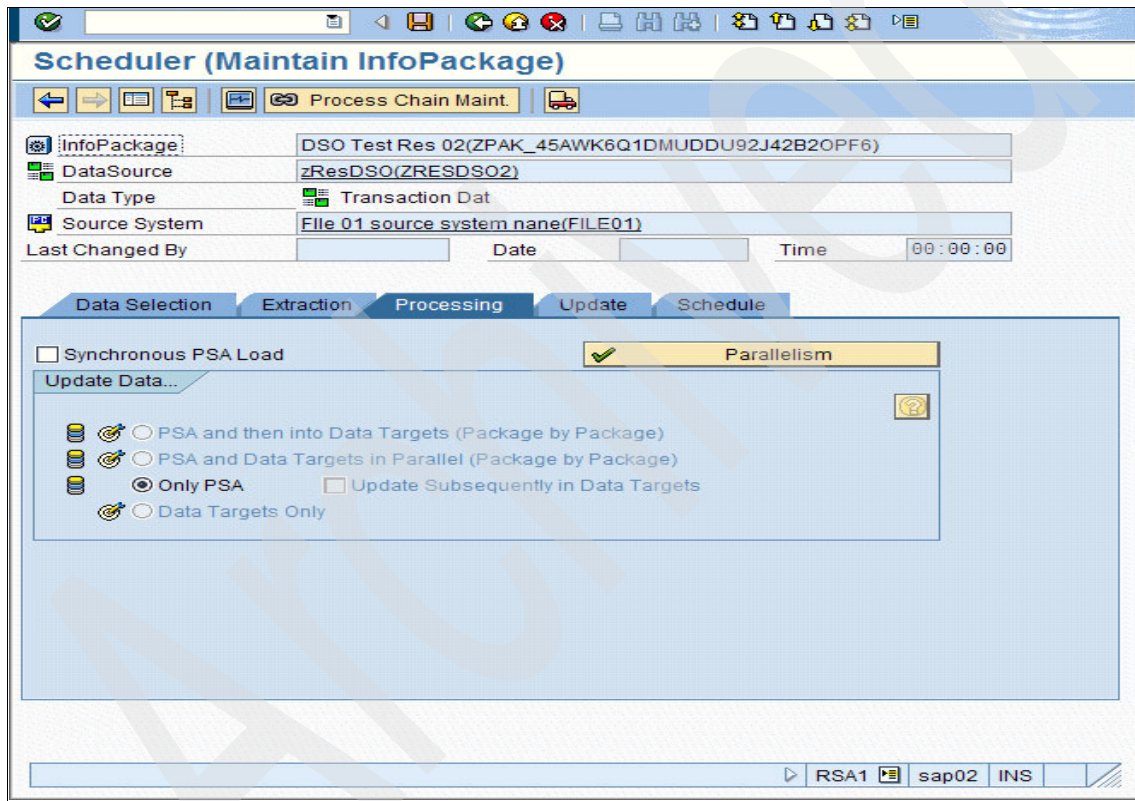


Figure 5-6 The processing options

During the first test case, we defined the load into DSO using the *Only PSA* option. Because no target object is defined, the BI system visualizes this case using the processing activities, as shown in Figure 5-7. First the PSA is loaded. In a second step the data target, such as DSO or InfoCube, can be populated.

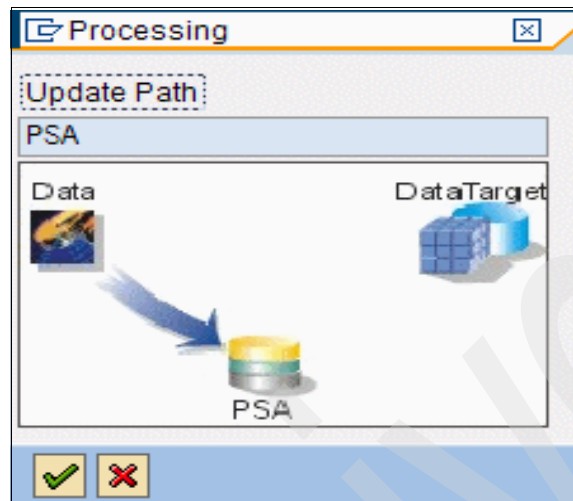


Figure 5-7 Loading PSA only (no target is defined)

Processing options

After all specifications are performed, the PSA load can be started using the start options. The following options are available:

- ▶ Start Data Load Immediately, which starts the defined activity in a dialog work process. The profile parameter `rdisp/max_wprun_time` is set to 3600 in a BI system.
- ▶ Start in Background, which starts a job assigned to a background (batch) work process.

Figure 5-8 shows the provided options.

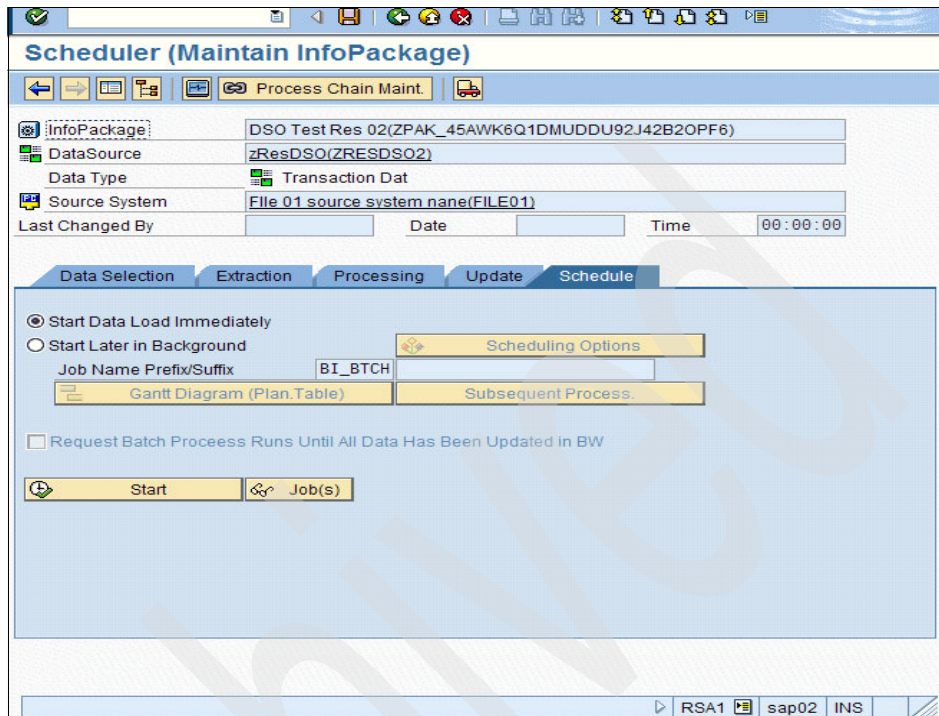


Figure 5-8 The scheduling options

The PSA load process was running successfully. Choosing the monitoring function from the Administrator Workbench or starting the RSMO transaction, the protocol of the data load is available as depicted in Figure 5-9.

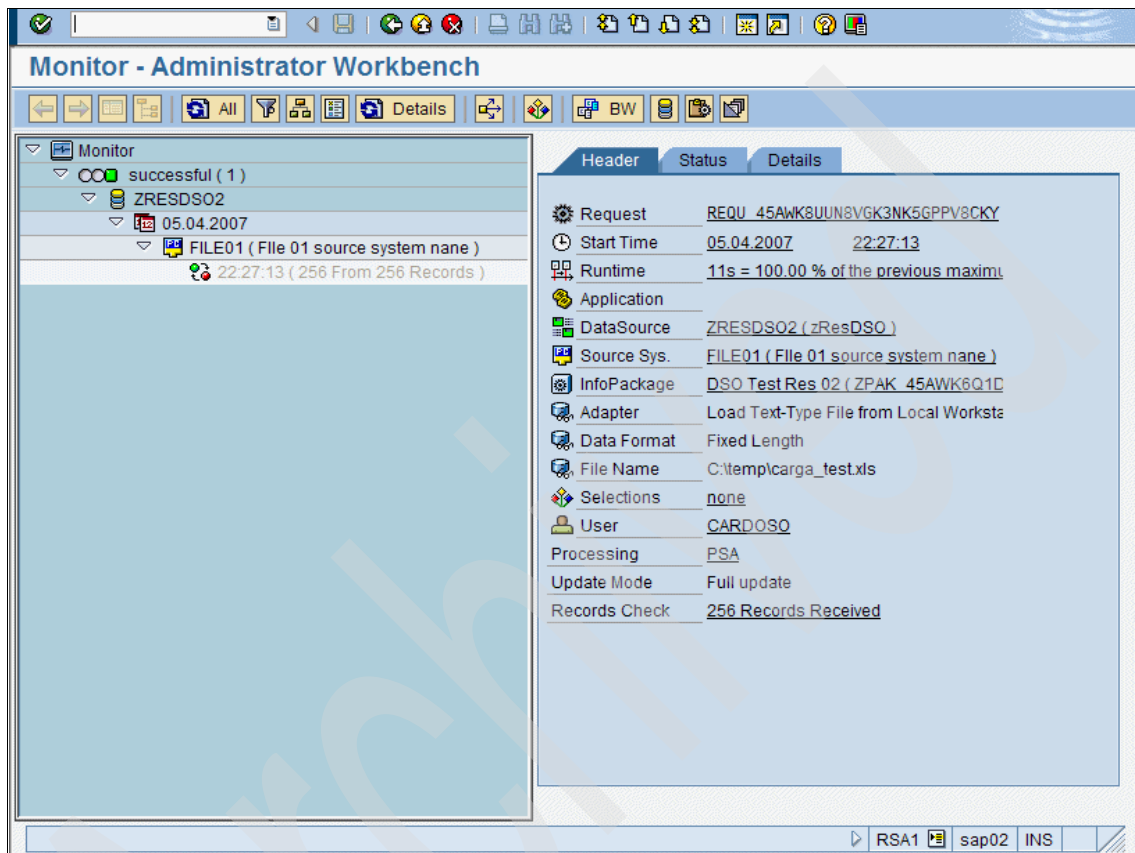


Figure 5-9 The Monitor functions (RSMO)

The TAB header provides information about the defined and used BI objects. Additionally, it gives an overview of the runtime parameters. Selecting the **Details** tab provides the single steps and the associated time periods.

Figure 5-10 shows the following:

- ▶ The phases of the load run, such as extraction, transfer, and load (ETL)
- ▶ The data package size
- ▶ The number of records loaded into the PSA
- ▶ The time used for the single step

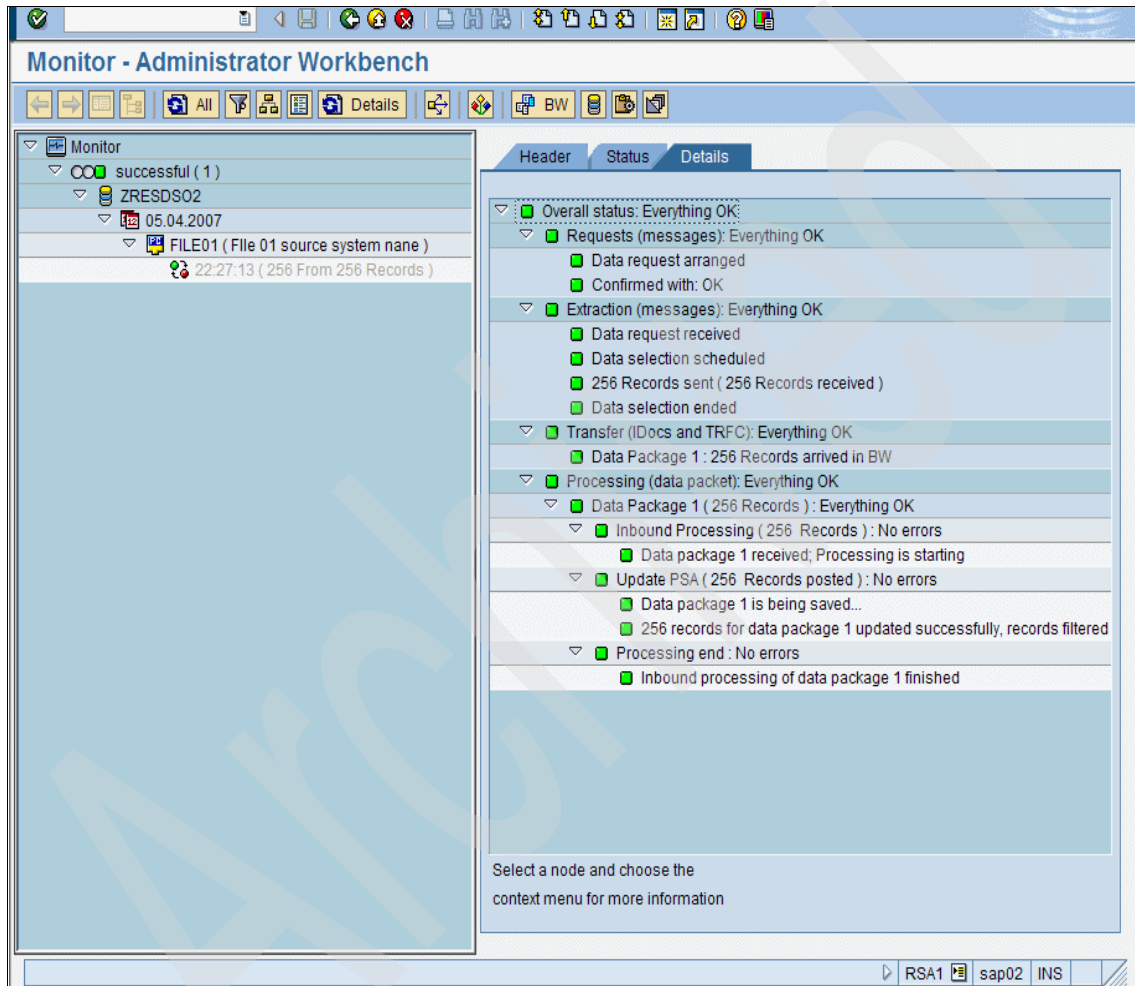


Figure 5-10 Monitoring the PSA load

The package size depends on the SAP BI customizing values in the rscustv6 transaction, as shown in Figure 5-11.

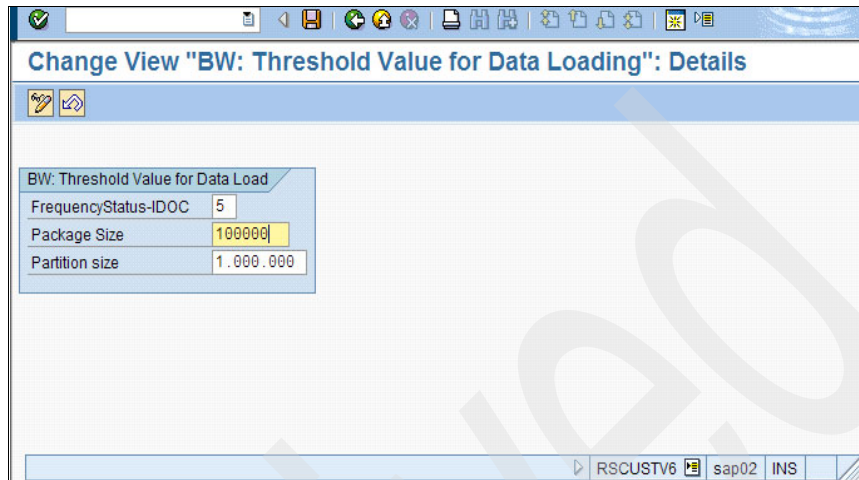


Figure 5-11 BI threshold value for data loading

The PSA table name is created automatically and is entered into the RSTSODS table. The values in Figure 5-12 include the technical name, which is /BIC/B0000110000. The naming convention shows /BIO/B000, or /BIC/B000, or the DSO tables.

Field	Value
ODSNAME	/BIC/B0000110
VERSION	0
DATETO	01.01.9999
DATEFROM	01.01.1998
OBJSTAT	ACT
ODSNAME TECH	/BIC/B0000110000
PROGRAM	GP45AWJ68JB9Q5WRVATAXZBCSZ6
MAINTPROG	
USERAPP	NEW_DS
USEROBJ	ZRESDS02 FILE01
TSTPNM	CARDOSO
TIMESTMP	20.070.405.201.258
PARTNO	2

Figure 5-12 PSA newly created table name

Field names in table RSTSODS are misleading, because the table actually contains information about PSA tables rather than DSO tables. In former times (up to Release 1.3) the PSA was called ODS. An ODS as known today did not exist.

With 2.x, SAP renamed ODS to PSA, but the control tables could not be renamed, otherwise an upgrade would not be possible. The time stamp reflects the date of 2004/11/08 15:27:42.

Now the PSA table is created and loaded and can be used for further processing such as loading into targets like DSO or InfoCubes.

5.1.2 Management of PSA

Data size in PSA can grow as data is loaded daily. Each time a request for data load in the Infopackage is loaded to PSA, a new table partition is automatically

added. If you do not manage PSA properly, you may have too many partitions and wasted storage space for a new load. A best practice is to reclaim usage of space occupied by obsolete data in the PSA. Reuse of the same number of partitions is greatly facilitated by DB2 partition rotation. Partition rotation is the process whereby the number of partitions is kept the same, but the *oldest* partition now becomes the *newest*. Essentially, this occurs when the old data is archived or deleted and the space is now required for reuse.

Note that SAP BW has a PSA tablespace created with default option COMPRESS YES, which results in DB2 data compression. Data loaded into the PSA is often repeated character strings, so storage size usage benefits from compression. In addition to the storage space usage benefit, performance benefits from data transfer with less I/O activities, improving utilization of the buffer pool, which improves overall performance.

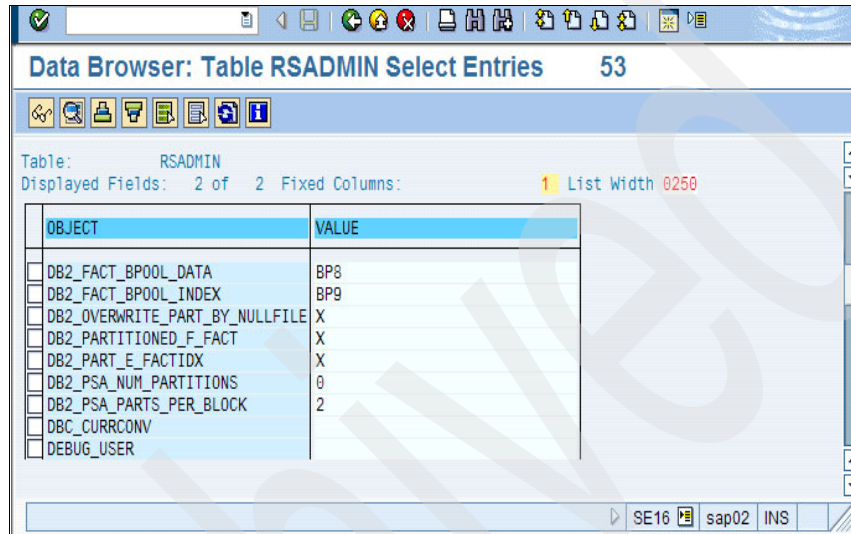
5.1.3 Load PSA DB2 compression

Because character strings are often repeated in the PSA table, DB2 compression usually improves the load performance. The repeated strings allow a high degree of compression for the data in the PSA table. In SAP BI, the tablespaces for PSA tables are created with the COMPRESS YES option and the DB2 data compression is activated. Compression leads to less I/O and improved utilization of the bufferpool which, in turn, enhances overall performance.

Compression can be turned off by setting DB2_PSA_COMPRESS = NO in the table RSADMIN, but this should be done on an exception basis.

5.1.4 PSA partitioning

Partitioning of the PSA table is advisable due to its large size. With DB2, the PSA table is always partitioned, so the parameter DB2_PSA_NUM_PARTITIONS in the RSADMIN table becomes obsolete. Figure 5-13 shows related PSA parameters.



The screenshot shows the SAP Data Browser interface for the RSADMIN table. The title bar reads "Data Browser: Table RSADMIN Select Entries 53". Below the title bar, there are navigation icons and a status bar indicating "Table: RSADMIN", "Displayed Fields: 2 of 2", "Fixed Columns:", and "List Width 0250". The main area contains a table with two columns: "OBJECT" and "VALUE". The table lists several parameters with their corresponding values.

OBJECT	VALUE
<input type="checkbox"/> DB2_FACT_BPOOL_DATA	BP8
<input type="checkbox"/> DB2_FACT_BPOOL_INDEX	BP9
<input type="checkbox"/> DB2_OVERWRITE_PART_BY_NULLFILE	X
<input type="checkbox"/> DB2_PARTITIONED_F_FACT	X
<input type="checkbox"/> DB2_PART_E_FACTIDX	X
<input type="checkbox"/> DB2_PSA_NUM_PARTITIONS	0
<input type="checkbox"/> DB2_PSA_PARTS_PER_BLOCK	2
<input type="checkbox"/> DBC_CURRCONV	
<input type="checkbox"/> DEBUG_USER	

At the bottom right of the window, there are navigation buttons labeled "SE16", "sap02", and "INS".

Figure 5-13 RSADMIN parameters

The number of partitions is not fixed. New partitions can be added if needed. SAP uses the DB2 feature of adding partitions. On DB2 the maximum number of partitions is 4096.

If all of the packages in the first partition are logically deleted (flagged as deleted), the partition is rotated and becomes the last partition. If all packages in any other partition are logically deleted, the partition is truncated.

To see the current number of partitions in DB2, execute the following SQL query (transaction DB2C):

```
SELECT PRT.PARTITION, PRT.LOGICAL_PART, PRT.CARD, PRT.LIMITKE
FROM SYSIBM.SYSTABLES TAB, SYSIBM.SYSTABLEPART PRT
WHERE
  TAB.TSNAME = PRT.TSNAME
  AND TAB.DBNAME = PRT.DBNAME
  AND TAB.CREATOR = <name of creator>
  AND TAB.NAME = <name of PSA table>
```


Using the SE14 transaction with the storage attributes, the number of partitions are shown in Figure 5-14.

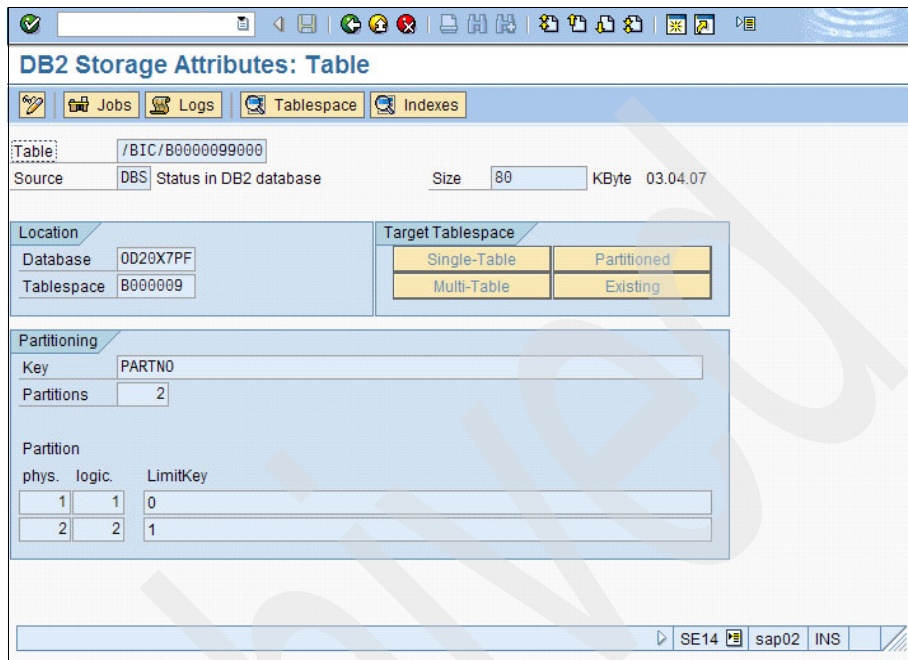


Figure 5-14 PSA table partitions

To see whether a package is flagged as deleted and how packages are distributed over partitions, see table RSTSODSPART with the PSA table name for ODSNAME_TECH. Note that column PARTNO in RSTSODSPART is always one more than LIMITKEY in SYSTABLEPART.

Archived

Best practices for InfoCubes

SAP BI 7.0 introduces several new features for the current InfoCubes. Some of these are discussed in this chapter, along with some of the new terminology with regards to InfoCubes.

This chapter covers the following areas:

- ▶ Terminology changes - types of InfoCube
- ▶ Repartitioning of InfoCubes
- ▶ Remodeling InfoProviders
- ▶ SAP compression

To start, we would like to give you a basic understanding of what an InfoCube is. InfoCubes are InfoProviders but are not necessarily data targets (they are only data targets if data is physically stored in them). By this definition, VirtualProviders (which previously were called Virtual Data Stores) are InfoProviders and not data targets. In SAP BI 7.0, the real-time characteristic can be assigned to a standard InfoCube. Real-Time InfoCubes (previously know as Transactional InfoCube) are used differently than standard InfoCubes.

An InfoCube, whether data target or InfoProvider, describes a self-contained data set. This could be used for reporting purposes. From a relational database standpoint, an InfoCube is a collection of relational tables defined in a star schema pattern: two fact tables (E and F) in the middle, surrounded by several dimension tables (see Figure 6-1).

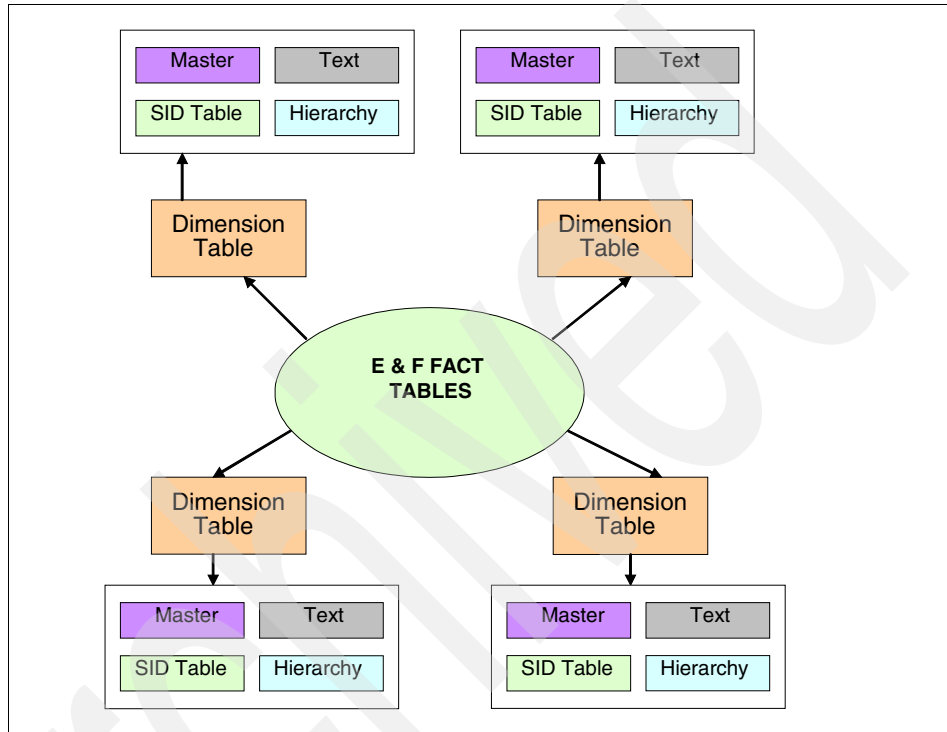


Figure 6-1 Star - schema

InfoCubes are supplied with data from one or more InfoSources, also known as DSOs, or with data from a different system. Optimizing the performance of loading data into the InfoCube object involves examining SQL efficiency and proper indexing.

Some of the terminology has changed and some new SAP BI capabilities have been added with this new release of SAP BI. Figure 6-2 shows a chart with those changes and additions.

Terminology Changes	
Old Term	New Term
Transactional InfoCube	Real-Time InfoCube
Virtual Data Stores	VirtualProviders (refers to RemoteCube, SAP RemoteCube and Virtual InfoCube)

New BI Capabilities for InfoCubes	
Repartitioning	
Remodeling	

Figure 6-2 Terminology changes and capability additions

6.1 Types of InfoCubes

Data targets are objects that store data physically. Some InfoCubes are data targets. These are *physical data stores*.

Other InfoCubes are not data targets, because there is no data physically stored in them. InfoCubes that are not data targets are *VirtualProviders*.

We look at these types in more detail.

6.1.1 Physical data stores (data targets)

InfoCubes that are physical data stores actually store the data physically and can be either basic InfoCubes or real-time InfoCubes.

Basic InfoCubes Basic InfoCubes store data physically. There are two fact tables per InfoCube: E and F. Data initially gets loaded into the F table. During SAP compression, this will be moved to the E- fact table.

Real-time InfoCubes These InfoCubes are both readable and writable, supporting parallel write access. Real-time InfoCubes are used for BI-BPS planning data. Standard InfoCubes can be converted to real-time InfoCubes if so required.

Real-time InfoCubes store data physically in two ways: by the transaction used for entering planning data and by using the BI staging.

Converting a standard InfoCube to a real-time InfoCube is used when the data in the standard InfoCube is no longer required. When the data in the InfoCube is required we then convert the standard InfoCube to a real-time InfoCube by using the process to convert a standard InfoCube to a real-time InfoCube.

6.1.2 VirtualProviders

The virtual providers are:

- ▶ RemoteCube

A RemoteCube is an InfoCube whose transaction data is not managed in the business warehouse, but rather externally. Only the structure of the RemoteCube is defined in BI. The data is read for reporting using a BAPI® from another system.

▶ SAP RemoteCube

In an SAP RemoteCube, data is stored in another SAP system.

▶ Virtual InfoCube with services

A virtual InfoCube with services is an InfoCube that does not have its own physical data source. The properties of the data source can be defined more precisely using a number of options. Depending on these properties, the data partitioning of the E- fact table can be defined while creating InfoCubes. To do this, select **RSA1** → **Create InfoCube** (see Figure 6-3).

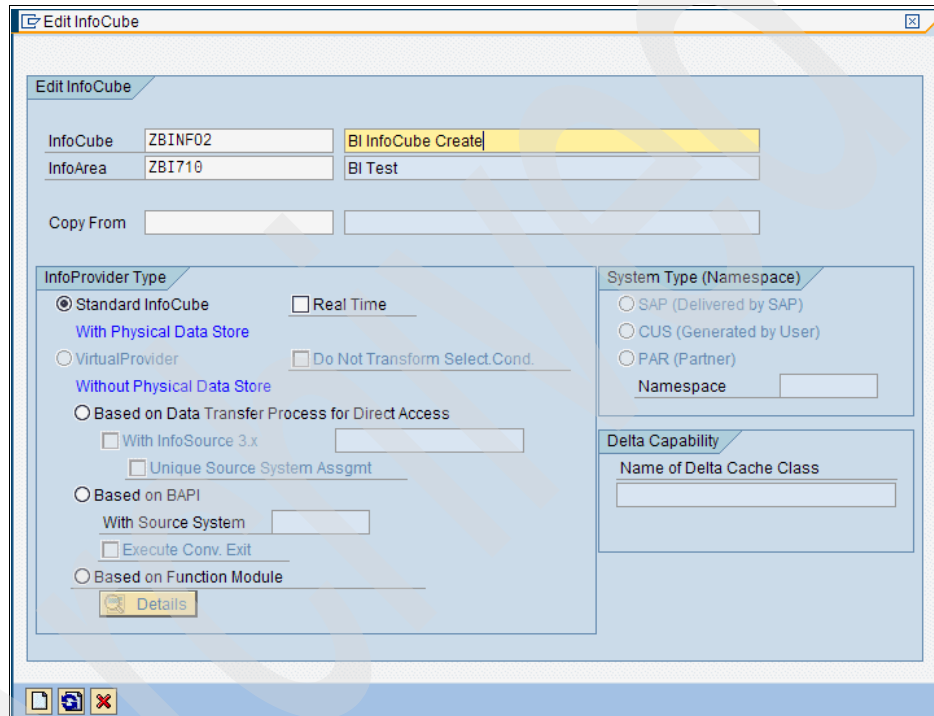


Figure 6-3 Create InfoCube

Specify the name of the InfoCube you would like to create and click the **Create** button at the bottom of the window. Define the cube with dimensions, key figures, and characteristics.

After selecting at least one time dimension, you can define the partitioning characteristics by selecting **Extras** → **Database Performance** → **Partitioning**. (See Figure 6-4.)

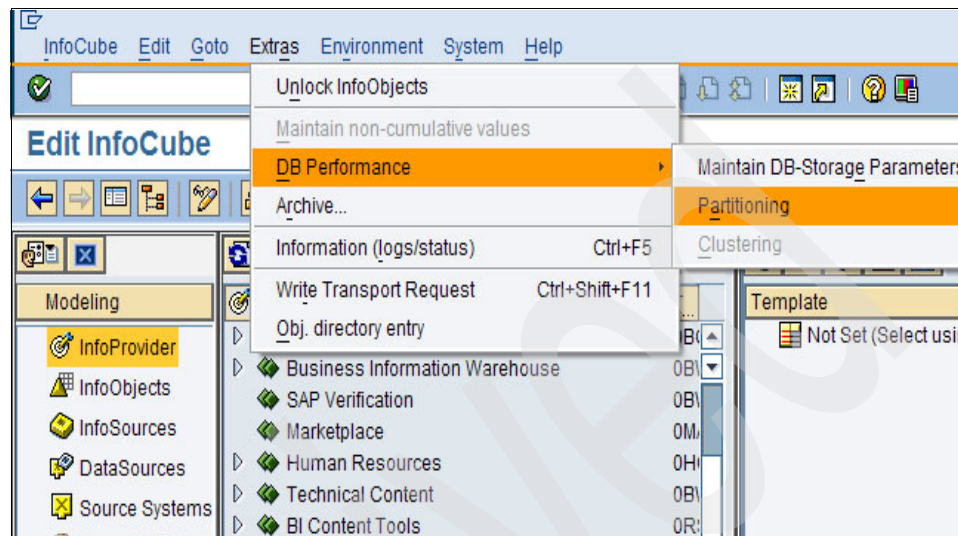


Figure 6-4 Partition InfoCube

Note the following points:

- ▶ Partitioning can only be done on InfoObjects 0CALMONTH and 0FISCPER.
- ▶ The E- fact table gets an additional column (SID_0CALMONTH or SID_0FISCPER), which it is partitioned on. This is necessary because all key columns contain dimension identifiers (DIMIDs), which are arbitrary and not known in advance (that is, ranges cannot be defined on dimids).
- ▶ For each query that has a restriction on time, an additional local predicate on the partitioning column is generated to allow partition elimination.

6.2 Repartitioning of InfoCubes

In this section we discuss repartitioning InfoCubes. This feature enables you to repartition InfoCubes that contain data and is available on BI 7.0 as of Support Pack 9 for DB2. (See SAP note 946641.)

The term *partitioning* in DB2 entails range partitioning. Partitioning helps both SAP compression and queries. Since the size of the E- fact table would grow consistently over time, SAP provides a feature to take advantage of DB2 partitioning. Currently SAP allows only time-based partitioning.

You would repartition an InfoCube when:

- ▶ The InfoCube was partitioned at the time of creation and the data has since grown to such an extent that you now require repartitioning the existing InfoCube.
- ▶ The InfoCube was not partitioned at the time of creation.

An InfoCube can be repartitioned in two ways on DB2, as DB2 does not support merging partitions. The following options for repartitioning are supported by DB2:

- ▶ Adding partitions

This feature enables you to add a partition to an E- fact table that already has been partitioned. This is the scenario: You had a partition condition from 01.2007 to 12.2007, but now require the partition until 02.2008. You will then need to add this new partitioning condition on that specific InfoCube.

- ▶ Complete partitioning

This type of repartitioning is when the complete data from the E-Fact and F-fact tables is copied to a temporary table and the new partition conditions are implemented by means of a complete repartitioning process, which is well documented in SAP Note 1008833.

6.2.1 Repartitioning implementation

To repartition an InfoCube from Data Warehousing (transaction RSA1) click the tab labelled **Administration** and then select **Repartitioning** (see Figure 6-5).

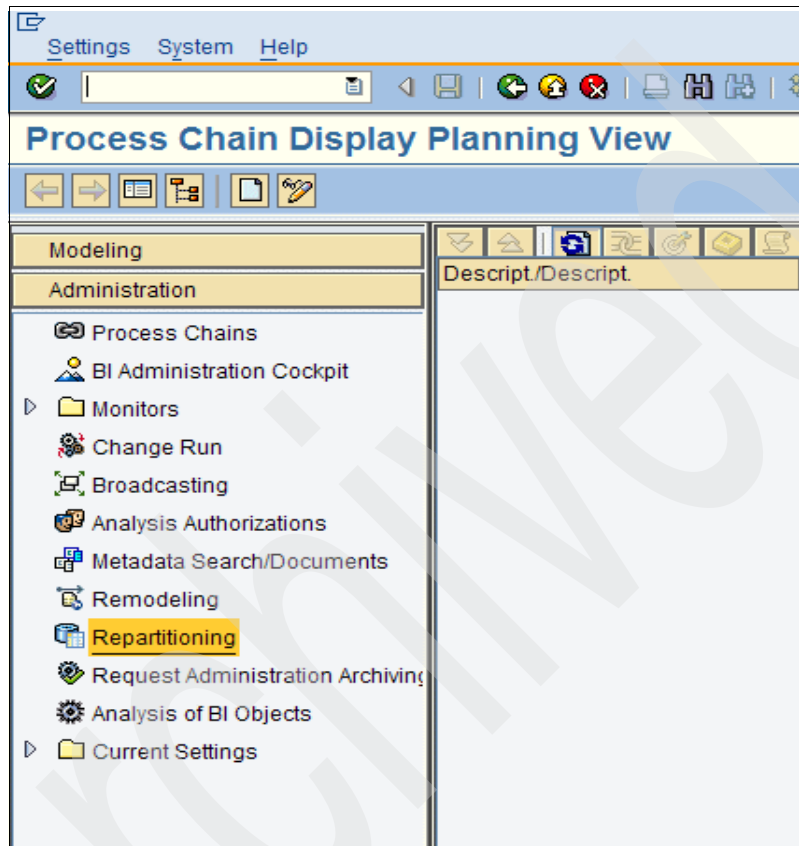


Figure 6-5 Repartitioning selection

At this stage (Figure 6-6) you will be able to repartition the InfoCube according to your requirements.

When presented with the repartitioning options, select the type of repartitioning. In our example we selected **Complete Repartitioning** and then clicked **initialize** to begin the process (shown in Figure 6-6).

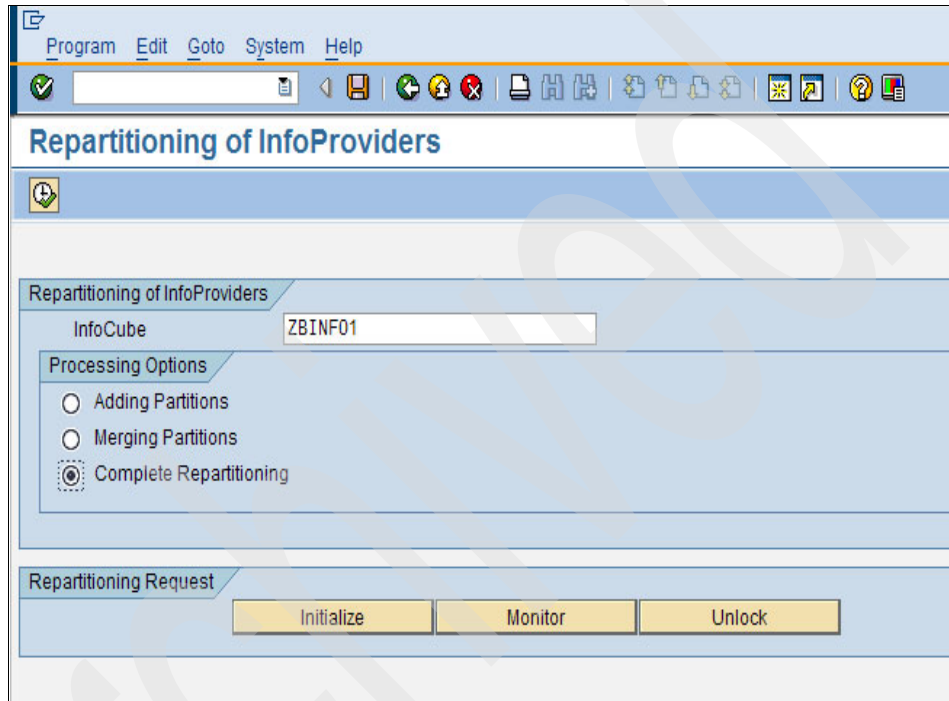


Figure 6-6 Repartitioning options

The system prompts you for confirmation of a backup of the system before you are able to continue with the repartitioning of the InfoCube (see Figure 6-7).

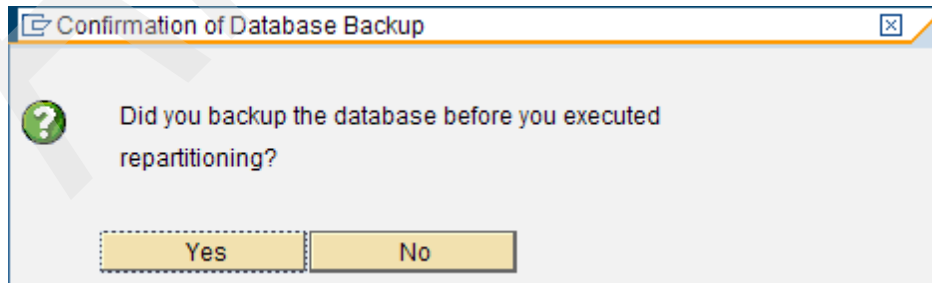


Figure 6-7 Confirmation of Database Backup

You will then be presented with a window that requires that you select a partitioning condition. This would be where you pick the time characteristic for partitioning (see Figure 6-8).

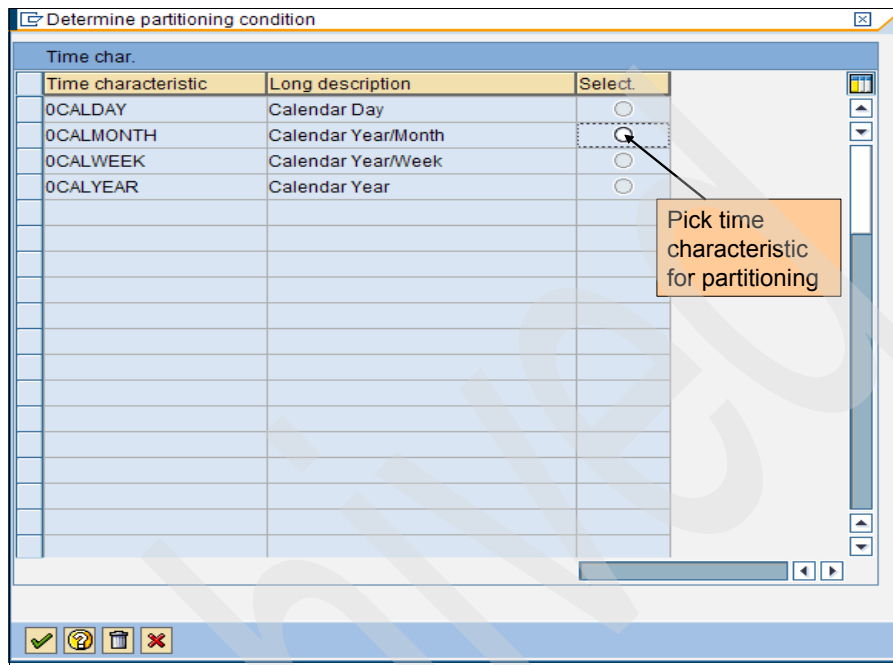


Figure 6-8 Determining partitioning condition

The repartitioning process can be monitored by using the monitor (previously shown in Figure 6-5 on page 80). The monitor gives you a picture of the current status of the repartitioning job and gives you detailed output of each step of the repartitioning process. You can also refer to SAP Note 1008833 for a detailed explanation for each of the repartitioning steps. See Figure 6-9 for an example of the details displayed while using the monitor for repartitioning.

The screenshot shows the SAP Monitor Requests window. The title bar includes menus for Monitor, Edit, Goto, Request, System, and Help. Below the title bar is a toolbar with various icons. The main content area is titled "Monitor Requests" and contains a tree view of the job steps. The tree view is expanded to show the "Complete Repartitioning" step, which is further detailed in a table.

Object Name	Description
MONITOR	Details of conversion
Complete Repartitioning	200701 << >> 200712
CREA_SHD_EFACT	Create shadow table /BIC/4EZBINFO1 for E fact table
CREA_SHD_FFACT	Create shadow table /BIC/4FZBINFO1 for F fact table
SPACE_CHECK	Check free space on DB
COPY_TO_SHD_EFACT	Copy all data records from /BIC/EZBINFO1 to /BIC/4EZBINFO1
COPY_TO_SHD_FFACT	Copy all data records from /BIC/FZBINFO1 to /BIC/4FZBINFO1
CREA_IDX	Create indexes on both shadow tables
SET_READ_LOCK	Set read lock for InfoCube ZBINFO1
INA_AGGR	Deactivate all active aggregates of InfoCube ZBINFO1
DELETE_FACTVIEW	Delete view of fact tables
CHECK_EFACT	Check data consistency of table /BIC/4EZBINFO1
CHECK_FFACT	Check data consistency of table /BIC/4FZBINFO1
SWITCH_EFACT	Swap E fact table /BIC/4EZBINFO1 with /BIC/EZBINFO1
SWITCH_FFACT	Swap F fact table /BIC/4FZBINFO1 with /BIC/FZBINFO1
CREA_FACTVIEW	Recreate view of fact tables
POST_ACT	Adapt BW metadata for InfoCube ZBINFO1
REPA_IDX	Repair indexes for both fact tables
ANALYZE	Calculate DB statistics for both fact tables
RELEASE_READ_LOCK	Reset read lock for InfoCube ZBINFO1
ACT_AGGR	Reactivate all aggregates of InfoCube ZBINFO1
CLEANUP	Various cleanup jobs

Figure 6-9 Image of the monitor for repartitioning

From within the monitor you are able to perform the following actions:

- ▶ Delete.
- ▶ Reset request.
- ▶ Reset step.
- ▶ Restart.

Information: For further information about the actions that can be performed from within the monitor, refer to:

<http://help.sap.com>

Search for Partitioning.

Figure 6-10 displays an example of an error in one of the steps of repartitioning an InfoCube. When there is an error in a step in the repartitioning process, you must double-click it so that you can view the details of the error and fix it accordingly.

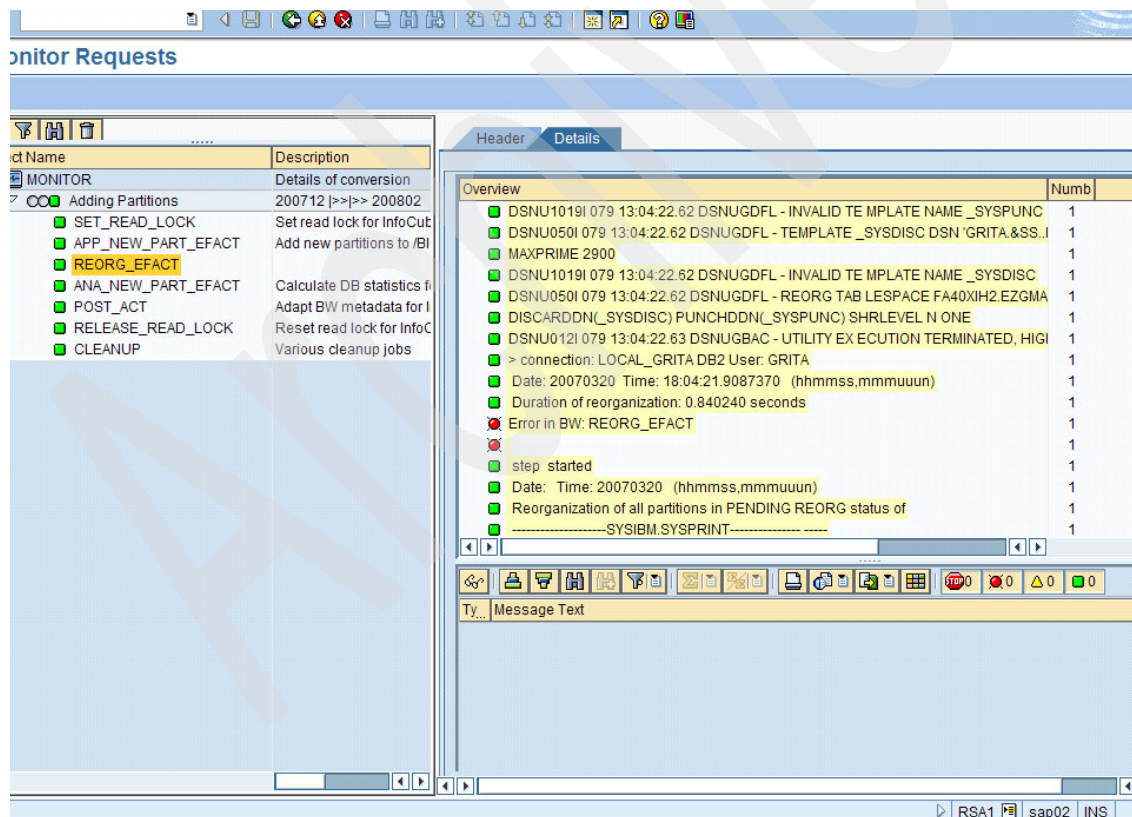


Figure 6-10 Display display of error in REORG_EFACT

In our example, our repartitioning process gave an error on step REORG_EFACT (highlighted with a red bullet in the example) where it tried to execute a reorganization. This type of error can be fixed by applying SAP note 842792, which must be implemented by the database administrator. Once the SAP note is applied, you can restart that step and your repartitioning process then runs to completion.

As in Figure 6-11, double-clicking any of the steps in the monitor allows you to retrieve more details on the step.

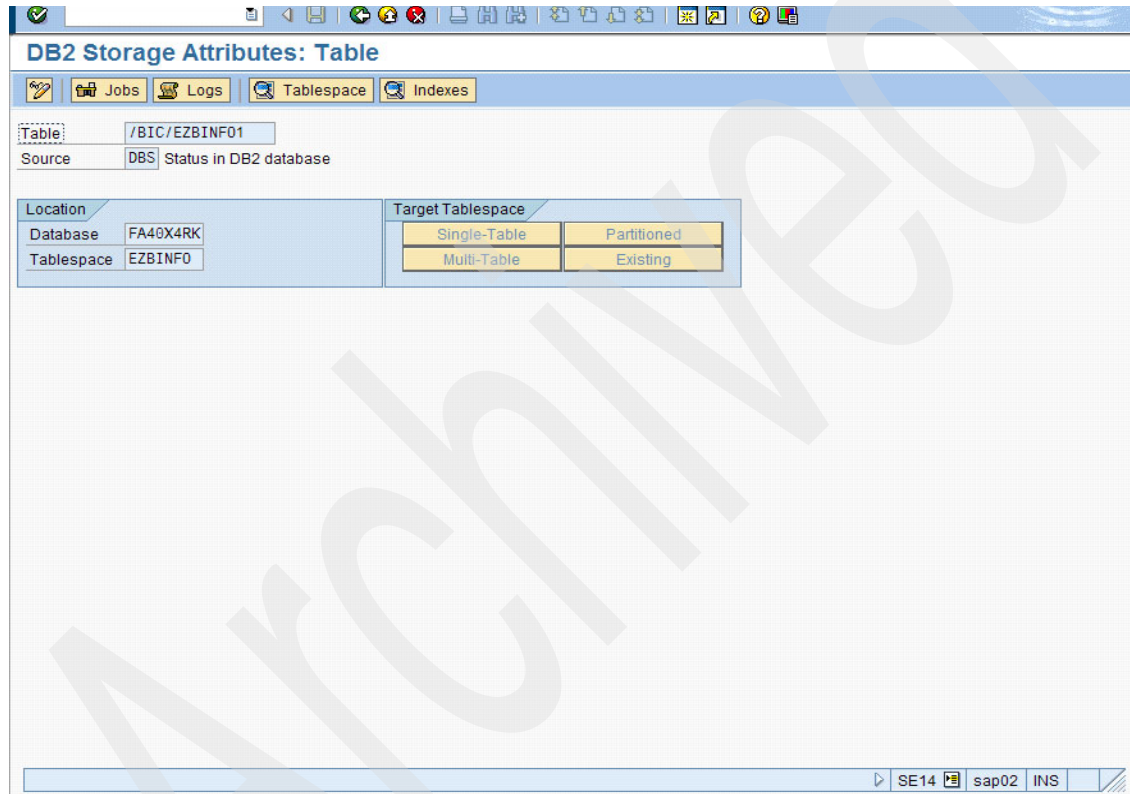


Figure 6-11 InfoCube E- fact table (/BIC/EZBINFO1) before repartitioning

Figure 6-11 also shows what the InfoCube E- fact table looked like before we did the complete repartitioning of the InfoCube. You see that the table has no partitions present. Executing transaction SE14 and entering the table name (which in our example is /BIC/EZBINFO1) will show this. Once you are on the window giving you the table information, select **Storage parameters** and you will be presented with the window shown in the example.

Once your repartitioning process is complete you can execute the same transaction shown in Figure 6-11 on page 85 to get an after picture of the table storage parameters.

Figure 6-12 shows the partitions created by the complete repartitioning process.

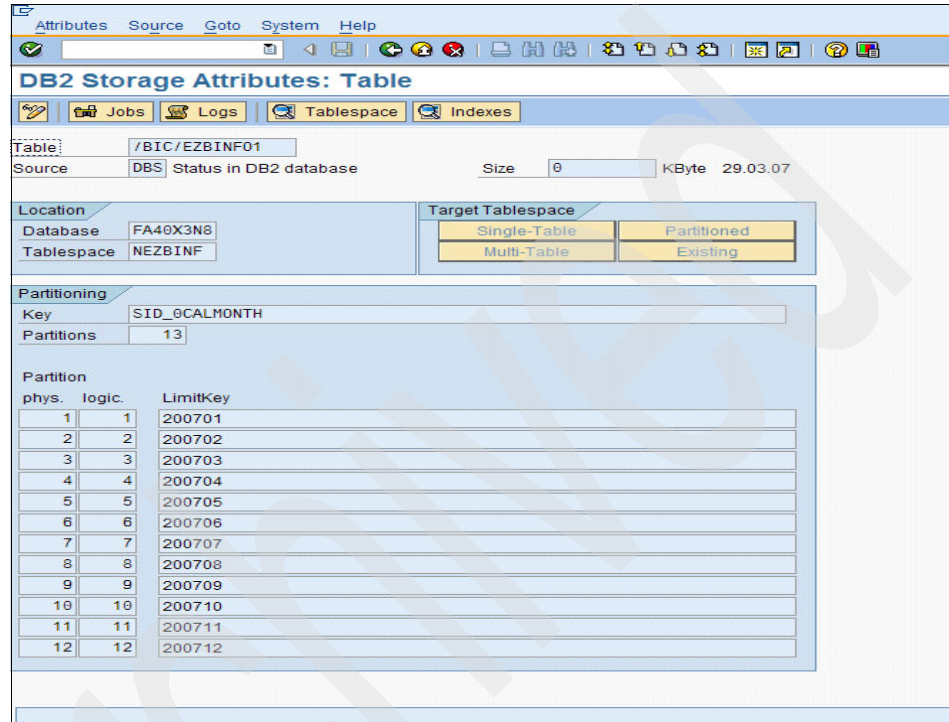


Figure 6-12 InfoCube E- fact table (/BIC/EZBINFO1) after repartitioning

You are also able to view the new partitions by executing transaction ST05 and doing a simple query on the table. In our example we executed the query “SELECT * FROM /BIC/EZBINFO1”. See Figure 6-13 to see the query panel resulting from executing transaction ST05 and the results of the query in Figure 6-14 on page 88.

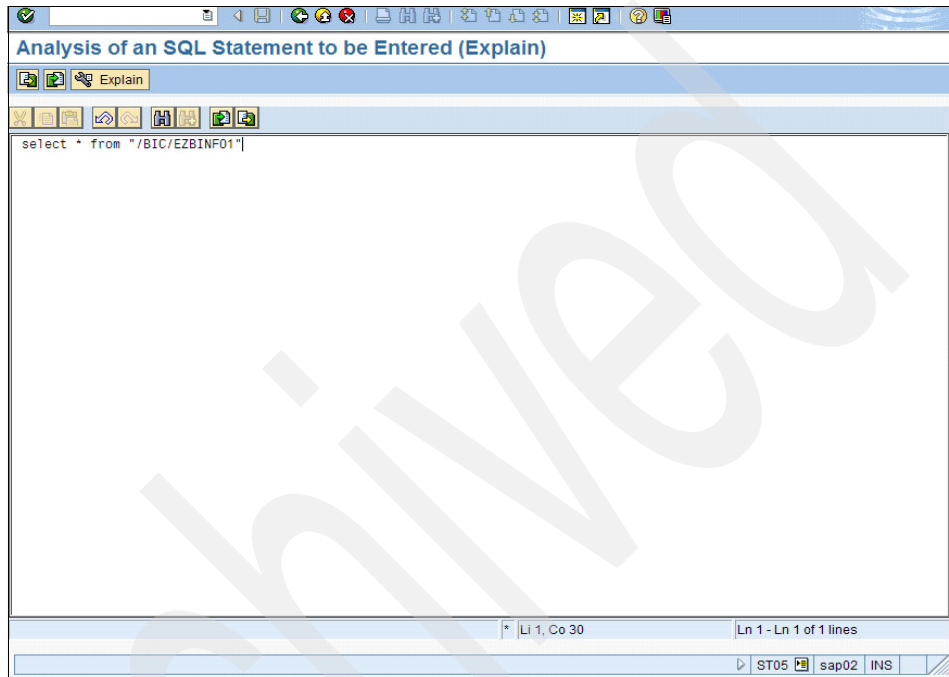


Figure 6-13 ST05 - running a simple query

Figure 6-14 is displayed once you have clicked the Explain icon on the menu bar (Figure 6-13 on page 87) and you have selected the **Table Information** tab. This tab displays the partitions that have been created by your repartitioning process.

The screenshot shows the 'Table Information' tab of the SAP DB2 Explain tool. The SQL statement is 'SELECT * from "/BIC/EZBINFO1"'. The table below shows the details of 13 partitions created for this table.

Schema	Name	Part	Rows	Pages	Row length	RUNSTATS?	REORG?	Bufferpool	VOLATI
BI9DB	/BIC/EZBINFO1	1	0	0	0	Not needed	Not needed	BP8	
BI9DB	/BIC/EZBINFO1	2	0	0	0	Not needed	Not needed	BP8	
BI9DB	/BIC/EZBINFO1	3	0	0	0	Not needed	Not needed	BP8	
BI9DB	/BIC/EZBINFO1	4	0	0	0	Not needed	Not needed	BP8	
BI9DB	/BIC/EZBINFO1	5	0	0	0	Not needed	Not needed	BP8	
BI9DB	/BIC/EZBINFO1	6	0	0	0	Not needed	Not needed	BP8	
BI9DB	/BIC/EZBINFO1	7	0	0	0	Not needed	Not needed	BP8	
BI9DB	/BIC/EZBINFO1	8	0	0	0	Not needed	Not needed	BP8	
BI9DB	/BIC/EZBINFO1	9	0	0	0	Not needed	Not needed	BP8	
BI9DB	/BIC/EZBINFO1	10	0	0	0	Not needed	Not needed	BP8	
BI9DB	/BIC/EZBINFO1	11	0	0	0	Not needed	Not needed	BP8	
BI9DB	/BIC/EZBINFO1	12	0	0	0	Not needed	Not needed	BP8	
BI9DB	/BIC/EZBINFO1	13	0	0	0	Not needed	Not needed	BP8	

Figure 6-14 Table Information tab - reflects partitions after running a simple query

6.2.2 Advantages of partitioning/repartitioning

In conjunction with SAP compression (see 6.4, “SAP compression for loads” on page 96), partitioning the E-Face and F- fact tables provides significant benefits for query performance:

- ▶ Partitioning enhances the ability of queries to execute in parallel, thus reducing overall elapsed time.
- ▶ When users specify a restriction based upon time, then SAP BI generates predicates against the partition limit keys to allow partition elimination. *Partition elimination* is the process of DB2 avoiding access to unnecessary partitions regardless of the E- fact table index used or whether a tablespace scan is used on the E- fact table. This has the benefit of allowing time-based filtering (at the partition level) to be combined with filtering from another dimension without requiring index ANDing or a Cartesian product of multiple dimensions before the E- fact table.

- ▶ Partitioning enables fast deletion when you are able to drop a particular partition.
- ▶ Query performance is enhanced when partitioning is used in conjunction with partition pruning.

Either or both partition elimination and parallelism act as a safety net to limit the negative effects of a poor access path choice. Thus, partitioning the E- fact table generally provides better overall query performance and stability.

Important: SAP Note 1008833 states that “You cannot transport repartitioning, because inconsistencies may occur in the target system. You must therefore execute repartitioning separately in each system of the system landscape with identical parameters before you can transport the InfoCube.”

6.3 Remodeling InfoProviders

With old versions of SAP BW we were unable to change the structure of the InfoProviders without deleting the data first.

Remodeling InfoProviders entails changing the structure of InfoCubes that already contain data, without losing any of your data and without having to rebuild the InfoCube.

The remodeling tool is available in SAP BI 7.0 as of BI support package 8. For one to be able to transport the remodeling rules from one system to another within the system landscape, you need be on SAP BI 7.0 Support Package 9. Refer to SAP Note 948798.

You may want to remodel an InfoCube when there has been a change in the company, that is, organizational changes. You may also want to remodel an InfoCube when you need to replace one InfoObject with another. Finally, you may want to remodel an InfoCube when the InfoObject characteristic/key is no longer required due to changed business processes or other reasons.

Important - You need to do the following before you start with your remodeling process:

1. Verify that the system has been successfully backed up.
2. Stop all process chains that are scheduled to run that impact/access the InfoProvider that you will be remodeling.

When you remodel an InfoProvider you have the options shown in Figure 6-15.

Choose from the following operations

<input checked="" type="radio"/> Add characteristic	<input type="radio"/> Add key figure
<input type="radio"/> Delete characteristic	<input type="radio"/> Delete key figure
<input type="radio"/> Replace characteristic	<input type="radio"/> Replace key figure

Figure 6-15 Remodelling options

6.3.1 Procedure for remodeling an InfoProvider

The Remodeling Tool is accessed from transaction RSMRT, or you can access it from the Administration menu bar by selecting **Remodeling** (see Figure 6-16).

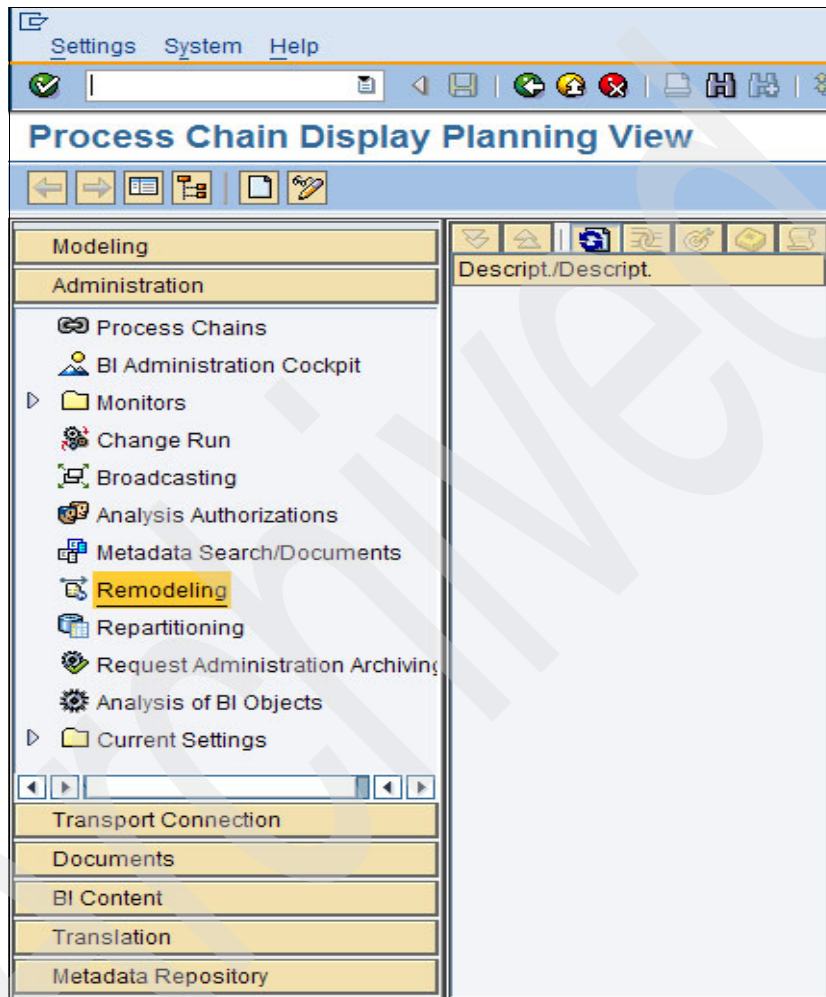


Figure 6-16 Administration - remodeling selection

Figure 6-17 demonstrates how you would identify a name for your remodeling rule and to which InfoCube you will be making the appropriate changes. Select **Create** to complete the creation of the remodeling rule.

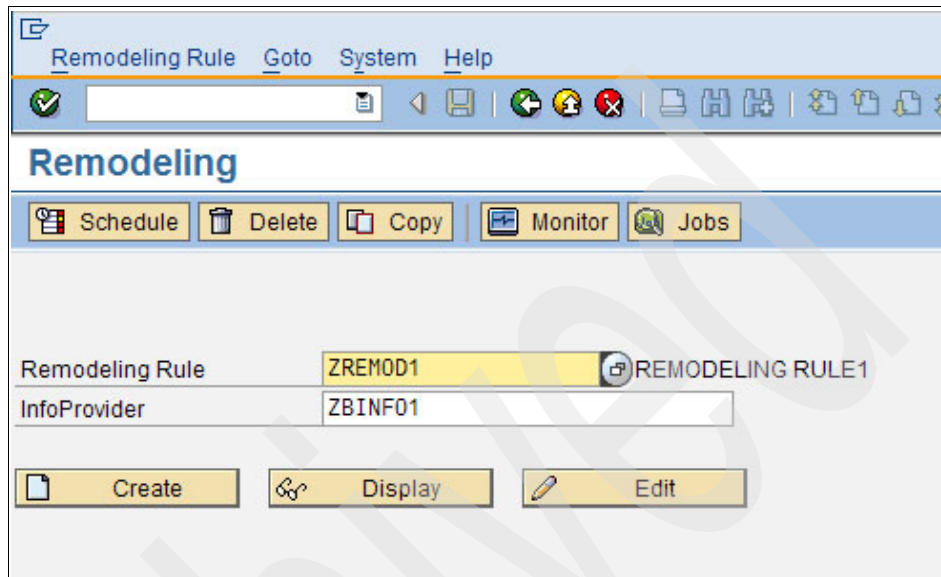


Figure 6-17 Creation of remodeling rule - step1

Step two of creating a remodeling room allows you to enter a description of your rule. Select **Transfer**. See Figure 6-18.

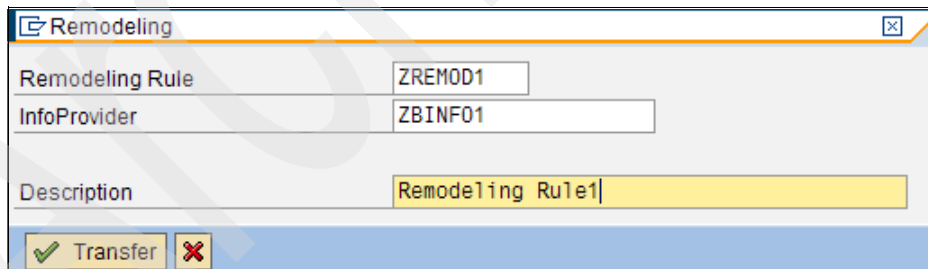


Figure 6-18 Creation of remodeling rule - step 2

Next you add an operation to the list by selecting **Add an operation to list** (or pressing Ctrl +F11). See Figure 6-19.

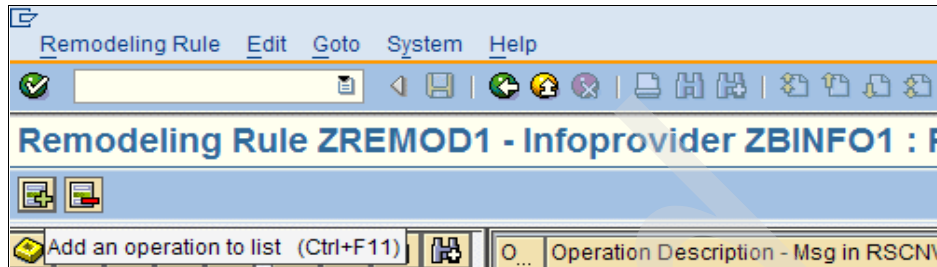


Figure 6-19 Select Add an operation to list

Figure 6-20 shows setting up the remodeling rule. When you have completed filling in the information required in this step, click **Transfer**.

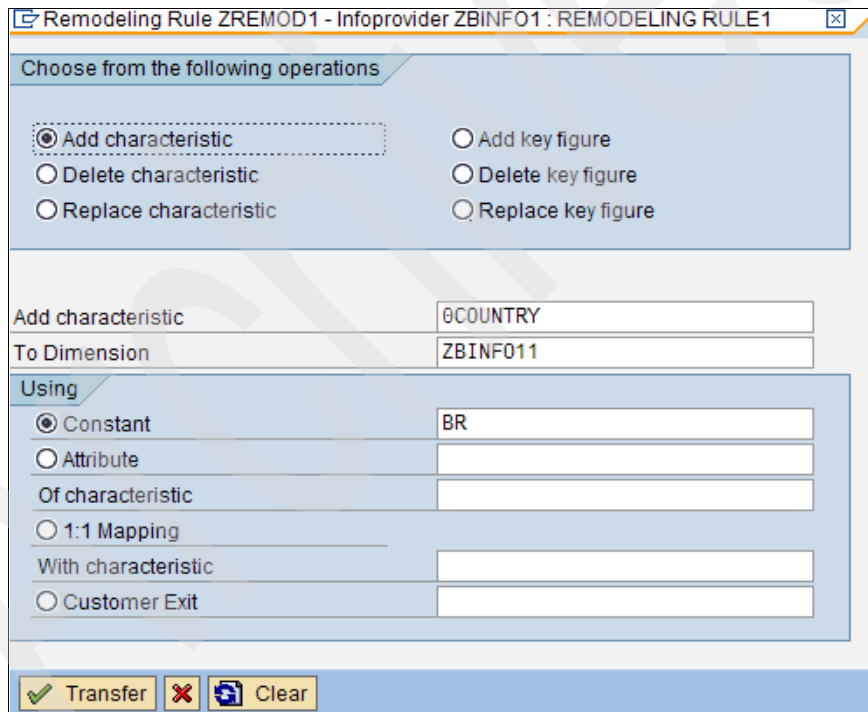


Figure 6-20 Set up remodeling rule (click Transfer)

At this step of the process you are able to execute a check on the remodeling rule you added (see Figure 6-21). Once the check has completed successfully without any errors you are able to click **Schedule**, which executes your remodeling rule. This job can be monitored from within the monitor, provided by SAP at this step of the process. The monitor will reflect exactly what is happening while the remodeling rule is being implemented. See Figure 6-22 on page 95 for an example of the remodeling monitor.

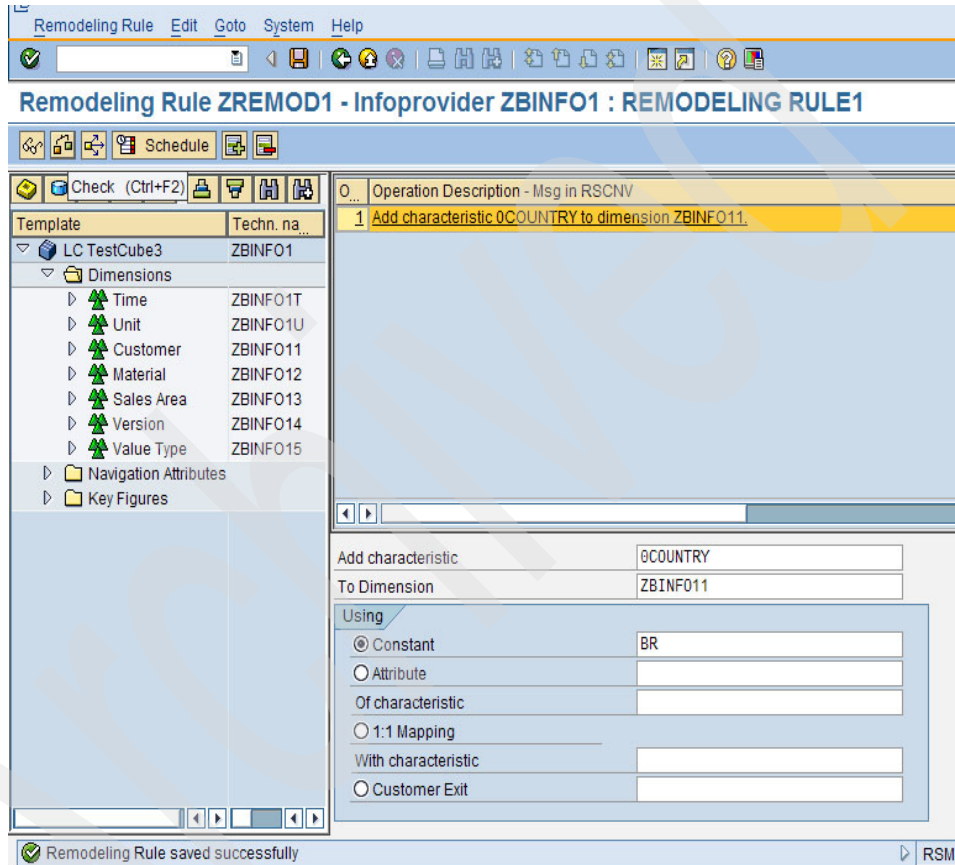


Figure 6-21 Remodeling rule check

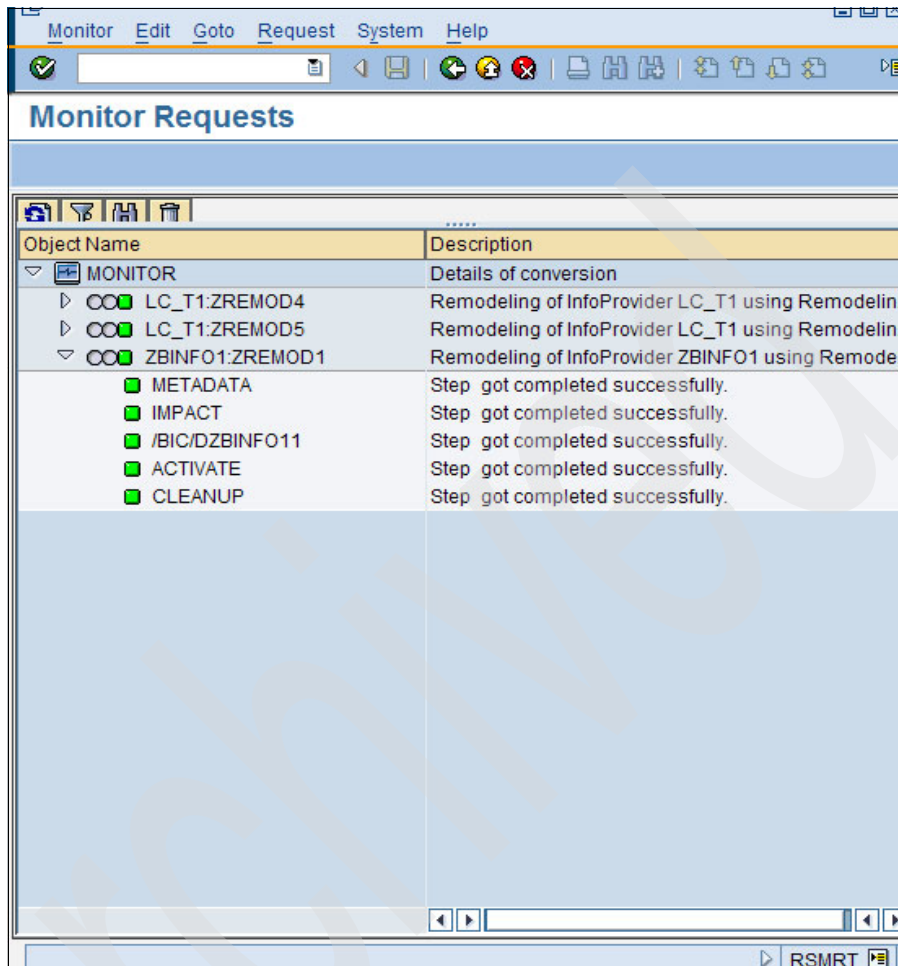


Figure 6-22 Remodeling monitor

Once the remodeling rule has been implemented successfully, you will be able to see your new characteristic/key figure that you have either added or replaced. If you delete the characteristic or key figure you will see that it removes it from your InfoProvider accordingly.

Note: Remember that any query, multiproviders, and son on that are associated with the InfoProvider may be deactivated during the remodeling process. You will therefore be required to reactivate these accordingly.

6.3.2 Benefits of remodeling InfoProviders

Remodeling allows you to add, delete, or replace a characteristic or key figure to an InfoCube without having to reload the data. Remodeling is not as involved as in previous versions, thus providing you with improved flexibility.

6.4 SAP compression for loads

SAP compression is the consolidation and summarization of data from an InfoCube's F- fact table into its E- fact table. Compression in this manner differs from traditional DB2 storage compression in that it does more than potentially reduce data storage requirements.

Implementing SAP compression from the F- fact table to the E- fact table, in conjunction with SAP partitioning of the E- fact table, is one of the most important factors in having good load and query performance.

A request is always loaded into the F- fact table. Compression copies the data into the E- fact table. Records in different requests that characteristics are all the same are merged into one record of the E- fact table. For non-cumulative cubes, an additional request containing the actual values of key figures is updated. These records are called reference points. At the end, the request is deleted from the F- fact table.

6.4.1 Advantages

There are several advantages to using SAP compression for loading data into E- fact tables. Some of those advantages are:

- ▶ SAP compression reduces the number of rows in the F- fact table (sometimes to zero).
 - A smaller F- fact table results in accelerated loading into the F- fact table.
 - SAP compression permits faster updating of the F- fact table indexes.
 - It accelerates aggregate rollups, since the F- fact table is the source of data for rollup.
 - It shortens RUNSTATS processing time on the F- fact table.
 - It reduces index REBUILD processing time if indexes are dropped as part of load.

- ▶ SAP compression places summarized records in the E- fact table.
 - The E- fact table can be partitioned (at the time the InfoCube is designed) to optimize query performance. If the InfoCube is not compressed, SAP partitioning will not take effect.
 - Data from overlapping packages (based upon time characteristics) is consolidated into E, and thus rows are somewhat *pre-summarized*.
 - Similarly, data from overlapping packages (based upon time characteristics) is consolidated into E, and thus rows are somewhat pre-summarized.
 - SAP compression simplifies query optimization of the InfoCube because the optimizer does not have to factor in filtering from the package ID.
- ▶ SAP compression resets the reference point for non-cumulative key figures. This can improve query performance, since fewer rows are needed to calculate key figure value queries for dates near to the reference point.
- ▶ SAP compression sometimes reduces data storage requirements when rows from more than one package have the same time value (packages 1 and 2 were loaded on the same day, for example). In situations in which the F- fact table contains packages that do not overlap time periods, SAP compression may not reduce disk storage requirements. But even in this situation, SAP compression is always recommended because of the other advantages.

6.4.2 Data validation

Data in the F- fact table must be carefully checked (via an acceptance and approval process) prior to compression. Compression results in the package ID in the F- fact table being replaced with a 00 or a 99 (see 6.4.3, “Cumulative data versus non-cumulative key figures” on page 98) and all records summarized into the E- fact table.

The replacement of the package ID means that these summarized records may not be deleted from the InfoCube after SAP compression using the corresponding package ID. For this reason, it is important that the data be carefully checked and validated prior to being summarized and prior to being compressed.

However, data can be deleted from the infoCube after SAP compression by other characteristics such as time. If packages have been consolidated due to overlapping time, then package deletion may not be possible.

Important: Packages cannot be deleted from the E- fact table using a package ID. Ensure that packages in the F- fact table have been reviewed prior to compression, since deletion from the F- fact table is much simpler.

6.4.3 Cumulative data versus non-cumulative key figures

There are two types of key figures in a fact table: cumulative key figures and non-cumulative key figures. If a cube has at least one non-cumulative key figure, it is called a non-cumulative cube. In the E- fact table, the data for non-cumulative key figures is stored in separate records, which are called reference points and have their own package DIMID to distinguish them from common records.

The E- fact table contains consolidated data from the F- fact table. For cumulative data, the key figure values are summed. For non-cumulative data, a function is applied to the key figures before the record is inserted in the E- fact table. Common functions include minimum value, maximum value, last value, and so on.

- ▶ *Non-cumulative key values* get assigned a package ID different from 0 in the E- fact table when the record is moved from the F- fact table to the E- fact table. For non-cumulative key figures, an additional row (the reference point) is inserted or updated in the E- fact table that has the same combination of dimension IDs, except for the time and package dimension. Rather than requiring summing all delta values from the F- fact table, the E- fact table reference point can be accessed directly, thus improving query performance.
- ▶ *Cumulative key values* get assigned a PID of 0 in the E- fact table. When a record in the F- fact table has the same key as a record already in the E- fact table a new record is not added to the E- fact table. Instead the cumulative key values for the record in the E- fact table are incremented by the value in the same record from the F- fact table.

6.4.4 Areas of impact

We recommend that SAP compression be implemented and updated on a regular basis (even if the ratio of compression appears to be insignificant) for the following reasons:

- ▶ Storage impact
 - SAP compression may result in reduced data storage requirements.
 - Occasionally, no change will occur.

- ▶ Load impact
 - Loading data into the InfoCube will be faster, because it usually involves inserting into an empty or small F- fact table and its dimension tables.
 - Additionally, if indexes on the F- fact table are dropped to improve insert performance, then REBUILD INDEX time will be much shorter due to a smaller F- fact table.

- ▶ Administration impact

Important: Lack of RUNSTATS on E and F is the major cause of poor compression performance.

- SAP compression results in a smaller F- fact table.
- SAP compression supports partitioning of the E- fact table.
- Administration of compressed InfoCubes is quicker because a smaller F- fact table means reduced RUNSTATS runtimes.
- Backups and restores are shorter in situations where SAP compression has resulted in reduced data storage requirements.
- Backups and restores can be shorter where partitioning of the E- fact table has been performed and the DBA chooses to back up/restore only changed partitions (since older partitions will not have changed since last backup).
- Deleting records from the F- fact table is faster due to the smaller size of the table.
- Updating of aggregates (rollup) is faster when the source InfoCube has been compressed due to the smaller size of the F- fact table.
- Reorganization is easier on a smaller F- fact table or current partitions only of the E- fact table.
- ▶ Query impact
 - Queries against the InfoCube are faster in situations in which summarization would normally be performed by the application during query time.
 - The summarized data will have fewer records to access.
 - Queries will also be accelerated due to E- fact table partitioning where queries can be defined against partitioning keys, or the query executes in parallel.
 - Query is also enhanced since compression results in less data being returned.

Important: After the initial compression of the F- fact table, ensure that RUNSTATS is run to maintain accurate statistics, as the E- fact table grows rapidly. The same applies as new partitions are populated.

6.4.5 Implementing SAP compression

Initially when you go live with new InfoCubes, it may be a good idea to wait a few weeks before deciding to compress regularly. As explained earlier, you could delete data from F- fact tables based on the package ID. So if you discover any data quality issues, you could reload the data after deleting one or more packages from the F- fact tables. Once you have solved all data quality issues, and you become confident with data loads, we recommend that you implement a periodic compression process.

Since compression will be adding rows to the E- fact table, which may make it necessary to drop/rebuild indexes and run runstats on the large E- fact table, it can be beneficial to compress periodically (for example, weekly, bi-weekly) rather than daily. This can help to reduce the administrative impact of compression.

6.4.6 Speed up compression of non-cumulative cubes

For non-cumulative cubes, the most difficult part is the update of reference points due to lack of a full, qualifying index. SAP Note 919597 describes an important code change to speed up this update of reference points. The solution was to replace the correlated subselect by a materialization into a temp table, which is then joined by the update.

6.4.7 Best practices for compressing large requests

Some best practices for compressing large requests are:

- ▶ Deletion of requests from an F- fact table is an expensive operation and often dominates elapsed time of compression. In order to speed up the deletion of requests:
 - Condense all requests.
 - When all requests are condensed, the F- fact table can be deleted completely, which is the fastest way.
 - Mass delete versus row deletion.
 - Be aware that request-save rollup cannot utilize mass delete.

- When not condensing all requests:
 - i. Drop the secondary indexes on the F- fact table.
 - ii. Call function module RSDU_INFOCUBE_INDEXES_DROP for the F- fact table.
 - iii. Condense the requests.
 - iv. Call function module RSDU_INFOCUBE_INDEXES_REPAIR.
This speeds up the deletion of the request from the F- fact table, because the deletion of entries from the secondary indexes is very expensive.
- Activate F- fact table partitioning.
See SAP notes 860830 and 894045.
- ▶ Tune the size of the bufferpool for the E- fact table index pages. If the bufferpool size is too small, it is the bottleneck for the insert into the E- fact table.
The default bufferpool for all index pages is BP3. The specific setting for fact table index pages via the RSADMIN parameter is DB2_FACT_BPOOL_INDEX (see SAP note 536074).
- ▶ When condensing many large requests:
 - a. Drop the secondary indexes on the E- fact table before starting the condensing.
 - b. Recreate the secondary indexes after all requests are compressed. Call function module RSDU_INFOCUBE_INDEXES_DROP and specify the E- fact table name.
 - c. Call function module RSDU_INFOCUBE_INDEXES_REPAIR to rebuild indexes.
Do not drop the secondary indexes if the E- fact table is already very large and the compression adds less than about 10% of new rows into the E- fact table.
- ▶ Restrict the size of the requests.
 - Requests should not be larger than 10 million rows.
 - Compression of huge requests results in long-running uncommitted UR (unit of recovery) and long elapsed time in case of a rollback.
- ▶ Use the partitioning of the F- fact table.
 - Refer to SAP note 860830 (DB2_PARTITIONED_F_FACT = X).
 - Configure the fast deletion of the partition for the request deletion from the F-Fact able (SAP note 894045).

- ▶ Use DPSIs for the cube's E- fact table.

With DPSIs the inserts into the smaller local index tree is faster (SAP note 917568).

Best practices for DataStore

In this chapter we discuss the best practices for optimizing SAP BI DataStore performance on a DB2 9 for z/OS platform. The DataStore is the new name as of SAP NetWeaver BI 2005 for the Operational Data Store (ODS). This chapter describes the following:

- ▶ Differences between DataStore and ODS
- ▶ Types of DataStores
- ▶ Performance aspects of DataStore query performance:
 - Indexing
 - Clustering
 - Partitioning

7.1 Differences between DataStore and ODS

As of Release BI 7.0, ODS objects are now called DataStore objects. If you are using standard ODS objects, there is no need to make any changes.

The changes introduced include a new type of DataStore object — the write-optimized DataStore object. The data from this new type of DataStore object is available immediately for further processing. You can specify separate runtime parameters for each DataStore object.

7.2 Types of DataStore

There are three DataStore object types available:

- ▶ Standard DataStore object
- ▶ DataStore object for direct update
- ▶ Write-Optimized DataStore object

7.2.1 Standard DataStore object

The Standard DataStore is the new name for the standard ODS object.

The DataStore object is stored in transparent flat tables. fact tables or dimensions are not created. The Standard DataStore creates three transparent tables, as shown in Table 7-1.

Table 7-1 DataStore tables

Position	PSA table	Naming convention
1	Active DataStore (A table)	/BIC/A<datastore_object>00
1	Activation queue	/BI{0IC}/B<10-digits>
1	Change log	/BIC/A<datastore_object>50

The activation queue contains the updated data records that are not activated yet. The change log contains the change history. The active DataStore table contains the active data. In the DataStore object for direct update, the data is immediately available in the active form, but it does not contain versions — it simply consists of a single table of active data.

7.2.2 DataStore object for direct update

The DataStore *for direct update* is the new name for the *transactional* ODS object.

The DataStore *for direct update* does not manage versions. For that reason it creates only the table of active DataStores. Because the data does not need to be activated, it is available immediately for analysis and reporting.

The data is retrieved from the external system via fill or delete APIs. It can be loaded concurrently from multiples users and the access is transactional.

7.2.3 Write-optimized DataStore object

The *write-optimized* DataStore is a new type of DataStore.

The *write-optimized* DataStore, like the DataStore *for direct update*, does not manage versions. For that reason, it creates only the table of active DataStores. The loaded data is available immediately for analysis and reporting.

The data is retrieved from data transfer processes. It can be used directly for reporting, but we recommend that this type of DataStore be used like a temporary consolidated storage area, a step for further load on target InfoProviders, standard DataStore objects, or InfoCubes.

7.3 Secondary indexes on DataStore objects

The primary index for DataStore objects is created automatically during DataStore objects creation. However, in order to tune query performance or partition DataStore object tables based on defined keys, you may need to create additional secondary indexes.

Figure 7-1 shows the initial status of the indexes on a new DataStore object.

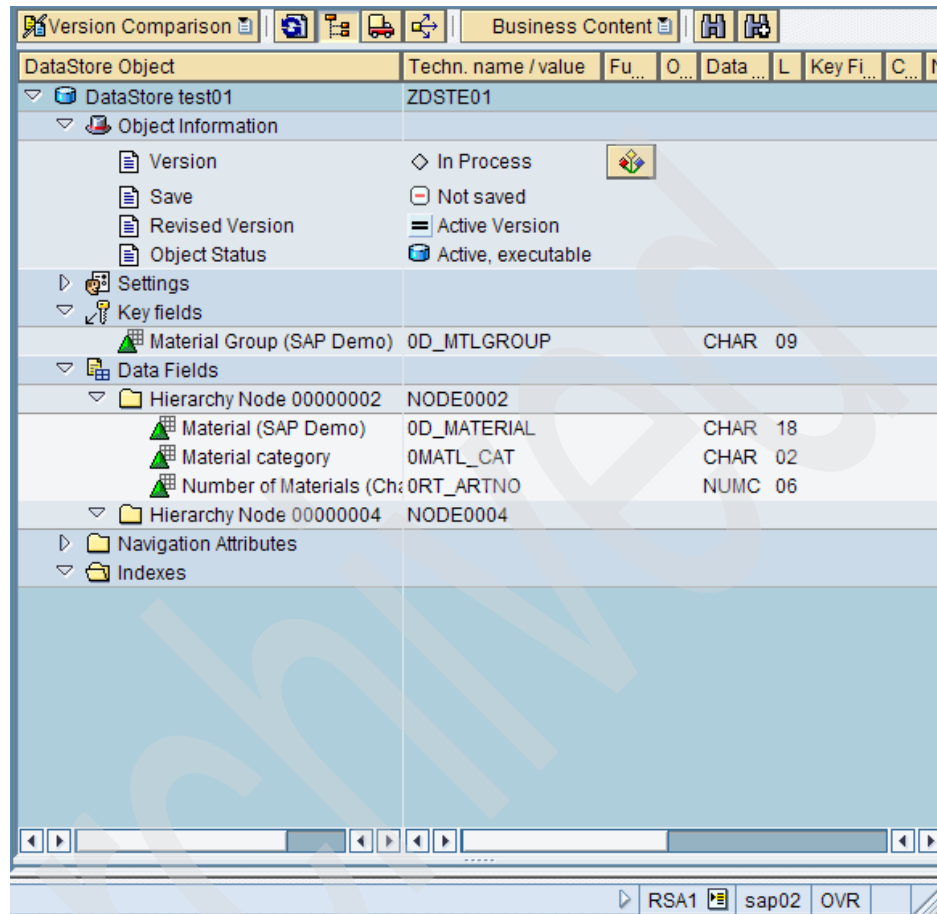


Figure 7-1 Initial status of indexes on DataStore object

The creating of a secondary index should be analyzed carefully.

Indexes, by nature, consume resources such as storage, memory, and CPU cycles to be maintained. Conversely, when the index is correctly selected, it improves performance. The DataStore objects are normally written on large data loads when having an index is an additional load. In this case we recommend that the index be deleted before the data load and then recreated for improved query performance. This procedure has to be analyzed, taking into consideration the time that the data load is penalized for the index maintenance compared to the time for deletion and recreation of the index.

7.3.1 Creating a secondary index on a DataStore object (DSO)

The secondary indexes can be created from the Edit DataStore Object window, as follows:

1. Call transaction RSA1 (**Administrator Workbench: Modeling** → **InfoProvider**). Double-click the DataStore object on which you want to create a secondary index.
2. On the Edit DataStore Object window, right-click **Indexes** and choose **Create New Indexes**, as shown in Figure 7-2.

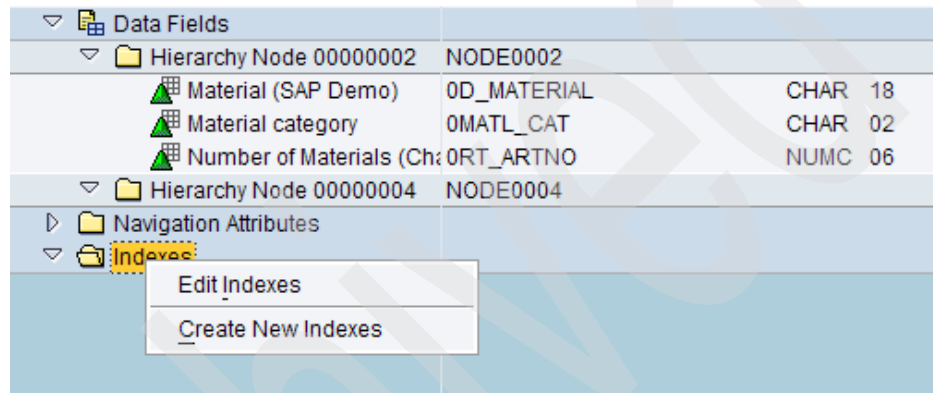


Figure 7-2 Context menu for the creation of index in a DataStore object

3. With your DataStore object in edit mode, you can use the context menu to create a new index, as shown in Figure 7-3.

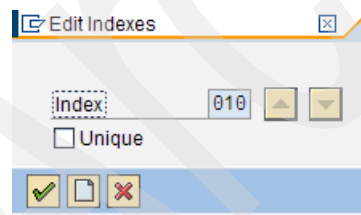


Figure 7-3 Select the name and unique attribute of a DSO secondary index

- By selecting the Create button, it will be created using the standard names with no fields inside. From this window you can create however many new indexes that you need. For example, we create two indexes (see Figure 7-4).

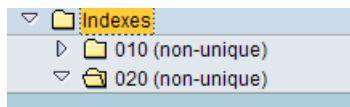


Figure 7-4 Field view of the DSO secondary index

In order to add the field to the definition, simply drag and drop the fields from the Key Field or Data Fields to each index. On our DEMO system we add two fields to each index, as shown in Figure 7-5.

Data Fields			
Hierarchy Node 00000002	NODE0002		
Material (SAP Demo)	0D_MATERIAL	CHAR	18
Material category	0MATL_CAT	CHAR	02
Number of Materials (Ch: 0RT_ARTNO		NUMC	06
Hierarchy Node 00000004	NODE0004		
Navigation Attributes			
Indexes			
010 (non-unique)			
Material (SAP Demo)	0D_MATERIAL	CHAR	18
Material category	0MATL_CAT	CHAR	02
020 (non-unique)			
Material (SAP Demo)	0D_MATERIAL	CHAR	18
Number of Materials (Ch: 0RT_ARTNO		NUMC	06

Figure 7-5 Added field to a DSO secondary index

Save and activate the DataStore object. This creates the new indexes in the database.

Run RUNSTATS to collect statistics for this new index. Use transaction DB13 (in the DBA Planning Calendar) to update statistics for one SAP object or transaction ST04, under the Performance and tuning tab, choose from the group **Tools** → **Update Statistics**.

7.4 Partitioning and clustering

With table-controlled partitioning, partitioning and clustering are separate concepts. The partitioning index no longer needs to be the clustering index as it does with index-controlled partitioning. For details on index-controlled

partitioning and table-controlled partitioning, see DB2 9 for z/OS documentation or the *DB2 Version 9.1 for z/OS Administration Guide*, SES1-2935-01.

Partitioning provides many benefits in the areas of data management, data availability, and query performance, especially for large tables like DataStore objects. The support from DB2 allows addition and rotation of partitions and provides more options for managing data. DB2 utilities can operate on a subset of the partitions, thus allowing the rest of the data to be available. Partitioned tables may be able to take advantage of parallelism for online queries, batch processing, and utilities. Query performance can improve if partitions can be eliminated from the search. The DB2 feature of data partitioned secondary index (DPSI) promotes partition-independence, which can reduce lock contention and improve index availability.

Clustering is the physical sequence in which records are stored in a table. The clustering sequence is defined by the clustering index for the table. DB2 tries to insert rows into the table to maintain the order of the clustering index. If you do not specify one of the table's indexes as clustering, then, by default, the clustering index is the first index that DB2 creates for the table.

You use the CLUSTER keyword on the CREATE INDEX statement to explicitly define the clustering index. When SAP creates tables, it arbitrarily chooses the primary index (index 0) as the clustering index. However, this may not necessarily be the best choice. The desired clustering sequence depends on how the data is accessed and used, which is mostly customer specific.

In DB2, the clustering index can be altered without the requirement to drop and recreate the affected indexes. In addition, with table-controlled partitioning, you have more flexibility over the choice of a clustering index, since it does not need to be the partitioning index.

When a range of rows is retrieved, the query performance can be improved if these rows are clustered and accessed by a clustering index. The rows can be read with fewer I/O operations, since the pages read are likely to contain more rows that need to be retrieved and DB2 can take advantage of its sequential prefetch feature.

Partitioning provides forced clustering because data within a partition cannot exceed the boundaries of that partition. Clustering is only a guide that DB2 tries to achieve. So, partitioning is more precise than clustering.

Also, partitioning can be used instead of indexing in some situations. Lower cardinality columns that are chosen for partitioning may then be removed from the index chosen as the clustering index.

Partitioning and clustering are both important performance tuning techniques for DataStore objects. DataStore objects are typically large, containing many rows of data. Queries against an DataStore object may retrieve thousands or more rows of data before a DB2 aggregate function, such as SUM, is applied. Queries that retrieve these large numbers of rows benefit most from partitioning and clustering.

7.4.1 Choosing the partitioning key

The partitioning key defines how a table is to be partitioned — it dictates which records are in which partitions. With table-controlled partitioning, the partitioning index is no longer required. The partitioning key and the limit key values for a table in a partitioned tablespace can be specified using the PARTITION BY clause and the PARTITION ENDING AT clause of the CREATE TABLE statement.

So, how do you choose the partitioning key? You can decide later if you also want the partitioning key to be a partitioning index. A *partitioning index* is defined as an index whose columns are the same as (and have the same collating sequence) or whose columns start with the columns in the PARTITION BY clause of the CREATE TABLE statement. In other words, it is an index that matches the partitioning key.

The characteristics of a good partitioning key are:

- ▶ It should provide partition elimination.

Users are not normally interested in all the data in a table. Instead, they are only interested in a subset of the data. If this subset of data is stored in one or a few partitions, then DB2 needs only to access these partitions and can eliminate the others from its search. Accessing and searching less data improves query performance.

- ▶ It should enable parallelism.

Each partition is a separate physical data set and has its own spacemap page. This allows the partitions to be operated on in parallel. In general, work done in parallel can be completed quicker than work done serially.

- ▶ It should be able to provide similar-sized partitions.

A key reason for partitioning is to handle large tables and to manage the size of the physical data sets behind the table. Having similar-sized partitions usually optimizes performance and availability.

In this section of the book we look at choosing the partitioning key to optimize query performance. Queries are executed against the active DataStore object table, so we are really discussing choosing the partitioning key for the

/BIC/A<DataStore_object_name>00) table. Keep in mind that this table also is involved in the activation of DataStore object data, so this must also be considered when deciding a partitioning scheme.

To choose an appropriate partitioning key, you need to understand your data and know how it is used and how it is managed. You should focus on how the data is commonly retrieved. Look at which key figure fields the users are using in their queries. Next look at which of these key figure fields are local predicates on the DataStore object and which are joins to master data. Only fields in the local predicates should be considered as columns in the partitioning key.

Unlike InfoCube queries, where SAP keeps a historical record of the InfoObjects referenced in a query in the table RSDDSTATAGGRDEF, you must trace (for example, ST05) and examine the SQL of a DataStore object query in order to analyze how the DataStore object is used by queries.

One common partitioning key for an DataStore object is date. This works if the data is commonly queried by date, for example, Display the sales volume by day or month. Partitioning by date also lends itself to data management. As time goes on, older data can be archived and deleted from the active DataStore object.

In addition to adding the indexes to improve query performance, we also recommend partitioning /BIC/AZOSASALE00 by CALMONTH.

If a greater number of partitions is required for reduced partition size or increased parallelism, then partitioning would be recommended by CURTYPE, CALMONTH. (Note, however, that the addition of CURTYPE as the leading partitioning column should only be made if CURTYPE *always* exists in queries against this DataStore object.)

After you choose a partitioning key, you need to decide on the number of partitions and the limit key value for each partition. The number of partitions should be determined by the anticipated table growth. The limit key value should be determined by what will give evenly sized partitions.

SAP provides a function (through transaction SE14) that collects limit key data based on a particular index. You specify an index and the desired number of partitions, and SAP determines the limit key value for each partition that will give evenly sized partitions.

Of course, this strategy only works for tables that are populated with data. It also requires that an index exists that will be the partitioning index. To use this function with table-controlled partitioning, you may have to define an index for your partitioning key. After using this index to collect the limit key data, you can drop the index if it is not needed for data access.

7.4.2 Choosing and changing the clustering index

The *clustering index* determines the order in which rows are stored in a table. It is defined with the CLUSTER keyword on the CREATE INDEX statement. The clustering index is also the clustering key. There can only be *one* clustering index on a table.

By default, in SAP, the primary index is always the clustering index. However, depending on specific customer situations, this is not always the best choice for the clustering index. This is especially true for DataStore objects.

To optimize the query performance on DataStore objects, you should consider explicitly defining a clustering index. Choosing the clustering index is similar to choosing the partitioning key — you need to understand your data and know how it is used. You should focus on how the data is commonly retrieved.

If there are queries that access a table index sequentially through an index that is not defined as clustering, and if this is a predominant way that the table is accessed, then the performance can be improved by specifying this index as clustering. Index-sequential access occurs in most cases for the range predicates (BETWEEN, >, <), for predicates that include only a prefix of an index, or for accesses through a non-unique index.

Changing a clustering index cannot be done through SAP. It must be done through DB2. To change the clustering index, use the ALTER INDEX command on both indexes. You can specify the CLUSTER option for the new clustering index or the NOT CLUSTER option for the index that you no longer want to be the clustering index. This change takes effect immediately. Any subsequent INSERT statements will use the new clustering index. The existing data remains clustered by the previous clustering index until the tablespace is reorganized.

Note that, since this change is not done through SAP, SAP is not aware of it. For this reason, a transport or SAP release upgrade that changes the structure of this table in some way that requires the table to be recreated will cause the changed clustering index to be lost on non-partitioned tables. However, this is *not* true for partitioned tables. All attributes of partitioned tablespaces, including clustering, are preserved in these cases.

7.4.3 Data Partitioned Secondary Indexes (DPSIs)

This feature allows physically partitioned secondary indexes. A DPSI index has as many index partitions as there are tablespace partitions. The index keys in partition *n* of the DPSI reference only data in partition *n* of the tablespace.

DPSI indexes provide an alternative to non-partitioned secondary indexes (NPSIs). There are some areas where NPSI indexes can cause performance and contention problems, namely, availability and partition-level operations. NPSIs can also result in significant contention while executing a query in parallel, since each parallel task must route through the same single B-tree index structure to access the separate data partitions. Therefore, DPSIs can improve parallelism performance. If a table and all its indexes are partitioned, then the data in separate partitions can be accessed in parallel.

However, DPSIs will not always improve query performance and may, in fact, cause query performance degradation. Queries against DPSIs that do not include predicates against the leading partitioning column are likely to experience performance degradation due to the need to probe each partition of the index for values that satisfy the predicate. Queries with predicates against the secondary index that also restrict the query to a subset of the partitions (by having predicates against the leading partition column) should see a benefit.

So again, you must know the queries that will be executing against your DataStore object in order to make an informed decision about whether to use DPSIs. If you do use them, you must decide which columns to include in the DPSI. The decision to use a NPSI or DPSI must take into account both data maintenance practices and the access patterns of the data. We recommend replacing an existing NPSI with a DPSI only if there are perceivable benefits, such as easier data or index maintenance, improved data or index availability, or improved performance.

Using DPSIs for DataStore objects

For DataStore objects, DPSIs should only be considered if queries against the DataStore object generally contain restrictive predicates against the partitioning columns (most important is the leading column of the partitioning key) or if the number of partitions is small (for example, 10 or less) and parallelism will be exploited.

Let us look again at the DataStore object query discussed earlier in this chapter. Assume that CALMONTH is chosen as the partitioning key. If it always exists in queries, then we recommend that the index CURTYPE, ACCT_ASGN, /BIC/ZMATLPLNT, PLANT be created as a data partitioned secondary index. The local predicates of CALMONTH (and CURTYPE) would restrict the query to a small subset of partitions.

SAP does not support the creation of DPSI indexes yet. They must be created with native SQL within DB2. Use the PARTITIONED keyword on the CREATE INDEX statement to create a data partitioned secondary index.

7.4.4 Partitioning a DataStore object

Ideally, an DataStore object would be partitioned as desired before it is populated with data. However, in reality it is often partitioned after it contains data (after it has become large and queries are found not to be performing well). Converting a DB2 table from non-partitioned to partitioned involves dropping and recreating the table, so you must take care to preserve the data in the table.

We recommend using a combination of SAP and DB2 tools to convert an DataStore object from non-partitioned to partitioned. According to the *SAP 6.40 DBA Guide*, DB2 utilities should be used to move data between tables if the table has more than one million rows or it is larger than 100 MB. In general, DataStore object tables meet this requirement. SAP table conversion techniques use SQL SELECTs and INSERTs to perform the data transfer, which is not as efficient as DB2 unload and load utilities when processing large amounts of data.

In the newer releases of SAP BI, this function will be implemented in the same way as InfoCube E- fact table repartitioning, described in 6.2, “Repartitioning of InfoCubes” on page 78.

The following procedure describes how to convert an DataStore object table from non-partitioned to partitioned using table-controlled partitioning. It is taken from the *SAP 6.40 DBA Guide*, but it has been expanded and tailored for table-controlled partitioning, as table controlled partitioning is the recommended method in DB2 9 for z/OS.

1. Call transaction SE16 (Data Browser) to check the number of rows in the table to be partitioned. (This is a reality check to verify that you have the same amount of data in the table before and after partitioning.)
2. Create a quiesce point for the entire DB2 subsystem. Be sure that you have full image copies for all the SAP data. Do not allow other users write access to the DB2 system until the table conversion is completed.
3. Use a DB2 utility to unload the data in the table to be partitioned to a sequential file. In this example, we used the DB2 UNLOAD utility. See Example 7-1 for sample JCL for the DB2 UNLOAD utility.

Another option is to use the DB2 REORG utility with the “UNLOAD EXTERNAL” option. (For complete details on the DB2 utilities, refer to *DB2 V9 Utility Guide and Reference*.)

Example 7-1 Example of JCL for DB2 UNLOAD utility

```
//UNLOAD JOB (999,POK), 'REORG', CLASS=A, REGION=OM,  
// MSGCLASS=T, MSGLEVEL=(1,1), NOTIFY=&SYSUID  
/*JOBPARM S=SC04  
// JCLLIB ORDER=(DB2TU.PROCLIB)  
//STEP1 EXEC DSNUPROC, UID=SMPLUNLD, UTPROC=, SYSTEM=DB2T
```

```
//SYSREC DD DSN=LYDIA.SMPLUNLD.SYSREC,  
// DISP=(NEW,CATLG,CATLG),  
// UNIT=SYSDA,SPACE=(TRK,(2,1))  
//SYSPUNCH DD DSN=LYDIA.SMPLUNLD.SYSPUNCH,  
// DISP=(NEW,CATLG,CATLG),  
// UNIT=SYSDA,SPACE=(TRK,(1,1))  
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *  
UNLOAD TABLESPACE OD60XSPU.XSAP FROM TABLE "BI9DB"."/BIC/ADSOTEST00"
```

After the data has been unloaded, call transaction SE14 (ABAP Dictionary: Database Utility). Specify the name of the table to be partitioned and click **Edit**. See Figure 7-6.

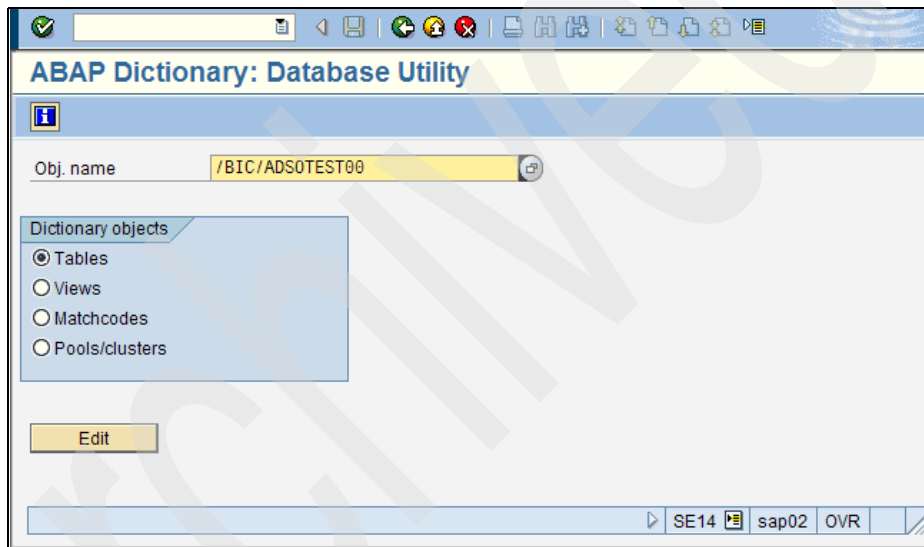


Figure 7-6 ABAP Dictionary: Database Utility

From the ABAP Dictionary: Utility for Database Tables window, choose **Storage Parameters**; see Figure 7-7.

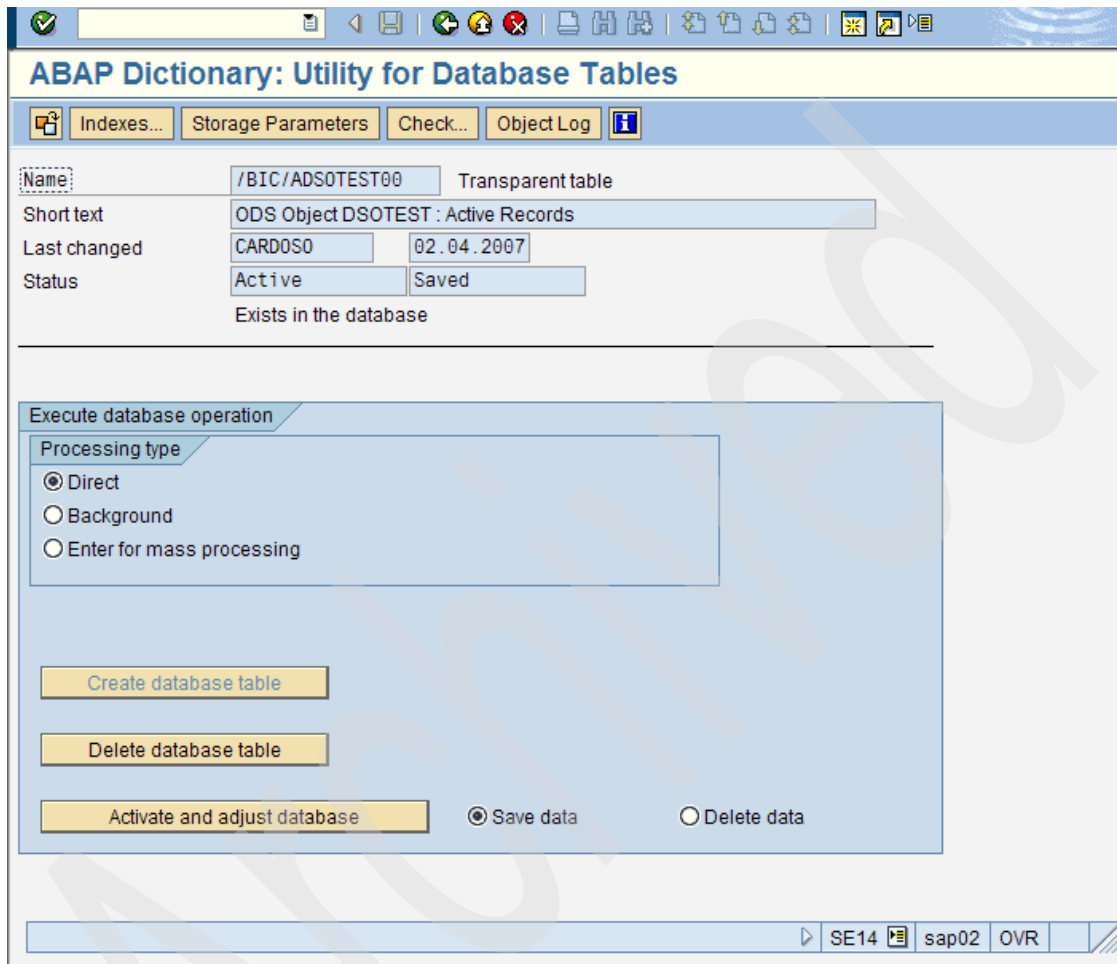


Figure 7-7 ABAP Dictionary: Utility for Database Tables

From the window DB2/390 Storage Attributes: Table, choose **Attributes** → **Display** → **Change**. Under Target Tablespace, choose **Partitioned**; see Figure 7-8.

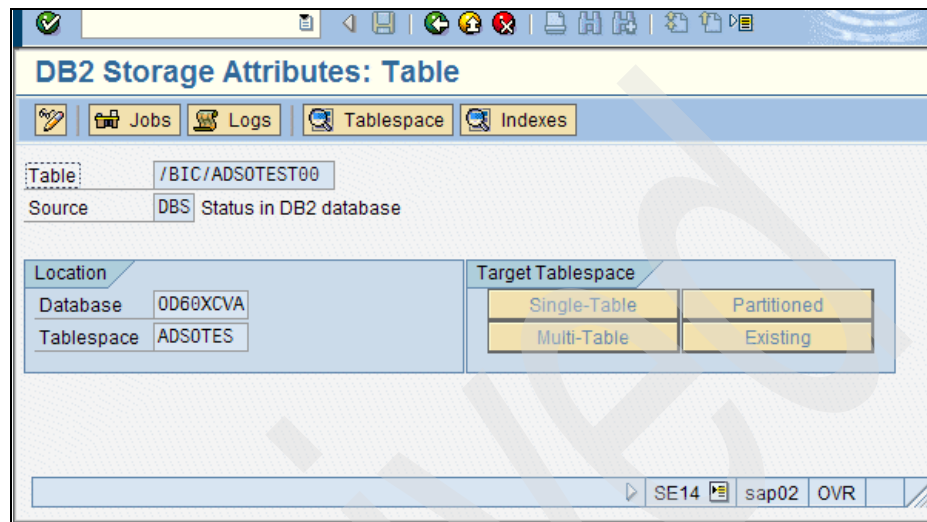


Figure 7-8 DB2/390 Storage Attributes: Table

At this point, you can choose whether to use LIMITKEY data for the partitioning. For this example, we chose not to use it since we are creating a table-controlled partitioned table. For more information about using LIMITKEY data, see “Using the LimitKey function” on page 122.

The next decision is to either choose table-controlled partitioning or an index for the partitioning. Choose table-controlled and then specify a partitioning key, the number of partitions, and for each partition, the limitkey value. Press Enter to get to the additional partition fields to fill in the limitkey. For this example, we arbitrarily chose six partitions and we chose limitkeys to create evenly sized partitions. Refer to Figure 7-9 on page 118.

Note: Keep in mind that this is just a test DataStore object, and does not represent real data. The goal of this section is to show the mechanics of partitioning a DataStore object. Other sections of this book discuss the considerations for designing a partitioning scheme.

When you are finished, choose **Attributes** → **Save**. SAP will use these saved storage parameters the next time it creates this table, which it will do in the next step.

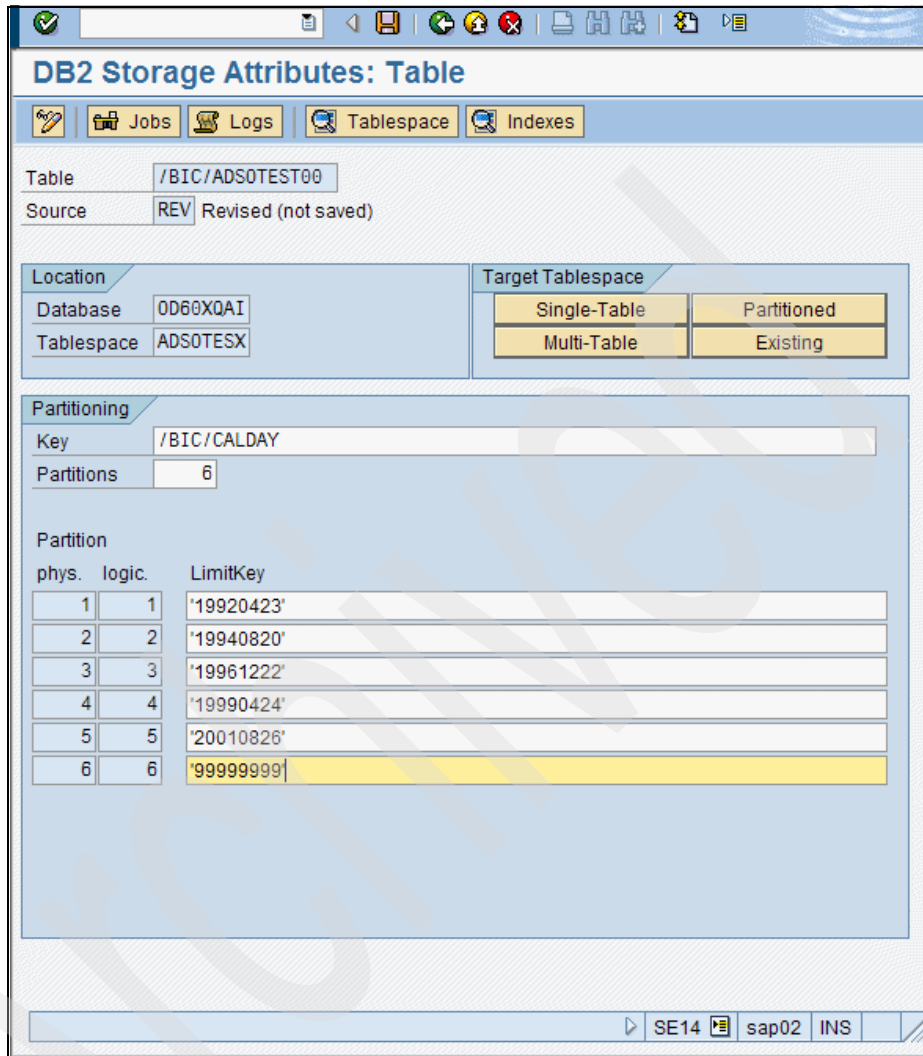


Figure 7-9 DB2/390 Storage Attributes: Table - 2

Now return to the window ABAP Dictionary: Utility for Database Tables and select options **Delete data** and **Activate and adjust database**. You can use the Delete data option here because you have already unloaded and saved the data in a sequential file using a DB2 utility. See Figure 7-10.

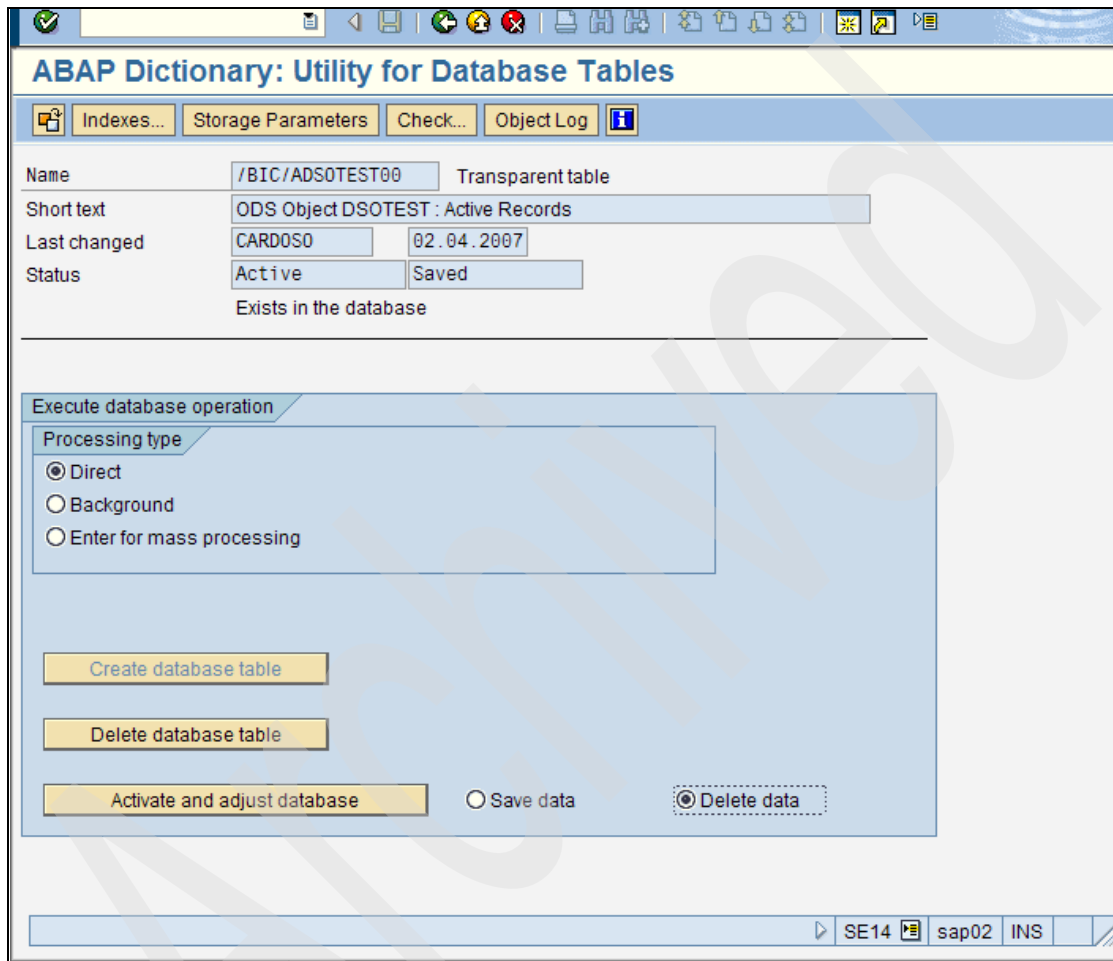


Figure 7-10 ABAP Dictionary: Utility for Database Tables - 2

During the activate and adjust database, SAP is dropping the table and recreating the tablespace, table, and indexes using the new storage parameters, which tell it to make it a partitioned table. Check the object log to see the DDL that was executed and the results.

The object log from this example is shown in “Example of an SAP object log for table-controlled partitioning” on page 125. It shows that table-controlled

partitioning was used. The CREATE TABLE statement has the PARTITION BY clause. The table /BIC/AODSTEST00 is partitioned by CALDAY and it has two indexes, /BIC/AODSTEST00~0 and /BIC/AODSTEST~01, neither of which are partitioned.

Now that the objects have been created in the database, use a DB2 utility to reload the data into the newly created partitioned table. In this example we used the DB2 LOAD utility. Use the LOG NO option for better performance.

Refer to Example 7-2 for sample JCL for the DB2 LOAD utility, and Example 7-3 for a sample control statement for the DB2 LOAD utility. For complete details on the DB2 utilities, refer to *DB2 V9 Utility Guide and Reference*.

Example 7-2 Example of JCL for the DB2 load utility

```
//LOAD JOB (999,POK), 'LOAD', CLASS=A, REGION=OM,
// MSGCLASS=T, MSGLEVEL=(1,1), NOTIFY=&SYSUID
/*JOBPARM S=SC04
// JCLLIB ORDER=(DB2TU.PROCLIB)
//*
//STEP1 EXEC DSNUPROC, UID=SMPLOAD, UTPROC=, SYSTEM=DB2T
//SYSERR DD DSN=LYDIA.LOAD2.STEP3.SYSERR,
// DISP=(NEW,CATLG,CATLG), UNIT=SYSDA,
// SPACE=(4096, (20,20) ,, ROUND)
//SYSDISC DD DSN=LYDIA.LOAD2.STEP3.SYSDISC,
// DISP=(NEW,CATLG,CATLG), UNIT=SYSDA,
// SPACE=(4096, (20,20) ,, ROUND)
//SYSMAP DD DSN=LYDIA.LOAD2.STEP3.SYSMAP,
// DISP=(NEW,CATLG,CATLG), UNIT=SYSDA,
// SPACE=(4096, (20,20) ,, ROUND)
//SYSUT1 DD DSN=LYDIA.LOAD2.STEP3.SYSUT1,
// DISP=(NEW,CATLG,CATLG), UNIT=SYSDA,
// SPACE=(4096, (20,20) ,, ROUND)
//UTPRINT DD SYSOUT=*
//SORTOUT DD DSN=LYDIA.LOAD2.STEP3.SORTOUT,
// DISP=(NEW,DELETE,CATLG), UNIT=SYSDA,
// SPACE=(4096, (20,20) ,, ROUND)
//SYSREC DD DSN=LYDIA.SMPLUNLD.SYSREC, DISP=SHR
//SYSIN DD *
```

Example 7-3 Example of a control statement for the DB2 LOAD utility

```
LOAD DATA INDDN(SYSREC)
  INTO TABLE "BI9DB"."/BIC/ADSOTEST00"
  ( "/BIC/VERSION"
    POSITION( 00003:00010) VARGRAPHIC
  , "/BIC/SOLD_TO"
    POSITION( 00011:00032) VARGRAPHIC
  , "/BIC/SALESORG"
```

```

        POSITION( 00033:00042) VARGRAPHIC
, "/BIC/VTYPE"
        POSITION( 00043:00050) VARGRAPHIC
, "/BIC/DISTR_CHAN"
        POSITION( 00051:00056) VARGRAPHIC
, "/BIC/DIVISION"
        POSITION( 00057:00062)
, "/BIC/MATERIAL"
        POSITION( 00063:00100) VARGRAPHIC
, "/BIC/CALDAY"
        POSITION( 00101:00118) VARGRAPHIC
, "/BIC/FISCVARNT"
        POSITION( 00119:00124) VARGRAPHIC
, "/BIC/STAT_CURR"
        POSITION( 00125:00136) VARGRAPHIC
, "/BIC/BASE_UOM"
        POSITION( 00137:00144) VARGRAPHIC
, "/BIC/INCORDVAL"
        POSITION( 00145:00153) DECIMAL
, "/BIC/INVCD_VAL"
        POSITION( 00154:00162) DECIMAL
, "/BIC/OPORDVALSC"
        POSITION( 00163:00171) DECIMAL
, "/BIC/RTNSVAL"
        POSITION( 00172:00180) DECIMAL
, "/BIC/CRMEM_VAL"
        POSITION( 00181:00189) DECIMAL
, "/BIC/INCORDQTY"
        POSITION( 00190:00198) DECIMAL
, "/BIC/INVCD_QTY"
        POSITION( 00199:00207) DECIMAL
, "/BIC/OPORDQTYBM"
        POSITION( 00208:00216) DECIMAL
, "/BIC/RTNSQTY"
        POSITION( 00217:00225) DECIMAL
, "/BIC/CRMEM_QTY"
        POSITION( 00226:00234) DECIMAL
, "/BIC/ORD_ITEMS"
        POSITION( 00235:00243) DECIMAL
, "/BIC/RTNS_ITEMS"
        POSITION( 00244:00252) DECIMAL
, "/BIC/INCORDCST"
        POSITION( 00253:00261) DECIMAL
, "/BIC/INVCD_CST"
        POSITION( 00262:00270) DECIMAL
, "/BIC/RTNSCST"
        POSITION( 00271:00279) DECIMAL
, "/BIC/CRMEM_CST"
        POSITION( 00280:00288) DECIMAL

```

```
, "/BIC/RECORDMODE"  
      POSITION( 00289:00292) VARGRAPHIC  
) ENFORCE NO
```

Now that the data has been reloaded back into the table, take a full image copy of the tablespace as a first backup. This will cancel the copy pending status of the tablespace.

Call transaction SE16 (Data Browser) to check the number of entries in the table. You should have the same number of entries as when you started the table conversion.

Using the LimitKey function

Note: You must have an index that matches your partitioning key in order to use the LimitKey function. So, even though you do not need an index that matches your partitioning key for table-controlled partitioning, you will need one if you want to use the LimitKey function.

To use the LimitKey function, call transaction SE14 (ABAP Dictionary: Database Utility). Specify the table name for which you want limit key data, click Edit (see Figure 7-6 on page 115), and follow these steps:

1. From the window ABAP Dictionary: Utility for Database Tables, choose **Storage Parameters**. See Figure 7-7 on page 116.
2. From the window DB2/390 Storage Attributes: Table window, choose **Goto** → **LIMITKEY Data**. See Figure 7-11. If limitkey data has been collected before, then it will be displayed.

To collect new limitkey data, specify the index on which you want to collect the limitkey data and the number of partitions that you want your table to have. Then choose **LIMITKEY Data** → **Collect**. A batch job is submitted. Monitor this job using the Jobs button or transaction SE37.

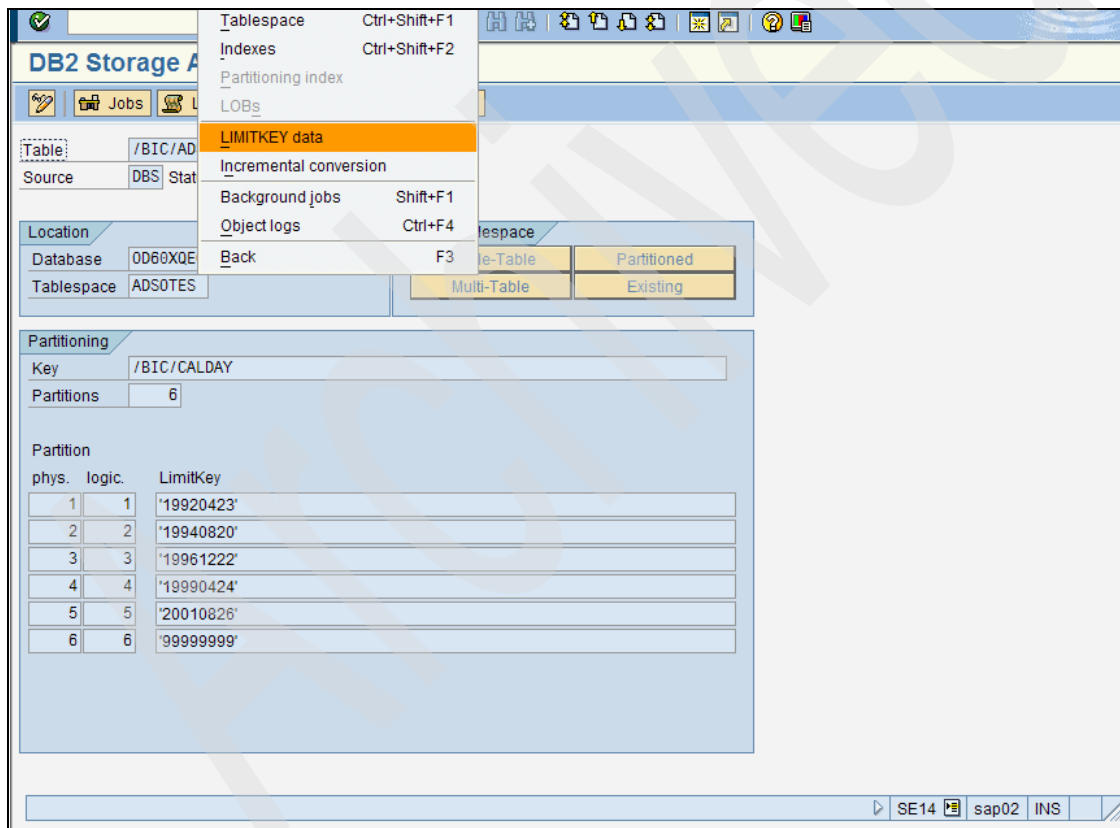


Figure 7-11 DB2/390 Storage Attributes: Table - LIMITKEY data

Upon completion of the job, you will see the LIMITKEY data displayed, as shown in Figure 7-12.

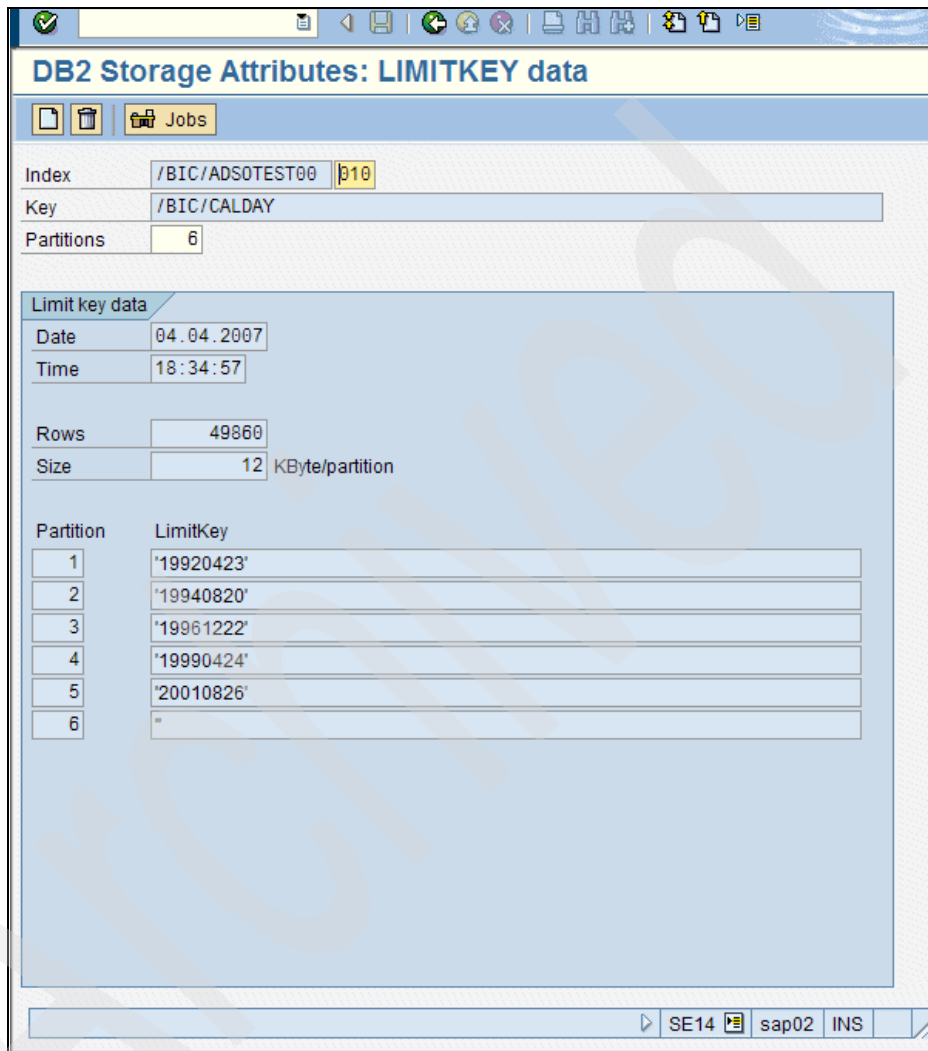


Figure 7-12 DB2/390 Storage Attributes: LIMITKEY data

Example of an SAP object log for table-controlled partitioning

Example 7-4 shows the SAP object log created when a table was converted from non-partitioned to partitioned using table-controlled partitioning.

Example 7-4 SAP object log from partitioning of a table using table-controlled partitioning

DB-TABL-BID-ADSOTEST00

```
-----  
Request: Delete and recreate Table /BIC/ADSOTEST00 (CARDOSO/03.04.07/01:26)  
Process: sap02_2  
Enhancement category for table missing  
Test activation of Table /BIC/ADSOTEST00 successful  
Activation and DDL statements for Table /BIC/ADSOTEST00 required  
-----
```

```
Activate table /BIC/ADSOTEST00 (CARDOSO/03.04.07/01:26)  
Enhancement category for table missing  
-----
```

```
Activate dependent table type /BIC/WADSOTEST00
```

```
Table type /BIC/WADSOTEST00 was activated
```

```
sql:
```

```
DROP TABLE "/BIC/ADSOTEST00" +LOCATION
```

```
DDL time(__1): .....52 milliseconds
```

```
sql:
```

```
CREATE TABLESPACE ADSOTESX IN OD20X9ZF
```

```
USING STOGROUP BI90DD PRIQTY 12
```

```
SECQTY 6827 FREEPAGE 0 PCTFREE 20
```

```
GBPCACHE CHANGED COMPRESS NO
```

```
LOCKPART YES DSSIZE 4 G Numparts 6 (
```

```
PART 1 USING STOGROUP BI90DD ,
```

```
PART 2 USING STOGROUP BI90DD ,
```

```
PART 3 USING STOGROUP BI90DD ,
```

```
PART 4 USING STOGROUP BI90DD ,
```

```
PART 5 USING STOGROUP BI90DD ,
```

```
PART 6 USING STOGROUP BI90DD )
```

```
BUFFERPOOL BP2 LOCKSIZE ROW
```

```
LOCKMAX 1000000 CCSID UNICODE
```

```
CREATE TABLE "/BIC/ADSOTEST00"
```

```
("BIC/VERSION" VARGRAPHIC (000003) NOT NULL
```

```
DEFAULT ' ' ,
```

```
"BIC/SOLD_TO" VARGRAPHIC (000010) NOT NULL
```

```
DEFAULT ' ' ,
```

```
"BIC/SALESORG" VARGRAPHIC (000004) NOT NULL
```

```
DEFAULT ' ' ,
```

```
"BIC/VTYPE" VARGRAPHIC (000003) NOT NULL
```

```
DEFAULT '000' ,
```

```
"BIC/DISTR_CHAN" VARGRAPHIC (000002) NOT NULL
```

```
DEFAULT ' ' ,
```

```

"/BIC/DIVISION" VARGRAPHIC (000002) NOT NULL
DEFAULT ' ' ,
"/BIC/MATERIAL" VARGRAPHIC (000018) NOT NULL
DEFAULT ' ' ,
"/BIC/CALDAY" VARGRAPHIC (000008) NOT NULL
DEFAULT '00000000' ,
"/BIC/FISCVARNT" VARGRAPHIC (000002) NOT NULL
DEFAULT ' ' ,
"/BIC/STAT_CURR" VARGRAPHIC (000005) NOT NULL
DEFAULT ' ' ,
"/BIC/BASE_UOM" VARGRAPHIC (000003) NOT NULL
DEFAULT ' ' ,
"/BIC/INCORVAL" DECIMAL (000017,000002) NOT NULL
DEFAULT 0 ,
"/BIC/INVCD_VAL" DECIMAL (000017,000002) NOT NULL
DEFAULT 0 ,
"/BIC/OPORDVALSC" DECIMAL (000017,000002) NOT NULL
DEFAULT 0 ,
"/BIC/RTNSVAL" DECIMAL (000017,000002) NOT NULL
DEFAULT 0 ,
"/BIC/CRMEM_VAL" DECIMAL (000017,000002) NOT NULL
DEFAULT 0 ,
"/BIC/INCORQTY" DECIMAL (000017,000003) NOT NULL
DEFAULT 0 ,
"/BIC/INVCD_QTY" DECIMAL (000017,000003) NOT NULL
DEFAULT 0 ,
"/BIC/OPORDQTYBM" DECIMAL (000017,000003) NOT NULL
DEFAULT 0 ,
"/BIC/RTNSQTY" DECIMAL (000017,000003) NOT NULL
DEFAULT 0 ,
"/BIC/CRMEM_QTY" DECIMAL (000017,000003) NOT NULL
DEFAULT 0 ,
"/BIC/ORD_ITEMS" DECIMAL (000017,000003) NOT NULL
DEFAULT 0 ,
"/BIC/RTNS_ITEMS" DECIMAL (000017,000003) NOT NULL
DEFAULT 0 ,
"/BIC/INCORDCST" DECIMAL (000017,000002) NOT NULL
DEFAULT 0 ,
"/BIC/INVCD_CST" DECIMAL (000017,000002) NOT NULL
DEFAULT 0 ,
"/BIC/RTNSCST" DECIMAL (000017,000002) NOT NULL
DEFAULT 0 ,
"/BIC/CRMEM_CST" DECIMAL (000017,000002) NOT NULL
DEFAULT 0 ,
"/BIC/RECORDMODE" VARGRAPHIC (000001) NOT NULL
DEFAULT ' ' )
PARTITION BY ( "/BIC/CALDAY" ) ( PART 1 VALUES (
'19920423' ) , PART 2 VALUES ( '19940820' )
, PART 3 VALUES ( '19961222' ) , PART 4 VALUES (

```



```

'19990424' ) , PART 5 VALUES ( '20010826' )
, PART 6 VALUES ( '99999999' ) )
IN OD60XZR2.ADSOTESX
DDL time(__1): .....435 milliseconds
sql:
CREATE TYPE 2 UNIQUE INDEX "/BIC/ADSOTEST00~0" ON "/BIC/ADSOTEST00"
("/BIC/VERSION" ,
"/BIC/SOLD_TO" ,
"/BIC/SALESORG" ,
"/BIC/VTYPE" ,
"/BIC/DISTR_CHAN" ,
"/BIC/DIVISION" ,
"/BIC/MATERIAL" ,
"/BIC/CALDAY" ,
"/BIC/FISCVARNT" ,
"/BIC/STAT_CURR" ,
"/BIC/BASE_UOM" )
USING STOGROUP BI90DI PRIQTY 16
SECQTY 40960 FREEPAGE 0 PCTFREE 10
GBPCACHE CHANGED
BUFFERPOOL BP3 COPY NO
PIECESIZE 2097152 K CLUSTER NOT PADDED
DDL time(__2): .....367 milliseconds
sql:
ALTER TABLE "/BIC/ADSOTEST00"
ADD PRIMARY KEY ("/BIC/VERSION",
"/BIC/SOLD_TO",
"/BIC/SALESORG",
"/BIC/VTYPE",
"/BIC/DISTR_CHAN",
"/BIC/DIVISION",
"/BIC/MATERIAL",
"/BIC/CALDAY",
"/BIC/FISCVARNT",
"/BIC/STAT_CURR",
"/BIC/BASE_UOM")
DDL time(__3): .....45 milliseconds
sql:
COMMIT WORK
DDL time(__4): .....3 milliseconds
sql:
DELETE FROM DDSTORAGE
WHERE DBSYSABBR = 'DB2'
AND TABNAME = '/BIC/ADSOTEST00'
AND INDEXNAME = '0'
DDL time(__5): .....4 milliseconds
sql:
COMMIT WORK
DDL time(__6): .....3 milliseconds

```

```

sql:
DELETE FROM DDSTORAGE
WHERE DBSYSABBR = 'DB2'
AND TABNAME = '/BIC/ADSOTEST00'
AND INDEXNAME = ''
DDL time(__7): .....3 milliseconds
sql:
CREATE TYPE 2 INDEX "/BIC/ADSOTEST00~01" ON "/BIC/ADSOTEST00"
("/BIC/CALDAY" )
USING STOGROUP BI90DI PRIQTY 16
SECQTY 40960 FREEPAGE 0 PCTFREE 10
GBPCACHE CHANGED
BUFFERPOOL BP3 COPY NO
PIECESIZE 2097152 K
DDL time(__8): .....263 milliseconds
sql:
COMMIT WORK
DDL time(__9): .....3 milliseconds
sql:
DELETE FROM DDSTORAGE
WHERE DBSYSABBR = 'DB2'
AND TABNAME = '/BIC/ADSOTEST00'
AND INDEXNAME = '010'
DDL time(__10): .....1 milliseconds
Request for /BIC/ADSOTEST00 executed successfully

```

Example of SAP object log for index-controlled partitioning

When converting a table from non-partitioned to partitioned, if you choose an index instead of table-controlled, then SAP creates DDL for index-controlled partitioning instead of table-controlled partitioning. See Example 7-5.

Example 7-5 SAP Object log from the partitioning of a table using an index

DB-TABL-BID-ADSOTEST00

```

-----
Request: Delete and recreate Table /BIC/ADSOTEST00 (CARDOSO/04.04.07/15:57)
Process: sap02_2
Enhancement category for table missing
Test activation of Table /BIC/ADSOTEST00 successful
Activation and DDL statements for Table /BIC/ADSOTEST00 required
-----
Activate table /BIC/ADSOTEST00 (CARDOSO/04.04.07/15:57)
Enhancement category for table missing
-----
Activate dependent table type /BIC/WADSOTEST00

```

```

Table type /BIC/WADSOTEST00 was activated
sql:
DROP TABLE "/BIC/ADSOTEST00" +LOCATION
DDL time(__1): .....52 milliseconds
sql:
CREATE TABLESPACE ADSOTESX IN OD20X9ZF
USING STOGROUP BI9ODD PRIQTY 12
SECQTY 6827 FREEPAGE 0 PCTFREE 20
GBPCACHE CHANGED COMPRESS NO
LOCKPART NO DSSIZE 4 G Numparts 6 (
PART 1 USING STOGROUP BI9ODD ,
PART 2 USING STOGROUP BI9ODD ,
PART 3 USING STOGROUP BI9ODD ,
PART 4 USING STOGROUP BI9ODD ,
PART 5 USING STOGROUP BI9ODD ,
PART 6 USING STOGROUP BI9ODD )
BUFFERPOOL BP2 LOCKSIZE ROW
LOCKMAX 1000000 CCSID UNICODE
CREATE TABLE "/BIC/ADSOTEST00"
("/BIC/VERSION" VARGRAPHIC (000003) NOT NULL
DEFAULT ' ' ,
"/BIC/SOLD_TO" VARGRAPHIC (000010) NOT NULL
DEFAULT ' ' ,
"/BIC/SALESORG" VARGRAPHIC (000004) NOT NULL
DEFAULT ' ' ,
"/BIC/VTYPE" VARGRAPHIC (000003) NOT NULL
DEFAULT '000' ,
"/BIC/DISTR_CHAN" VARGRAPHIC (000002) NOT NULL
DEFAULT ' ' ,
"/BIC/DIVISION" VARGRAPHIC (000002) NOT NULL
DEFAULT ' ' ,
"/BIC/MATERIAL" VARGRAPHIC (000018) NOT NULL
DEFAULT ' ' ,
"/BIC/CALDAY" VARGRAPHIC (000008) NOT NULL
DEFAULT '00000000' ,
"/BIC/FISCVARNT" VARGRAPHIC (000002) NOT NULL
DEFAULT ' ' ,
"/BIC/STAT_CURR" VARGRAPHIC (000005) NOT NULL
DEFAULT ' ' ,
"/BIC/BASE_UOM" VARGRAPHIC (000003) NOT NULL
DEFAULT ' ' ,
"/BIC/INCORVDVAL" DECIMAL (000017,000002) NOT NULL
DEFAULT 0 ,
"/BIC/INVCD_VAL" DECIMAL (000017,000002) NOT NULL
DEFAULT 0 ,
"/BIC/OPORDVALSC" DECIMAL (000017,000002) NOT NULL
DEFAULT 0 ,
"/BIC/RTNSVAL" DECIMAL (000017,000002) NOT NULL
DEFAULT 0 ,

```

```

"/BIC/CRMEM_VAL" DECIMAL (000017,000002) NOT NULL
DEFAULT 0 ,
"/BIC/INCORDQTY" DECIMAL (000017,000003) NOT NULL
DEFAULT 0 ,
"/BIC/INVCD_QTY" DECIMAL (000017,000003) NOT NULL
DEFAULT 0 ,
"/BIC/OPORDQTYBM" DECIMAL (000017,000003) NOT NULL
DEFAULT 0 ,
"/BIC/RTNSQTY" DECIMAL (000017,000003) NOT NULL
DEFAULT 0 ,
"/BIC/CRMEM_QTY" DECIMAL (000017,000003) NOT NULL
DEFAULT 0 ,
"/BIC/ORD_ITEMS" DECIMAL (000017,000003) NOT NULL
DEFAULT 0 ,
"/BIC/RTNS_ITEMS" DECIMAL (000017,000003) NOT NULL
DEFAULT 0 ,
"/BIC/INCORDCST" DECIMAL (000017,000002) NOT NULL
DEFAULT 0 ,
"/BIC/INVCD_CST" DECIMAL (000017,000002) NOT NULL
DEFAULT 0 ,
"/BIC/RTNSCST" DECIMAL (000017,000002) NOT NULL
DEFAULT 0 ,
"/BIC/CRMEM_CST" DECIMAL (000017,000002) NOT NULL
DEFAULT 0 ,
"/BIC/RECORDMODE" VARGRAPHIC (000001) NOT NULL
DEFAULT ' ' )
IN OD20X9ZF.ADSOTESX
DDL time(__1): .....435 milliseconds
sql:
CREATE TYPE 2 UNIQUE INDEX "/BIC/ADSOTEST00~01" ON "/BIC/ADSOTEST00"
("/BIC/CALDAY" )
USING STOGROUP BI9ODI PRIQTY 16
SECQTY 40960 FREEPAGE 0 PCTFREE 10
GBPCACHE CHANGED CLUSTER (
PART 1 VALUES( '19920423' )
USING STOGROUP BI9ODI ,
PART 2 VALUES( '19940820' )
USING STOGROUP BI9ODI ,
PART 3 VALUES( '19961222' )
USING STOGROUP BI9ODI ,
PART 4 VALUES( '19990424' )
USING STOGROUP BI9ODI ,
PART 5 VALUES( '20010826' )
USING STOGROUP BI9ODI ,
PART 6 VALUES( '99999999' )
USING STOGROUP BI9ODI )
BUFFERPOOL BP3 COPY NO
DDL time(__2): .....231 milliseconds
sql:

```

```

COMMIT WORK
DDL time(__3): .....4 milliseconds
sql:
DELETE FROM DDSTORAGE
WHERE DBSYSABBR = 'DB2'
AND TABNAME = '/BIC/ADSOTEST00'
AND INDEXNAME = '010'
DDL time(__4): .....1 milliseconds
sql:
CREATE TYPE 2 UNIQUE INDEX "/BIC/ADSOTEST00~0" ON "/BIC/ADSOTEST00"
("/BIC/VERSION",
"/BIC/SOLD_TO",
"/BIC/SALESORG",
"/BIC/VTYPE",
"/BIC/DISTR_CHAN",
"/BIC/DIVISION",
"/BIC/MATERIAL",
"/BIC/CALDAY",
"/BIC/FISCVARNT",
"/BIC/STAT_CURR",
"/BIC/BASE_UOM")
USING STOGROUP BI90DI PRIQTY 16
SECQTY 40960 FREEPAGE 0 PCTFREE 10
GBPCACHE CHANGED
BUFFERPOOL BP3 COPY NO
PIECESIZE 2097152 K
DDL time(__5): .....134 milliseconds
sql:
ALTER TABLE "/BIC/ADSOTEST00"
ADD PRIMARY KEY ("/BIC/VERSION",
"/BIC/SOLD_TO",
"/BIC/SALESORG",
"/BIC/VTYPE",
"/BIC/DISTR_CHAN",
"/BIC/DIVISION",
"/BIC/MATERIAL",
"/BIC/CALDAY",
"/BIC/FISCVARNT",
"/BIC/STAT_CURR",
"/BIC/BASE_UOM")
DDL time(__6): .....45 milliseconds
sql:
COMMIT WORK
DDL time(__7): .....3 milliseconds
sql:
DELETE FROM DDSTORAGE
WHERE DBSYSABBR = 'DB2'
AND TABNAME = '/BIC/ADSOTEST00'
AND INDEXNAME = '0'

```

```

DDL time(__8): .....4 milliseconds
sql:
COMMIT WORK
DDL time(__9): .....4 milliseconds
sql:
DELETE FROM DDSTORAGE
WHERE DBSYSABBR = 'DB2'
AND TABNAME = '/BIC/ADSOTEST00'
AND INDEXNAME = ''
DDL time(__10): .....3 milliseconds
sql:
COMMIT WORK
DDL time(__11): .....2 milliseconds
Request for /BIC/ADSOTEST00 executed successfully
-----

```

The CREATE INDEX statement has the VALUES clause to determine the partitions. As a result of executing this DDL:

- ▶ Table /BIC/ADSOTEST00 is a partitioned table.
- ▶ Index /BIC/ADSOTEST00~01 is a partitioned index — and it is also a partitioning index.
- ▶ Index /BIC/ADSOTEST00~0 is a non-partitioned primary index.

We do not recommend using index-controlled partitioning. Table-controlled partitioning is the recommended method, because it provides more support and flexibility for partitioning tables and indexes.

7.5 DB2 compression

Since DataStore objects tend to be large (especially the active DataStore object data, the /BIC/A<DataStore_object_name>00 table), we recommend using DB2 compression for these objects. There are several benefits to using DB2 compression, especially in the area of data management.

However, DB2 compression can also improve query performance on queries that retrieve a large number of rows, like typical queries on DataStore objects. When the data is compressed, more rows fit on a page. Therefore, more rows can fit into the bufferpools providing a better buffer hit ratio and more rows can be retrieved with less I/O.

7.5.1 Materialized query tables

Material query tables (MQTs) can be used to improve DataStore objects query performance. MQTs offer a way to hold the results from previous queries so the results do not need to be recomputed for future queries. The query results are materialized and saved in MQTs.

SAP provides this function for InfoCubes using aggregates. SAP does not support aggregates for DataStore objects, so the function of MQTs can be used to do this for DSO objects.

Archived

Archived



Query and Load performance

In this chapter we describe how to analyze query and load performance in your SAP BI DB2 system on a z/OS platform. We provide a road map to take the reader through some general system-wide health checks. Additionally, we provide an approach to analyzing load performance.

8.1 Troubleshooting general system performance

In this section we provide a road map to take the reader through some general system-wide health checks. These ensure that there is no general problem that may compromise DB2 performance within SAP BI. It is worth taking the time to perform this task before doing more specific in-depth analysis.

It is beyond the scope of this book to provide a comprehensive guide to troubleshooting general SAP and z/OS DB2 system performance problems. However, we provide a process flow to perform a basic system health check.

We primarily focus on methodology and refer you, when necessary, to publications where you can find a more detailed discussion about these subjects. For example, if you are running on a DB2 data sharing environment, refer to *SAP on DB2 for z/OS and OS/390: High Availability and Performance Monitoring with Data Sharing*, SG24-6950.

When users complain about performance, you need to follow a road map (such as Figure 8-1) or a procedure to capture the relevant information necessary to begin your analysis and follow it to a conclusion.

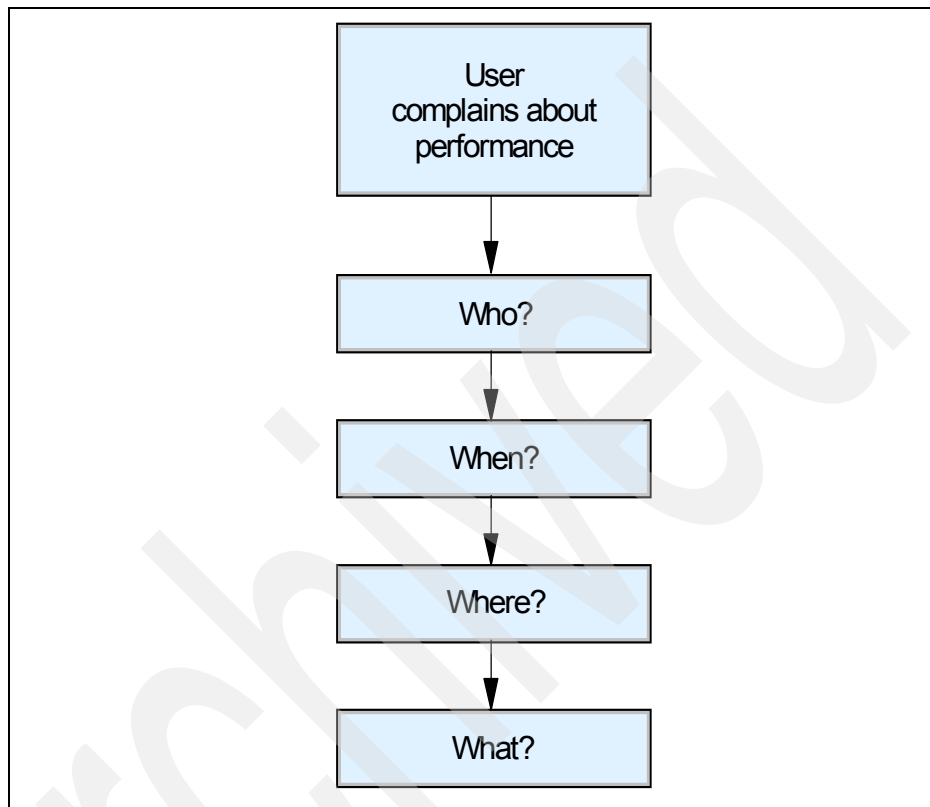


Figure 8-1 Performance strategy road map

You need to ascertain:

- ▶ Which users are complaining? (You need to know their user IDs.)
- ▶ On what date and time did the users have the problem?
- ▶ Which server was the user was logged onto or which server was the batch/process chain executing on?
- ▶ Most importantly, which query or load process is causing problems?

After establishing the E-Facts, you can begin your analysis. If many users are complaining of poor performance or system hangs, or all the load processes are overrunning, for example, then this may be a system-wide problem. If all users on one particular server are complaining of poor performance, then the problem may be only with that particular server.

Figure 8-2 illustrates a road map for analysis of server-wide and system-wide performance problems.

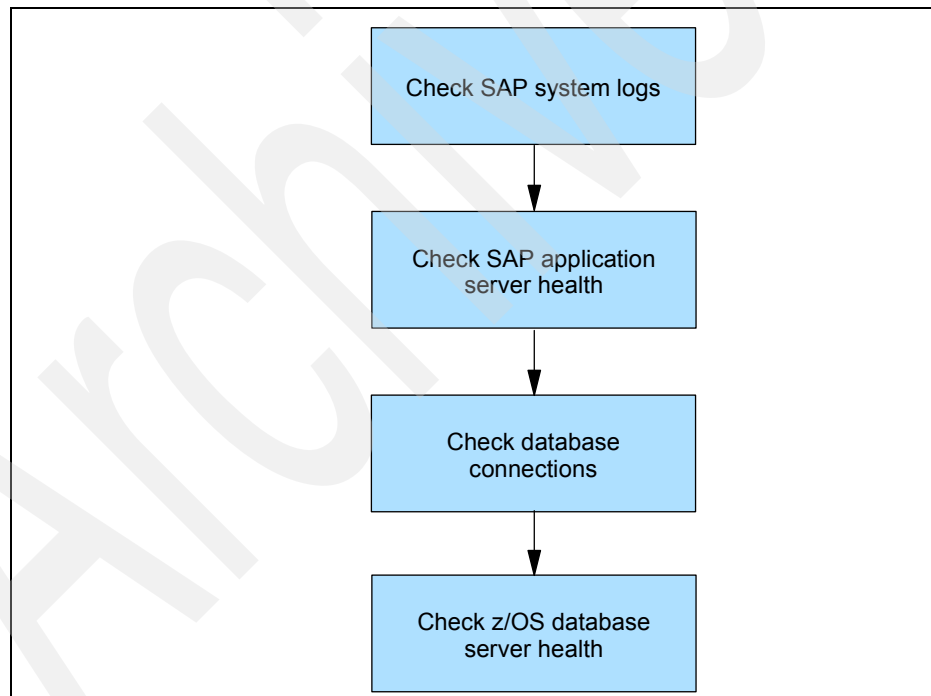


Figure 8-2 Sample road map for analysis of performance problems

We will look at each step in a little more detail:

1. Check SAP system logs.

Use transaction SM21 to check your SAP system logs for any error situations that may be causing performance problems. Use the menu path **Systemlog** → **Choose** → **All remote system locks**.

2. Check application server health.

If there are no obvious errors on the SAP system logs, begin checking the health of the application servers:

- a. Use transaction ST06 to check the CPU consumption. From the menu path **Goto** → **Current data** → **Snapshot** → **Top CPU processes**, you can determine whether a user transaction or report is looping and using all of the available CPU.
- b. Check the health of the SAP buffers using transaction ST02.

3. Check the database connection.

If there are no problems with the servers in question, confirm that the database connection from the server to the DB2 data sharing member is performing well:

- a. Check the DRDA® network connection by using SAP transaction DB2, selecting the **DB2 Performance Tuning** tab, then selecting **DB2 ping** to test the network response time.

This should show a response time of 1–2 milliseconds to send 500 bytes of data. If this response time is longer, then there is a problem with the link.

You can investigate this further by clicking **DB2 connect diagnostics** on the DB2 Performance Tuning tab.

- b. Continue the investigation by asking the network support group to investigate the problem.

4. Check the z/OS database server health.

Performance analysis on OS/390® z/OS is beyond the scope of this book, but there are some basic checks that you can perform:

- Check the OS/390 z/OS system logs for any error situations that may cause problems on the LPAR.
- Check the DB2 MSTR job log for any error situations in DB2 that may be causing performance problems.
- Check the LPAR CPU utilization and look for looping address spaces consuming CPU.
- Use RMF™ Monitor III to check your workloads for any delay.

Further information about the subject of general SAP performance analysis for the z/OS DB2 platform can be found in the following IBM whitepaper:

<http://www-1.ibm.com/support/techdocs/atmastr.nsf/WebIndex/WP100287>

If you are running on a DB2 data sharing environment, then refer to *SAP on DB2 for z/OS and OS/390: High Availability and Performance Monitoring with Data Sharing*, SG24-6950.

Figure 8-3 presents a road map for analyzing performance issues. Use it to determine what tools may be of use to you in determining the performance problem bottleneck.

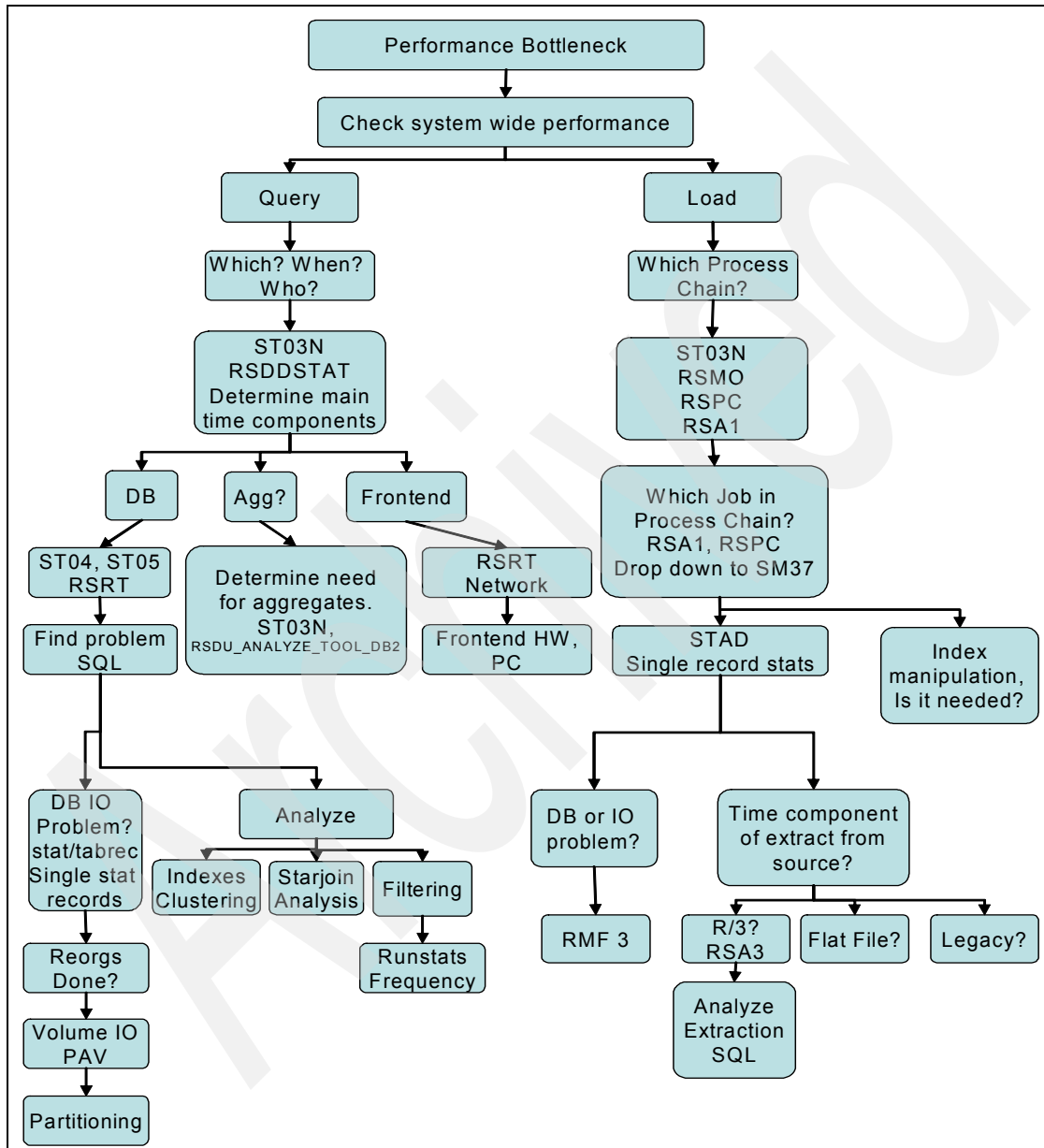


Figure 8-3 Analyzing performance - road map

8.2 Analyzing load performance

In this section we discuss an approach for analyzing load performance. Figure 8-4 shows the load portion of the performance analysis road map.

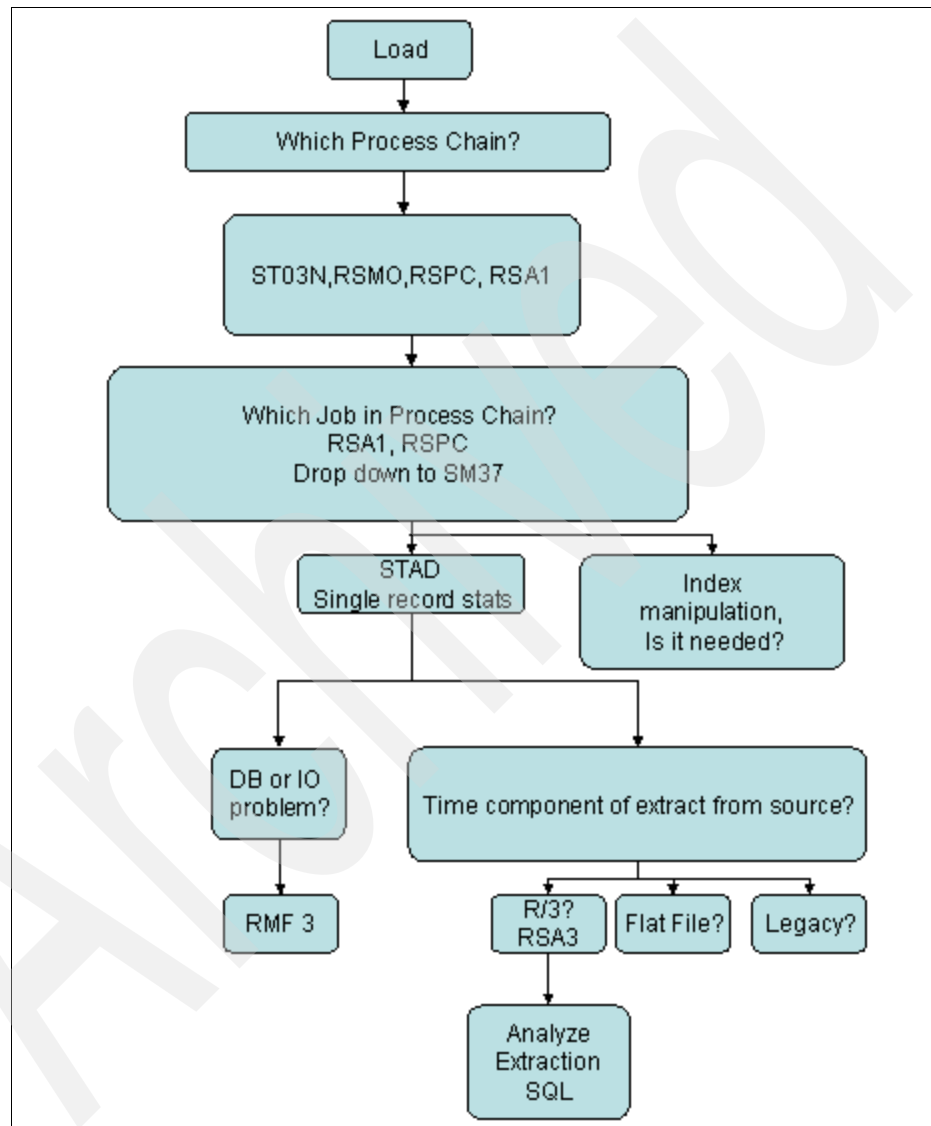


Figure 8-4 Analyzing performance road map for load

We assume a scenario where there is no system-wide performance problem, but only an individual load of data with unacceptably slow performance. The root

cause could be any phase of the load, any part of the process chain. It could be during extraction from the source (R/3 system, earlier system, flat file system, PSA, ODS), transfer rules, update rules, or target (PSA, ODS, or InfoCube).

There may be several methodologies to troubleshoot the cause of the performance problem. The choice can depend on environmental restrictions, such as a business reason for not allowing tracing in a production environment. Our methodology begins with determining which process chain and which component of the process chain takes the longest portion of elapsed time. Then we try to determine which general area is the cause and what can be done to alleviate it.

There are several SAP transactions that we can use for debugging the problem. We have captured sample windows from these transactions to give you some ideas.

First, we check system workload statistics. The statistics provide, for example, total, average, and percentage of elapsed times for each part of the load. You have information about which request ID and which component (source, target, transfer rules, or update rules) takes the most elapsed times. Note that source could be within the system, for example, PSA or ODS.

We use transaction **ST03** → **Expert Mode** → **BW System Load** → **Select time window** → **Select Load Data under Analysis View pane** → **Select Aggregation** → **Select Request**. Figure 8-5 shows a sample window.

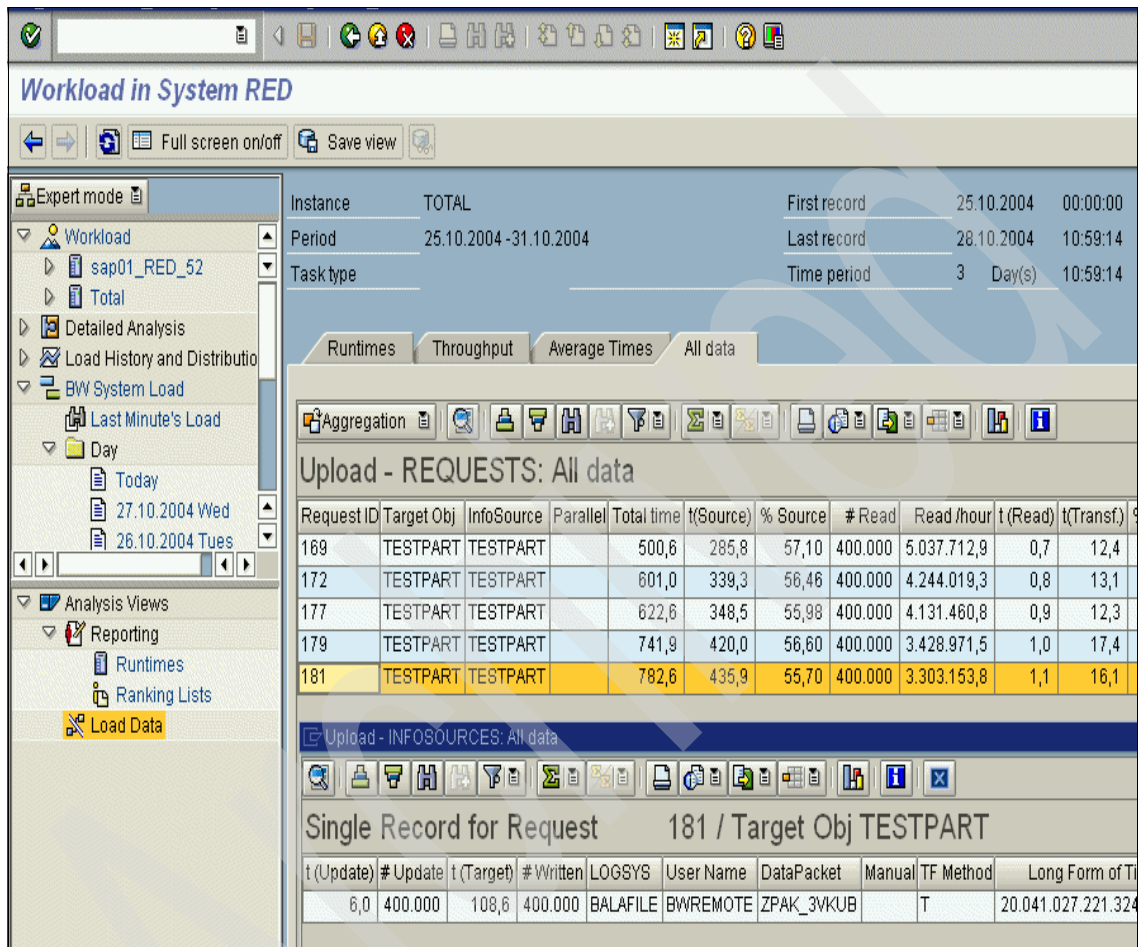


Figure 8-5 BW load data statistics from ST03

Note that in this sample window, the target and source have the same name for different objects. Normally, they should have different names.

Additionally, we can get more information by using SAP transaction RSMO, the Administrator Workbench Monitor. Figure 8-6 displays a sampled load. Here you have access to header information for a load job.

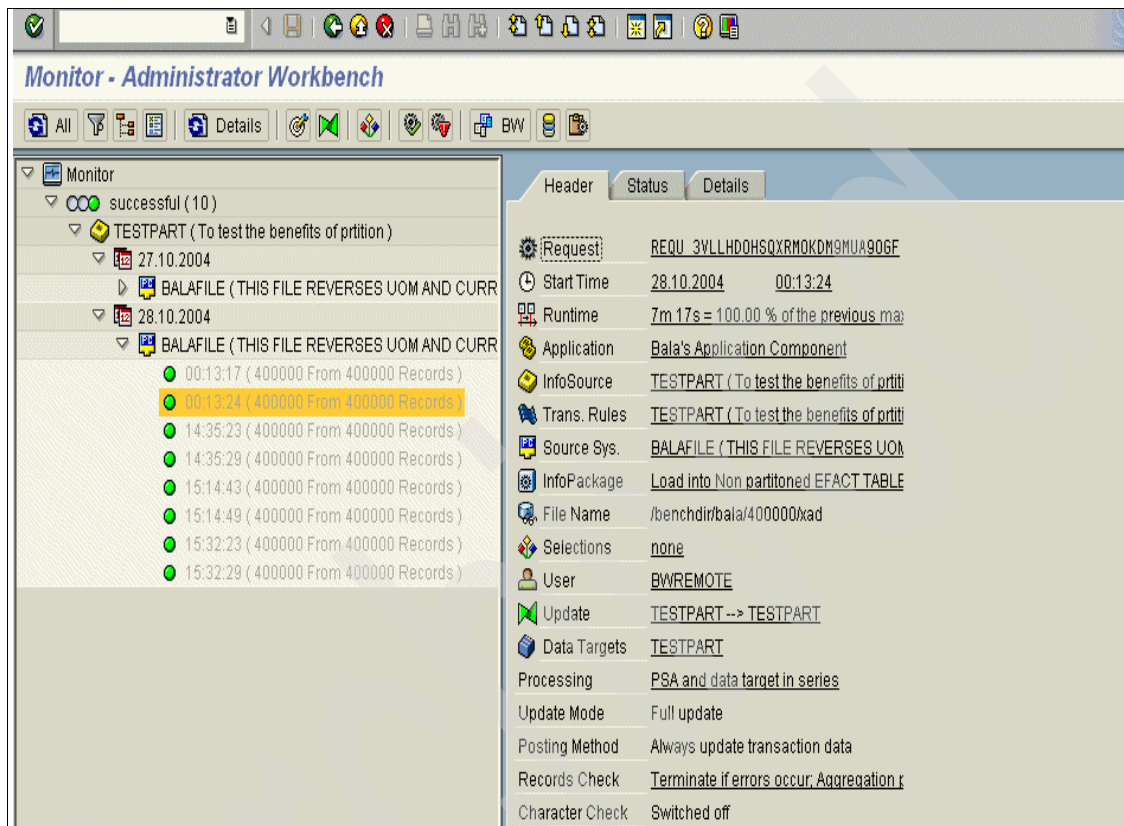


Figure 8-6 Header information for a load job by RSMO

Click the **Details** tab of the RSMO window to display more information for each component of the load. See Figure 8-7.

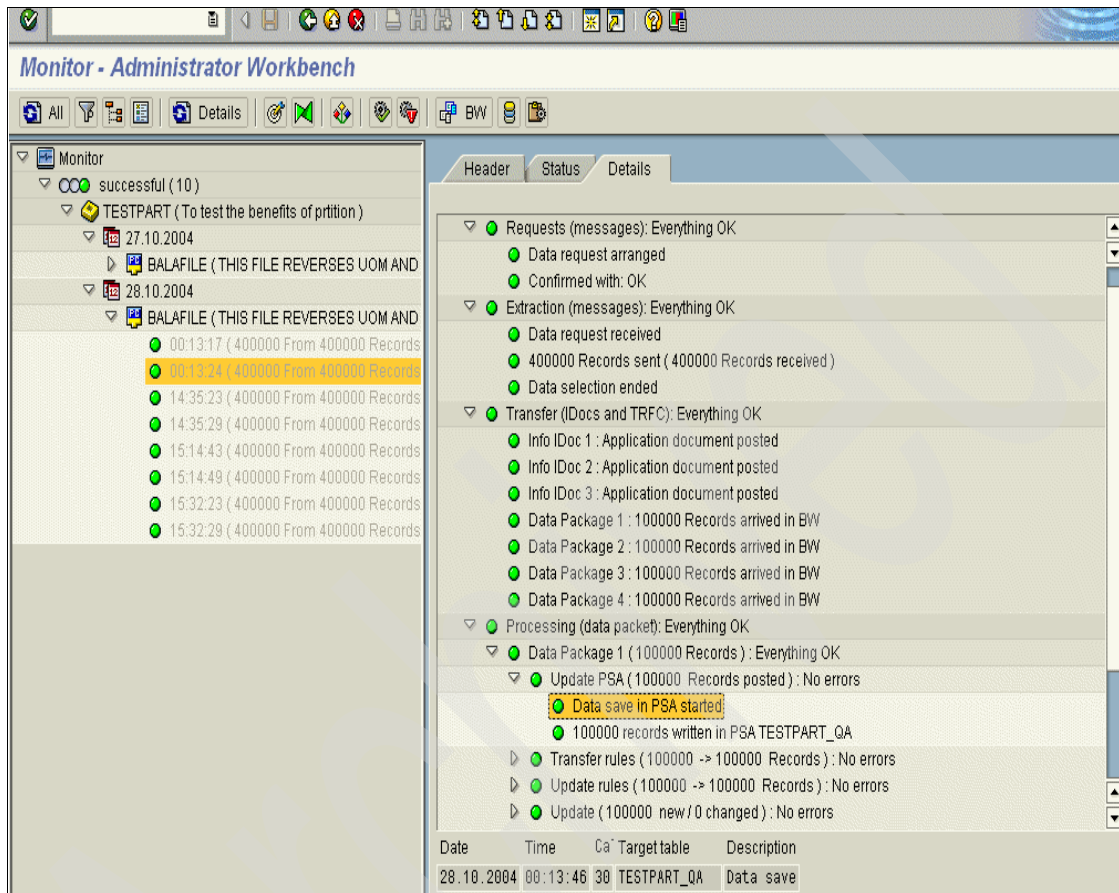


Figure 8-7 Details of load job by RSMO

The load is broken into four smaller data packets for parallel processing. You can see detailed information for request, extraction, transfer, and processing (transfer rules, update rules, upload) components. Observe the times to determine which component takes most of the elapsed time.

Next, if there is defined process chain, we can display the graphical view of the process chain for load by invoking Process Chain Maintenance Menu (SAP transaction RSPC).

Here you can see whether there is index manipulation, and collapse (compression), for example, see Figure 8-8.

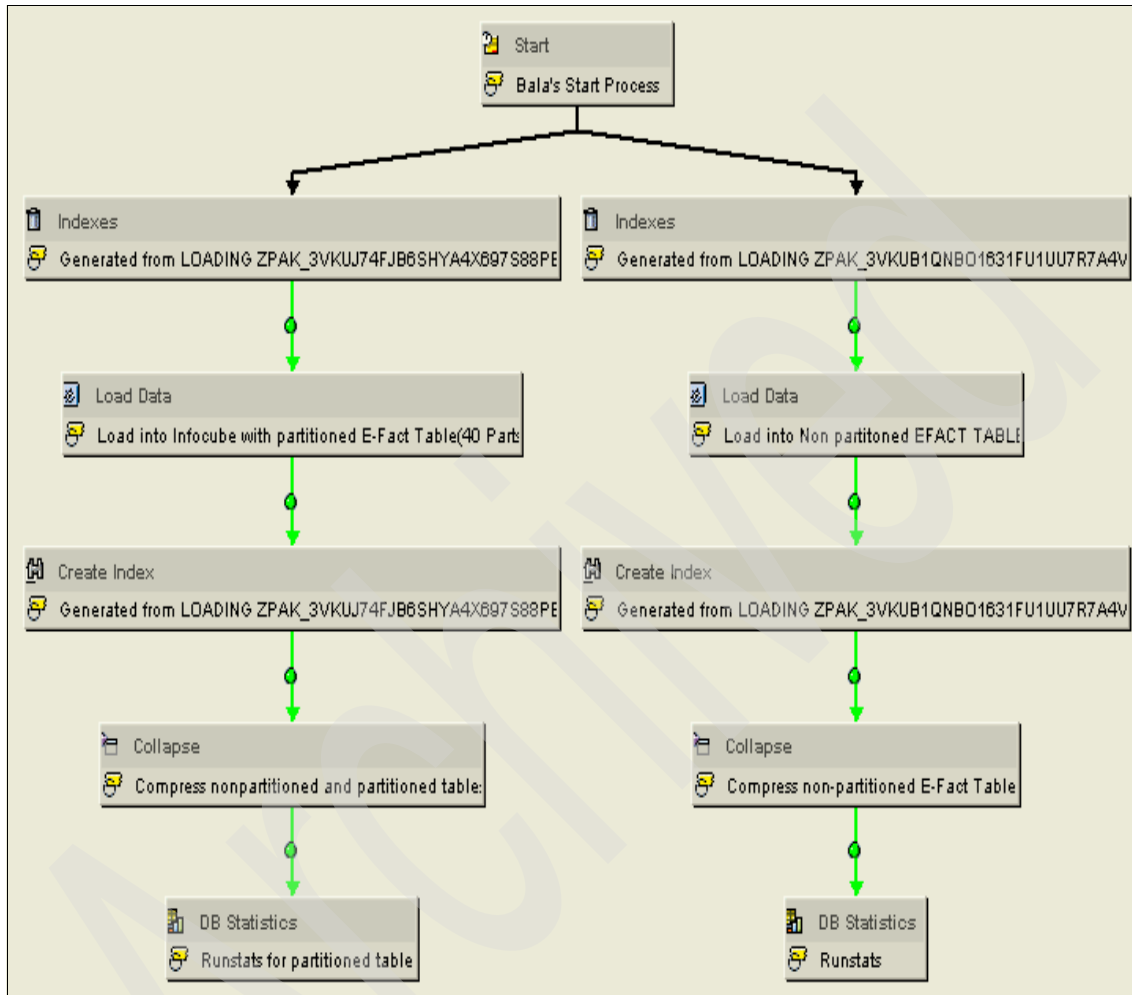


Figure 8-8 Process chain flow diagram of load job from RSPC

Select the component of interest to drop to the SM37 job log to check for more details of its activities. This can be done by right-clicking the Load Data icon to display all jobs. Figure 8-9 shows dropping the index for a certain table.

Job Log Entries for BI_PROCESS_DROPINDEX / 17530402					
Long text Previous page Next page					
Job log overview for job: BI_PROCESS_DROPINDEX / 17530402					
Date	Time	Message text	Message class	Message no.	Me
27.10.2004	17:55:10	Job started	00	516	
27.10.2004	17:55:10	Step 001 started (program RSPROCESS, variant &0000000000170, user ID BWREMOTE)	00	550	
27.10.2004	17:55:29	SQL: 27.10.2004 17:55:29 BWREMOTE	DBMAN	99	
27.10.2004	17:55:29	DROP INDEX "/BIC/FETBLPART~020"	DBMAN	99	
27.10.2004	17:55:29	SQL-END: 27.10.2004 17:55:29 00:00:00	DBMAN	99	
27.10.2004	17:55:29	SQL: 27.10.2004 17:55:29 BWREMOTE	DBMAN	99	
27.10.2004	17:55:29	DROP INDEX "/BIC/FETBLPART~030"	DBMAN	99	
27.10.2004	17:55:30	SQL-END: 27.10.2004 17:55:30 00:00:01	DBMAN	99	
27.10.2004	17:55:30	SQL: 27.10.2004 17:55:30 BWREMOTE	DBMAN	99	
27.10.2004	17:55:30	DROP INDEX "/BIC/FETBLPART~040"	DBMAN	99	
27.10.2004	17:55:30	SQL-END: 27.10.2004 17:55:30 00:00:00	DBMAN	99	
27.10.2004	17:55:30	SQL: 27.10.2004 17:55:30 BWREMOTE	DBMAN	99	
27.10.2004	17:55:30	DROP INDEX "/BIC/FETBLPART~050"	DBMAN	99	
27.10.2004	17:55:31	SQL-END: 27.10.2004 17:55:31 00:00:01	DBMAN	99	
27.10.2004	17:55:31	SQL: 27.10.2004 17:55:31 BWREMOTE	DBMAN	99	
27.10.2004	17:55:31	DROP INDEX "/BIC/FETBLPART~060"	DBMAN	99	
27.10.2004	17:55:31	SQL-END: 27.10.2004 17:55:31 00:00:00	DBMAN	99	
27.10.2004	17:55:31	Indexes for InfoCube ETBLPART have been deleted successfully	RSMP	87	
27.10.2004	17:55:32	Event RSPROCESS with parameter 3VLLI4Z0WUE4N6J1Y0TVM4XSF successfully triggered	RSMP	90	
27.10.2004	17:55:38	Job finished	00	517	

Figure 8-9 Detailed activities of job log from SM37

Furthermore, you can get more statistical information about your load job by invoking the SAP transaction STAD. Here you have an option to filter statistics from your load job according to user, time window, and certain performance metric values, as shown in Figure 8-10 and Figure 8-11 on page 150.

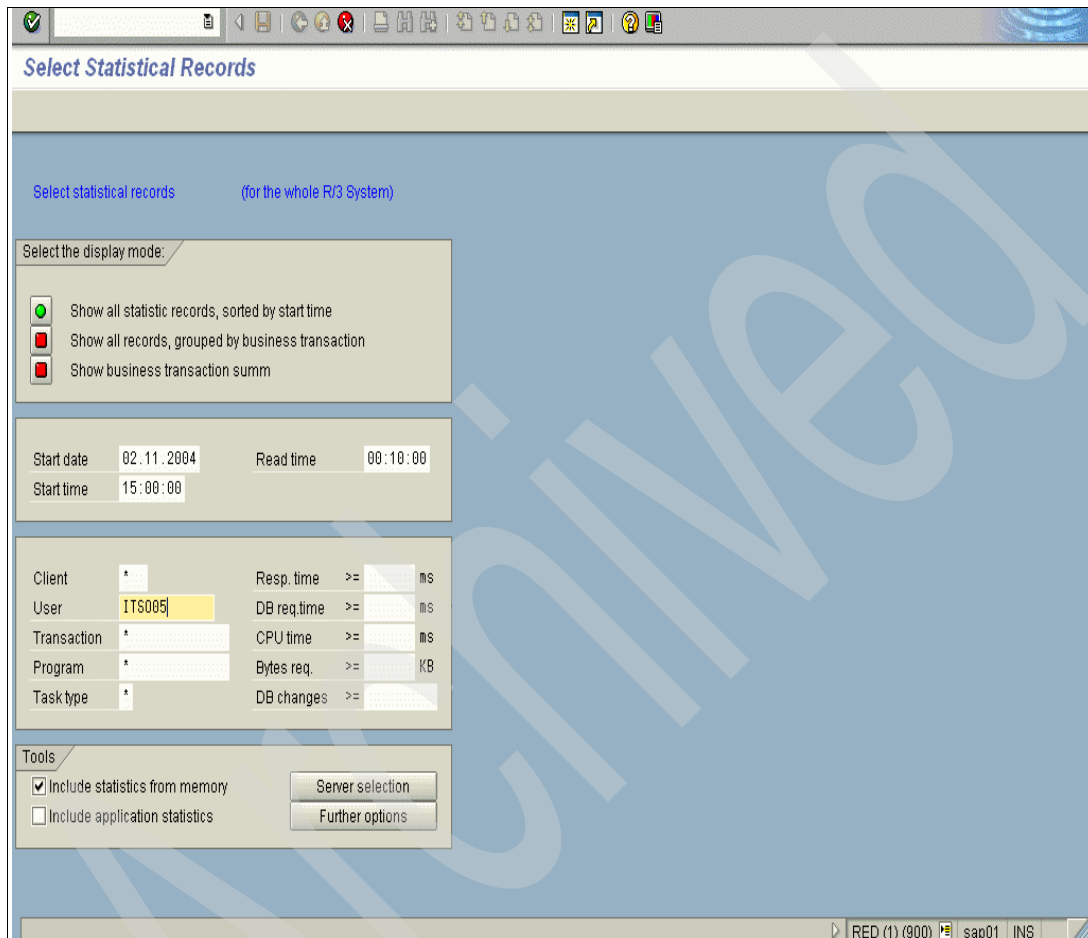


Figure 8-10 STAD window

Workload: Statistical Records

Download [Icons] Disp. mode [Icons] Sel. fields [Icons] Server ID [Icons]

System: RED Number of RFCs which responded (without errors): 1 (1)
 Analysed time: 02.11.2004 / 15:00:00 - 02.11.2004 / 15:40:00 Time frame: +/- 00:08:00
 Display mode: All statistic records, sorted by time

Started	Server	Transaction	Program	T Scr. Wp	User	Response time (ms)	Time in WPs (ms)	Wait time (ms)
	*		*	*	ITS005	0		
15:28:13	sap01_RED_52	RSA1	RSABW_START_NEW	D 1010 1	ITS005	9	9	
15:28:14	sap01_RED_52	RSA1	RSABW_START_NEW	D 1010 0	ITS005	5	5	
15:28:15	sap01_RED_52	RSA1	RSABW_START_NEW	D 1010 0	ITS005	4	4	
15:28:17	sap01_RED_52	RSA1	RSABW_START_NEW	D 1010 0	ITS005	5	5	
15:28:17	sap01_RED_52	RSA1	RSABW_START_NEW	D 2100 0	ITS005	36	36	
15:28:25	sap01_RED_52	RSA1	RSABW_START_NEW	D 2100 0	ITS005	16	16	
15:28:26	sap01_RED_52	RSA1	RSABW_START_NEW	D 0101 0	ITS005	283	282	
15:28:30	sap01_RED_52	RSA1	RSABW_START_NEW	D 2100 0	ITS005	24	24	
15:28:35	sap01_RED_52	RSA1	RSABW_START_NEW	D 2100 0	ITS005	7	7	
15:28:37	sap01_RED_52	RSA1	RSABW_START_NEW	D 2100 0	ITS005	6.307	6.288	
15:28:38	sap01_RED_52		ALE	L 3004 1	ITS005	133	132	
15:28:44	sap01_RED_52		RFC	R 3004 0	ITS005	2.547	2.546	
15:28:44	sap01_RED_52		ALE	L 3004 1	ITS005	178	177	
15:28:44	sap01_RED_52		ALE	L 3004 2	ITS005	332	330	
15:28:46	sap01_RED_52		RSDELREQ1	B 32	ITS005	2.141	2.141	
15:28:46	sap01_RED_52		RSBTCRTE	B 32	ITS005	101	101	
15:28:48	sap01_RED_52		RFC	R 3004 0	ITS005	2.037	2.036	
15:28:54	sap01_RED_52	RSM0	RSM01_RSM2	D 0100 0	ITS005	2.314	2.210	
15:28:58	sap01_RED_52	RSM0	RSM01_RSM2	D 0100 0	ITS005	102	61	
15:29:03	sap01_RED_52	RSM0	RSM01_RSM2	D 0100 0	ITS005	16	9	
15:29:04	sap01_RED_52	RSM0	RSM01_RSM2	D 0100 0	ITS005	182	99	
15:29:08	sap01_RED_52	RSM0	RSM01_RSM2	D 0100 0	ITS005	1.213	1.193	
15:29:14	sap01_RED_52	RSM0	RSM01_RSM2	D 0601 0	ITS005	17	17	
15:29:23	sap01_RED_52	RSM0	RSM01_RSM2	D 0100 0	ITS005	109	87	
15:29:28	sap01_RED_52	RSM0	RSM01_RSM2	D 0100 0	ITS005	387	387	

Figure 8-11 All statistical records from STAD

Double-click to go to the Single Record Statistic window to determine which component takes most of the response time, for example, database request time to total response time. Also, check whether there is a high insert number. If so, then determine which table, tablespace, and volser apply. Then you can check RMF reports to find if there is an I/O bottleneck. The cause could be having too few parallel I/Os, stripings, or partitions implemented.

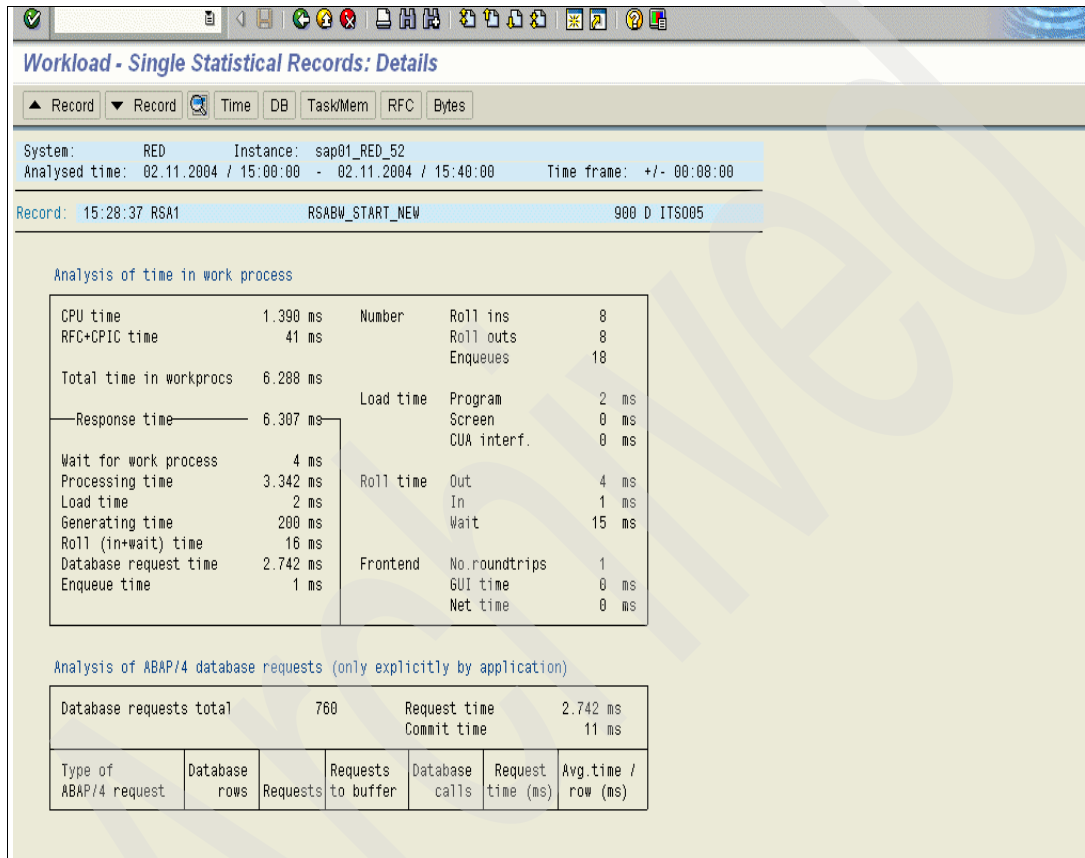


Figure 8-12 Single statistical record - work process time

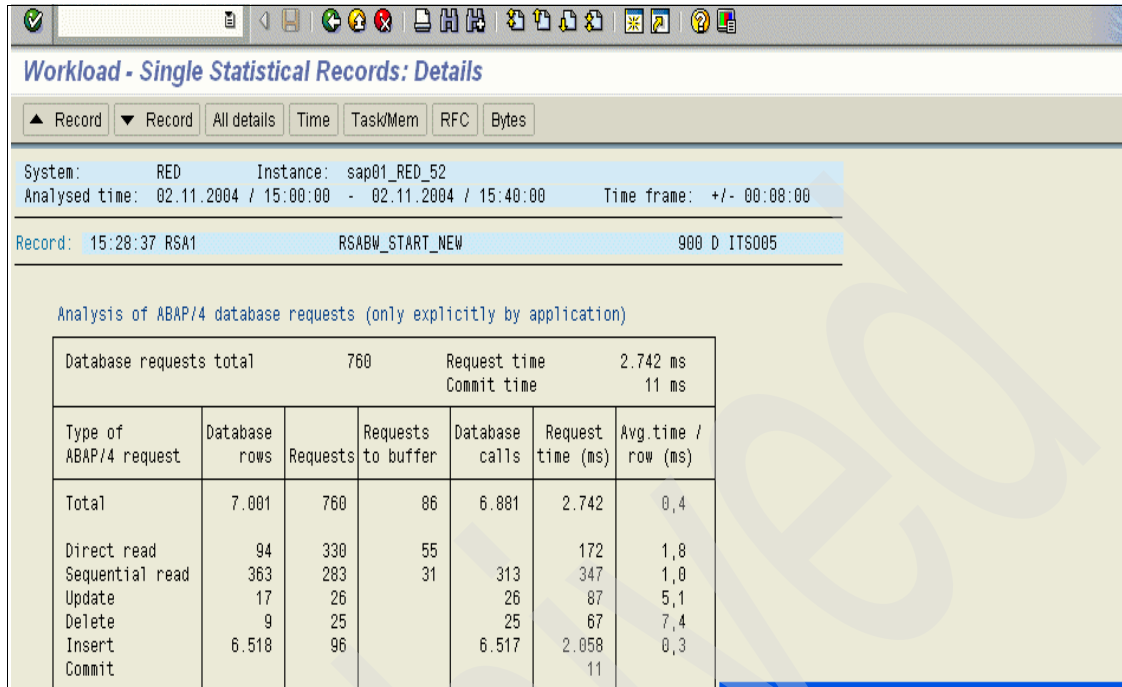


Figure 8-13 Single statistical record - DB request times

How do you determine which table? If the job is still running, you can check Global Work Process Overview (SM66) to determine which table is being processed for a specific user and a specific job. If the job is no longer running, you can check which table is being processed in the SM37 job of the process chain graphical view. See Figure 8-14.

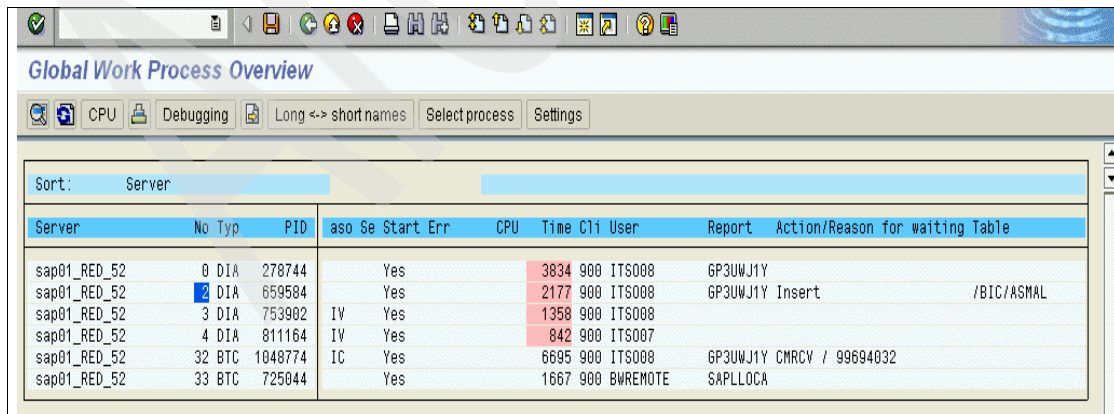


Figure 8-14 Display current work process activities from SM66

Alternatively, if temporary trace is permitted for individual analysis, there is an SAP profile parameter `stat/tabrec` that can be activated to get more statistics on table access or transactions. You can change the online parameter `stat/tabrec`, which specifies the maximum number of table access statistics subrecords that the kernel can write for each transaction step. The N tables with the highest DB request time for each dialog step are stored.

If you only want to activate the table access statistics for specific transactions, you can specify a maximum of five transactions in the online parameters `stat/tcode1` to `stat/tcode5`. The table access statistics are then only produced for these transactions.

After tracing is done, it should be set to 0 (zero) at all times. Otherwise, the number of statistics records rises greatly, which can lead to performance problems in the statistics collector.

To change parameter `tabrec` online:

1. Start the workload monitor by calling transaction ST03.
2. Choose **Expert** → **User Mode**.
3. Choose **Collector & Perf. Database** → **Statistics Records and File** → **Online Parameters** → **Dialog Step Statistics**.
4. The system displays the Change Runtime Parameters for Statistics Collection dialog window.

To look at table access statistics:

1. Start the workload monitor by calling transaction ST03.
2. Choose the user mode.
3. In the Workload tree, choose the instance to be analyzed and the time period.
4. Choose **Analysis Views** → **Table Access Statistics**.

At this point, information from ST03, RSMO, RSPC, and STAD transactions should provide you with a general idea of where the potential cause is. Now what can you do to alleviate it?

If extraction takes the most elapsed time

For this condition you can do a few things on the source data and source system to improve extraction. Source data can be from an earlier system, flat file, or R/3 system. Generally, these are beyond your control, especially an earlier system, which may have slow disks, not enough CPUs, and less memory. However, you can have some influence on loading from a flat file and from a R/3 system, as described here.

Loading from a flat file

You can avoid I/O performance by placing very large flat files on the application server, instead of remotely on a client machine over a slow network or from tape, or on the same disk that has heavy I/O activity. In addition, you can try to use fixed-length ASCII format instead of CVS, where it internally converts into fixed length format. If possible, split large flat files for parallel processing.

Loading from an R/3 system

If loading from an R/3 system, you can use SM50, SM51, and SM66 to look for long-running processes to determine whether there is a resource problem on the source system. Also, you can use Extractor Checker (RSA3) to check for any extraction problem in the first place.

There are specific SAP notes addressing application areas, and you may want to refer to them for guidance. You can temporarily turn on an ST05 trace to analyze SQL statements to identify expensive SQL statements.

Determine whether indexes of datasource tables are okay, and not missing or needed. Having indexes on the source tables avoids full table scans.

For ST05 trace capture, make sure that you filter only with user-running extraction and make sure no other parallel extraction is in progress. Next, check whether runstats are up-to-date for tables that are read from.

If there are more application servers, you may want to spread the load to other servers in order to avoid bottlenecks from a single server's CPU and memory resources. You can do this by configuring table ROIDOCPRMS.

Also note that the data package size defined in this table impacts the frequency of database commits. In general, define smaller package size for resource-constrained systems, and larger ones for larger systems (but make sure that they are not so large that they impact communication processes and result in a holdup of work processes on the source system). Refer to SAP note 409641 for more details.

A number of other factors can contribute to high extraction time (for example, the general performance of the R/3 system, a slow network between the R/3 system and the BW system, resource constraints like not enough CPU, or the memory resources of the R/3 system).

If custom enhancement exits are used for extraction, the ABAP Runtime Analysis tool (SE30) can be used to help with debug performance of ABAP coding. It can help isolate whether the problem is in the area of ABAP, I/O, or SQL by looking at the time spent in each subroutine.

If transfer time takes the most elapsed time

For this condition, use transaction SM50 to determine whether there is resource constraint. Check whether there are enough work processes, especially for loading to PSA and data targets in parallel. Parallel processing can compete for resources. Use the ST06 transaction to determine whether the bottleneck is in CPU, memory, or network.

For parallel processing (load balancing) to multiple BW application servers, synchronization among them is necessary because during large data load, swapping of buffers of master data can occur on a BW application server. Synchronization overhead of buffers among them can impact performance. You can use ST05 to check the SQL trace for log table activities. If this problem occurs, you can try to shut down other application servers and set `rdisp/bufrefmode=sendoff,exeauto` during the load phase.

If upload (update) to a PSA takes the most elapsed time

Potential root causes for this condition could be I/O contention or the partitioning configuration. I/O contention is a possible result of high volumes of insertion during large data loads. Check for database I/O, disk layout, and striping configurations to see whether you can improve it.

The partitioning configuration has an impact as well. If partitions are defined too large, there may be no parallel database subprocesses used. If partitions are defined too small, there may be *too many* parallel database subprocesses used. Packet size and partition size can be done from transaction RSCUSTV6.

Note that in DB2 V8, the partition size defined in RSCUSTV6 will not influence the number of partitions. SAP BW table makes use of the DB2 V8 partition addition feature (for more details refer to 5.1.2, “Management of PSA” on page 68).

The PSA table partition is automatically created for each data request in an InfoPackage. If multiple data requests on the same Infopackage are loaded to PSA, then the PSA table partition will have additional table partitions per the number of loads of data requests. If the same data request load is repeated, a new partition will be added.

Therefore, to influence parallelism, split large loads into multiple Infopackages that each have only one data request. For more information refer to SAP note number 485878 for “DB2/390: BW Partitioning the PSA tables”.

Transfer rules and update rules takes too long

For this condition you can use the simulation tool to debug the problem. For example, in the RSMO detailed window, under Processing, select **Data Package**

and right-click to select the **Simulation Update** tool. Additionally, you can use SM30 or ST05 to identify the ABAP and SQL used in expensive update and transfer rules.

Loading data targets (ODS/InfoCube) takes too long

For this condition try to load master data before transactional data for ODS and InfoCube. By creating all necessary SIDs and populating master data tables first, there will be significant performance improvement because you avoid expensive SID creation during transactional data load.

In addition, try to delete before load, if possible. Performance is gained by deleting existing data before a complete replacement load. Deleting data from PSA helps to reduce PSA access times. Deleting data from InfoCube helps to reduce deletion and compression times.

You can also influence performance improvement by parallel processing. For BW 2.x, it is not possible to have data packets/requests loaded into an ODS object in parallel. In BW 3.x, you can load data from different requests in parallel to an ODS activation queue. Transaction RSCUSTA2 can be used to control data ODS data load. Here you can define a maximum number of parallel dialog work processes, a minimum number of records per package, a server group for load balancing, and so on.

For non-reporting ODS objects, if possible, turn off the BEx Reporting setting. Load will be faster as Master Data SID tables do not have to be read and linked to the ODS data. You can do this by selecting **RSA1** → **InfoProvider**, then double-check the ODS object and expand settings to turn off the BEx Reporting box.

For initial large data load, there are significant database accesses to the NRIV table to fulfill a number of range requests. You can reduce application server access to the database by buffering the SID number range for InfoCube. This can be done using transaction SNRO to increase the number range buffer. After load completion, you may want to reset the number range buffer to its original values to minimize unnecessary memory allocation. Refer to SAP note 130253 for more details.

From the RSPC graphical view of the process chain, you can drop down to the SM37 job log to see more detailed activities. You can see whether there is index manipulation, and which tables are being processed.

Regarding index manipulation, you need to ask whether this is necessary for your particular loading scenario. Generally, index manipulation is effective for the initial full load of large data because of a high volume of insertions. For the initial load of very large data, consider dropping secondary indexes to improve loading

performance. Then rebuild them for query performance. For subsequent daily load, which is usually a delta (small) load to update an existing, very large table, it may be no longer effective because loading and rebuilding indexes may take too long.

You also need a large sort workspace. In addition, runstats for each load may take too long and consume too much CPU resource. Periodic runstats are still necessary, but should not be too frequent.

8.3 Analyzing query performance

In this section we discuss an approach for analyzing query performance, following the road map laid out in Figure 8-1 on page 137.

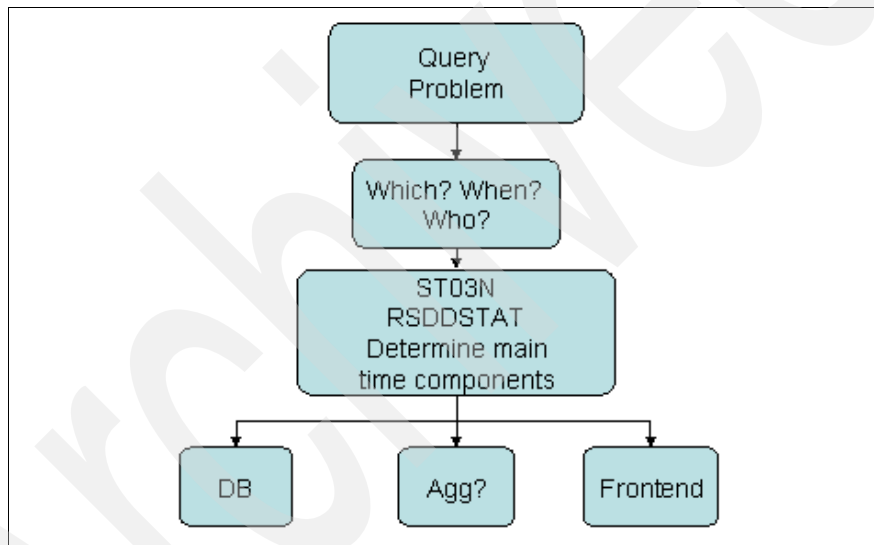


Figure 8-15 Query analysis road map

Although there are some relevant changes from SAP BI 3.x to 7.x in respect to where query statistics data are stored, most of the items discussed and shown in this chapter are also applicable in SAP BI 7.x. If you are running SAP BI 7.x there is a tool available called SAP Query Analyzer that we discuss in 8.5, "SAP Query Analysis Tool" on page 178.

Note: You must activate the saving of statistical data for each InfoCube by the following path: **BW Admin Workbench** → **Tools** → **BW Statistics for InfoProviders** → **Mark field “OLAP” for the InfoCube**. If you do not activate statistics, then you will have no data recorded to allow you to analyze query performance.

Be aware that when you enable the saving of statistics, it will result in more data being stored in RSDSTAT and the cube 0BWTC_C02.

8.3.1 Starting the analysis from ST03

The starting point we use for analyzing query performance is the ST03 Workload Monitor transaction for statistics on individual query executions. Alternatively, on SAP BI 3.x systems you can look at the raw data in table RSDSTAT using transaction SE16.

From ST03 you can see an overview of all of the queries that have been executed over a variety of time periods. The information displayed by ST03 shows a breakdown of the response time of the query by its constituent parts.

Using this information, you can identify the component that contributes the most time to the response time. In our analysis road map, we have broken the query response time into three major components:

- ▶ Database processing time
- ▶ Aggregates required?
- ▶ Front-end processing time

Figure 8-16 shows the data provided by ST03 for a number of queries. To reach this display, execute the following actions in ST03:

1. Select **Expert Mode** from the Change user mode tab in the top left corner of the window.
2. Expand the submenu **BW System Load**.
3. Select the relevant time period for your analysis. See Figure 8-16.

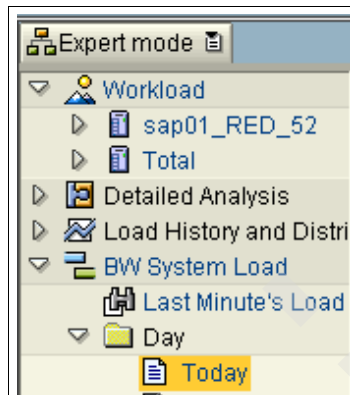


Figure 8-16 Selecting time period for the analysis

4. In the right-hand panel of the window, change the aggregation mode to query.

5. In the right-hand panel of the window, select **All Data** (Figure 8-17).

The screenshot shows a software interface with a header section containing 'Instance: TOTAL', 'Period: 27.10.2004', and 'Task type'. Below this are three tabs: 'Share of Runtime', 'Average Times (AVG)', and 'All data'. The 'All data' tab is selected. A toolbar with various icons is visible above a table titled 'Reporting - QUERIES: All Data'. The table has five columns: 'InfoCube', 'Name of Query', 'No. of Nav.', 'Total time', and 'Ø Total'. The row for 'JOCUBE LONGRUNNING' is highlighted in yellow.

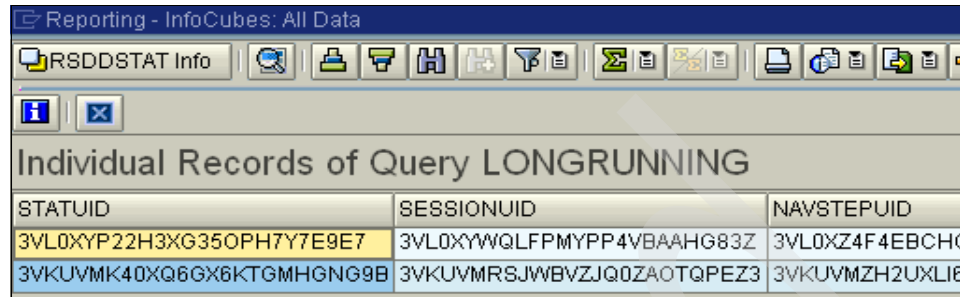
InfoCube	Name of Query	No. of Nav.	Total time	Ø Total
TOTAL	TOTAL	23	962,3	41,8
TESTPART	NOPARTETABLE	14	833,1	59,5
JOCUBE	LONGRUNNING	2	67,7	33,9
ODSBENCH	ZODSBENCH	2	20,5	10,3
ODSCOPYS	ZTEST	1	19,1	19,1
TESTPART	TESTNOPART	1	15,4	15,4
ETBLPART	PEFACTTBL	3	6,5	2,2

Figure 8-17 Selecting All data tab

You will now see the aggregated response time data for the query that you want to analyze. This data gives you an indication of (aggregated) major components of the response time for the query you are analyzing.

For a comprehensive description of the fields of this display, read OSS note 130696. If each execution of the query by the users is the same, then this gives you enough information to identify the major component of the response time.

To drill down to an individual execution of a particular query, double-click the query name. See Figure 8-18.



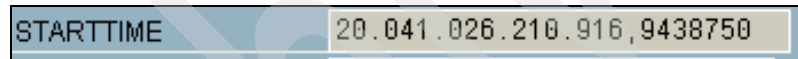
The screenshot shows a window titled 'Reporting - InfoCubes: All Data'. Below the title bar is a toolbar with various icons. The main content area displays 'Individual Records of Query LONGRUNNING' above a table with three columns: STATUID, SESSIONUID, and NAVSTEPUID. The first two rows of data are highlighted in yellow and blue respectively.

STATUID	SESSIONUID	NAVSTEPUID
3VL0XYP22H3XG35OPH7Y7E9E7	3VL0XYWQLFPMYPP4VBAAHG83Z	3VL0XZ4F4EBCHC
3VKUVMK40XQ6GX6KTGMHNG9B	3VKUVMRSJWBVZJQ0ZAOTQPEZ3	3VKUVMZH2UXLIE

Figure 8-18 Sample list of executions of a query

You then get a pop-up window with a row of data for each individual execution of the query. In the top left corner of the pop-up, switch the view from Standard Info to RSDSTAT Info.

You can identify the execution of the query you are interested in by the UNAME column and the STARTTIME columns. See Figure 8-19.



The screenshot shows a pop-up window with a single row of data. The first column is labeled 'STARTTIME' and the value is '20.041.026.210.916,9438750'.

STARTTIME	20.041.026.210.916,9438750
-----------	----------------------------

Figure 8-19 Example STARTTIME time in RSDSTAT

This time is taken from the database server of your BW system. On your z/OS server there are two times: LOCAL and UTC/GMT. (Note that this is the UTC/GMT time on z/OS, not the LOCAL time.) The format of the start time is *YYYYMMDDhhmmssmmuuun*. In this example, this start time represents 2004 October 26 21:09:16.

You can now analyze your query to identify the major component of the response time. In the following sections we discuss how to identify and analyze our three main components of response time:

- ▶ Aggregates required?
- ▶ Database processing time
- ▶ Front-end processing time

8.3.2 Determining whether aggregates are required

In this section we discuss how to identify whether your query response time can be improved by the use of aggregates.

If your query has long run times that are unacceptable to the user, and if the query summarizes many rows, then using aggregates is often the most effective way to speed up the query. To determine whether your query would benefit from aggregates, you can look at the detailed breakdown of the response time on ST03.

QDBSEL/QDBTRANS

If the QDBTIME component of the response time is very high, examine the fields QDBSEL/QDBTRANS to determine whether the query is reading too many records from the cube in relation to the amount of rows that are transferred back to the OLAP engine.

The field QDBSEL is the number of rows that satisfied the predicates. The field QDBTRANS tells you how many records were transferred from the database to the application server after the GROUP BY statement in the SQL.

As a rule of thumb (RoT), if the ratio between QDBSEL/QDBTRANS is > 10 , then you may be able to get larger improvements in performance through the use of aggregates than through other database or SQL tuning. However, this does not mean that you should ignore other methods for further improving performance by analyzing the SQL used by the query.

QDBSEL/QDBTRANS is the number of rows selected for each row returned to the query user. When this is high, it means that DB2 must summarize many selected rows into each result row, so an SAP aggregate table would probably help performance, since the aggregate table would contain pre-summarized data. After the E- fact table is summarized into an aggregate table, the query will retrieve fewer rows to create the result.

The SAP rule of thumb is that when this ratio is greater than 10, an aggregate may be appropriate. Use the frequency of query execution, the importance of performance for the query, and the time available for aggregate roll-ups in conjunction with this rule of thumb when evaluating the need for new aggregate tables.

For instance, if an aggregate is not frequently used, and if the query is very slow but time and CPU are available in the aggregate roll-up window, then creating an aggregate is a way to move part of the report generation time to the nighttime.

If the ratio QDBSEL/QDBTRANS is low (which indicates that aggregates may not help) and the QDBTIME is still very high, then you should look at the relationship of the fields QDBSEL/QTIMEDB. This is a measure of how fast (rows per second) the rows are retrieved from the database. As a rule of thumb, if the figure is 300–400 or lower, then you should further analyze the database to evaluate the way the SQL is executing. Continue down the database leg of our

road map to determine the cause of the slow retrieval of records and high database time.

RSDU_ANALYZE_TOOL_DB2

Another tool to help you analyze the need for aggregates, or to determine whether further database analysis is needed, is the function module RSDU_ANALYZE_TOOL_DB2. You can execute this function module from transaction SE37.

This tool provides you with information about your query, such as the number of rows examined by DB2, selected rows (before the aggregation), and transported rows (after the aggregation). In addition, the performance indicators we discussed are also calculated:

- ▶ Examined rows per selected row
- ▶ Selected rows per transported rows
- ▶ Selected rows per second

Note that some values in the list are determined from the statistics of the DB2 Dynamic Statement Cache. These values are 0 if the corresponding SQL statement is no longer in the cache. For more information about this tool, read OSS note 633832.

The next steps

Once you have determined that aggregates may help the performance of the query, you can discuss creating aggregates for the InfoCube with the BW functional consultant responsible for the design of the InfoCube.

Keep in mind that, while aggregates help query performance, the updating of these aggregates with data adds time to your data load process, and requires extra system resources during the build process. However, as this is normally done during off-peak hours (usually overnight), then for most installations it does not create any problems.

In this book we do not go into detail about the proposal and building of aggregates. However, we introduce the RSA1 tool, which can be used to do this. For a more detailed discussion of this subject, refer to the following IBM whitepaper: *SAP BW Query Performance with DB2 on zSeries*:

<http://www-1.ibm.com/support/docview.wss?uid=tss1wp100322>

Using RSA1 to propose and build aggregates

The BW workbench tool RSA1 can be used to help propose and generate aggregates for an InfoCube. Use the following menu access path to reach this functionary: **RSA1** → **Modeling** → **InfoProvider**, then right-click the InfoCube

that the query runs against → **Maintain Aggregates**. At the top of the window select **Propose** → **propose** (statistics, usually query).

Enter a time range in the pop-up window. BW analyzes the queries run against this InfoCube, lists existing aggregates, and proposes new aggregates and the number of queries that would benefit from the proposed aggregate.

8.3.3 Front-end processing time

In this section we discuss how to identify whether your query response time is being impacted adversely by slow network front-end processing on a user's personal computer. We discuss which response time components to look for when analyzing the query response time components in ST03.

From the query response time breakdown in ST03, the column that records the time taken in the network and the front-end PC is entitled Frontend. In the table RSDSTAT, the column that records this time is entitled QTIMEEXCEL.

This time measurement starts as soon as the data has been formatted in the BW server and is sent to the front-end Excel® application (Bex). As soon as the front-end (Bex) has inserted all the data into the worksheet, the time measurement ends. Therefore, this time includes all network time and the time taken to process and present the results in the front-end personal computer.

Further analysis

If the front-end processing time is lengthy, then you can do further analysis to determine which subcomponent is contributing the most time. Follow this process.

Find the IP address of the front-end personal computer that the user is logged on to by using the transaction AL08. In the output display of AL08, you will see the user's front-end address in the column entitled Terminal.

After obtaining the terminal address, you can go to the next step. To check the connection between the BW server and the user's PC, use the following menu path: **ST06** → **Additional Functions** → **LAN check by Ping** and select **Presentation server**.

You will be presented with a list of connected presentation (front-end) servers. Highlight the server found in the AL08 transaction by clicking the name and then click the **Pick** button at the top of the window.

You can now ping that presentation server to check the network response time between the BW application server and the presentation server. If this time is excessive, contact your network support group for more detailed investigation.

Analysis of the processing on the personal computer can be done by switching on the Windows resource monitor while the query is running. (This needs to be done locally at the personal computer.) If the CPU and the memory of the personal computer is overused, then the long network time may be caused by slow personal computers.

8.3.4 Database performance

If the main component of your query response time is database time, and you have already analyzed the need for aggregates and found that they are either already present or will not help. The next stage of your analysis is to look at the SQL and the performance of the database.

The starting point we can use for analyzing database performance is shown in the flow diagram in Figure 8-20.

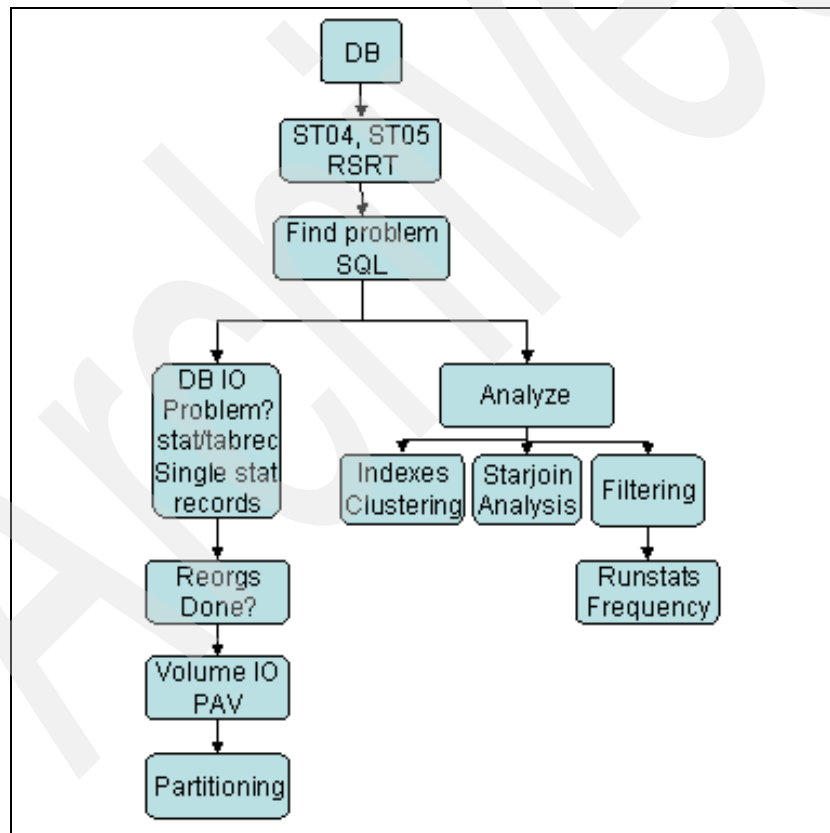


Figure 8-20 DB performance analysis

If the main component of query performance is database time. Continue your analysis using the following tools:

RSRT Query monitor. Use to run queries on InfoCubes or ODS in debug mode.

ST04 Database administration. Ensure that the IFCID(318) trace is started. The command `-START TRACE(P) CLASS(30) IFCID(318) DEST(SMF)` must be issued either by your DBA or from transaction ST04.

ST05 SQL performance analysis. Use to trace an active query or coordinate start of trace with user to trace a known problem query.

RSRT query monitor

The RSRT transaction can be used to run queries on InfoCubes or ODS. See Figure 8-21.

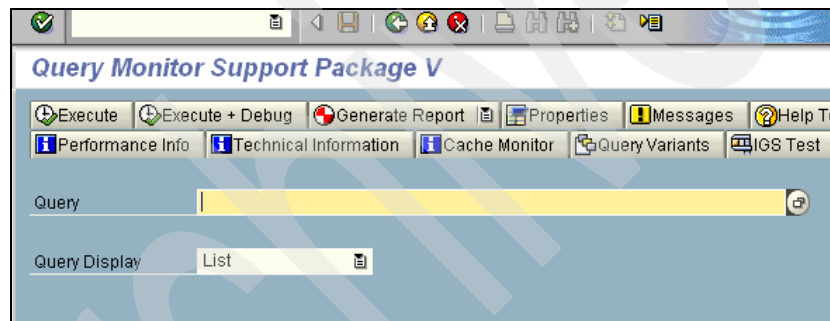


Figure 8-21 RSRT main window

Type in the query name that you would like to test or use the F4 key to generate a query value help box. Click **Execute + Debug** to display the options that can be selected. Refer to Figure 8-22.

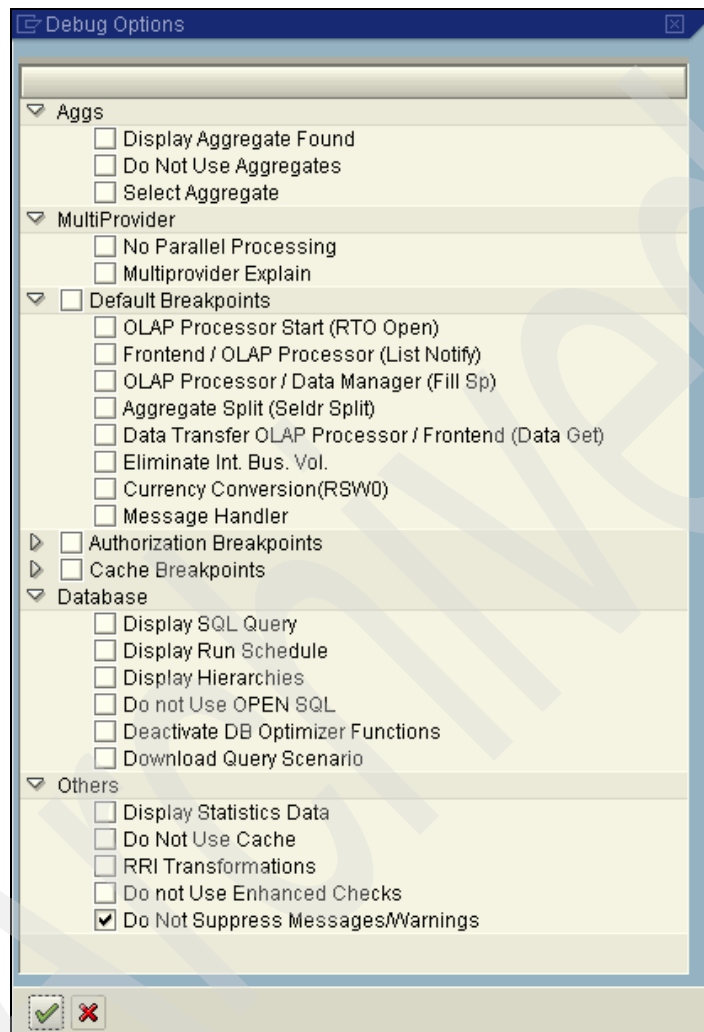


Figure 8-22 RSRT execute and debug options

Some commonly used options are:

- ▶ Display SQL Query (to display the statement including literals before the statement is executed).
- ▶ Display Run Schedule (to explain the statement before the statement is executed).

- ▶ Display Statistics Data (to show the RSDDSTAT statistics with time breakdown).
- ▶ Do Not Use Cache (to bypass the OLAP cache and force the SQL to be executed in DB2).
- ▶ Do Not Use Aggregates (to bypass use of aggregates).

Initially, run the query using the options Display Statistics and Do Not Use Cache. The report output will be displayed first. See Figure 8-23.

BW - output test

Relational browse | Key Figure Definition | Key Figure Detail | Menu

BW - output test

Messages:

'JOPROD			
'DESKTOP	824,417 ST	91,567,693.00	EUR
'MONITOR	822,206 ST	91,358,507.00	EUR
'NOTEBOOK	827,019 ST	91,739,067.00	EUR
'PRINTER	825,914 ST	91,353,013.00	EUR
'SOFTWARE	823,535 ST	91,546,740.00	EUR
'Overall Result	4,123,091 ST	457,565,020.00	EUR

Additional information:

Free chars:

InfoObject	Name	Hierarchy Name
JOCUST	JOCUST	
0CALMONTH	Calendar Year/Month	
JOPROD	JOPROD	

Attributes:

InfoObject	Attribute	HiText

Structures:

3TRKQ3E074Y047M9STNDXPAWH

Text symbols:

Description	Value
Author	C5001828
Last Changed by	C5001828
InfoProvider	JOCUBE
Query Technical Name	LONGRUNNING
Query Description	
Key Date	28.10.2004
Changed At	18.06.2004 13:03:37
Status of Data	14.05.2004 14:22:03
Current User	ITS001
Last Refreshed	28.10.2004 17:47:56
Calendar Year/Month	02.2003..12.2003
JOCUST	CUST0002..CUST9993
JOPROD	DESKTOP..SOFTWARE

Figure 8-23 RSRT query output

Pressing the green arrow displays the RSDDSTAT statistics. See Figure 8-24.

STATUID	QA6GRUSED	QDBSEL	QDBTRANS	QNUMCELLS	QNUMRANGES	RECCHAVLRE	QTIMEDLAPI	QTIMEOLAP	QTIMEDB	QTIMEVAROP	Q
3VLSS1PP3K2WANISZ2AZA1027		880,091	5	21	10	13	0.253907	0.195312	29.019531	0.000000	0

Figure 8-24 RSRT RSDDSTAT

Note that QTIMEDB can contain time spent in the RSRT steps for *explaining* and *viewing* the SQL, so QTIMEDB and the elapsed time in the ST04 statement cache may be different from each other.

You can use STATUID to link to the SQL statement cache and display the runtime statistics for this statement. Run **ST04** → **Cached statement statistics**, which will give you the window shown in Figure 8-25.

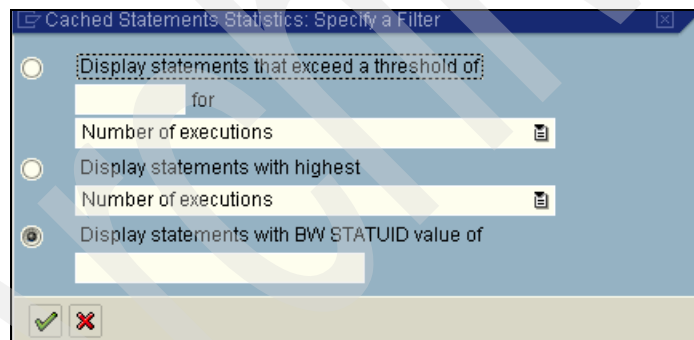


Figure 8-25 ST04 - Cached Statements Statistics pop-up

Select the appropriate button, enter the STATUID, and press Enter. Refer to Figure 8-26.

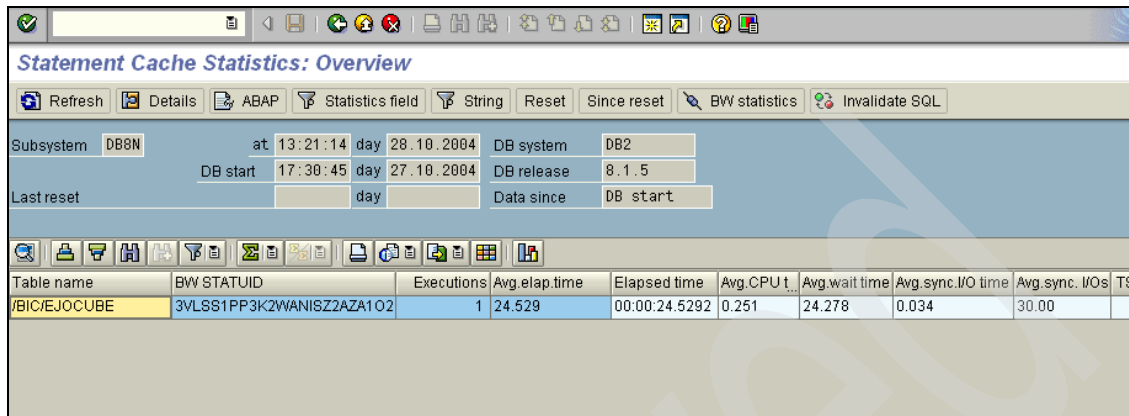


Figure 8-26 ST04 cache statistics selected by STATUID

Note that if this functionality fails, check OSS note 633075.

Select a statement and click **BW Statistics**, which will give you the full RSDDSTAT details. From these, you can determine the DB access time, the number of rows examined, number of rows selected, and so on. For full details of the data returned from RSDDSTAT, refer to OSS note 130696.

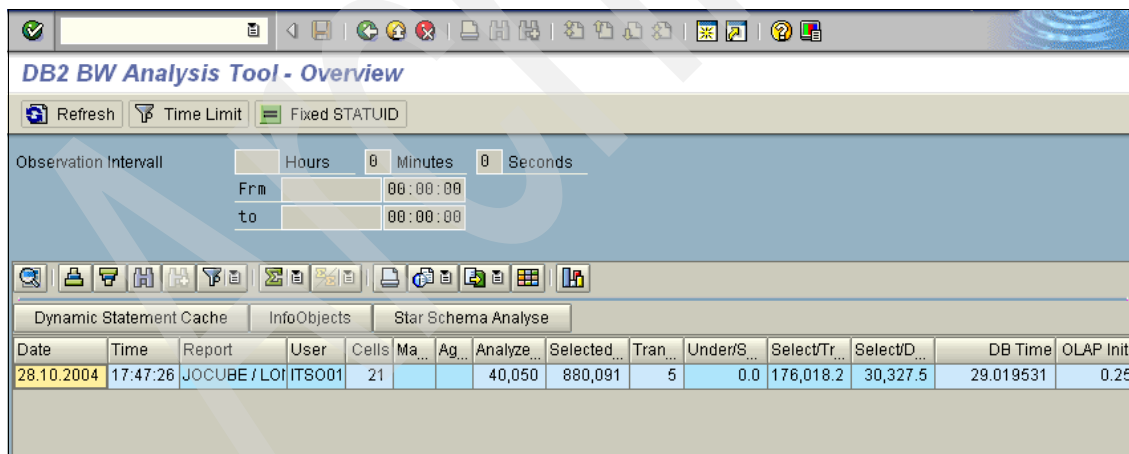


Figure 8-27 ST04 statement cache BW statistics

This example query used 29 seconds of DB time, selected 880,091 rows that satisfied the predicates, and then transferred 5 rows to the output window. Looking back at the aforementioned rule of thumb (rows selected divided by rows

transferred is greater than 10), the aggregate may be useful. Refer to 8.3.2, “Determining whether aggregates are required” on page 161, for more detailed explanations.

Note that in Figure 8-27 on page 170, the number of rows analyzed is shown as 40,500. This is obviously incorrect (given that 880,091 rows were selected), and is due to query parallelism. This can be checked by returning to the ST04 overview window, highlighting the line you selected and pressing **Details** → **Avg Execution Statistics**. Then check the Avg Parallel Groups field, as shown in Figure 8-28.

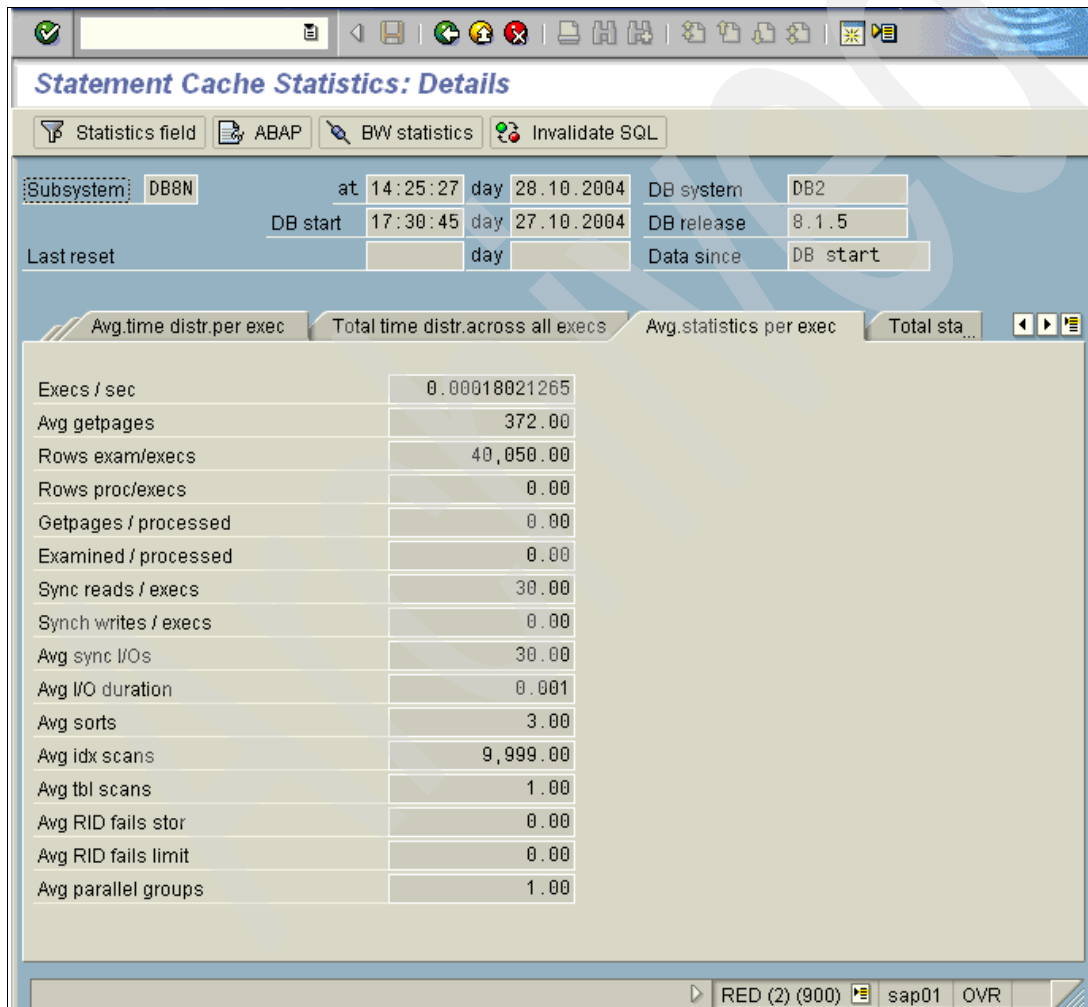


Figure 8-28 ST04 query parallelism

To check the access path used by the SQL, from the statement cache statistics window, select a statement and press **Details**. Refer to Figure 8-29.

The screenshot shows the 'Statement Cache Statistics: Details' window in SAP. At the top, there are buttons for 'Statistics field', 'ABAP', 'BW statistics', and 'Invalidate SQL'. Below this, a summary table provides system and timing information:

Subsystem	DB8N	at	15:24:02	day	28.10.2004	DB system	DB2
		DB start	17:30:45	day	27.10.2004	DB release	8.1.5
Last reset				day		Data since	DB start

Below the summary, there are tabs for 'Statement text', 'Identification and status', 'Avg.time distr.per exec', 'Total time distr.across all execs', and 'Avg.statistics per exec'. The 'Statement text' tab is active, showing an 'Explain' section with a list of SQL statements. The first statement is highlighted in yellow:

```
SELECT "S____002", "S____005", "S____006", "1ROWCOUNT", "JO_QTY",
"JO_REV" FROM ( SELECT "D1"."SID_JOPROD" AS "S____002"
,"DU"."SID_OBASE_UOM" AS "S____005", "DU"."SID_OCURRENCY" AS "S____006"
, COUNT(*) AS "1ROWCOUNT", SUM ("F"."/BIC/JO_QTY") AS "JO_QTY",
SUM ("F"."/BIC/JO_REV") AS "JO_REV" FROM "/BIC/EJOCUBE" "F"
"/BIC/DJOCUBE1" "D1", "/BIC/SJOPROD" "S1", "/BIC/DJOCUBEU" "DU"
,"/BIC/DJOCUBET" "DT", "/BIC/DJOCUBEP" "DP", "/BIC/DJOCUBE3" "D3"
"/BIC/SJOCUST" "S2" WHERE "F"."KEY_JOCUBE1" = "D1"."DIMID" AND
"D1"."SID_JOPROD" = "S1"."SID" AND "F"."KEY_JOCUBEU" = "DU"."DIMID" AND
"F"."KEY_JOCUBET" = "DT"."DIMID" AND "F"."KEY_JOCUBEP" = "DP"."DIMID"
AND "F"."KEY_JOCUBE3" = "D3"."DIMID" AND "D3"."SID_JOCUST" = "S2"."SID"
```

Below the SQL statement, there is a 'Prepare attributes' section with the text 'WITH HOLD' highlighted in yellow. At the bottom right of the window, the status bar shows 'RED (2) (900) sap01 OVR'.

Figure 8-29 Statement details

Select **Explain**. Answer **YES** to the pop-up, Turn on parallelism. Refer to Figure 8-30.

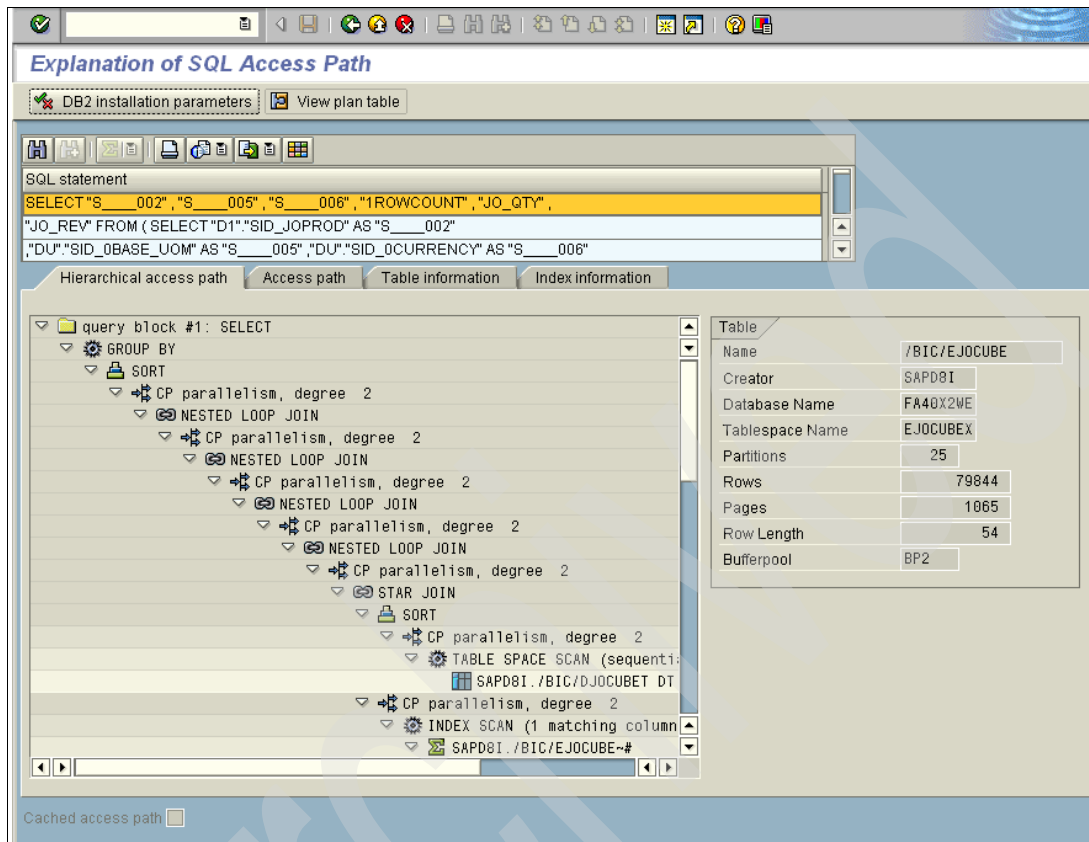


Figure 8-30 ST04 Explain output

The Explain window gives you a graphical representation of the SQL access path and options to view table/index information and the plan table.

For further details on interpreting the plan table output, refer to the DB2 UDB for z/OS SQL Reference publication. For a more detailed discussion on this subject, refer to IBM whitepaper *SAP BW Query Performance with DB2 on ZSeries*:

<http://www-1.ibm.com/support/docview.wss?uid=tss1wp100322>

ST04 - database administration

Use table RSDDSTAT to identify the STATUID for the query and filter ST04 Cached Statement Statistics using this value. This shows you the SQL that is relevant to the query.

The ratio (ST04 statement cache “rows examined”/QDBSEL) is the number of rows examined for each row that satisfied the predicates. If this is high, it can be a sign of access path problems caused by predicate columns without indexes, or that DB2 is not able to use a filtering predicate early in the query to reduce the number of candidate rows.

When a query is run with indexes that filter the result early and effectively, this ratio may be as low as 5 or 10. By itself, this ratio is *not* a reliable indicator of performance problems. For example, if a query contains a small table without a good index, the table may be repeatedly scanned and this ratio will be high, but the query may still run very quickly, since scanning a few dozen rows of a table in memory is fast. If you find a query that is slow, and where this ratio is high (for example, 100–500, or more), then use the ST04 Explain tool.

8.4 DB2 Visual Explain

There is an alternative way to interpret the plan table output manually. You may want to consider using DB2 Visual Explain. It is a software tool that lets you view and analyze DB2 access paths graphically to facilitate your analysis and tuning of SQL queries or statements.

The tool offers suggestions for improving the performance of your SQL queries or statements. You can change an SQL statement and dynamically explain it to see whether the access path is improved by the change. You can also use this tool to browse the current values of the subsystem parameters.

DB2 Visual Explain makes Distributed Relational Database Architecture™ (DRDA) queries through a DB2 client on the (for example, Microsoft® Windows) workstation to get the information it needs. The subsystem parameter values are retrieved by calling a stored procedure on the host, which makes an IFI call to the appropriate DB2 trace record and returns them to the workstation.

Basically, DB2 Visual Explain helps database administrators and application developers do the following:

- ▶ See the access path for a given SQL statement.
- ▶ Tune SQL statements for better performance.
- ▶ View the current values for subsystem parameters.

Figure 8-31, Figure 8-32 on page 176, and Figure 8-33 on page 177 show examples of DB2 Visual Explain windows.

QUERYNO	QBLOCKNO	APPLNAME	PROGNAME	PLANNO	METHOD	CREATOR	TNAME	TABNO	ACESSTYPE	MATCHCOLS	ACCESSCREATOR	ACCESSNAME	INDEXONLY	SORTN_LINK
1	1		SQLLF000	1	0	SYSIBM	SYSPLAN	1	I	1	SYSIBM	DSNPPH01	N	N
2	1		SQLLF000	1	0	SYSIBM	SYSINDEXES	1	R	0			N	N
2	1		SQLLF000	2	1	SYSIBM	SYSTABLES	2	I	2	SYSIBM	DSNDTX01	Y	N

Figure 8-31 Visual Explain Plan Table - all rows displayed

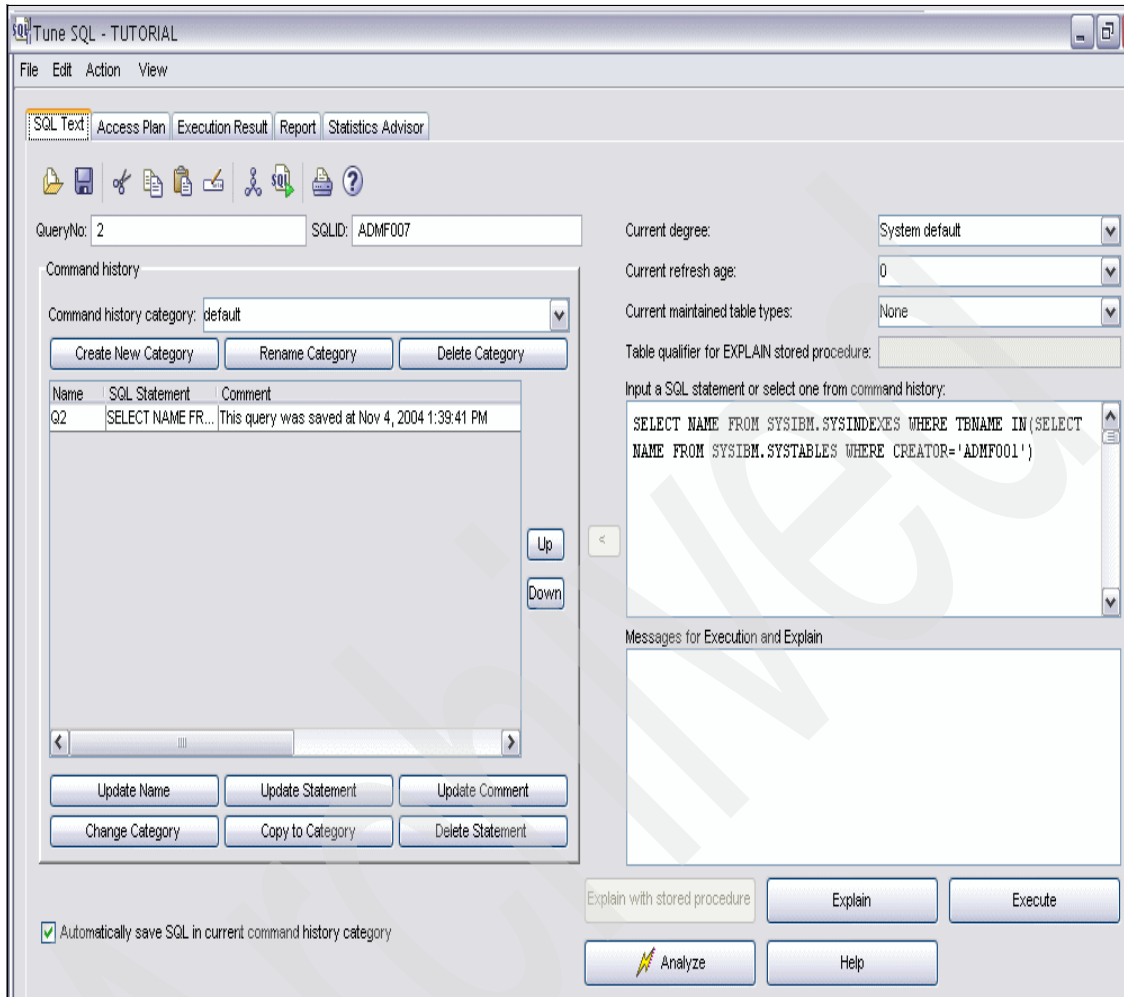


Figure 8-32 Visual Explain - SQL text displayed

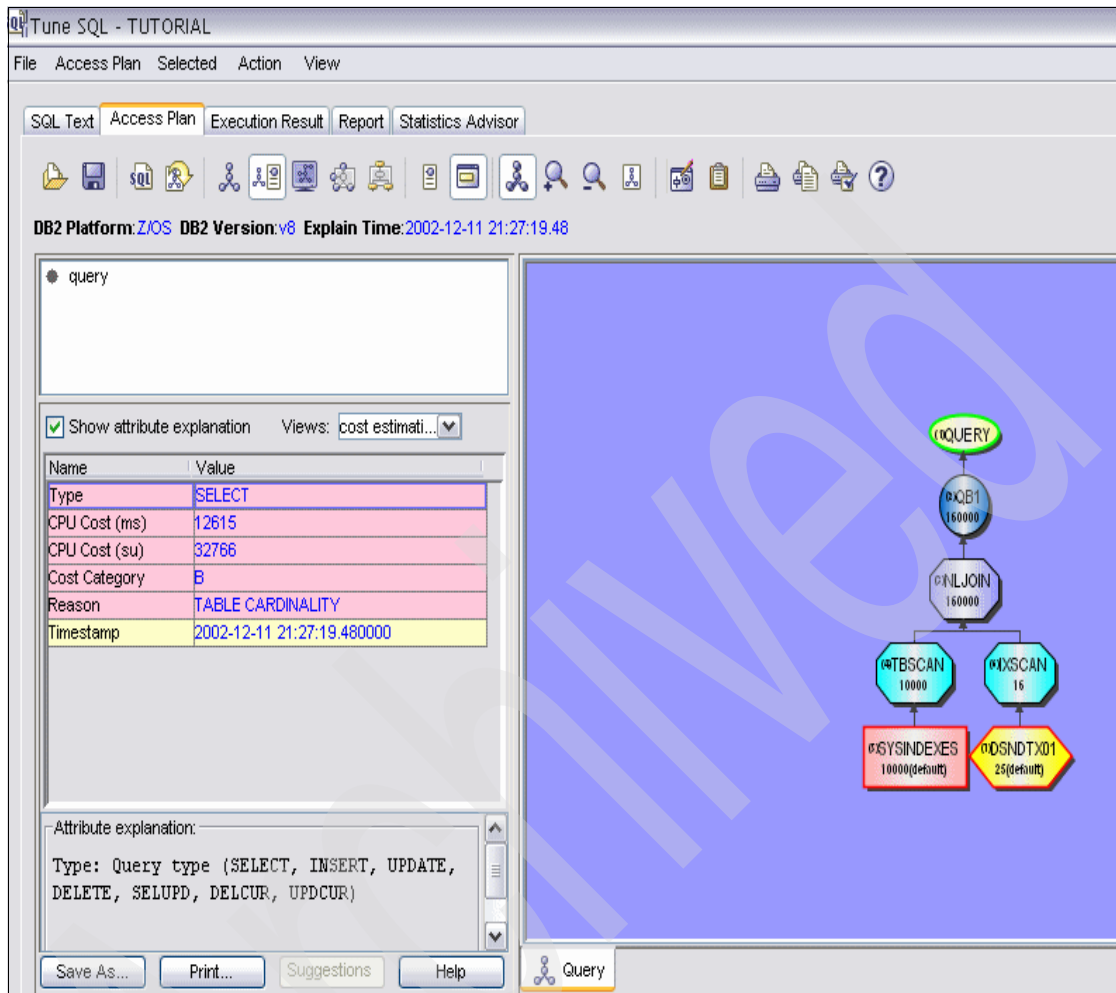


Figure 8-33 Visual Explain - graphical display of access plan

8.5 SAP Query Analysis Tool

The SAP Query Analysis tool is one of the new features delivered in BI 7.00 support package 10.

The SAP Query Analysis Tool is embedded in the BI framework and can be accessed from transaction RSRT.

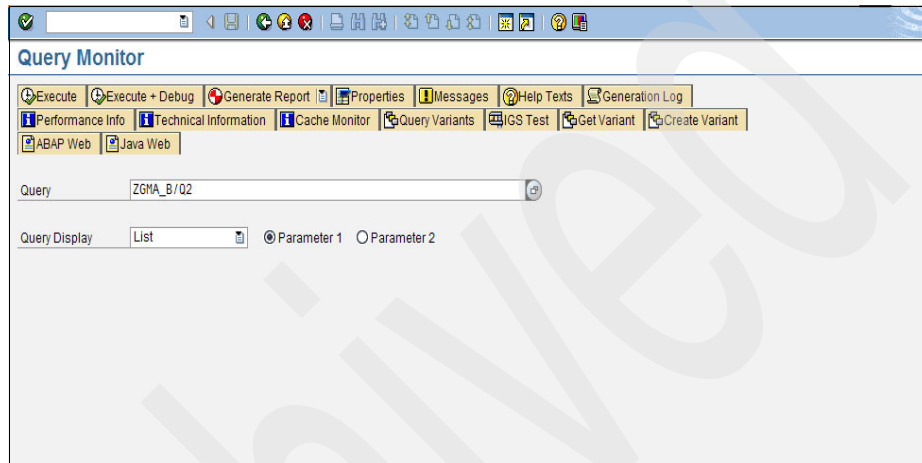


Figure 8-34 SAP Query Analysis Tool - RSRT

The SAP Query Analysis Tool provides:

- ▶ The facility to analyze complex SQL statements.
- ▶ It supplements the db tools such as EXPLAIN.
- ▶ A graphical view of the query.

The SAP Query Analysis Tool enables you to analyze the database request for a query and the impact on performance so that you are able to improve the performance accordingly. When you execute transaction RSRT, select the query to be analyzed and then click the **Execute and Debug** tab. You are then presented with the window shown in Figure 8-35, from which you select option **Display SQL/BIA Query**. You are then presented with the window shown in Figure 8-35, which offers you the following two options to analyze the query:

- ▶ Display of SQL access plan
- ▶ SQL Query Visualizer

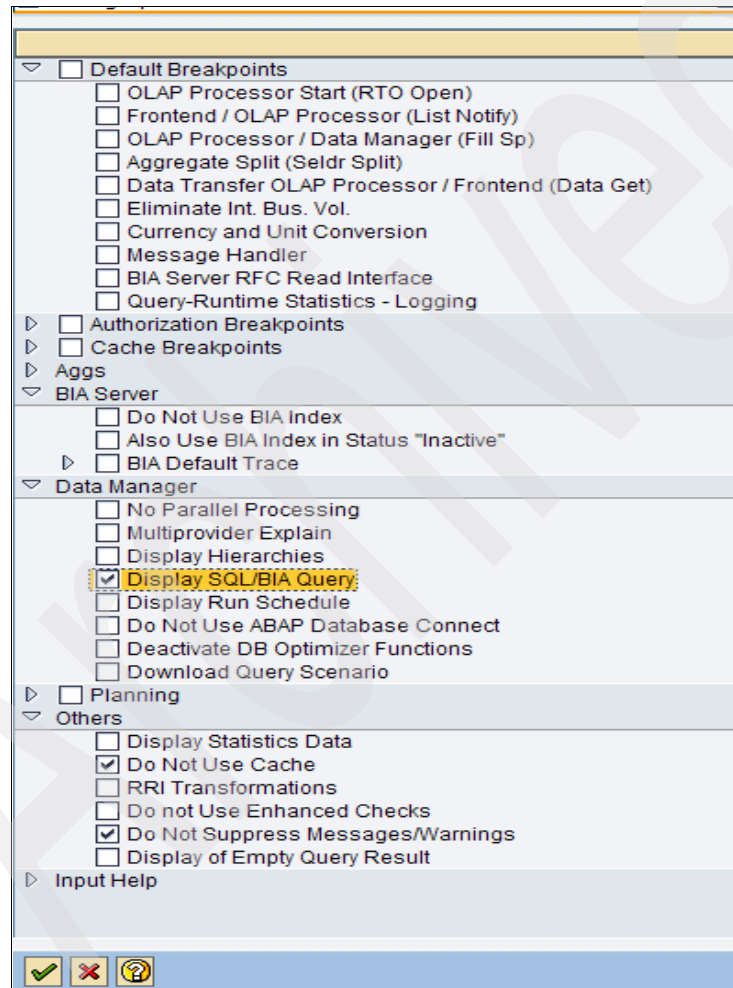


Figure 8-35 Options available when executing in debug mode

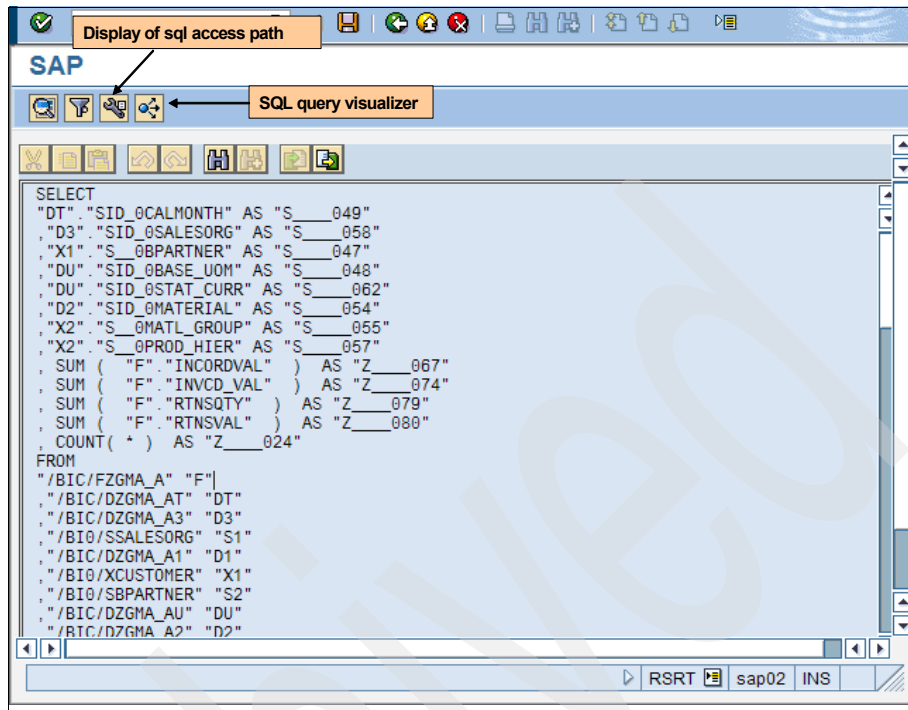


Figure 8-36 Window Displaying the Query

8.5.1 Display of SQL access plan

The “Display of sql access plan” on the SAP Query Analysis tool enables you to learn what access plan is used by the database to execute the query. The “Display of sql access plan” option shows the complete access plan used by the database to execute the query based on the current system state.

Because BI queries are executed with SET CURRENT DEGREE=“ANY”, always choose **Enable Parallelism for Explain** when analyzing BI queries.

For join performance it is essential that the filtering dimensions are accessed before the E- fact table, and the dimensions providing no essential filtering are joined afterwards to the result. See Figure 8-37.

System Help

Explanation of SQL Access Path

DB2 installation parameters View plan table

SQL statement

```
SELECT 'DT'.'SID_0CALMONTH' AS 'S___049', 'D3'.'SID_0SALESORG' AS 'S___058', 'X1'.'S_0BPARTNER' AS 'S___047', 'DU'.'SID_0BASE_UOM' AS 'S___048', 'DU'.'SID_0STAT_CURR' AS 'S___062', 'D2'.'SID_0MATERIAL' AS 'S___054', 'X2'.'S_0MATL_GROUP' AS 'S___055', 'X2'.'S_0PROD_HIER' AS 'S___056'
```

Hierarchical access path Access path Table information Index information

- NESTED LOOP JOIN
 - NESTED LOOP JOIN
 - NESTED LOOP JOIN
 - INDEX SCAN (1 matching column)
 - BI9DB./BIC/DZGMA_A4-010
 - BI9DB./BIC/DZGMA_A4 D4
 - TABLE SPACE SCAN (sequential prefetch)
 - BI9DB.DSN_DIM_TBLX(02)
 - INDEX SCAN (1 matching column)
 - BI9DB./BIC/FZGMA_A-000
 - BI9DB./BIC/FZGMA_A F
 - INDEX SCAN (1 matching column)
 - BI9DB./BIC/DZGMA_A1~0
 - BI9DB./BIC/DZGMA_A1 D1
 - INDEX SCAN (2 matching columns)
 - BI9DB./BI0/XCUSTOMER-0
 - BI9DB./BI0/XCUSTOMER X1
 - INDEX SCAN (1 matching column)
 - BI9DB./BI0/SBPARTNER-001
 - BI9DB./BI0/SBPARTNER S2
 - INDEX SCAN (1 matching column)

Cached access path

Figure 8-37 Display of SQL access path

This provides you with a graphical representation of the SQL access path, options to view table/index information, and a view the plan table. It also gives you detailed information of the SQL statement executed by the query.

The access path is displayed hierarchically. By double-clicking a particular table or index you are able to view the relevant data for the table/index. As displayed in the example below we are able to retrieve information about the table /BIO/SSALESORG. The information retrieved included the table name, creator, database name, name of the tablespace where the table resides, and so on. The relevant statistical values are also displayed, such as the number of rows and pages in the table. See Figure 8-38 and Figure 8-39 on page 183.

The screenshot shows the 'Explanation of SQL Access Path' window. The SQL statement is:


```
SELECT "DT"."SID_0CALMONTH" AS "S_049", "D1"."SID_ZCUSTOMER" AS "S_120", "D2"."SID_ZMA
    IAL" AS "S_110", "X1"."S_0MATL_TYPE" AS "S_121", "X1"."S_0RPA_WGH1" AS "S_122", "X1"
    ."S_0RT_COLOR" AS "S_123", "DU"."SID_0STAT_CURR" AS "S_062", SUM ( "F"."RTNSVAL" ) A
```

 The 'Table information' tab is selected, showing a hierarchical access path. The table /BIO/SSALESORG S6 is highlighted. The right-hand pane displays the following table information:

| | |
|-----------------|----------------|
| Name | /BIO/SSALESORG |
| Creator | BI9DB |
| Database Name | A043XLZ8 |
| Tablespace Name | XSAP |
| Partitions | 0 |
| Rows | 1001 |
| Pages | 9 |
| Row Length | 33 |
| Bufferpool | BP2 |

Figure 8-38 Display of table /BIO/SSALESORG information

| | |
|-----------------|----------------|
| Name | /BIO/SSALESORG |
| Creator | BI9DB |
| Database Name | A043XLZ8 |
| Tablespace Name | XSAP |
| Partitions | 0 |
| Rows | 1001 |
| Pages | 9 |
| Row Length | 33 |
| Bufferpool | BP2 |

Figure 8-39 Table /BIO/SSALESORG information

The table information tab shown in Figure 8-40 gives you the following details:

- ▶ Table names that are used by the query.
- ▶ Runstats information - Is runstats on the table need?
- ▶ Reorg - Is there a requirement to reorg the table?
- ▶ Bufferpool - gives you the name of the bufferpool that the table is using.
- ▶ Part - This column gives you information of the partition used. From the Table information tab you are able to navigate directly in the DDIC editor (SE11), to the storage parameters (SE14), and to the data set space information of the relevant tablespace (DB02).

The screenshot shows the 'Explanation of SQL Access Path' window. The SQL statement is:


```
SELECT 'DT':'SID_0CALMONTH' AS 'S____049', 'D1':'SID_ZCUSTOMER' AS 'S____120', 'D2':'SID_ZMA
    IAL' AS 'S____110', 'X1':'S_0MATL_TYPE' AS 'S____121', 'X1':'S_0RPA_WGH1' AS 'S____122', 'X1'
    'S_0RT_COLOR' AS 'S____123', 'DU':'SID_0STAT_CURR' AS 'S____062', SUM ( 'F':'RTNSVAL' ) A
```

 The 'Table information' tab is active, showing a table with the following data:

| Schema | Name | Part | Rows | Pages | Row length | RUNSTATS? | REORG? | Bufferpool | VOLATILE |
|--------|-----------------|------|---------|-------|------------|------------|-------------|------------|----------|
| BI9DB | /BI0/SDIVISION | 0 | 11 | 1 | 29 | Not needed | Recommended | BP2 | No |
| BI9DB | /BI0/SMATL_TYPE | 0 | 108 | 1 | 33 | Not needed | Recommended | BP2 | No |
| BI9DB | /BI0/SRPA_WGH1 | 0 | 8.336 | 91 | 43 | Not needed | Recommended | BP2 | No |
| BI9DB | /BI0/SRT_COLOR | 0 | 8.317 | 128 | 61 | Not needed | Recommended | BP2 | No |
| BI9DB | /BI0/SSALESORG | 0 | 1.001 | 9 | 33 | Not needed | Recommended | BP2 | No |
| BI9DB | /BI0/SVERSION | 0 | 101 | 1 | 31 | Not needed | Recommended | BP2 | No |
| BI9DB | /BIC/DZGMA_Z11 | 0 | 10.001 | 40 | 16 | Not needed | Recommended | BP6 | No |
| BI9DB | /BIC/DZGMA_Z12 | 0 | 10.001 | 40 | 16 | Not needed | Recommended | BP6 | No |
| BI9DB | /BIC/DZGMA_Z13 | 0 | 99.994 | 592 | 24 | Not needed | Recommended | BP6 | No |
| BI9DB | /BIC/DZGMA_Z14 | 0 | 101 | 1 | 16 | Not needed | Not needed | BP6 | No |
| BI9DB | /BIC/DZGMA_Z1P | 0 | 64 | 1 | 24 | Not needed | Not needed | BP6 | No |
| BI9DB | /BIC/DZGMA_Z1T | 0 | 3.289 | 23 | 28 | Not needed | Recommended | BP6 | No |
| BI9DB | /BIC/DZGMA_Z1U | 0 | 2 | 1 | 20 | Not needed | Not needed | BP6 | No |
| BI9DB | /BIC/FZGMA_Z1 | 1 | 200.000 | 4.445 | 89 | Not needed | Not needed | BP8 | |
| BI9DB | /BIC/FZGMA_Z1 | 2 | 200.000 | 4.445 | 89 | Not needed | Not needed | BP8 | |

Figure 8-40 Table information

The Index information tab shown in Figure 8-41 gives you the details below:

- ▶ Index - names of the indexes that are used by the query.
- ▶ Index Columns - Tells you which key is used for the index.
- ▶ Reorg - Is there a requirement to do a reorg?
- ▶ Part - This column gives you information of the partition used.

The screenshot shows the 'Explanation of SQL Access Path' window. At the top, there's a title bar and a menu bar. Below that, there's a section for 'DB2 installation parameters' and a 'View plan table' button. The main area shows the 'SQL statement' and a list of indexes. The 'Index information' tab is selected, showing a table with the following data:

| Schema | Index | Part | Index columns | Clusterratio [%] | 1st key card. | Full key card. | Tree levels | Leaf pages | Index |
|--------|--------------------|------|---------------|------------------|---------------|----------------|-------------|------------|-------|
| BI9DB | /BIO/SDIVISION-0 | 0 | DIVISION | 100 | 11 | 11 | 2 | 1 | |
| BI9DB | /BIO/SDIVISION-001 | 0 | SID | 100 | 11 | 11 | 2 | 1 | |
| BI9DB | /BIO/SMATL_TYPE-0 | 0 | MATL_TYPE | 100 | 108 | 108 | 2 | 1 | |
| BI9DB | /BIO/SMATL_TYPE-00 | 0 | SID | 100 | 108 | 108 | 2 | 1 | |
| BI9DB | /BIO/SRPA_WGH1-0 | 0 | RPA_WGH1 | 39 | 8.336 | 8.336 | 2 | 82 | |
| BI9DB | /BIO/SRPA_WGH1-001 | 0 | SID | 100 | 8.336 | 8.336 | 2 | 23 | |
| BI9DB | /BIO/SRT_COLOR-0 | 0 | RT_COLOR | 30 | 8.317 | 8.317 | 3 | 129 | |
| BI9DB | /BIO/SRT_COLOR-001 | 0 | SID | 100 | 8.317 | 8.317 | 2 | 23 | |
| BI9DB | /BIO/SSALESORG-0 | 0 | SALESORG | 100 | 1.001 | 1.001 | 2 | 5 | |
| BI9DB | /BIO/SSALESORG-001 | 0 | SID | 100 | 1.001 | 1.001 | 2 | 3 | |
| BI9DB | /BIO/SVERSION-0 | 0 | VERSION | 100 | 101 | 101 | 2 | 1 | |
| BI9DB | /BIO/SVERSION-001 | 0 | SID | 100 | 101 | 101 | 2 | 1 | |
| BI9DB | /BIC/DZGMA_Z11-0 | 0 | DIMID | 100 | 10.001 | 10.001 | 2 | 28 | 10 |
| BI9DB | /BIC/DZGMA_Z11-010 | 0 | SID_ZCUSTOMER | 100 | 10.001 | 10.001 | 2 | 34 | 10 |
| BI9DB | /BIC/DZGMA_Z12-0 | 0 | DIMID | 100 | 10.001 | 10.001 | 2 | 28 | 10 |

Figure 8-41 Index Information

From the Index information tab you are able to navigate directly to the storage parameters (SE14) and to the data set space information (DB02). The DB2 installation parameters can be accessed directly from within the Explanation of SQL Access Path window.

The most important question is whether the execution plan seems to be good. But this is not easy, especially if the query consists of many tables. But because we know that an SAP BI query on a cube is represented by a snowflake schema, usually the best execution plan is the one that accesses the most filtering dimensions first, which means before the E- fact table access. So for query analysis it is important to know the cardinalities and filter factor of dimension branches. For getting this information more easily, SAP developed a new tool, the SAP Query Visualizer, that makes it very simple to get all the filtering information with only a few mouse clicks.

As a DB2 performance expert you would like to see all of the information from the DB2 plan table for the corresponding query. For this you have to click the button **View plan table** (Figure 8-37 on page 181). Figure 8-42 shows an example of the rows in the DB2 plan table for a given query. In this example, the optimizer does a star join (JOIN_TYPE = S) and creates a sparse index for the materialized dimension branch (ACCESSTYPE = T).

| DB2 Plan Table | | | | | | | | | |
|----------------|----------|--------|--------|------------------|-------|------------|-----------|--------------------|-----------|
| QBLOCKNO | APPLNAME | PLANNO | METHOD | TNAME | TABNO | ACCESSTYPE | MATCHCOLS | ACCESSNAME | INDEXONLY |
| 1 | | 1 | 0 | /BIC/DJOCUBET | 8 | T | | /BIC/DJOCUBET~0 | N |
| 1 | | 2 | 1 | DSN_DIM_TBLX(03) | 13 | T | | | N |
| 1 | | 3 | 1 | /BIC/DJOCUBE2 | 10 | T | | | N |
| 1 | | 4 | 1 | /BIC/EJOCUBE | 1 | I | | /BIC/EJOCUBE~0 | N |
| 1 | | 5 | 2 | DSN_DIM_TBLX(02) | 12 | R | | | N |
| 1 | | 6 | 1 | /BIC/DJOCUBEU | 5 | I | | /BIC/DJOCUBEU~0 | N |
| 1 | | 7 | 1 | /BIC/DJOCUBEP | 9 | I | | /BIC/DJOCUBEP~0 | N |
| 1 | | 8 | 3 | | | | | | N |
| 2 | | 1 | 0 | /BI0/0200000003 | 11 | R | | | N |
| 2 | | 2 | 1 | /BI0/XCUSTOMER | 3 | R | | | N |
| 2 | | 3 | 1 | /BIC/DJOCUBE3 | 2 | I | | /BIC/DJOCUBE3~010 | N |
| 2 | | 4 | 1 | /BI0/SACCNT_GRP | 4 | I | | /BI0/SACCNT_GRP~00 | N |
| 3 | | 1 | 0 | /BI0/XMATERIAL | 7 | I | | /BI0/XMATERIAL~0 | N |
| 3 | | 2 | 1 | /BIC/DJOCUBE1 | 6 | I | | /BIC/DJOCUBE1~010 | N |

| TNAME | SRTN_JOIN | SRTC_JOIN | SRTC_GRPBY | PREFETCH | ACCESS_DEG | PARA_MODE | CORR_NAME | PAGE_RNG | JOIN_TYPE |
|------------------|-----------|-----------|------------|----------|------------|-----------|-----------|----------|-----------|
| /BIC/DJOCUBET | N | N | N | | 0 | | DT | | S |
| DSN_DIM_TBLX(03) | Y | N | N | | 0 | | | | S |
| /BIC/DJOCUBE2 | Y | N | N | | 0 | | D2 | | S |
| /BIC/EJOCUBE | N | N | N | | 0 | | F | Y | S |
| DSN_DIM_TBLX(02) | Y | Y | N | S | 0 | | | | S |
| /BIC/DJOCUBEU | N | N | N | | 0 | | DU | | |
| /BIC/DJOCUBEP | N | N | N | | 0 | | DP | | |
| | N | N | Y | | 0 | | | | |
| /BI0/0200000003 | N | N | N | S | 0 | | H1 | | |
| /BI0/XCUSTOMER | N | N | N | S | 0 | | X1 | | |
| /BIC/DJOCUBE3 | N | Y | N | | 10 | C | D3 | | |
| /BI0/SACCNT_GRP | N | N | N | | 10 | C | S1 | | |
| /BI0/XMATERIAL | N | N | N | | 0 | | X3 | | |
| /BIC/DJOCUBE1 | N | N | N | | 0 | | D1 | | |

Figure 8-42 DB2 plan table

8.5.2 SAP Query Visualizer

In this section we discuss the SAP Query Visualizer

How you I call the SAP Query Visualizer

Execute the BI query with transaction RSRT and click the option, as shown in Figure 8-35 on page 179. Then, when the program stops by showing one of the generated SQL queries, click the button for the **SAP Query Visualizer**, as shown in Figure 8-43 on page 187.

The SAP query visualizer gives you a graphical view of the query and insight into the structure of the query. You are also able to calculate several counts and filter factors. By default, the cardinality of the involved database objects is calculated and the filter factors of the local conditions on these objects are displayed.

By clicking a link or by selecting the appropriate link from the Detail Tree of an object, the filter factors of the corresponding joins can be calculated by clicking the **Calculate** button in the detail table. These filter factors denote the percentage of the whole table occurring in the result set of the join.

The windows shown in Figure 8-43 and Figure 8-44 on page 188 consist of the query text, query block display (graphical display), query menu, detail tree, and detail menu.

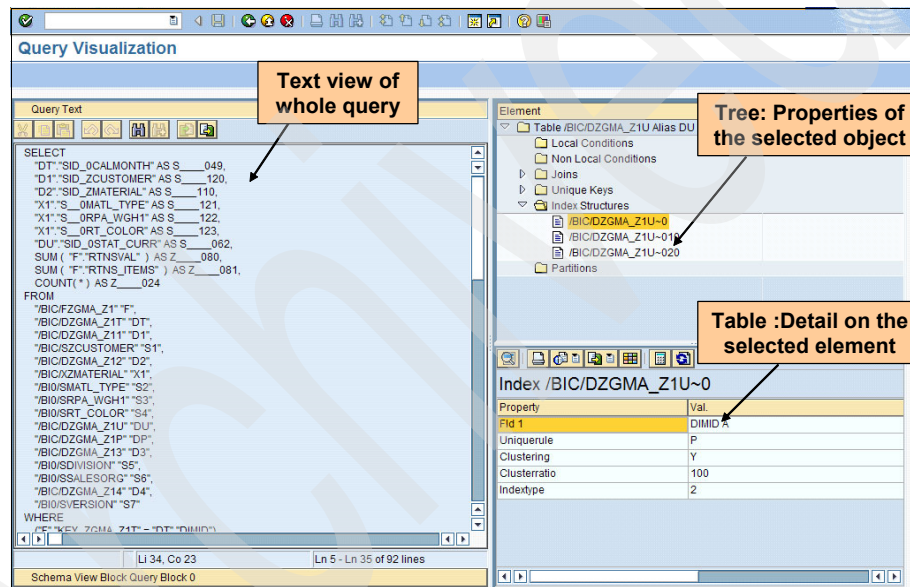


Figure 8-43 SAP Query Visualizer (window 1)

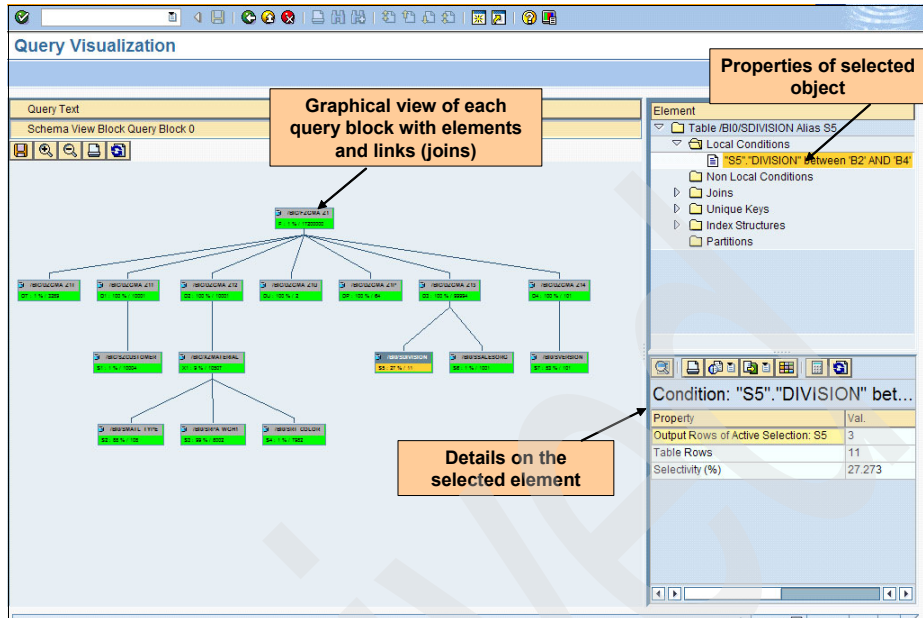


Figure 8-44 SAP Query Visualizer (window 2)

The graphical view is obtained by clicking **Schema View Block Query Block** (see Figure 8-45).

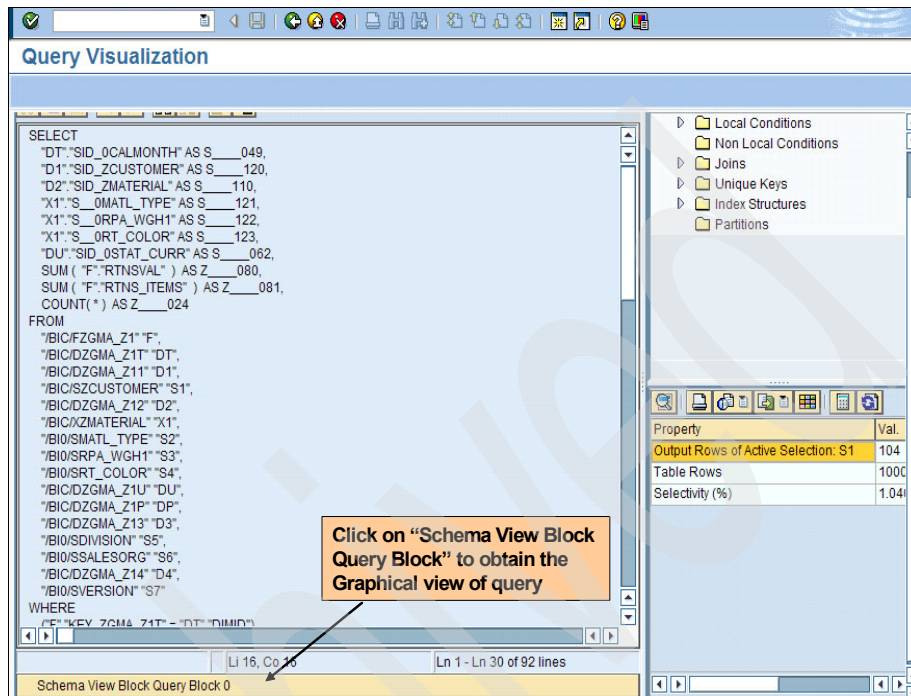


Figure 8-45 How to get the graphical view of the query

Now you see the graphical view of the query (see Figure 8-46).

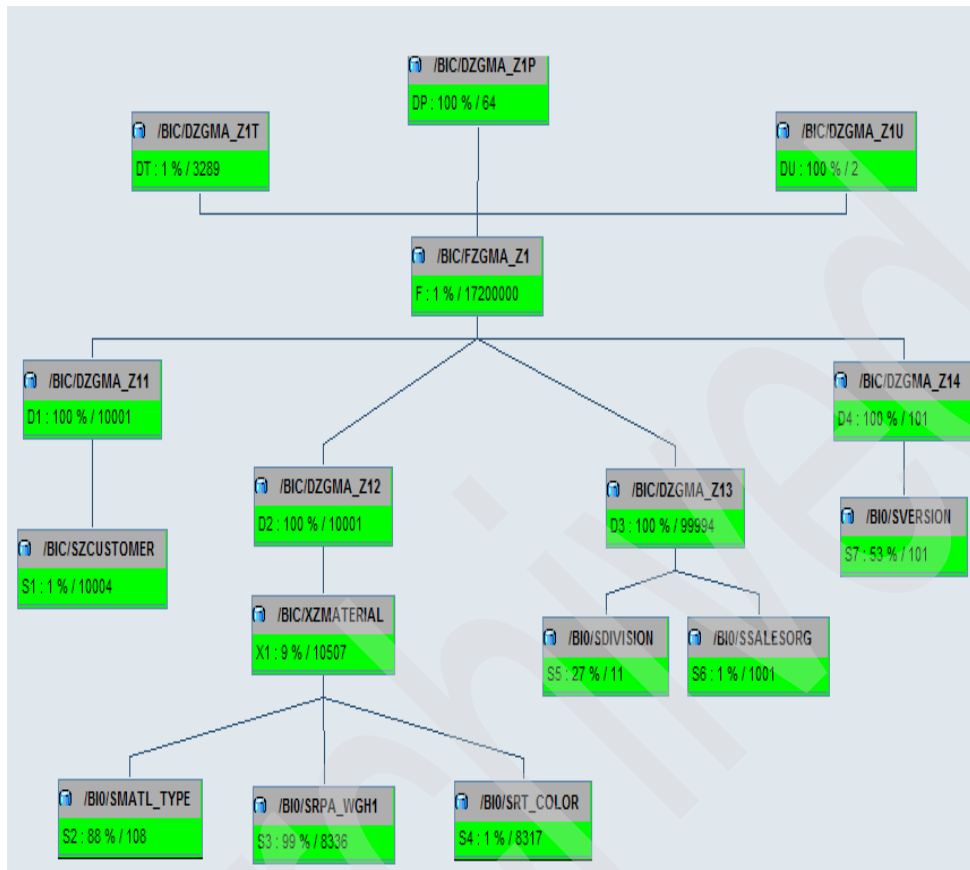


Figure 8-46 Graphical view of query

In the graphical view you are able to drag and drop the query blocks to the position where you would like so that you are able to view all dimension branches with the corresponding tables in a well organized manner in one window.

8.5.3 Features of SAP Query Visualizer

Some of the features of the SAP Query Visualizer include¹:

- ▶ The Text view displays the complete query (including subqueries and NTEs).
- ▶ The Graphical view displays individual query blocks.

¹ Dr. Matthias Gimbel, SAP AG, Boebligen, Germany

- ▶ Table cardinalities and selectivity of local predicates are calculated at startup.
- ▶ Join selectivities are calculated on demand in the background with an adjustable degree of parallelism.
- ▶ The fact alias flag separates dimension paths when calculating top-down selectivity.
- ▶ Correlation analysis is possible by selecting multiple links starting from selected Table Alias.

Note: SAP Note 935815 has a document attached to the note that details the functionality of the SAP Query Analysis Tool.

8.5.4 Example of using the query analysis tool

In Figure 8-47 when you select the query block by double-clicking it, the details of the query block are shown on the right-hand side. This displays the objects of the selected query block (table, local conditions, non local conditions, joins, index structures, unique keys, partitions, and so on).

In the example in Figure 8-47, the selectivity calculated on the query block we have selected indicates that on the table /BIO/SRT_COLOR 1.395% out of 8317 records were selected by the local conditions.

The screenshot displays the 'Schema View Block Query Block 0' interface. The main area shows a query block tree with several nodes, each representing a table and its associated conditions. The selected node is '/BIO/SRT_COLOR' with condition 'S4 : 1 % / 8317'. The right-hand side shows a detailed view of the selected table, including a table with properties and values.

| Property | Val. |
|-------------------------------------|-------|
| Output Rows of Active Selection: S4 | 116 |
| Table Rows | 8317 |
| Selectivity (%) | 1.395 |

Figure 8-47 Selecting query block for analysis

The link between the objects as indicated by the selected line in Figure 8-48 indicates the filter factors of the conditions. You are able to calculate the filter factor of this link by clicking the calculator icon on the right-hand side of the graphical view and then refresh to see the calculated filter factor. The filter factor for our example has been calculated and the link of the query block that we selected has a filter factor of 1% (this has been rounded off) of Join X1 and 0% of S4.

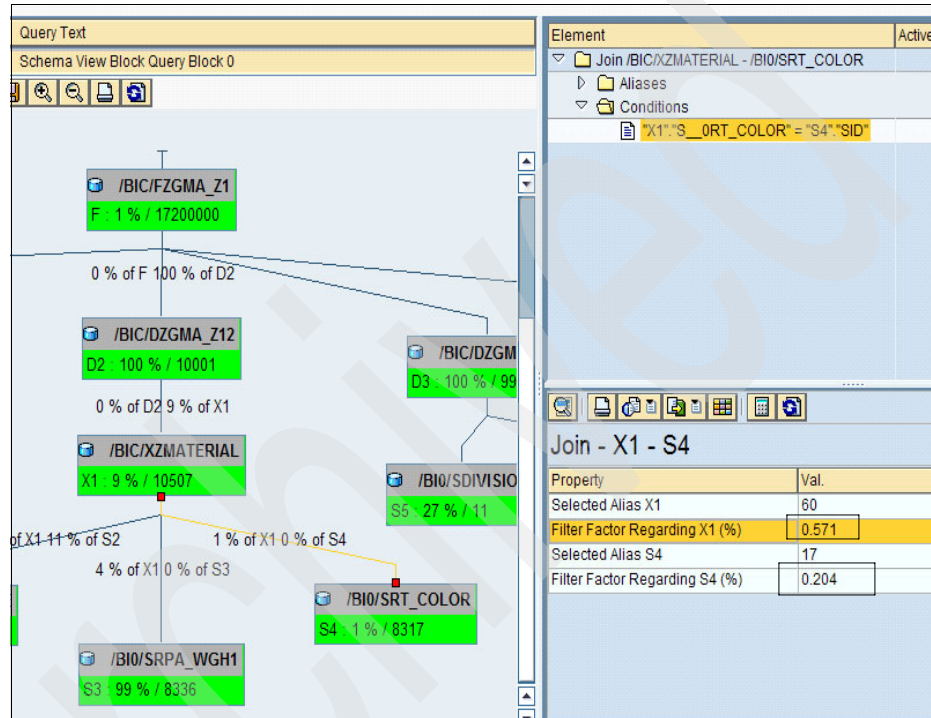


Figure 8-48 Link between objects

Archived

SAP BI Accelerator

This chapter provides an overview and introduction to the SAP BI Accelerator (SAP BIA), which is available from BI version 7.0 and later.

SAP BIA was introduced by SAP as High-Performance Analytics and is based on SAPs TREX search technology adapted for the BI needs.

The SAP BI Accelerator is a separate SAP instance. It is an appliance and runs on Blade servers with Linux SUSE on an Intel® Xeon using 64-bit architecture. This requires only the SAP BIA License, SUSE Linux and General Parallel File System™ (GPFS™). There is no database installation.

9.1 The BIA architecture

The BIA is based on the principles of cutting a big problem into smaller pieces. To do that it uses multiple techniques:

- ▶ Vertical decomposition
- ▶ Horizontal partitioning
- ▶ Smart compression
- ▶ Massive parallel execution

The architecture of BIA and its components are shown in Figure 9-1.

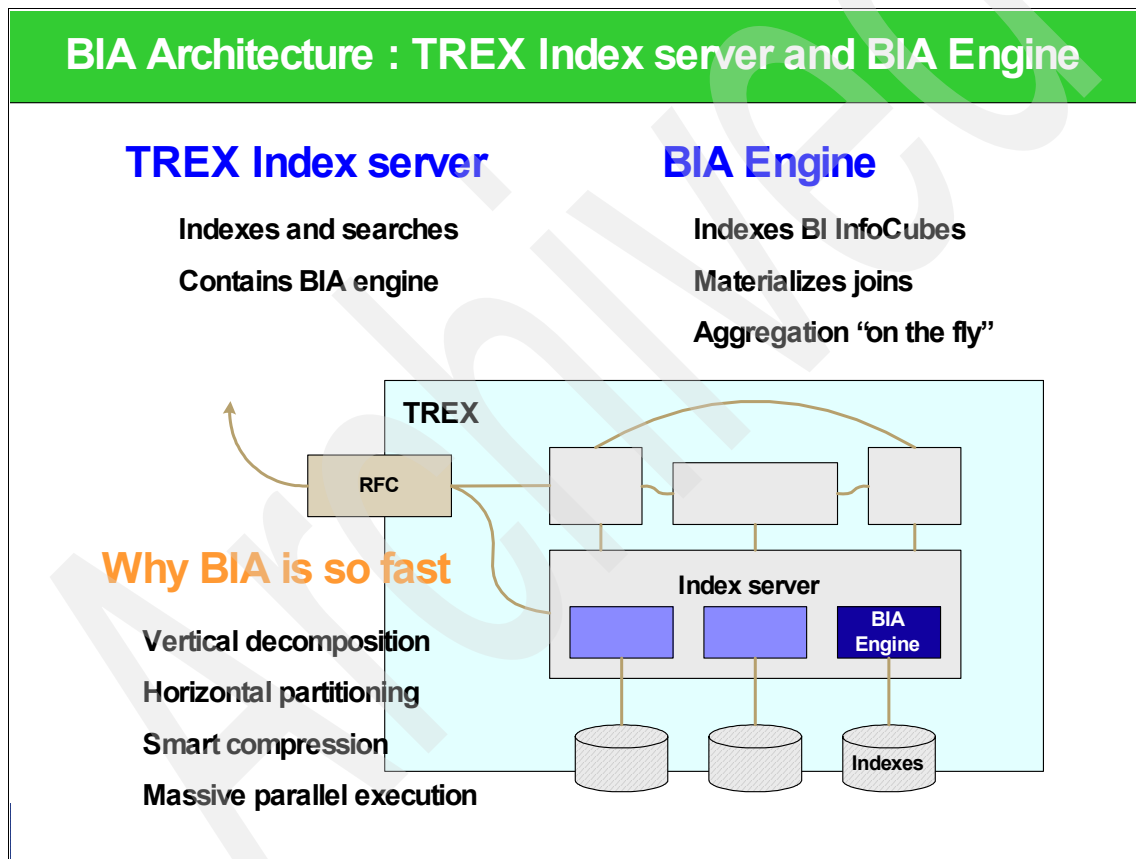


Figure 9-1 BIA architecture

Vertical decomposition

The BIA stores the data based on columns, a process called *vertical decomposition*. This increases the query's performance, minimizing the data

managed because only it retrieves the data for the selected columns, instead of the traditional approach that gets the whole row.

Horizontal partitioning

The sizing of the BIA, covered in 9.2, “Sizing BIA” on page 197, is based on the memory needs of the data to be managed by the BIA. Based on that is the number of Blades that can manage that memory. For each Blade’s CPU, it creates one partition. Each partition can hold one or a part of a InfoCube/dimension data, depending on its size.

Smart compression

The BIA uses dictionary-based compression allowing it to reduce the size of the data stored. It also allows it to manage all the data in memory. For this reason it is important do correct sizing. If some paging is involved, the BIA performance is going to be affected.

Massive parallel execution (MPE)

These techniques mentioned prepare the system for MPE. The queries and loads can be partitioned on small independent parts and run on the different partitions. Every step uses its own data on its own process and does not affect any other step. The number of parallel sessions that can run on a BIA system is one of the metrics of capacity sizing.

9.2 Sizing BIA

The purpose of the BIA is to improve the performance of queries on selected InfoCubes. The customer has to evaluate the potential InfoCubes selected for using BIA. SAP recommends not using the BIA for all InfoCubes. It should be used for large InfoCubes (thousands of millions of rows) to speed up query performance.

For the sizing process SAP Note 917803 explains how to calculate the memory consumption using report ZZ_BIAMEMCONSUMPTION_BW3X.

Note: We recommend using the option P_DETAIL to get more precise results for sizing.

To get the number of Blades needed for the BIA system you can use the following algorithm:

Result of report x 2 (GB) / amount of memory on Blade (GB) = number of Blades

The BIA system sizing is valid worldwide as of February 2007 and is based on the calculated BIA memory needs. We can see this in Figure 9-2.

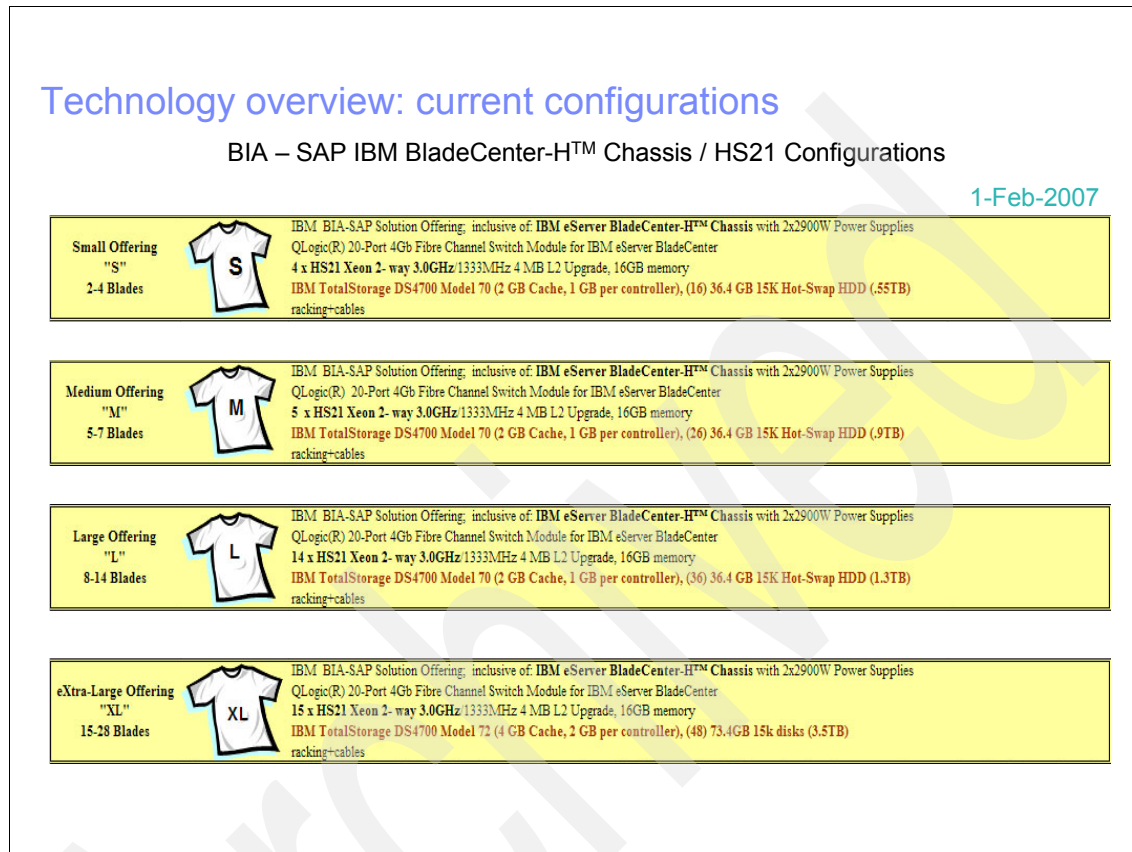


Figure 9-2 BIA t-shirt sizing

9.3 System z requirements

From the software side of a BIA implementation, you must have at least SAP NetWeaver 2004s with at least BI 7.0.

You can reach this target in different ways:

- ▶ New BI 7.0 fresh installation.
- ▶ From an upgraded or technical upgrade from SAP BW (2.x or 3.x) to NetWeaver 2004s.

This is actually a BI 7.0 installation, the same as scenario 1. For BIA it makes no difference whether it is a fresh installation or came from an upgrade.

- ▶ You can Data Mart from your existing SAP BW (2.x or 3.x) to a new SAP NetWeaver 2004s.

In this case, the queries are migrated and run on the BI 7.0 system. We can call this option a partial upgrade of the SAP BW (2.x or 3.x).

In either case, the queries are running from a NetWeaver 2004s environment, and only in scenario 3, the data does not reside in a BI 7.0 system. Based on that, our recommendation is that if you are not already on BI 7.0, at least do a technical upgrade to it. In case for some reason that is not possible, scenario 3 is technically possible to do, but involves more administration and infrastructure resources.

These scenarios are detailed in Figure 9-3.

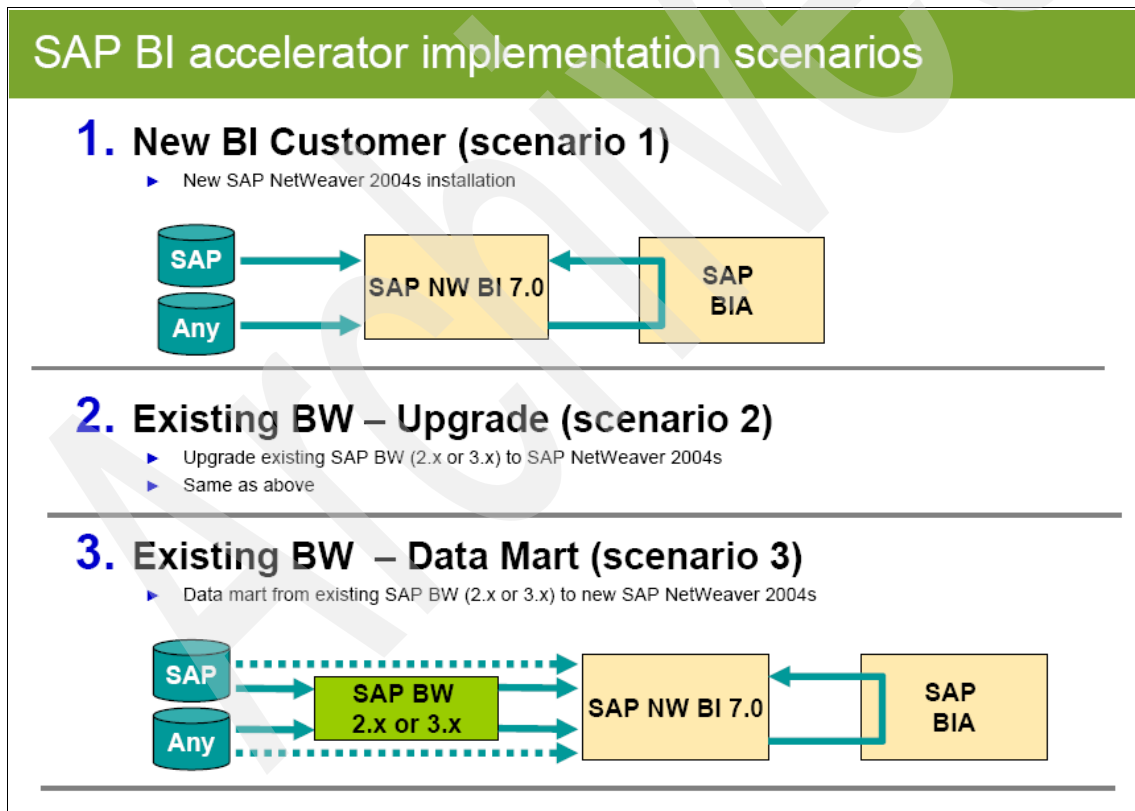


Figure 9-3 BIA implementation scenarios

From the hardware side, the BIA is an appliance. It came pre-installed and ready for configuration. From the System z side, the BIA has to be connected using a GigaBit Ethernet. You can check the network configuration by sending a 15,000-byte packet through the network connection and you must be able to receive a response in the order of a second.

An example of the configuration is shown in Figure 9-4.

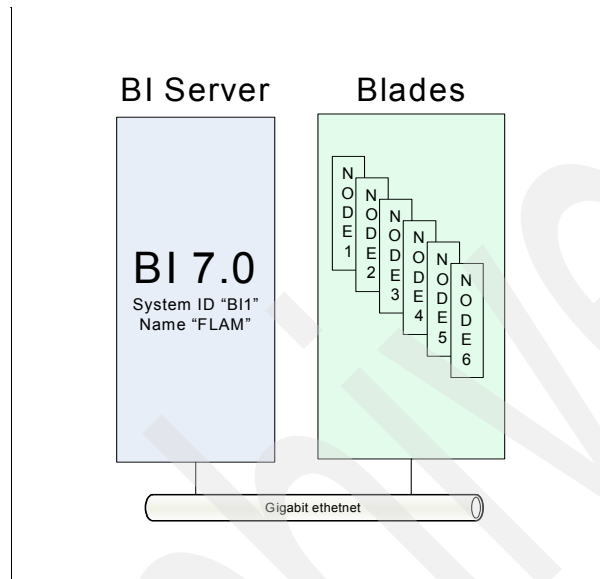


Figure 9-4 System z connectivity

9.4 BIA installation

The BIA appliance came pre-installed. It requires simple customizing steps regarding the BI system to it will be connected.

For the installation you need the following data:

- ▶ SAP system identification (SID) for new TREX instance
- ▶ New TREX option to create a new RFC connection on SAP BI system
- ▶ Instance number
- ▶ Total number of blades installed
- ▶ Confirmation of that use for host naming the <local_hostname+1>
- ▶ SAP BI SID
- ▶ SAP BI instance number
- ▶ SAP BI application server
- ▶ SAP BI client

- ▶ SAP BI user
- ▶ SAP BI password

You also must choose this data:

- ▶ Password for TREX administrator
- ▶ Name of the shared directory
- ▶ Root password for SUSE on blades

And you are asking to create a new RFC connection on an SAP BI system.

The installation itself must be done by an IBM specialist and consists of four steps:

1. Get parameters for installation.
2. Install BIA (TREX) on first blade.
3. Clone the installed BIA on the remaining blades.
4. Configure the RFC connection to the SAP BI system.

You can check the installation with the command “**checkBIA.sh**”. If the SAP BIA was stopped you can manage it with the TREX graphic admin tool running the shell “**TREXAdmin.sh**”. If it was already running you can manage it from inside SAP BI with the transaction “**TREXADMIN**”.

9.5 BIA administration

Administration of the BIA was integrated on the management workbench of the BI server. From the BIA side, there only needs to be maintenance and current updates on installed software: Linux SUSE, BIA, and GPFS software, and some HW maintenance: Blades, storage, and networking.

The impact on the administration comes on the BI side. The tuning of the query performance and fewer indexes and aggregates makes the administration of BI easier.

9.6 Benefits of BIA

BIA provides a different approach to improve query performance than using aggregates. BIA improves query performance on selected InfoCubes and also cuts administering tasks and the related resources.

The improvement of the performance has a variance between 5 and 200 times better. It depends on three E-Factors:

- ▶ How the InfoCubes are designed
- ▶ How queries are optimized
- ▶ Whether the main cause of the performance bottlenecks is generated in the access to the data

On a better designed InfoCube and optimized queries the results of running BIA are going to have less impact. The gains in this case are going to be the administration of BI and the predictability of BIA.

But when bad performance is not caused by data access, using BIA has no effect. For example, if a query uses complex formulae or exception aggregation of formulae and calculated key figures or formula variables with replacement from attributes and has a long response time, running it with a BIA will not help.

9.7 Overview

The BIA is a complementary appliance to help BI to improve performance queries for selected InfoCubes in a more predictable way, analyzing a very large amount of data (thousands of millions of rows) and gives a shorter and more timely response to the business.

The BIA also makes the administration in the BI server easier and gives freedom to the design of infoCubes.



Tips for SQL efficiency

This chapter provides tips that will help you enhance the efficiency of your SQL processing in both DSOs and InfoCubes. It also describes how to set up your indexes for those components.

10.1 SQL efficiency for DSO

Queries against the DSO can be long and complicated. They usually involve joins to one or more master data tables, as well as local predicates on the master data tables and DSO. Understanding and analyzing your queries is an important step in performance tuning.

In this section we illustrate a method that you can use to break down a query. This kind of analysis is helpful in determining the requirements for indexing, clustering, partitioning, and statistics collection. Without first breaking down and simplifying the query, it is not possible to adequately understand these requirements.

Figure 10-1 shows the query that we analyze in this section. It is a real-life customer query.

ODS Query Example

```
SELECT "S" _065, "S" _028, "S" _097, "S" _109, "S" _179, "S" _180, "0ADVERTISNG", "0G_APRABA", "0G_ASTDPR", "0G_AVTRGK",
"0G_AVVCD", "0G_AVVCD", "0G_AVVCHG", "0G_AVVCTG", "0G_AVVDED", "0G_AVVFGT", "0G_AVVFIX", "0G_AVVIN", "0G_AVVIVS", "0G_AVVLST",
"0G_AVVMS", "0G_AVVOR", "0G_AVVRET", "0G_AVVROY", "0G_AVVSRB", "0G_AVVVLB", "0G_AVVVOT", "0G_AVVVRM", "0G_AVVWHG", "0G_QABSMG",
"0G_QVVTN", "0G_QVVUOM", "0SALES_CMSN", "0SALES_MARK", "1ROWCOUNT", "ZCOGS", "ZEXTREV", "ZFRD_D", "ZGRMRGN", "ZGRMRGN_F",
"ZGROSSSL", "ZG_AVVFTC", "ZINTERDV", "ZINTRADV", "ZNETSALES", "ZNETSL_F", "ZSLSD", "ZTODD", "ZT_OTD", "ZT_REBATE"
FROM ( SELECT "DB2A0001"."SID" AS "S" _065, "BIO/SCURRENCY"."SID" AS "S" _028, "A0001"."SID" AS "S" _097, "A0002"."SID" AS "S" _109
,"BIO/SREFER_DOC"."SID" AS "S" _179, "BIO/SREFER_ITM"."SID" AS "S" _180, SUM ( "BIO/AZOSASALE00"."ADVERTISNG" ) AS "0ADVERTISNG", SUM (
,"BIO/AZOSASALE00"."G_APRABA" ) AS "0G_APRABA", SUM ( "BIO/AZOSASALE00"."G_ASTDPR" ) AS "0G_ASTDPR", SUM (
,"BIO/AZOSASALE00"."G_AVTRGK" ) AS "0G_AVTRGK", SUM ( "BIO/AZOSASALE00"."G_AVVCD" ) AS "0G_AVVCD", SUM (
,"BIO/AZOSASALE00"."G_AVVCHG" ) AS "0G_AVVCHG", SUM ( "BIO/AZOSASALE00"."G_AVVCTG" ) AS "0G_AVVCTG", SUM (
,"BIO/AZOSASALE00"."G_AVVDED" ) AS "0G_AVVDED", SUM (
,"BIO/AZOSASALE00"."G_AVVFGT" ) AS "0G_AVVFGT", SUM ( "BIO/AZOSASALE00"."G_AVVFIX" ) AS "0G_AVVFIX", SUM ( "BIO/AZOSASALE00"."G_AVVIN" )
AS "0G_AVVIN", SUM ( "BIO/AZOSASALE00"."G_AVVIVS" ) AS "0G_AVVIVS", SUM ( "BIO/AZOSASALE00"."G_AVVLST" ) AS "0G_AVVLST", SUM (
,"BIO/AZOSASALE00"."G_AVVMS" ) AS "0G_AVVMS", SUM ( "BIO/AZOSASALE00"."G_AVVOR" ) AS "0G_AVVOR", SUM (
,"BIO/AZOSASALE00"."G_AVVRET" ) AS "0G_AVVRET", SUM ( "BIO/AZOSASALE00"."G_AVVROY" ) AS "0G_AVVROY", SUM (
,"BIO/AZOSASALE00"."G_AVVSRB" ) AS "0G_AVVSRB", SUM ( "BIO/AZOSASALE00"."G_AVVVLB" ) AS "0G_AVVVLB", SUM ( "BIO/AZOSASALE00"."G_AVVVOT"
) AS "0G_AVVVOT", SUM ( "BIO/AZOSASALE00"."G_AVVVRM" ) AS "0G_AVVVRM", SUM ( "BIO/AZOSASALE00"."G_AVVWHG" ) AS "0G_AVVWHG", SUM (
,"BIO/AZOSASALE00"."G_QABSMG" ) AS "0G_QABSMG", SUM ( "BIO/AZOSASALE00"."G_QVVTN" ) AS "0G_QVVTN", SUM (
,"BIO/AZOSASALE00"."G_QVVUOM" ) AS "0G_QVVUOM", SUM ( "BIO/AZOSASALE00"."SALES_CMSN" ) AS "0SALES_CMSN", SUM (
,"BIO/AZOSASALE00"."SALES_MARK" ) AS "0SALES_MARK", COUNT(*) AS "1ROWCOUNT"
FROM "BIO/PDIVISION"."BIO/SCO_AREA", "BIO/SCURRENCY", "BIO/SREFER_DOC", "BIO/SREFER_ITM", "BIO/SUNIT"."A0002", "BIO/SUNIT"
"A0001", "BIO/SUNIT"."DB2A0001", "BIO/AZOSASALE00", "BIO/XZMATPLNT", "BIO/XZRLNSP"
WHERE "BIO/AZOSASALE00"."CURRENCY" = "BIO/SCURRENCY"."CURRENCY" AND "BIO/AZOSASALE00"."G_UVVUOM" = "A0001"."UNIT" AND
" "BIO/AZOSASALE00"."G_UABSMG" = "A0002"."UNIT" AND "BIO/AZOSASALE00"."REFER_DOC" = "BIO/SREFER_DOC"."REFER_DOC" AND
" "BIO/AZOSASALE00"."REFER_ITM" = "BIO/SREFER_ITM"."REFER_ITM" AND "BIO/AZOSASALE00"."CO_AREA" = "BIO/SCO_AREA"."CO_AREA" AND
" "BIO/AZOSASALE00"."PROFIT_CTR" = "BIO/SREFER_ITM"."PROFIT_CTR" AND "BIO/AZOSASALE00"."CO_AREA" = "BIO/SREFER_ITM"."CO_AREA" AND
" "BIO/AZOSASALE00"."BICZA_SADIV" = "BIO/PDIVISION"."DIVISION" AND "BIO/AZOSASALE00"."BICZMATPLNT" = "BIO/XZMATPLNT"."BICZMATPLNT" AND
" "BIO/AZOSASALE00"."PLANT" = "BIO/XZMATPLNT"."PLANT" AND "BIO/AZOSASALE00"."BICZRLNSP" = "BIO/XZRLNSP"."BICZRLNSP" AND
" "BIO/AZOSASALE00"."G_UVVTON" = "DB2A0001"."UNIT" AND "BIO/PDIVISION"."OBJVERS" = 'A' AND "BIO/XZMATPLNT"."OBJVERS" = 'A' AND
" "BIO/XZRLNSP"."OBJVERS" = 'A' AND ( ( "BIO/AZOSASALE00"."ACCNT_ASGN" = '00' OR "BIO/AZOSASALE00"."ACCNT_ASGN" = '01' OR
" "BIO/AZOSASALE00"."ACCNT_ASGN" = '04' OR "BIO/AZOSASALE00"."ACCNT_ASGN" = '06' ) AND ( "BIO/AZOSASALE00"."CALMONTH" BETWEEN '200311' AND
'200404' ) AND ( ( "BIO/SCO_AREA"."SID" = 3 ) AND ( ( "BIO/AZOSASALE00"."CURTYPE" = 'B0' ) AND ( "BIO/SREFER_ITM"."SID" = 31 ) ) AND ( (
" "BIO/PDIVISION"."BICZREPTDIV" = 5X' ) AND ( ( "BIO/XZMATPLNT"."S_ZMATLONLY" = 360013 ) ) AND ( ( "BIO/XZRLNSP"."S_ZA_SASGRP" = 80 ) ) )
GROUP BY "DB2A0001"."SID", "BIO/SCURRENCY"."SID", "A0001"."SID", "A0002"."SID", "BIO/SREFER_DOC"."SID", "BIO/SREFER_ITM"."SID", "BIO/03000411419"
FOR FETCH ONLY WITH UR
```

Figure 10-1 Example customer query against an ODS

To break down a query, follow these steps:

1. Build a join graph.

For this process, we can ignore the SELECT list, FROM clause, and GROUP BY clause, because we are comfortable that SAP has generated a valid SQL statement. Therefore, we focus on the WHERE clause, which shows the table join relationships and filtering predicates.

Within the WHERE clause, we first focus on the join predicates to build the join graph. See Figure 10-2. We use the local predicates in a later step to determine where the filtering occurs.

Step 1 - Join Analysis

```

WHERE "/BIC/AZOSASALE00"."CURRENCY" = "/BI0/SCURRENCY"."CURRENCY"
AND "/BIC/AZOSASALE00"."G_UVVUOM" = "A0001"."UNIT"
AND "/BIC/AZOSASALE00"."G_UABSMG" = "A0002"."UNIT"
AND "/BIC/AZOSASALE00"."REFER_DOC" = "/BI0/SREFER_DOC"."REFER_DOC"
AND "/BIC/AZOSASALE00"."REFER_ITM" = "/BI0/SREFER_ITM"."REFER_ITM"
AND "/BIC/AZOSASALE00"."CO_AREA" = "/BI0/SCO_AREA"."CO_AREA"
AND "/BIC/AZOSASALE00"."PROFIT_CTR" = "/BI0/SPROFIT_CTR"."PROFIT_CTR"
AND "/BIC/AZOSASALE00"."CO_AREA" = "/BI0/SPROFIT_CTR"."CO_AREA"
AND "/BIC/AZOSASALE00"."BIC/ZA_SADIV" = "/BI0/PDIVISION"."DIVISION"
AND "/BIC/AZOSASALE00"."BIC/ZMATLPLNT" = "/BIC/XZMATLPLNT"."BIC/ZMATLPLNT"
AND "/BIC/AZOSASALE00"."PLANT" = "/BIC/XZMATLPLNT"."PLANT"
AND "/BIC/AZOSASALE00"."BIC/ZRLNSP" = "/BIC/XZRLNSP"."BIC/ZRLNSP"
AND "/BIC/AZOSASALE00"."G_UVVTON" = "DB2A0001"."UNIT"
AND "/BI0/PDIVISION"."OBJVERS" = 'A'
AND "/BIC/XZMATLPLNT"."OBJVERS" = 'A' AND "/BIC/XZRLNSP"."OBJVERS" = 'A'
AND ((( "/BIC/AZOSASALE00"."ACCNT_ASGN" = '00' OR "/BIC/AZOSASALE00"."ACCNT_ASGN" = '01' OR
"/BIC/AZOSASALE00"."ACCNT_ASGN" = '04' OR "/BIC/AZOSASALE00"."ACCNT_ASGN" = '06' ))
AND (( "/BIC/AZOSASALE00"."CALMONTH" BETWEEN '200311' AND '200404' ))
AND (( "/BI0/SCO_AREA"."SID" = 3 ))
AND (( "/BIC/AZOSASALE00"."CURTYPE" = 'B0' ))
AND (( "/BI0/SPROFIT_CTR"."SID" = 31 ))
AND (( "/BI0/PDIVISION"."BIC/ZREPTDIV" = '5X' ))
AND (( "/BIC/XZMATLPLNT"."S__ZMATLONLY" = 360013 ))
AND (( "/BIC/XZRLNSP"."S__ZA_SASGRP" = 80 )))

```

Focus on Join Predicates

Defer Local Predicates to later

Figure 10-2 Step 1 - join analysis

The process of building the join graph involves taking each join predicate and mapping the table join relationship.

The first join predicate is "/BIC/AZOSASALE00"."CURRENCY" = "/BI0/SCURRENCY"."CURRENCY", which maps to a join relationship between the tables "/BIC/AZOSASALE00" and "/BI0/SCURRENCY". See Figure 10-3.

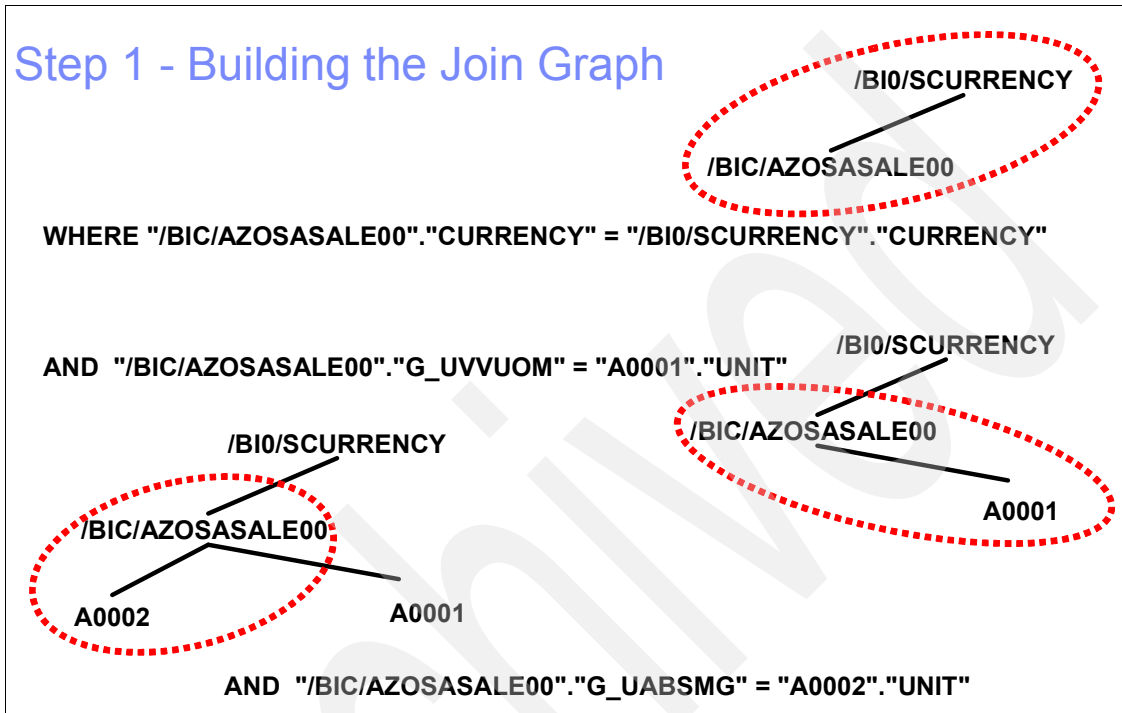


Figure 10-3 Building the join graph

The second predicate joins "/BIC/AZOSASALE00" with "A0001". This relationship is added to the first join relationship. The third is then added, and the join graph continues to grow until all join relationships are added. See Figure 10-4.

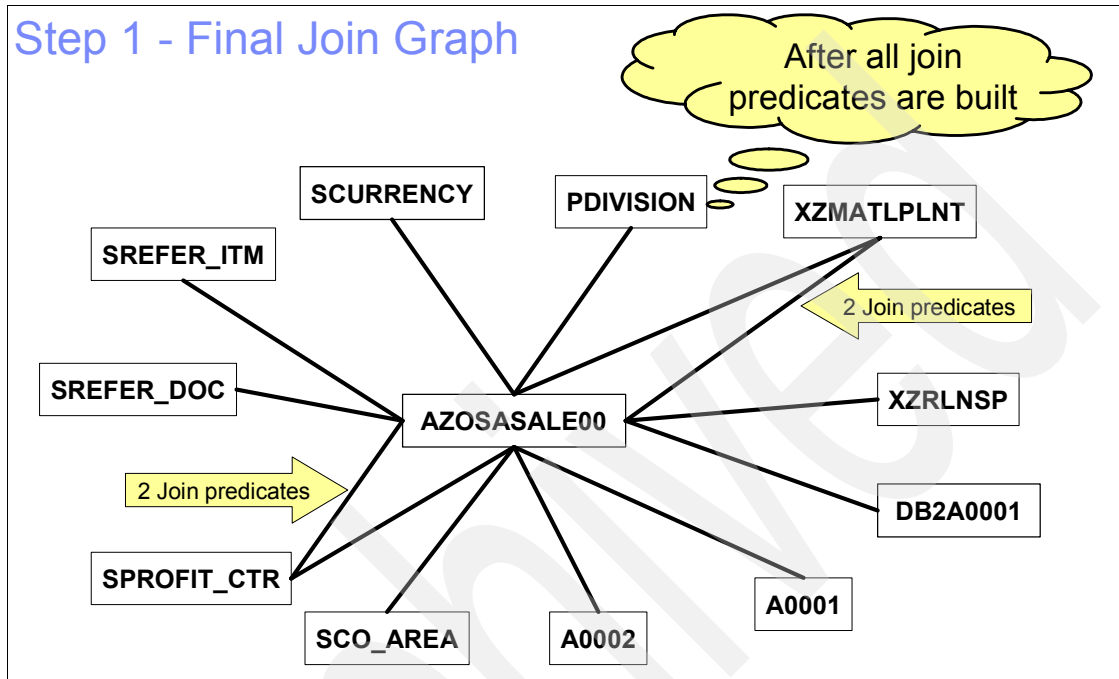


Figure 10-4 Final join graph

2. Map each local predicate in the query to the table that it refers to on the join graph.

After all table join relationships are represented, map each local predicate in the query to the table that it refers to on the graph. Refer to Figure 10-5.

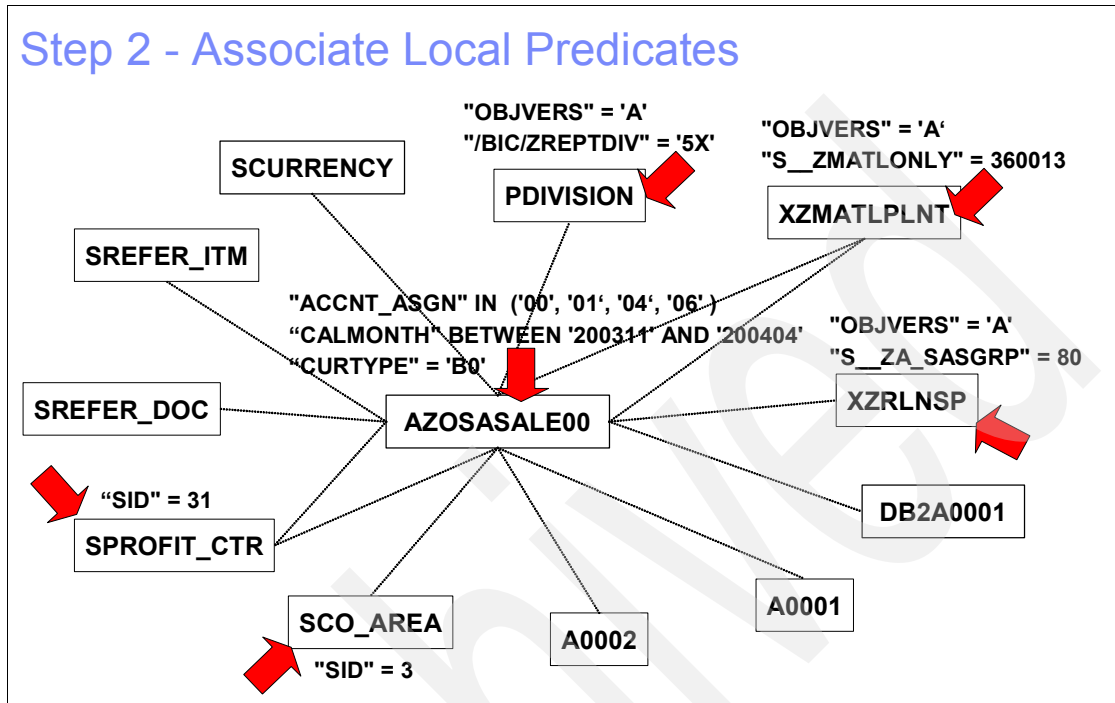


Figure 10-5 Associate local predicates

What we have done is taken the original complex query and broken it down into an easily understandable diagram that depicts the table join relationships and filtering predicates. This diagram can be used in conjunction with the original SQL for further detailed analysis.

- Determine where the filtering is occurring for this SQL statement. Currently, it is not clear which predicates provide the greatest degree of filtering.

This step involves breaking apart the original query to count the number of rows retrieved for each master data table joined to the central ODS table. Include the local predicates for the tables in the join on the count SQL statement. Figure 10-6 shows the SQL count statement and the returned count for the PDIVISION/AZOSASALE00 join relationship.

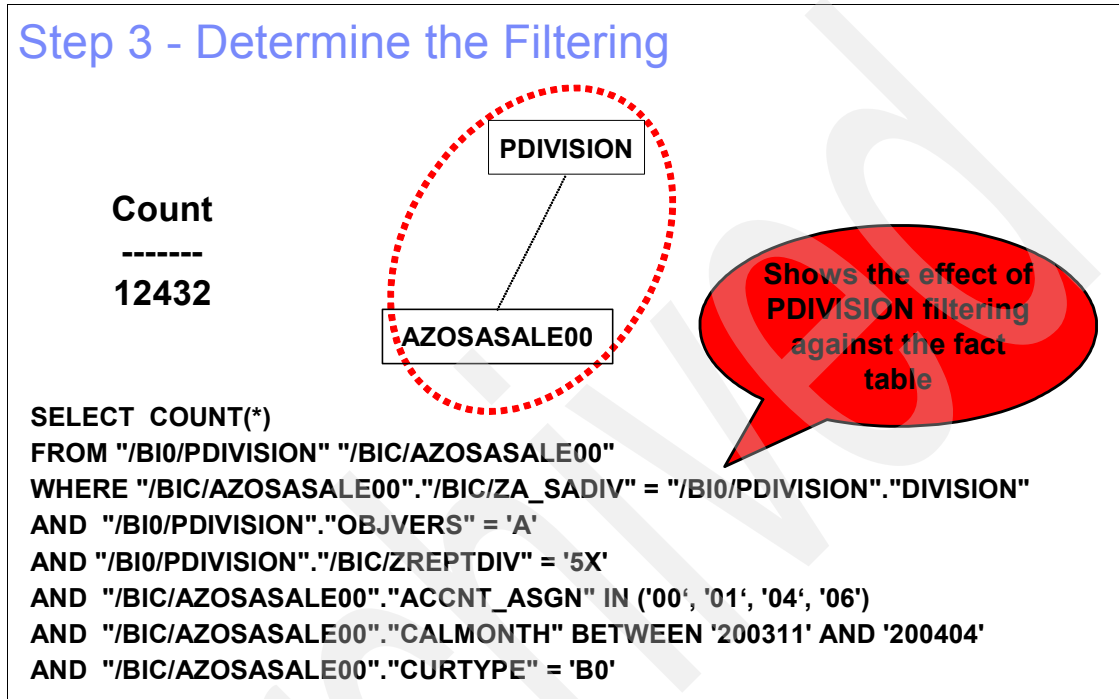


Figure 10-6 Determine the filtering

This count demonstrates that the combined filtering of local predicates on the PDIVISION and AZOSASALE00 tables return 12,432 rows. This is considered considerable filtering, given that the original DSO table has 7 million rows.

Step 3 is repeated for each master data table. A count should also be returned for the DSO table without any joins (and local predicates only), and finally a count of the total filtering from the entire query. These counts will be used to compare all filtering combinations. See Figure 10-7 for these counts.

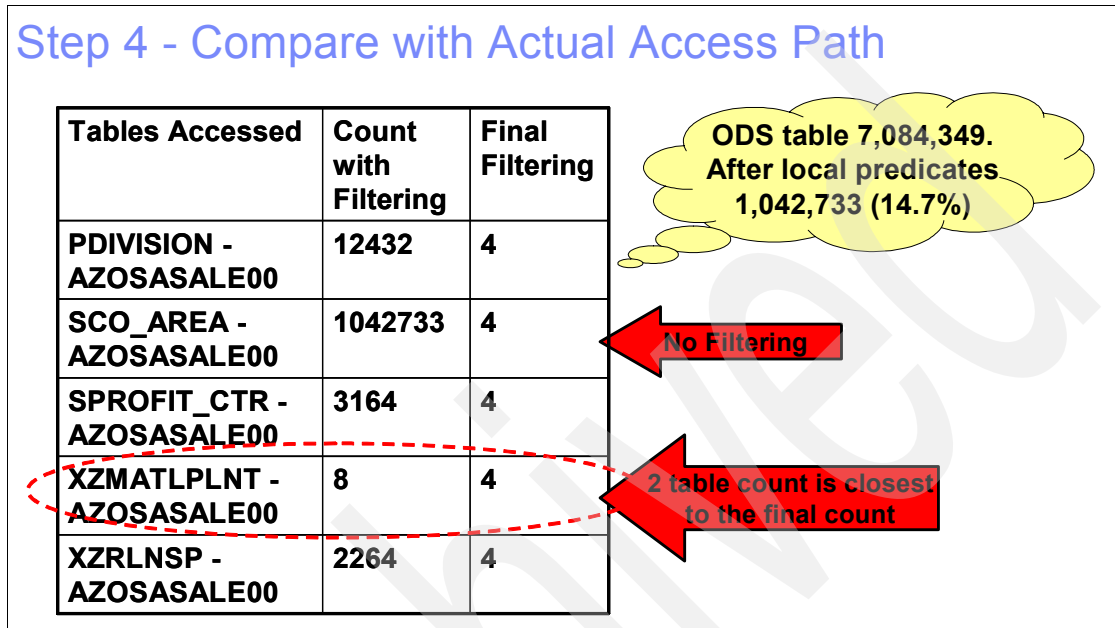


Figure 10-7 Compare with actual access path

The original DSO table contains 7,084,349 rows. After local predicates are applied, the result is 1,042,733 rows, and the final count for the query including all filtering is four rows.

4. Compare the filtering provided by each master data table joined to the DSO.

In this example we can see that SCO_AREA provides no filtering (because the count equals the count of local DSO predicates only), but all other master data tables provide strong filtering. It is clear, however, that XZMATLPLNT joined to AZOSASALE00 provides the strongest filtering (8 rows returned, when the entire query only returns 4 rows).

Any indexing, clustering, and partitioning decisions should take into consideration the patterns of all queries against the DSO (if possible), although — for this example only — the decisions should be made to support a table join sequence of table XZMATLPLNT accessed first, and joined to AZOSASALE00.

10.2 SQL efficiency for InfoCube

Now let us take a look at SQL efficiency for InfoCube. We recommend a method to simplify the analysis of a query against an InfoCube. An example of using this method follows a general description.

1. Analyze the joins and build a join graph.
 - a. Ignore the SELECT list. We assume that all the columns that are selected are required. Note that we do not have much choice on this if the query is *generated SQL*. If it is user-written SQL, then double-check the **SELECT** list.
 - b. Ignore the FROM clause. We assume that all the tables that are listed are required. Note that we do not have much choice on this if the query is generated SQL. If it is user-written SQL, then double-check the **FROM** list.
 - c. Ignore the GROUP BY clause. The columns in the GROUP BY clause must be equal to or a superset of the columns in the SELECT list. Any column in the SELECT list that is not contained within a column function must be in the GROUP BY clause. Note that all the columns in the GROUP BY clause do not need to be in the SELECT list.
 - d. Focus on the WHERE clause.
 - i. For each join predicate, map the table join relationship.
 - ii. Ignore the local predicates for now.
 - iii. Check that all the tables in the FROM clause are accounted for in the join graph. This verifies that each table has at least one join predicate.
2. Associate local predicates to their table by mapping each local predicate to the appropriate table on the join graph.
3. Determine the filtering.

Count the number of rows retrieved for each master data table, then each master data joined to its corresponding dimension, and finally joined to the facttable. Include the local predicates on the tables. This shows the effect of filtering of the dimension/snowflake against the facttable.
4. Compare the filtering of the each of dimensions/snowflakes joined to the facttable. This data should show where the access path should start for the query and provide a method to compare with the actual access path seen for the query.

Let us look at an example of using this process to analyze a real-life customer query.

Using the InfoCube query example in Figure 10-8, we use the described method to break down the query and determine the requirements for indexing and statistics collection to ensure that the optimal access path is chosen. Without first breaking down and simplifying the query, it is not possible to adequately understand these requirements.

Infocube Query Example

```
SELECT "S___637", "S___566", "S___581", "S___561", "S___565", "S___574", "1ROWCOUNT", "Z_BQ", "Z_CNFCST",
"Z_COMMQ", "Z_FCBP", "Z_FCMK", "Z_FCSLN", "Z_NSQ", "Z_RBQ" FROM ( SELECT "D5"."SID_Z_DOCTYP" AS "S___637"
,"X1"."S___Z_PROTIND" AS "S___566", "D6"."SID_Z_FACTWK" AS "S___581", "X2"."S___Z_BTGRPNO" AS
"S___561", "X1"."S___Z_MODEL" AS "S___565", "X1"."S___Z_MAJCAT" AS "S___574", COUNT( * ) AS "1ROWCOUNT", SUM (
"F"."/BIC/Z_BQ" ) AS "Z_BQ", SUM ( "F"."/BIC/Z_CNFCST" ) AS "Z_CNFCST", SUM ( "F"."/BIC/Z_COMMQ" ) AS "Z_COMMQ", SUM (
"F"."/BIC/Z_FCBP" ) AS "Z_FCBP", SUM ( "F"."/BIC/Z_FCMK" ) AS "Z_FCMK", SUM ( "F"."/BIC/Z_FCSLN" ) AS "Z_FCSLN", SUM (
"F"."/BIC/Z_NSQ" ) AS "Z_NSQ", SUM ( "F"."/BIC/Z_RBQ" ) AS "Z_RBQ"
FROM "/BIC/FZIC_SFC" "F", "/BIC/DZIC_SFC5" "D5", "/BIC/DZIC_SFC3" "D3", "/BIC/XZ_PRODCD" "X1", "/BIC/DZIC_SFC6" "D6"
,"/BIC/DZIC_SFC1" "D1", "/BIC/XZ_SHIPTO" "X2", "/BIC/DZIC_SFCP" "DP", "/BI0/0600000005" "Z1"
WHERE "F"."KEY_ZIC_SFC5" = "D5"."DIMID"
AND "F"."KEY_ZIC_SFC3" = "D3"."DIMID"
AND "D3"."SID_Z_PRODCD" = "X1"."SID"
AND "F"."KEY_ZIC_SFC6" = "D6"."DIMID"
AND "F"."KEY_ZIC_SFC1" = "D1"."DIMID"
AND "D1"."SID_Z_SHIPTO" = "X2"."SID"
AND "F"."KEY_ZIC_SFCP" = "DP"."DIMID"
AND "X2"."S___Z_BTGRPNO" = "Z1"."SID"
AND ((( "DP"."SID_0RECORDTP" = 0 )) AND ( ( "DP"."SID_0REQUID" <=76419 ))
AND ( ( "D5"."SID_Z_DOCTYP" <> 2000008999 ) AND NOT ( "D5"."SID_Z_DOCTYP" = 6 ))
AND ( ( "D6"."SID_Z_FACTWK" BETWEEN 200327AND 200352 ))
AND ( ( "X1"."S___Z_MAJCAT" IN (103, 119, 150, 98 )))
AND ( ( "X1"."S___Z_PROTIND" <> 2000008999 ) AND NOT ( "X1"."S___Z_PROTIND" = 1))))
AND "X1"."OBJVERS" = 'A' AND "X2"."OBJVERS" = 'A' AND "Z1"."SID" <> 2000008999
GROUP BY "D5"."SID_Z_DOCTYP", "X1"."S___Z_PROTIND", "D6"."SID_Z_FACTWK", "X2"."S___Z_BTGRPNO", "X1"."S___Z_MODEL",
"X1"."S___Z_MAJCAT" ) "/BI0/0300422840" FOR FETCH ONLY WITH UR
```

Figure 10-8 Example InfoCube query

As listed, step 1 is to build the join graph. For this process, we can ignore the SELECT list, FROM clause, and GROUP BY clause, because we can be comfortable that SAP has generated a valid SQL statement. Therefore we focus on the WHERE clause, which shows the table join relationships and filtering predicates.

Within the WHERE clause, we first focus on the join predicates to build the join graph, and use the local predicates in a later step to determine where the filtering occurs. See Figure 10-9.

Step 1 - Join Analysis

```
WHERE "F"."KEY_ZIC_SFC5" = "D5"."DIMID"
AND "F"."KEY_ZIC_SFC3" = "D3"."DIMID"
AND "D3"."SID_Z_PROD CD" = "X1"."SID"
AND "F"."KEY_ZIC_SFC6" = "D6"."DIMID"
AND "F"."KEY_ZIC_SFC1" = "D1"."DIMID"
AND "D1"."SID_Z_SHIPTO" = "X2"."SID"
AND "F"."KEY_ZIC_SFCP" = "DP"."DIMID"
AND "X2"."S_Z_BTGRPNO" = "Z1"."SID"
AND ((( "DP"."SID_ORECORDTP" = 0 ))
AND (( "DP"."SID_OREQUID" <=76419 ))
AND (( "D5"."SID_Z_DOCTYP" <> 2000008999 ))
AND NOT ("D5"."SID_Z_DOCTYP" = 6 ))
AND (( "D6"."SID_Z_FACTWK" BETWEEN 200327AND 200352 ))
AND (("X1"."S_Z_MAJCAT" IN (103, 119, 150, 98 )))
AND (("X1"."S_Z_PROTIND" <> 2000008999 ))
AND NOT ( "X1"."S_Z_PROTIND" = 1))))
AND "X1"."OBJVERS" = 'A'
AND "X2"."OBJVERS" = 'A'
AND "Z1"."SID" <> 2000008999
```

Focus on Join Predicates

Defer Local Predicates to later

Figure 10-9 Deciding on the join graph - focusing on the predicates

The process of building the join graph involves taking each join predicate and mapping the table join relationship.

The first join predicate in Figure 10-10 is "F". "KEY_ZIC_SFC5" = "D5"."DIMID". This maps to a join relationship between F ("/BIC/FZIC_SFC") and D5 ("/BIC/DZIC_SFC5") tables. This relationship is represented in the diagram.

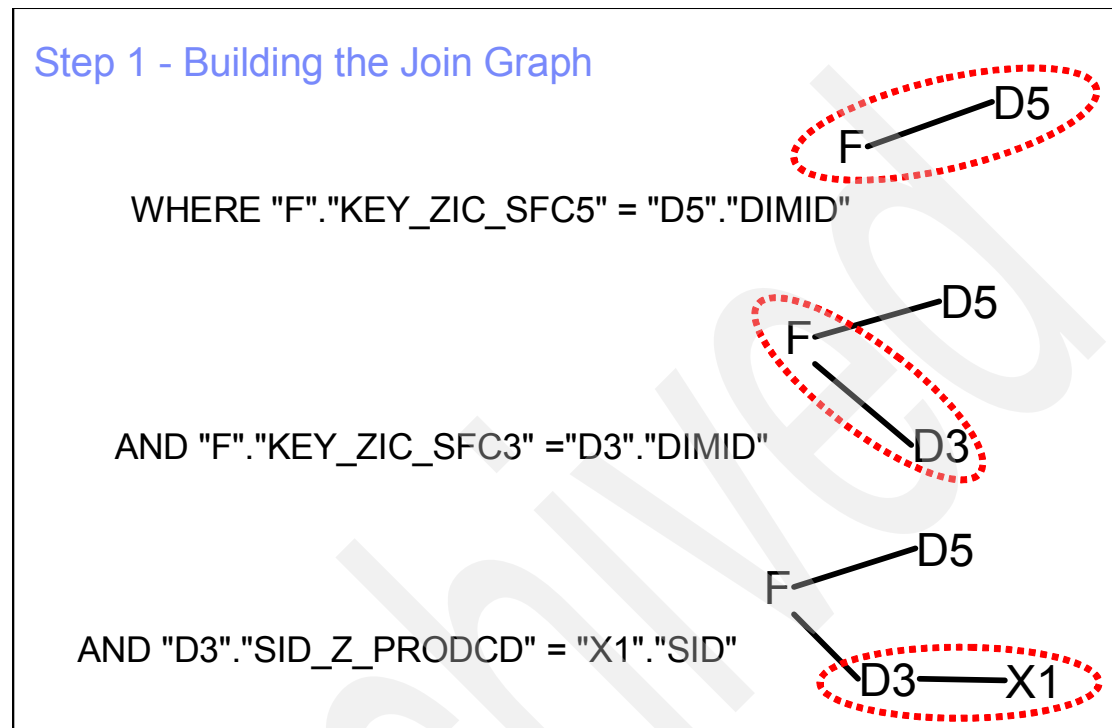


Figure 10-10 Building the join graph

The second predicate joins F with D3. The relationship is added to the first join relationship. The third is then added, and the join graph continues to grow until all join relationships are added.

With all table join relationships represented, the next step is to determine where the filtering occurs for this query. The initial step in this process is to take each local predicate to the table that it refers to on the join graph. See Figure 10-11.

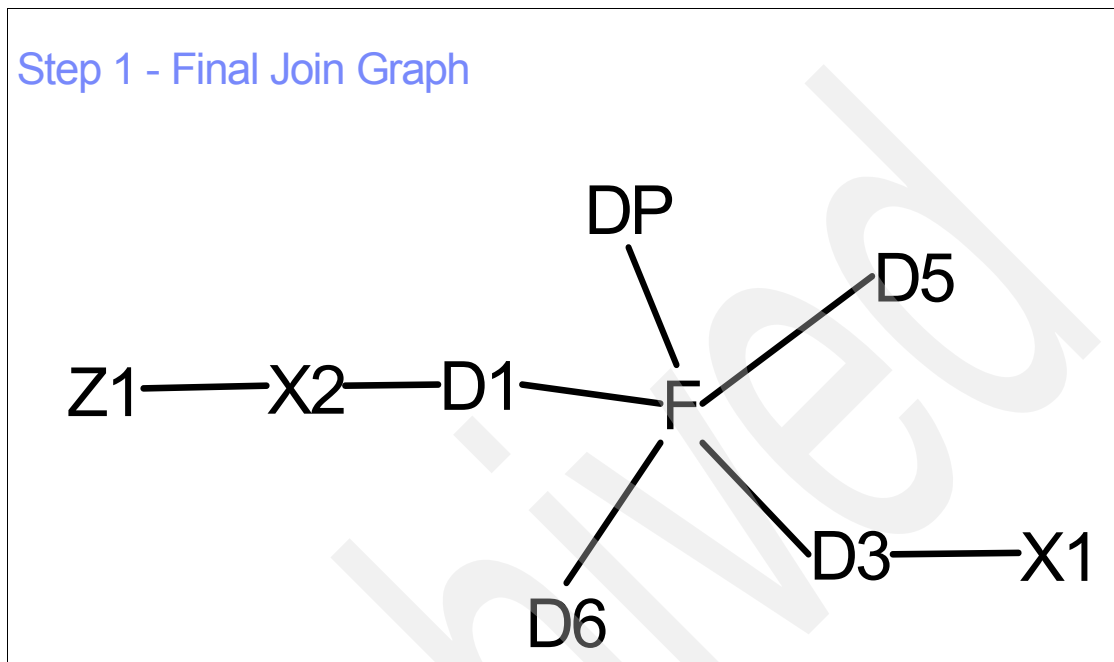


Figure 10-11 The final join graph

Thus, we have broken down the original complex query into an easily understandable diagram that depicts the table join relationships and filtering predicates. This diagram can be used in conjunction with the original SQL for further detailed analysis. See Figure 10-12.

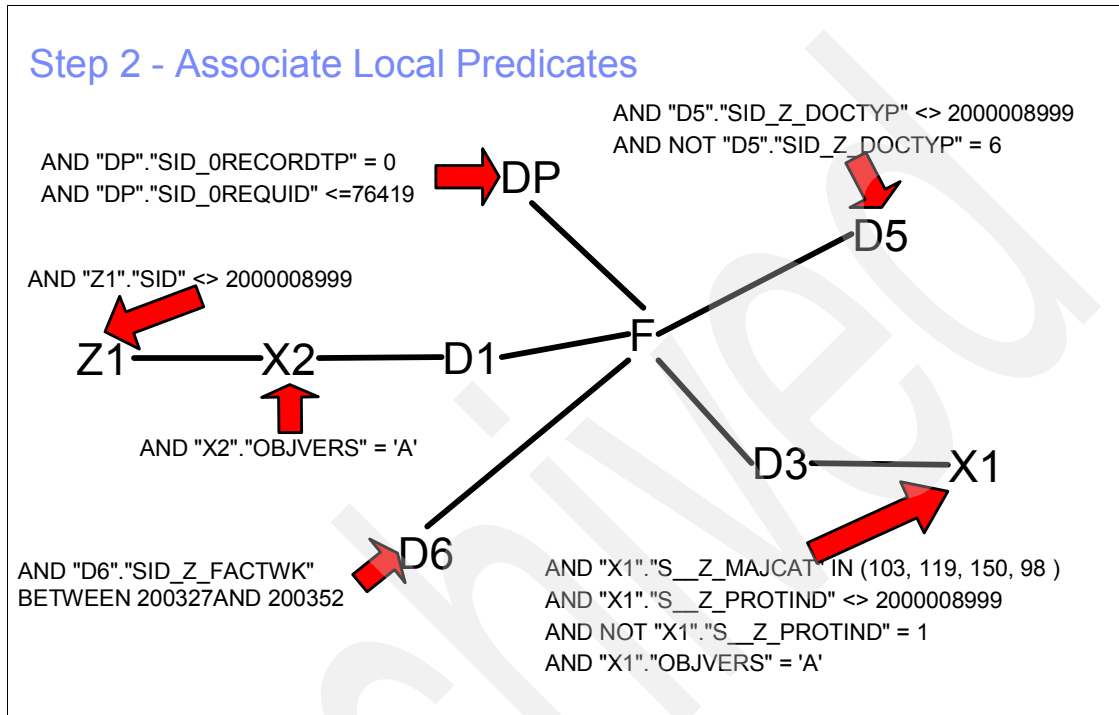


Figure 10-12 Associate local predicates

The next step is to determine where the filtering is occurring for this SQL statement. Currently it is not clear which predicates provide the greatest degree of filtering.

This step involves breaking apart the original query to count the number of rows retrieved for each dimension and master data table independently, and joined to the central facttable. See Figure 10-13.

Step 3 - Determine the Filtering

D6

```
SELECT COUNT(*)  
FROM "/BIC/DZIC_SFC6" "D6"  
WHERE "D6"."SID_Z_FACTWK" BETWEEN 200327 AND 200352
```

Shows the filtering
of this predicate

Count

37

```
SELECT COUNT(*)  
FROM "/BIC/FZIC_SFC" "F" ,"/BIC/DZIC_SFC6" "D6"  
WHERE "F"."KEY_ZIC_SFC6" = "D6"."DIMID"  
AND "D6"."SID_Z_FACTWK" BETWEEN 200327AND 200352
```

Shows the effect of
filtering against the fact
table

F
/
D6

Count

1,317,164

Figure 10-13 Determining the filtering

The counts in Figure 10-14 demonstrate that filtering of local predicates on the D6 table (37 rows), and combined D6 to F table (1,317,164 rows).

This step is repeated for each set of dimension/snowflakes independently and joined with the facttable. These counts will be used to compare all filtering combinations.

Step 4 - Compare Filtering

| Join Sequence | Count with Filtering | Table CARDF | % Rows Returned |
|---------------|----------------------|-------------|-----------------|
| Z1 | 13 | 13 | 100% |
| Z1-X2 | 26,514 | 353,413 | 7.50% |
| Z1-X2-D1 | 171,787 | 1,565,004 | 10.98% |
| Z1-X2-D1-F | 1,702,101 | 10,743,216 | 15.84% |
| X1 | 258 | 43,511 | 0.59% |
| X1-D3 | 281 | 30,932 | 0.26% |
| X1-D3-F | 511,179 | 10,743,216 | 4.76% |
| DP | 432 | 432 | 100% |
| DP-F | 10,743,216 | 10,743,216 | 100% |
| D5 | 598 | 598 | 100% |
| D5-F | 10,743,216 | 10,743,216 | 100% |
| D6 | 37 | 423 | 8.75% |
| D6-F | 1,317,164 | 10,743,216 | 12.26% |

Figure 10-14 Comparing filtering

The original facttable contains 10,743,216 rows.

Comparing the filtering provided by each dimension/snowflake joined to the fact, we can see in Figure 10-14 that neither DP nor D5 provide any filtering (because the count of join to the fact equals the count of the fact only). The snowflake containing D3 provides the strongest filtering, followed by the dimension D6, and the snowflake containing D1.

Therefore, if the current access path does not involve the strongest filtering dimension (D3, accessed before the facttable), then the access path may be suboptimal.

If the access path does not reflect the filtering as outlined by the counts, then the first step is to analyze the statistics and ensure that RUNSTATS was run. (This may seem like an obvious suggestion, but it is surprising how often RUNSTATS

is run on an empty table (dimension or fact), and statistics have not been updated since.)

If statistics are current, and the access path does not match with the counts such that the most filtering dimension/snowflake is accessed before the facttable, then the most likely scenario is either:

- ▶ The dimension/snowflake chosen before the facttable has had its filtering overestimated (so the filtering looks better than it actually is).
- ▶ The good dimension/snowflake (not chosen before the facttable) has had its filtering underestimated (so the filtering looks worse than it actually is).

For information about analyzing and determining the actual cause, refer to 8.3, “Analyzing query performance” on page 157.

10.3 Indexing of DSOs

Query performance on DSO objects can be greatly influenced by having appropriate indexes on the DSO tables. When a DSO object is created within SAP, only one index is created. This is a primary index, which is usually comprised of all the key fields of the DSO object.

This index most likely will not provide optimal performance for the queries being executed against the DSO. Secondary indexes should be created. However, keep in mind that there is a cost to maintaining indexes. This cost is mostly incurred during the load DSO phase. Therefore, index design is important and it depends on the queries that will be executed against the DSO. The point is to create indexes that are worth the cost of maintenance.

The objective of indexing on the DSO is to minimize the number of DSO table rows that must be accessed to produce the final result. The closer the number of rows retrieved is to the number of rows returned by the query, the better the performance should be. Therefore, indexes should be designed to support the most filtering of rows. This is more important than the number of matching columns in an index.

Let us look at the DSO query that is analyzed in 10.1, “SQL efficiency for DSO” on page 204, and see how additional indexes on the tables accessed can influence the performance of the query.

The WHERE clause of this query consists of the following predicates, which proved to provide the greatest degree of filtering:

```
WHERE "/BIC/AZOSASALE00"."/BIC/ZMATLPLNT" =  
      "/BIC/XZMATLPLNT"."/BIC/ZMATLPLNT"
```

```

AND "/BIC/AZOSASALE00"."PLANT" = "/BIC/XZMATLPLNT"."PLANT"
AND "/BIC/AZOSASALE00"."ACCNT_ASGN" IN ('00', '01', '04', '06')
AND "/BIC/AZOSASALE00"."CALMONTH" BETWEEN '200311' AND '200404'
AND "/BIC/AZOSASALE00"."CURTYPE" = 'B0'
AND "/BIC/XZMATLPLNT"."OBJVERS" = 'A'
AND "/BIC/XZMATLPLNT"."S__ZMATLONLY" = 360013

```

The join sequence that we wish to encourage is table /BIC/XZMATLPLNT accessed first, and joined to /BIC/AZOSASALE00. Therefore, local predicates on the /BIC/XZMATLPLNT should be indexed to encourage this as the leading table to be accessed.

```

AND "/BIC/XZMATLPLNT"."OBJVERS" = 'A'
AND "/BIC/XZMATLPLNT"."S__ZMATLONLY" = 360013

```

Note that the OBJVERS column has only one possible value, and therefore does not provide any filtering. The only reason to add this to the index is if index-only access is desired.

The index recommendation for table /BIC/XZMATLPLNT is shown in Table 10-1.

Table 10-1 Index recommendation

| Column | Reason |
|----------------|---|
| S__ZMATLONLY | Required to support local filtering |
| /BIC/ZMATLPLNT | Optional to support additional joins (see 10.3.1, "Index design for post-DSO access" on page 223) |
| PLANT | Optional to support additional joins (see 10.3.1, "Index design for post-DSO access" on page 223) |

Only local predicates are available as index matching on the leading table accessed. Join predicates are not. Thus the index design for BIC/XZMATLPLNT incorporates the local predicate first.

The join predicates are not valid as filtering predicates on the leading table (since the join value is not known when the first table is accessed), but they do provide filtering if this table is ever chosen as a non-leading table in the join sequence. The join predicates in the index ensure that the subsequent table accessed will be joined in the sequence of the index, which can improve sequential retrieval for the non-leading table data.

To support /BIC/AZOSASALE00 as the second table in the join sequence, the following predicates must be considered:

```

WHERE "/BIC/AZOSASALE00"."/BIC/ZMATLPLNT" =
"/BIC/XZMATLPLNT"."/BIC/ZMATLPLNT"
AND "/BIC/AZOSASALE00"."PLANT" = "/BIC/XZMATLPLNT"."PLANT"

```

```

AND "/BIC/AZOSASALE00"."ACCNT_ASGN" IN ('00', '01', '04', '06')
AND "/BIC/AZOSASALE00"."CALMONTH" BETWEEN '200311' AND '200404'
AND "/BIC/AZOSASALE00"."CURTYPE" = 'B0'

```

For the non-leading tables in the join, local predicates and join predicates are treated equally from the perspective of the ability to match index columns. Equal and IN predicates must precede range (BETWEEN, LIKE, <, <=, >, >=) predicates in the index, however, since range predicates will stop the index matching, and subsequent predicates will be applied as index screening.

Similarly, a second IN predicate will also force matching to stop (at the column immediately preceding the second IN list) and revert to index screening. Therefore, where possible, priority should be given to equal predicates for indexing.

Whether local predicates appear before join predicates depends on whether the local predicates will always appear in the query. If so, then the preference is to order local equal and IN columns before join columns in the index. This allows these columns to be used as matching as the leading or non-leading table in a join.

Also, local predicates as the leading columns will ensure that any matching access to this index (as leading or non-leading table) will be restricted to the bounds of the equal predicate, which can provide a better bufferpool hit ratio and exploitation of index lookaside.

To improve flexibility of local predicates, an alternative is to partition by a commonly used local predicate. A good example is often a time-based column in a data warehouse environment. Data warehouses (including SAP BI DSO and InfoCubes) generally contain many months or years of data, but users often query only the most recent, thus making separation of current and historical (using partitioning) a wise choice.

The indexing options for table /BIC/AZOSASALE00 are shown in Table 10-2.

Table 10-2 Indexing options

| Column | Local or join | Operator | Partition candidate | Index column sequence | Mandatory/optional |
|----------------|---------------|----------|---------------------|-----------------------|--------------------|
| CURTYPE | Local | = | Possibly | 1 | Optional |
| ACCNT_ASGN | Local | IN | No | 2 | Optional |
| /BIC/ZMATLPLNT | Join | = | No | 3 | Mandatory |
| PLANT | Join | = | No | 4 | Mandatory |

| Column | Local or join | Operator | Partition candidate | Index column sequence | Mandatory/optional |
|----------|---------------|----------|---------------------|-----------------------|--------------------|
| CALMONTH | Local | BETWEEN | Yes | 5 | Optional |

The mandatory columns to support the join and the majority of the filtering are the join predicates (if you recall, the local predicates filtered from 7 million to 1 million rows, but the join predicates filtered down to 8 rows). All local predicates, therefore, are listed as optional.

CALMONTH is a strong candidate to be the partitioning column. This provides many benefits for maintenance (REORG, backup, and so on, only required on the most recent partitions, as older partitions do not change over time), and also for exploitation by other queries without the column required in the index (with DB2 table controlled partitioning).

CURTYPE has only two possible values, which makes this a relatively poor index choice, but it could be a good partitioning candidate (preceding CALMONTH) to add greater granularity of each partition if required. CURTYPE should only be added to the index, or as a partitioning column, if an equals predicate always exists in queries against the DSO.

ACCT_ASGN is also not a strong filtering column, and has the disadvantage of an IN list rather than equal predicate coded against it. IN list predicates are not preferred for index matching for lower cluster ratio indexes (less than 80%) because list prefetch cannot be exploited with matching IN list access. Therefore, this column may not be a good choice for indexing unless the column is always existing in queries and the column belongs to a high cluster ratio index.

The index recommendations for table /BIC/AZOSASALE00 are shown in Table 10-3.

Table 10-3 Index recommendation

| Column | Reason |
|----------------|---|
| CURTYPE | Only if local predicate always exists in queries, and column is not chosen as a partitioning column |
| ACCT_ASGN | Only if local predicate always exists in queries, and index provides good clustering |
| /BIC/ZMATLPLNT | To support join filtering |
| PLANT | To support join filtering |
| CALMONTH | Only if not chosen as partitioning column |

10.3.1 Index design for post-DSO access

For each table not chosen to provide filtering before the DSO is joined, you must ensure that these tables have indexes that support the join and local filtering predicates. This will ensure efficient nested loop join performance for all tables.

Given the following WHERE clause from one of the remaining tables from the DSO example:

```
WHERE "/BIC/AZOSASALE00"."PROFIT_CTR" = "/BIO/SPPROFIT_CTR"."PROFIT_CTR"  
AND "/BIO/SPPROFIT_CTR"."SID" = 31
```

The relevant columns from table /BIO/SPPROFIT_CTR are PROFIT_CTR (join predicate) and SID (local predicate).

For smaller master data tables (less than a few data pages) or master data tables with local predicates that provide very little filtering, it may be sufficient to ensure that an index exists on the join columns only. This provides the greatest degree of flexibility, especially since the local predicates are not going to significantly limit the number of data pages accessed.

As the master data table becomes larger and the local predicates provide more filtering, it is important to create indexes that support both the local and join predicates. Local predicates are preferred as the leading index columns because they support this table accessed as the leading table accessed or non-leading in the table join sequence. Additionally, they allow frequency statistics to be collected on the local predicate column with default RUNSTATS. Both the local and join predicates should be included in the index because both columns provide filtering as part of the join.

Therefore, for each smaller table (a few pages) we recommend having an index that supports the join predicates. This is also true for larger tables with local predicates that provide minimal filtering. Larger tables with local filtering predicates should have indexes on local (equal and IN) followed by join predicates. If local predicates are range predicates, then these should appear after join columns in the index.

The index recommendation for table /BIO/SPPROFIT_CTR is shown in Table 10-4.

Table 10-4 Index recommendation

| Column | Reason |
|------------|-----------------------------------|
| SID | Local (equal) filtering predicate |
| PROFIT_CTR | Filtering join predicate |

10.4 Indexing of InfoCubes

The default indexes for an InfoCube facttable are generally sufficient for optimal query performance. This is especially true when combined with partition elimination that can be exploited by DB2, such as a query with a time restriction on a partitioned E facttable.

The InfoCube query example shown in Figure 10-1 on page 204 actually contains a custom time dimension (D6), and therefore the query does not exploit the SAP-provided time dimension (referred to as DT in queries). Since E facttable partitioning is by the time characteristic, then InfoCubes with custom time dimensions may not benefit from partition elimination. This makes custom indexing a possible requirement.

10.4.1 facttable indexing

From Figure 10-1 on page 204, the filtering dimensions (in order of filtering) are D3, D6, and D1. A facttable index may be created to support the combined filtering, although you must consider the Cartesian product size of before facttable dimensions.

After filtering, the dimension/snowflake sizes are:

- ▶ D3 - 281
- ▶ D6 - 37
- ▶ D1 - 171,787

A Cartesian product of $D3 \times D6 \times D1 = 1.78$ billion is not desirable. The size of the D1 snowflake is too large to include in a Cartesian product. Given that this dimension provides the least filtering of the three listed, this can be discarded in choosing the optimal post-facttable access.

A Cartesian product of $D3 \times D6 = 10,397$ is very desirable given the facttable size of 10 million rows. At most, the facttable will be probed 10,397 times from this Cartesian result (potentially less, if not all combinations of the two dimensions exist in the facttable).

This query may benefit from an index on KEY_ZIC_SFC3, KEY_ZIC_SFC6.

10.4.2 Index design for dimension/master data tables

Indexes on dimension/master data tables provide two main functions:

- ▶ Efficient data access and join performance
- ▶ Easy statistics collection

For efficient join performance, it is important that an index exists to support the join predicates, regardless of the table size (small or large). For anything other than very small dimensions/master data tables (a few data pages), you must ensure that these tables have indexes that support the join and local filtering predicates. This will ensure efficient nested loop join performance for all tables.

Local predicates are preferred as the leading index columns because they support this table accessed as the leading table accessed or non-leading of a table join sequence. Additionally, they allow frequency statistics to be collected on the local predicate column with default RUNSTATS. Both the local and join predicates should be included in the index because both columns provide filtering as part of the join — that is, unless the dimension/master data table is very small, in which case an index on the join predicate alone may be sufficient for join performance.

Aside from data access and join performance, statistics collection on the dimension and master data tables becomes very important. An incorrect estimation of filtering from dimension/snowflake will propagate to the large facttable. Thus a 10% error in estimation on a 100-row dimension is only 10 million rows, but when joined to the facttable (10 million), the error factor becomes 1 billion rows.

Single column indexes that support the local predicates on the dimension and master data tables allow frequency statistics to be collected by default. For dimensions/master data tables that have multiple filtering predicates, indexes that contain all filtering columns allow default collection of correlation information on these tables. This information is crucial for correct access path selection (and table join sequence).

Dimensions and master data tables (large or small) can support many more indexes than can multi-million row facttables. This should be encouraged for the purpose of efficient data access/join performance, and also for effective statistics collection.

Archived

Project Jupiter: large BI Accelerator scalability evaluation

Jupiter is a joint project between IBM and SAP, focused on SAP NetWeaver 2004's Business Intelligence (BI) and SAP BI Accelerator¹, implemented on an IBM server infrastructure. Our solution demonstrates the capabilities of BI Accelerator in a very large IBM configuration. We tested scalability with increasing data volumes and resources.

This chapter describes the implementation and results of our project, including:

- ▶ The Jupiter solution configuration
- ▶ The SAP BI database load of three data sizes: 5 TB, 15 TB, and 25 TB
- ▶ Test case scenarios for BI Accelerator index creation and multi-user query loads, using the three data sizes with different BI Accelerator configurations

This chapter presents the test results, which were validated by an independent third party, WinterCorp². It features the setup and tuning techniques that we used for our large SAP BI Accelerator solution.

¹ SAP's BI Accelerator is an application that runs on Blade servers with Linux SuSE on an Intel Xeon® using 64-bit architecture. It requires an SAP BI Accelerator License, SuSE Linux, and a shared file system, such as IBM General Parallel File System (GPFS).

11.1 Background

To be more competitive in a dynamic business environment, large global companies are focused on improving their ability to identify and react to business changes. This responsiveness demands that operational personnel are able to quickly identify any developing business issues. These include detecting an inventory problem, identifying fraudulent purchases in real time, or identifying preferred customers at every point of contact. This requires that personnel across the lines of business have the correct information, at the correct time, to make the right business decisions. To satisfy this need for information, business intelligence solutions are evolving to become more proactive, delivering actionable information to the front lines of business operations.

To meet the challenges of an operational business intelligence environment, System z offers many advantages to ensure information availability for all. The System z environment is able to manage workloads from thousands of users, while securing data from external and internal threats, a critical capability for every company. System z environments offer a highly reliable, available, and manageable DB infrastructure to support huge query workloads that are typical of an operational BI environment.

Leveraging a mainframe's self-management characteristics, system workload control, and overall cost efficiencies can significantly improve the business value of a solution. With its unique scalability and security, combined with its data availability characteristics, System z provides the capability to seamlessly create an operational BI environment that meets the business requirements of today's 24/7 economy.

Combining these strengths with the performance enhancements of BM Systems Solution for SAP Business Intelligence Accelerator, companies have the ability to leverage the strengths of the mainframe, with the performance and flexibility benefits of a turnkey solution, meeting the demanding requirements of today's business climate.

11.1.1 Project overview

In support of the operational BI requirements of a business, IBM and SAP worked together to demonstrate the extreme scalability possible with this fully integrated solution. This project incorporated System x™ Blades that provide fast parallel processing of queries, complemented by System z and DB2 for z/OS as the secure, high availability data repository for the volumes of customer's InfoCube data. System p was used to support the SAP BI application software.

² <http://www.wintercorp.com/>

This very powerful combination of technologies provides an integrated environment capable of handling business intelligence today, and far into the future. This server infrastructure, coupled with DS8300 storage systems, proved to be a very resilient and highly available configuration over the course of the project. There were no hardware failures during the six months of this project.

During this project large volumes of customer and benchmark InfoCube data were loaded into DB2. The InfoCube data in DB2 was then indexed and stored in the BI Accelerator for fast query processing. All queries were directed to the BI Accelerator since the focus of the project was to test the scalability of the BI Accelerator. As with other database servers, some complex ad hoc queries directed to DB2 would require creation of aggregates for fast query response time. The beauty of BI Accelerator is that aggregates are not required.

The scale of our test environment was several magnitudes larger than the largest BI Accelerator blade configuration size (28 blades) previously tested. We chose to test three database sizes: 5 terabytes (TB), 15 TB, and 25 TB. Our project demonstrated that the SAP BI Accelerator on IBM blades can deliver very fast execution for a wide variety of complex queries with consistent response times. This was achieved using a highly compressed BI Accelerator index data structure and efficient parallel in-memory query processing.

11.2 Jupiter environment infrastructure

This section describes the Jupiter environment infrastructure that we built at the IBM System z Benchmark Center at Poughkeepsie, New York. It includes our:

- ▶ Hardware and software configuration
- ▶ Network infrastructure

By design, some hardware components in this environment were oversized, because we did not have all the information required for capacity planning at the beginning of the project. This large configuration was designed to ensure that there would be no resource constraints during any of the scalability tests.

11.2.1 Hardware and software configuration

Figure 11-1 gives an overview of the system configuration that we used to run our test case scenarios.

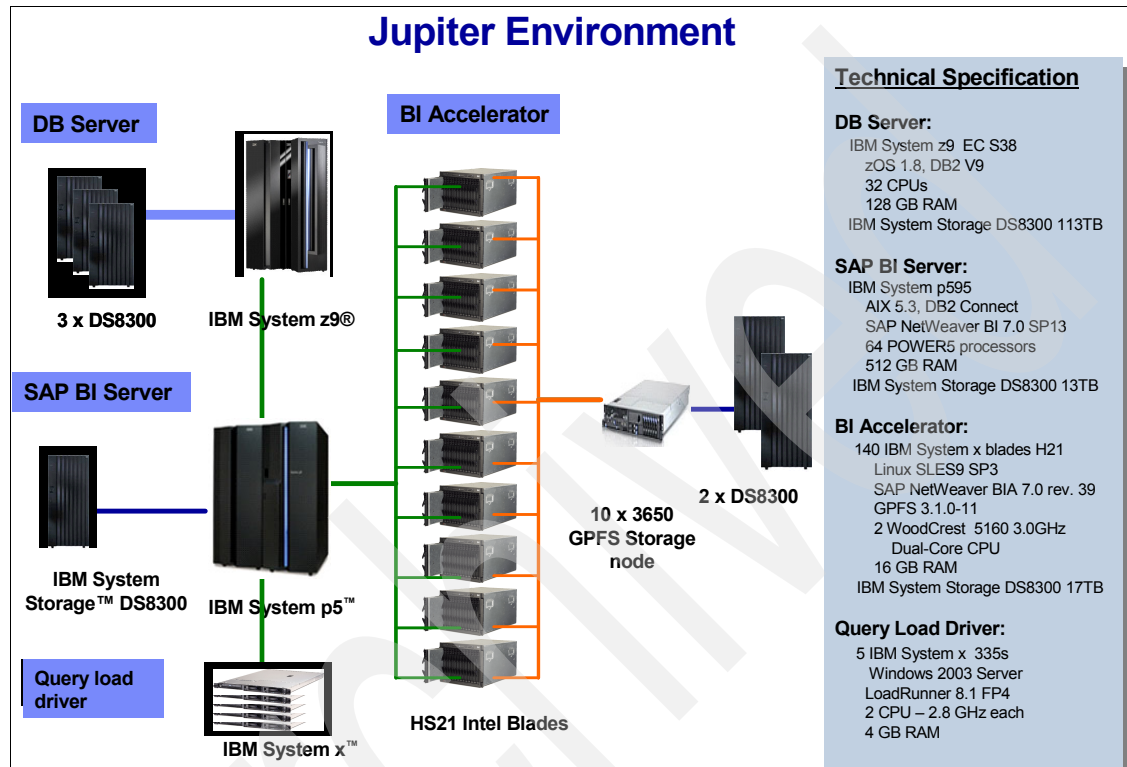


Figure 11-1 Jupiter environment infrastructure

There are four major components for the test configuration: database server, SAP BI server, BI Accelerator, and query load driver.

Server infrastructure hardware

The server infrastructure hardware is:

- ▶ For the database server, we used IBM System z9™ Enterprise Class S38 with 32 CPs and 128 GB of main storage.
- ▶ The SAP BI server ran on IBM System p5™ 595, which has 64 POWER5™ processors and 512 GB RAM.

- ▶ The BI Accelerator was installed on 140 IBM HS21 blades. Each blade had two WoodCrest 5160 3.00 GHz dual-core CPUs and 16 GB RAM, for a total of about 2.2 TB RAM across all blades.

Note that one backup blade was configured for every 27 blades. So our configuration consisted of 135 production and five backup blade servers.

- ▶ Five IBM System x 335 workstations with two CPUs, 2.8 GHz, and 4 GB of RAM were used for the query load driver.
- ▶ There were six IBM System Storage™ DS8300 systems in this configuration. Three storage systems were attached to the System z9, one to the System p5, and two to the blades. The RAID-5 configured storage for System z was 129 TB. The RAID-5 configured storage for System p was 14 TB. The RAID-10 configured storage for the blades was 29 TB. That is the total storage before logical volumes were formatted. The capacities for these storage systems after being initialized were 113 TB, 13 TB, and 17 TB, respectively.

Software

Here is the list of software installed on each hardware component:

- ▶ System z9: z/OS V1.8, DB2 for z/OS V9
- ▶ System p5: AIX 5.3, SAP Netweaver BI 7.0 SP13, DB2 Connect™ V9
- ▶ Blades: Linux SLES9 SP3, SAP Netweaver BI Accelerator 7.0 rev. 39, GPFS 3.1.0-11

11.2.2 Network infrastructure

The network configuration for the Jupiter environment is shown in Figure 11-2. In this diagram, you can see that the following technologies were used for our network infrastructure: FICON® channel, GbE, 10 GbE, InfiniBand®, and Fibre Channel.

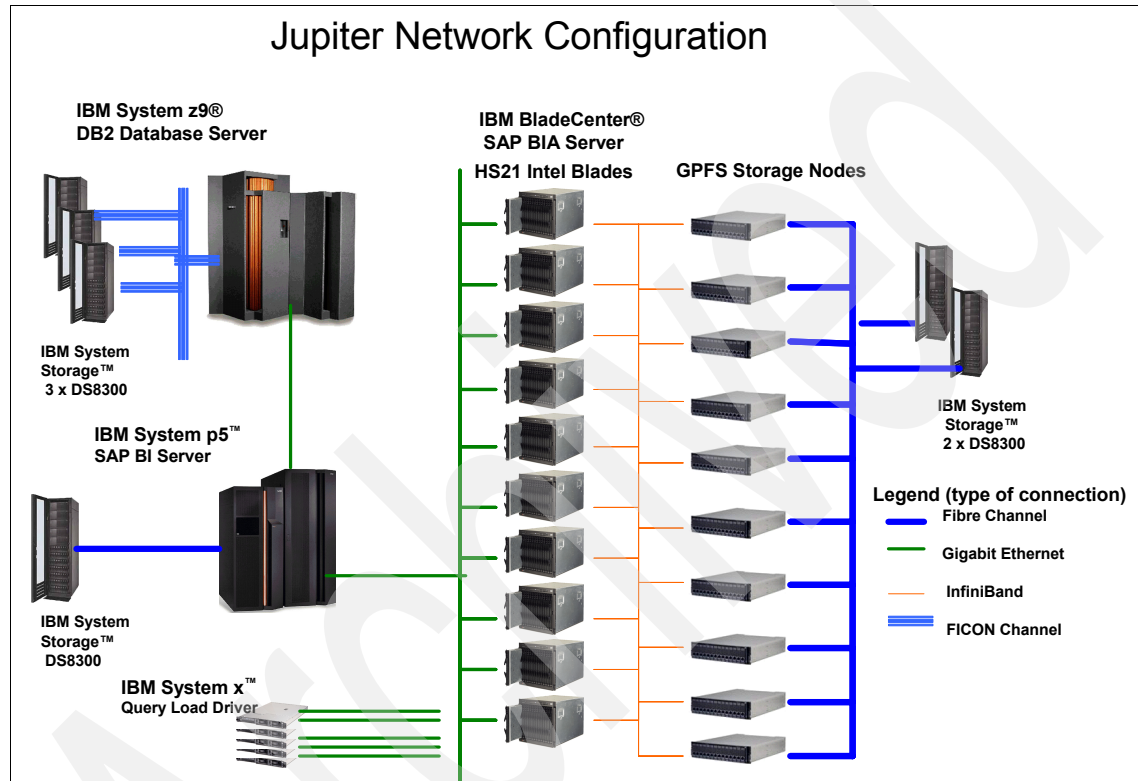


Figure 11-2 Jupiter network configuration

Connection between System z and DS8300

There were 24 FICON channel connections between System z and the three DS8300 storage systems (eight connections between System z and each DS8300 storage system). They were critical for writing data to the DB2 database during InfoCube load and reading data during the BI Accelerator index creation.

Connection between System p and DS8300

There were eight Fibre Channel connections between System p and the DS8300 storage system. Multipath Subsystem Device Driver (SDD) was used. This enabled parallel access to the file systems on the DS8300 storage system. Since

the input files for the InfoCube load resided on this storage system, it was critical to get the maximum bandwidth to the storage system.

Connection between System p and System z

There were eight 10 Gigabit Ethernet (GbE) lines between System p and System z. We could have used fewer than eight connections. However, the precise data rate requirement was not known at the beginning of the project. It was safer to over-configure than to under-configure, to ensure that there were no network constraints during InfoCube load and during BI Accelerator index creation phases. The data rate was not expected to be close to the aggregated bandwidth of these eight connections. Multiple connections mitigated (but did not totally prevent) delays due to latency from channel busy, hardware implementation, and physical media.

To simplify the environment, our System p was configured as a single system image. Since multiple SAP application servers can run in one partition, our Jupiter project employed multiple instances for a more realistic customer environment. This affected our network implementation, because each SAP application server can use only one SAPDBHOST connection, which is usually associated with one physical network connection. To make use of all eight connections, we combined them into one virtual connection using a virtual IP address (VIPA).

This approach utilizes network trunking, such as EtherChannel or MultiPath. EtherChannel is supported on System p but not on System z. So instead we used a MultiPath network connection, which has similar functionality to the EtherChannel. Example D-1 on page 326 shows our setup definition of static multipath. Each connection is configured with its own subnet on separate VLANs. This ensures that traffic is sent and returned over the same physical connection, and also provides better analysis and monitoring of performance data. Also, a jumbo frame size was configured for the large block sizes transmitted.

For load balancing, there is an algorithm to distribute traffic to all connections when several are combined into one virtual connection. This method provides higher availability, since another available physical connection associated with the virtual connection can be used if one of the physical connections is down. For details and guidance on high availability network implementation, refer to *SAP on IBM System z: High Availability for SAP on IBM System z Using Autonomic Computing Technologies*, SC33-8206.

Connection between System p and blade cluster

There was one 10 GbE connection from System p to our network switch, and an EtherChannel of four GbE connections from the switch to each blade chassis. Within the chassis, there was one GbE interface for each of the 14 blades within

the chassis for blade-to-blade connections. Data traffic between System p and the BI Accelerator cluster of blades was mainly through Remote Function Calls (RFCs). A standard Maximum Transmission Unit (MTU) size of 1500 was used because the data package traffic was small.

Connection between blade cluster and DS8300

A GPFS file system was mounted and accessible to all 140 blades. Generally, a Storage Area Network (SAN) is effective with 64 blades or fewer. As the number of blades scales up, GPFS topology with its cluster of network shared disk (NSD) server nodes performs better. In our large environment with 10 BladeCenter® chassis and 140 blades in total, 10 GPFS NSD servers were employed. They were System x3650 machines (two dual-core CPUs at 3.0 GHz with 4 GB RAM) running Linux SLES9 SP3.

Since InfiniBand (IB) has better bandwidth and latency than GbE, we chose IB for the network between the blades and the GPFS NSD servers. Each configured HS21 blade chassis had two embedded Cisco SDR IB switches. Blades 1 through 7 were configured to use the first IB switch, and blades 8 through 14 were on the second one, to have a non-blocking configuration. Each SDR IB daughter card on each blade is rated at 10 Gbps, or 4x, where x = 2.5 Gbps. Each SDR IB switch had two external 12x ports and two external 4x ports, with an aggregate bandwidth of 80 Gbps. Each GPFS NSD server had 4x or 10 Gbps IB connection.

On the other end, the 10 GPFS NSD servers were connected to GPFS storage (DS8300) via twenty 4 Gbps Fibre Channel connections, two connections per GPFS NSD server.

Since each blade had GbE and IB interfaces, blade-to-blade communication could use GbE or IB. In our case, IB was mainly used for reading and writing to the GPFS file system and GbE was used for blade-to-blade communication.

11.3 SAP BI database and InfoCube load

One of the major challenges for project Jupiter was to build a large, complex (25 TB) BI database within a very short period of time. According to the plan, a minimum InfoCube data load rate of 1 TB per day was needed in order to meet the aggressive schedules. In the following sections, we describe the database structures and the InfoCube load techniques that we implemented to meet the project requirements.

11.3.1 SAP BI database

The four most relevant objects used for SAP BI (as described in Chapter 2, “Overview of SAP BI and related data” on page 9) are the master data, persistent staging area (PSA), data store (DSO), and InfoCubes (or simply cubes).

For our Jupiter project, three database sizes were created and populated to supply the data volume requirements for the suite of tests. These are 5 TB, 15 TB, and 25 TB.

In reference to the BI database characteristics, it is important to note that:

- ▶ The growth in database size was reached by increasing the size of the InfoCubes only. The data volume for PSA, DSO, and master data remained constant.
- ▶ The data volume refers to cubes (F fact table and indexes) only. The actual SAP BI database size was larger, if we include the statistics for PSA, DSO, and master data.
- ▶ DB2 data compression was not turned on for this environment. Since the focus of the test was the performance scalability of the BI Accelerator, and since this feature is not available in all database platforms, SAP opted not to enable this DB2 feature.

11.3.2 BI database InfoCube load

The initial build of the SAP BI database involved the population of master data, PSA, DSO, and InfoCubes. There were several challenges encountered during this phase of the study. Some were caused by the level of DB2 code that we had installed at the time. Others were due to the characteristics of the database, such as the large volume of data being inserted and the concurrency of the jobs during data population.

The execution of this project took six months. As standard benchmark practice, the software code level was not updated from the initial level unless we encountered a defect. However, due to the duration of this project, we did apply a few maintenance updates.

The focus of the study was to showcase the performance scalability of the BI Accelerator in various data volumes. However, the amount of time needed to load the InfoCubes was a significant portion of the project schedule. Thorough planning for disk storage allocation, layout, and backup/restore procedures was essential to meet the demanding milestones.

As part of the pre-benchmark planning activities, we had calculated the total storage required to hold the 25 TB database plus two backup copies. We configured additional disk space for other DB2 tables, z/OS systems, and monitoring data.

We utilized IBM Storage Management System (SMS) to manage and allocate storage space needed for the DB2 data, as well as other system data sets such as log and work files. We defined a dataclass for *extended addressability*. This enabled us to allocate space for linear VSAM data sets larger than 4 GB, which most of the tablespace partitions required.

In the 25 TB BI database, 78 InfoCubes were populated with a total of 30 billion rows, all of which were loaded into the BI Accelerator. These included:

- ▶ Thirty cubes for 10 years of data each for sales orders, billing, and delivery
- ▶ Forty-eight cubes for 48 calendar months for customers

Although the focus of the BI Accelerator tests was on the 78 InfoCubes, more than 32,300 DB2 tables were built on DB2 for z/OS. The 78 cubes comprised 80–90% of the total database size. The largest table group was about 425 GB in size, with nearly 500 million rows per table. This table also had the longest average row length. See Table D-1 on page 329.

During the InfoCube load process, we overcame some of the challenges by implementing changes to the DB2 objects and by applying maintenance to the DB2 code. The modifications to the definition of InfoCube and PSA tables and indexes were as follows:

- ▶ Increasing the DSSIZE from 4 GB to 8 GB to accommodate the larger row size of some of the DB2 tables
- ▶ Reallocating tables and indexes to 32 K bufferpools
- ▶ Increasing primary allocation of DB2 tablespaces to ensure availability of space during InfoCube load and to minimize the overhead associated with VSAM secondary extents

These enabled us to successfully build the 5 TB, 15 TB, and 25 TB BI databases. The 5 TB and 15 TB databases were similar to the 25 TB BI database in structure, except for scaled-down InfoCube sizes.

For the SAP and DB2 BI database objects, various bufferpools with page sizes of 4 K, 8 K, 16 K, and 32 K were used. For example, DB2 bufferpool BP0 was used for DB2 system objects, BP1 for 4 K sortworks, BP2 for tables, and BP3 for indexes. The bufferpools with 8 K and 16 K pages were used for other SAP system objects.

On the other hand, the bufferpools with 32 K pages were allocated for InfoCubes, PSA tables and indexes, and 32 K DB2 sortworks. The use of 32 K page tablespaces minimizes the amount of leftover space per page for rows with an average length of few hundred to thousand bytes. This may also reduce the amount of getpage activity for selects during the InfoCube and BI Accelerator index loads. For example, if there are 910 bytes per row on a table, with a 32 KB page size, it can fit 36 rows on a page. On the other hand, if a 4 KB page is used for this table, the same number of rows will reside on nine pages, since each page can only hold four rows.

Moreover, we took advantage of the DB2 bufferpool page fix feature by setting PGFIX to YES. For I/O intensive workloads, this could reduce page fixing overhead. However, the trade-off is higher demand on real memory usage. Since our z9 processor had plenty of real storage, the benefits of PGFIX=YES outweighed the costs.

To get a summary of the bufferpool setup, refer to the following tables in Appendix D, “Project Jupiter: evaluation details” on page 325:

- ▶ Table D-2 on page 333
- ▶ Table D-3 on page 333
- ▶ Table D-4 on page 333
- ▶ Table D-5 on page 334

InfoCube loads have characteristics similar to an insert-intensive batch workload. Two common types of overhead in DB2 for this type of workload are locking and logging. To minimize these, we implemented the following DB2 database tuning tips:

- ▶ All secondary indexes for the F fact tables were dropped prior to the InfoCube loads. The secondary indexes were rebuilt, using CREATE INDEX SQL, after the F fact tables were populated.
- ▶ PSA (staging) tables were defined to use PAGE instead of ROW locks to minimize the number of locks acquired.
- ▶ MEMLIMIT for the IRLM PROC, Lock Manager, was raised to 16 GB to accommodate the large number of locks held.
- ▶ PSA table entries were deleted using the LOAD utility with REPLACE DD DUMMY options for fast PSA cleanup.

There may be other tuning options to improve the InfoCube load rate, but we did not pursue this after we successfully exceeded the minimum load rate required for the project.

11.4 Test case scenarios

The goal of the Jupiter project was to prove the scalability of a BI Accelerator for increasing data volumes and increasing blade resources. The test plan consisted of a suite of tests on three database sizes (5 TB, 15 TB, and 25 TB), various blade configurations (27, 81, and 135 blades), the BI Accelerator index creation, and a multi-user query workload.

Note that for every 27 blades, one backup blade was configured. This brought our total blade requirements to 28, 84, and 140 for the three database sizes.

11.4.1 BI Accelerator index creation

When all the data was populated into the InfoCubes in a DB2 database on System z, the next step was the BI Accelerator index creation. This phase read data from a row-based database and transformed it into in-memory column-based files in a cluster of blades. In the Jupiter study, all InfoCube data from a DB2 database was indexed to BI Accelerator blades before it was available for query processing.

Objectives

The purpose of the BI Accelerator index creation test scenarios was to measure key performance indicators (KPIs) on 5 TB, 15 TB, and 25 TB database size points with blade resources of 27, 81, and 135 systems, respectively. These KPIs were:

- ▶ Elapsed time to create indexes into the BI Accelerator
- ▶ BI Accelerator index creation throughput rate
- ▶ CPU utilization on BI server and blade servers

Description

Since the goal of BI Accelerator index creation is to read data from a database server to a GPFS on blades, certain sizing and configuration settings need to be considered. Proper BI Accelerator blade memory sizing is an important consideration. SAP note 917803 provides an official sizing estimate and guideline for blades. In this study, 135 blades plus five backups (10 HS21 chassis of 14 Woodcrest blades each) were used. Each blade had 16 GB RAM and two dual-core processors at 3 GHz. Thus, 140 blades had about 2.2 TB of memory (RAM). A rule of thumb is to allocate 50% of blade memory for storing the indexed cube data and 50% for the Linux system for processing runtime objects and storing query result sets. This avoids performance impacts from memory swapping to disk.

Table 11-1 lists our Jupiter BI Accelerator resource requirements.

Table 11-1 BI Accelerator requirements

| Total DB size | RAM | # blades @16 GB RAM | Number of rows |
|---------------|---------|---------------------|----------------|
| 5 TB | 432 GB | 27 + 1 backup | 6 B |
| 15 TB | 1296 GB | 81 + 3 backups | 18 B |
| 25 TB | 2160 GB | 135 + 5 backups | 30 B |

In addition, the minimum GPFS size requirement is three times the total blade memory size, which is approximately 6 TB. IBM DS4000™ servers are normally employed when using a smaller number of blades. In this study, the DS8300, an enterprise class storage system, was used.

Our next consideration was the configuration settings for SAP parameters. They can influence data transfer performance: BATCHPARA, PKSIZE, NUMPROC, and SUBPKSIZE. They are defined in the RSDDTREXADMIN table, which contains some of the BI Accelerator administration settings. For details and guidance, refer to the SAP document *Technical Operations Manual for BI Accelerator 7.0*. Here is a brief description of these parameters:

- ▶ BATCHPARA defines the number of parallel background processes used to read data from the fact table of each InfoCube in the database into the data buffer at the SAP BI application server. This parameter essentially governs the degree of parallelism and can require significant system resource.
- ▶ PKSIZE defines the data size in bytes that is transferred from the fact table in the database to the data buffers at the SAP BI application server for each batch process. Parameters PKSIZE and BATCHPARA can affect significant BI system resource requirements. Make sure that a minimum memory size equal to (PKSIZE x BATCHPARA x number of cubes to be indexed in parallel) is available in the BI system.
- ▶ NUMPROC defines the number of dialog processes for each batch process to transfer data from the SAP BI data buffer to the BI Accelerator via Asynchronous Remote Function Call (aRFC). It is important to ensure that there are enough dialog processes and RFC connections defined.
- ▶ SUBPKSIZE defines the data size, in number of records, to be sent from the SAP BI data buffer to the BI Accelerator via aRFC.

Figure 11-3 explains the data flow from database server to BI Accelerator relationship of these parameters and the calculation for the required number of Remote Function Call (RFC) connections.

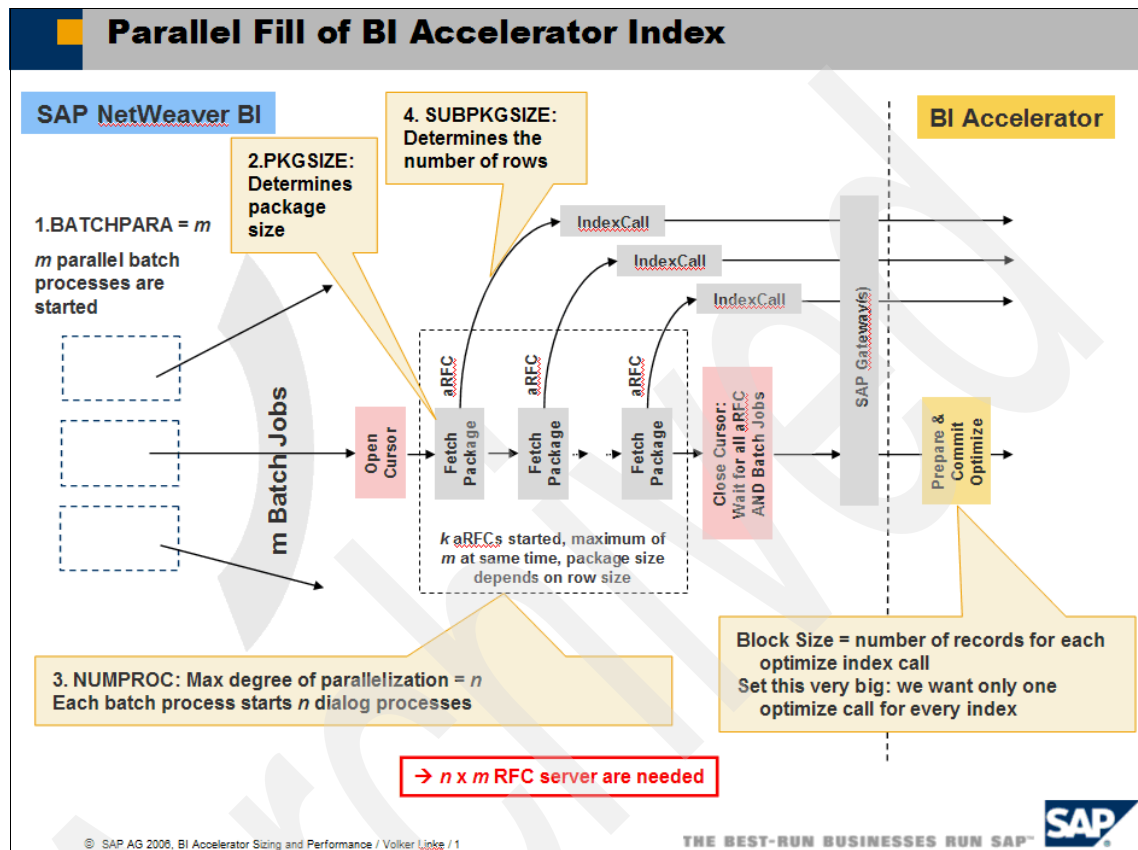


Figure 11-3 Parallel fill of BI accelerator index

To describe how the parameters influence the BI Accelerator creation process, we use one Jupiter example, an index creation of 10 cubes in parallel with the following settings: BATCHPARA=6, PKGSIZE=100,000,000, NUMPROC=5, SUBPKGSIZE=20,000 rows.

During the initial index creation phase, data was read from the F fact tables in the database to data buffers in the SAP BI application server. Since BATCHPARA=6, six concurrent background processes were started for each cube. Each the data transfer process moved data in 100 MB chunks, as specified in the PKGSIZE parameter. Since we chose to create indexes for 10 cubes in parallel, a total of 60 background processes run concurrently, if enough SAP

batch work processes are available. For our project, we defined 256 SAP batch work processes.

Once the data was in the data buffer of the SAP BI application server, the data package size PKGSIZE was subdivided into SUBPKSIZE. Each package of 20,000 rows was transferred to the BI Accelerator via Asynchronous Remote Function Calls (aRFC) employing five dialog processes (NUMPROC). Thus, for each batch process, there were five dialog processes defined. Therefore, the total number of actively used dialog processes could be up to 300 (BATCHPARA x NUMPROC x number of cubes).

Archived

In addition, each cube, especially large ones, can be configured to distribute to a set number of blades for indexing, depending on the configured number of parts. The cube is considered to be very large if it contains several hundred million rows. Since different cubes could have different numbers of columns, the number of cells is a more accurate method of deciding whether a cube is large. By default, 100 million cells (number of rows x column key figures) is a threshold to determine that the cube is large, and thus a candidate for horizontal partitioning. Figure 11-4 illustrates the horizontal partitioning concept.

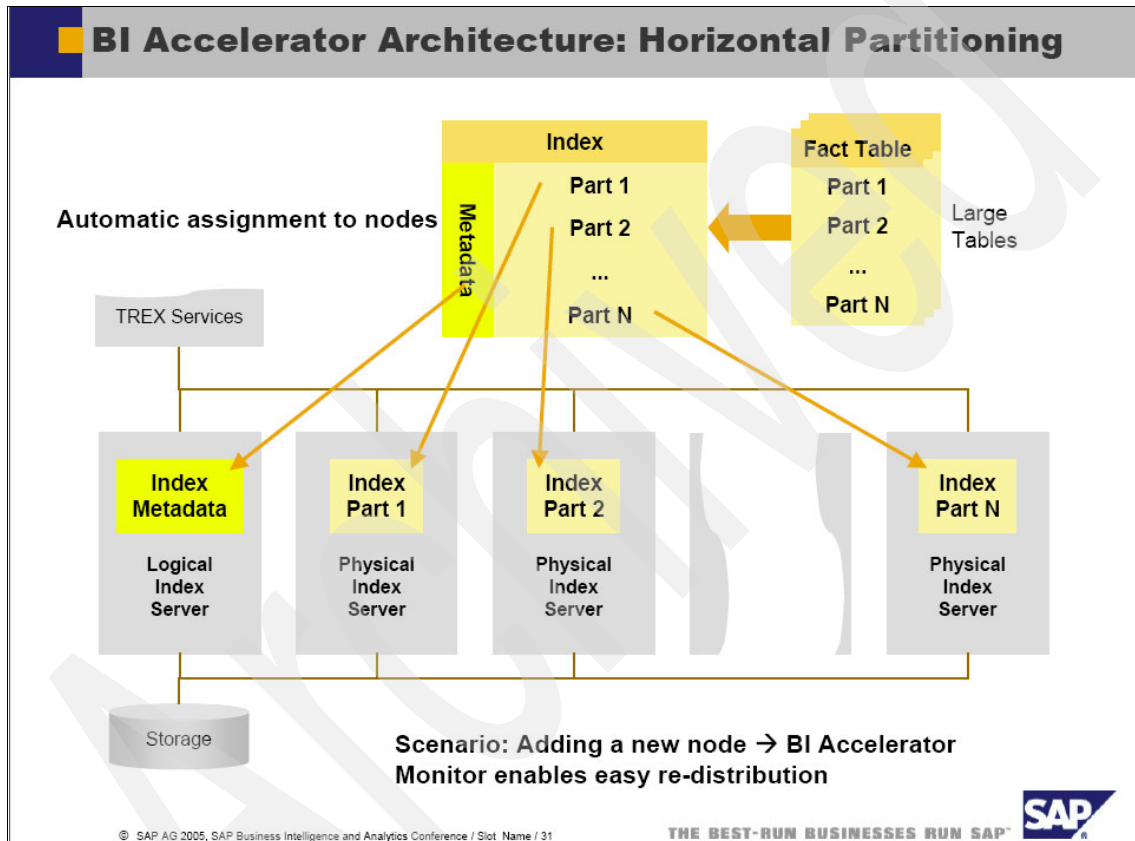


Figure 11-4 Horizontal partitioning

All InfoCubes used in this study were in this large cube category and were candidates for partitioning. Our study threshold setting was 250 million cells. Certain tests were conducted with 27 parts and others with 40 parts. The number of parts is a trade-off between having more parallel tasks and having more interblade communication traffic. This setting can be implemented with the TREXAdmin tool:

Tree > index > scrolling down to the bottom

(max_size=250,000,000, and max_split_parts=40).

When the data arrives at the blade, the RFC compressed data is uncompressed. Then gzip compression is done before writing temporary objects to GPFS, in the form of *.tmpx files. Each blade has its own unique storage location/ subdirectories for the same cube. The next phase is to read these temporary objects that were just written to GPFS. Initial indexing and optimizing take place. This uncompresses the data, reorganizes it, and compresses again with a different algorithm. Lastly, the data is written as a whole to the GPFS as final persistent objects in the form of *.bin files.

In Appendix D, “Project Jupiter: evaluation details” on page 325, Examples D-6 through D-8 display a list of BI Accelerator indexed cubes, a cube example with 40 part horizontal partitioning, a list of all columns of an indexed cube, and their corresponding temporary and persistent objects. They illustrate SAP BI Accelerator horizontal partitioning, vertical decomposition, and compression.

The BI Accelerator index creation process is very I/O intensive to the GPFS storage shared by the blades. During the test, we started with the default value of 256 K for the GPFS I/O block size. We added a second DS8300 storage system to handle the high GPFS I/O activity. Our further investigation found that the high GPFS I/O activity was actually due to the large block size of 256 K, so we experimented and determined that blocksize=64K and maxblocksize=64K yielded better GPFS performance.

The two DS8300 storage systems were configured using RAID-10 because of the write intensity of the BI Accelerator index creation. In general, RAID-10 is recommended for write-intensive workloads. However, it requires more disk space.

Measurement results

Table 11-2 provides the resulting GPFS data sizes after BI Accelerator index creation for our 5 TB, 15 TB, and 25 TB databases. Note that although the final space used in GPFS is much smaller than configured, more space is actually used for storing BI Accelerator temporary objects. Additionally, more than three times the space is required if backups or other activities are planned. Most importantly, this study shows that from 0.6 TB to 1.3 TB of fact table data per hour can be indexed to the BI Accelerator, which is quite impressive.

Table 11-2 Index creation results: throughput, duration, and space

| Database volume size and blade resources | Throughput (terabytes/hr) | Duration (hh:mm:ss) | GPFS size |
|---|----------------------------------|----------------------------|------------------|
| 5 TB index creation 27 blades | 0.6 | 6:40:30 | 300 GB |
| 15 TB index creation 81 blades | 1.1 | 11:07:07 | 800 GB |
| 25 TB index creation 135 blades | 1.3 | 15:04:29 | 1.1 TB |

The CPU usage for the database server on System z was rather light during BI Accelerator index creation. The average CPU rates at steady state were 11.84%, 25.00%, and 26.72% for 5 TB, 15 TB, and 25 TB, respectively.

The next three figures show the CPU utilization of SAP BI on System p and SAP BI Accelerator on blades for the three BI Accelerator index creation measurements taken at steady state. The average high load utilization on the charts refers to the average CPU utilization at steady state.

Figure 11-5 shows that the average CPU utilization during the 5 TB BI Accelerator index creation is 28.72% on System p and 75.32% on blades.

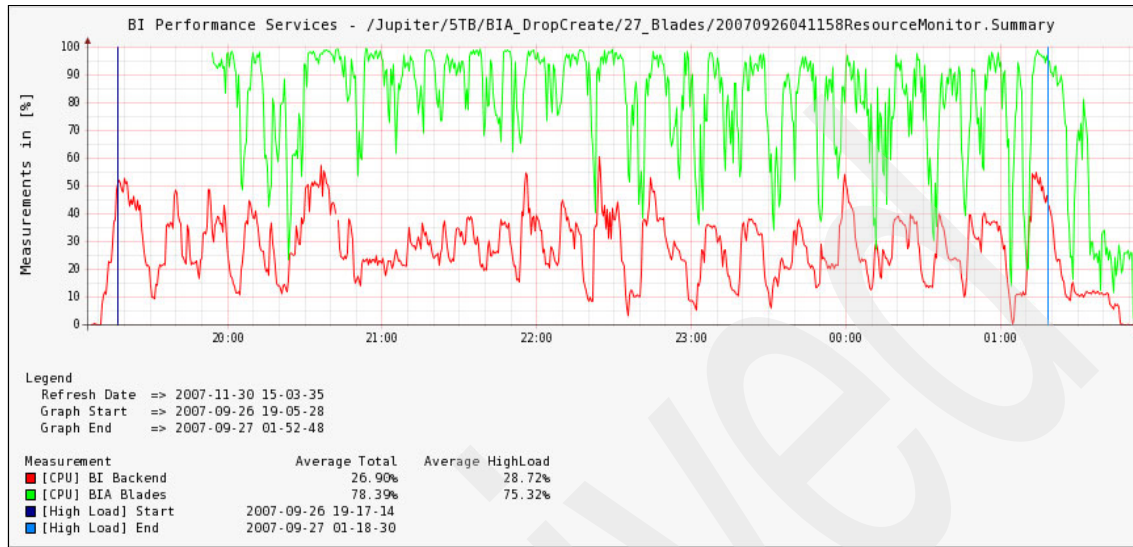


Figure 11-5 Index ceation: average CPU load of 5 TB, 27 blades, 5 InfoCubes parallel

Figure 11-6 shows the average CPU utilization during the 15 TB BI Accelerator Index creation. These are 61.65% for System p and 69.86% for the blades.

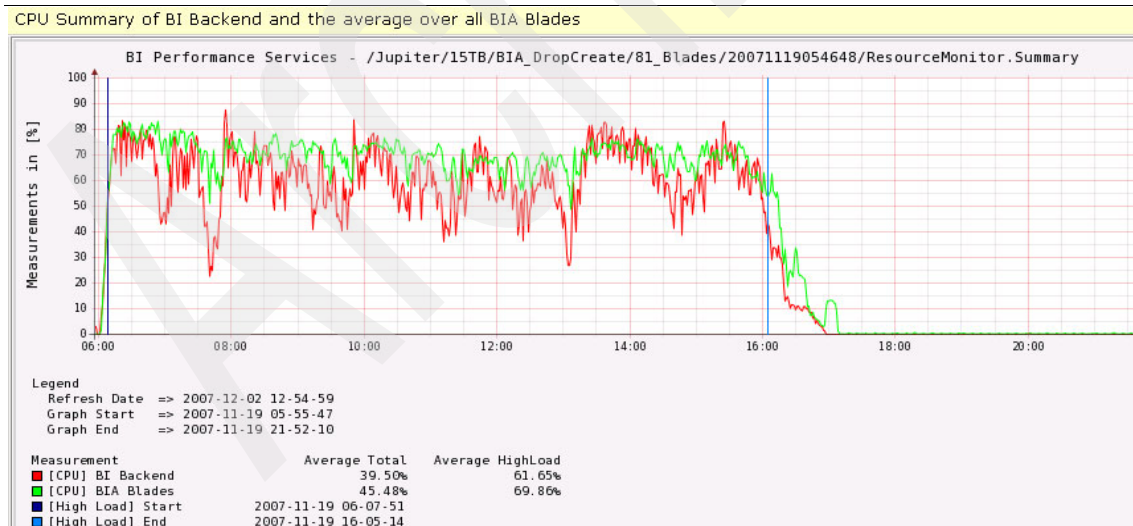


Figure 11-6 Index creation: average CPU load of 15 TB, 81blades, 10 InfoCubes parallel

Figure 11-7 shows the average CPU utilization for the 25 TB BI Accelerator Index creation. These are 75.24% and 57.17% for System p and blades, respectively.

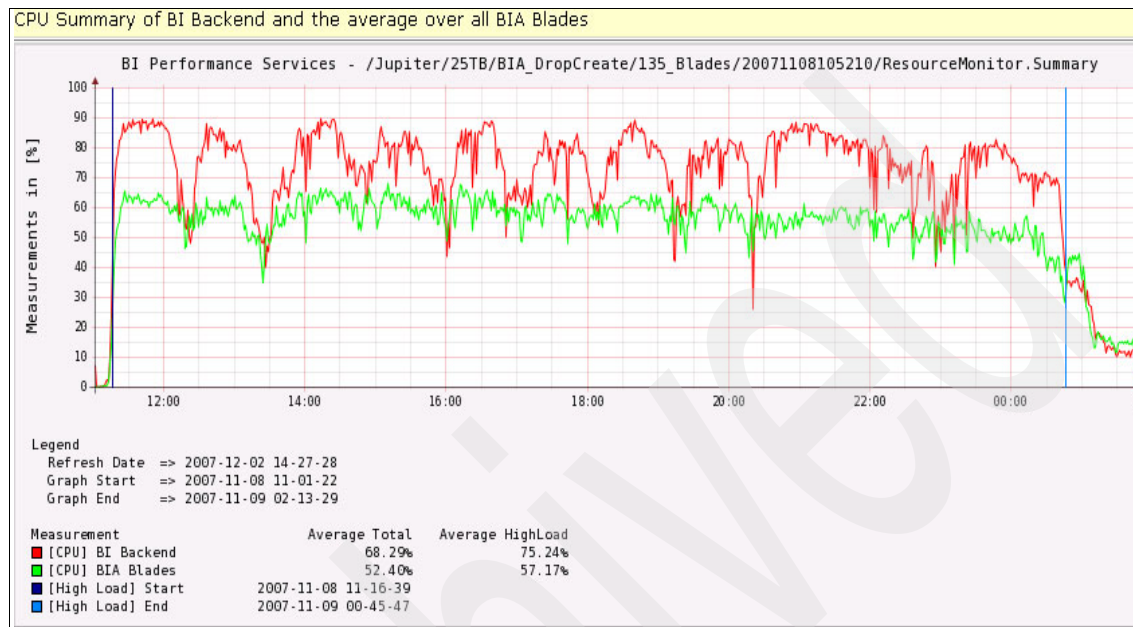


Figure 11-7 Index creation: average CPU load of 25 TB, 135 blades, 10 InfoCubes parallel

Note that the overall blade utilization figure decreased for each larger database size. This is because we added more blades to the accelerator each time that we enlarged the database.

11.4.2 Multi-user query

To test multi-user query scenarios, a load driver was used to simulate multiple users issuing queries to the SAP BI system.

Objectives

The goal for the multi-user query scenarios is to measure the following KPIs for 5 TB on 27 blades, 15 TB on 81 blades, and 25 TB on 135 blades,

- ▶ Query throughput rate in terms of business reports per hour and InfoCube queries per hour
- ▶ Average number of records touched per query in the BI Accelerator
- ▶ Average report response time
- ▶ CPU consumption on blades and BI server

Description

When a query is issued by a user, SAP BI on System p checks whether the data to be queried has been BI Accelerator indexed. If it has, query processing is directed to SAP BI Accelerator. Then if the data is not in BI Accelerator blade memory, it must be read from GPFS into blade memory. In preparation for query processing, all cube data can be pre-loaded from GPFS into blade memory during index creation or off-shift hours. This speeds up the query processing by eliminating I/O during production.

For our query measurements, all BI Accelerator indexes were pre-loaded into memory.

Of all the records in the InfoCubes, only selected rows (QDBSEL) are relevant to the query. As with RDBMS, filtering of all records in the InfoCube is done to identify the ones relevant to the query for further processing. Typically, indexes are used to do this. In this case, BI Accelerator column-based processing and compression have a clear advantage over an RDBMS in identifying and processing the relevant records. The selected records are summarized or aggregated concurrently by a number of blades, then the records are transferred to an OLAP engine in an SAP BI appserver residing on System p. The number of records transferred is referred to as QDBTRANS. At the SAP BI appServer, server rendering is performed before sending the query report to users.

Note that the queries that we designed for this study had high selectivity, with an average QDBSEL of 26–46 million records per query, for our 25 TB tests. Still, the BI Accelerator does tremendously well with an average response time of less than 20 seconds.

Measurement results

One benefit to using BI Accelerator is that no explicit tuning for the cubes is required to prepare for query processing. So there are no aggregates to be defined and no additional secondary indexes are needed.

The number of queries per hour shown in Table 11-3 is the number of business reports (user queries) per hour.

Table 11-3 BI Accelerator query test results: throughput, selectivity, response time

| DB/blade size | Throughput (queries/hr) | Response time (seconds) | QDBSEL (records) | QDBTRANS (records) | Part provider factor |
|------------------|-------------------------|-------------------------|------------------|--------------------|----------------------|
| 5 TB/27 blades | 100,404 | 4.5 | 5,906,534 | 303 | 1.69 |
| 15 TB/81 blades | 100,940 | 4.2 | 22,186,286 | 440 | 1.68 |
| 25 TB/135 blades | 100,712 | 4.2 | 36,767,954 | 520 | 1.69 |

The average CPU utilization on the database server on System z for this workload at steady state was less than 10%. They were 6.7%, 7.2%, and 6.2% for 5 TB, 15 TB, and 25 TB, respectively.

The next three figures show the average CPU load utilization of our SAP BI back end on System p and SAP BI Accelerator on the cluster of blades for the three measurement points during multi-user test scenarios. The average high load is the measured CPU utilization for approximately 15 minutes of peak steady state.

Figure 11-8 shows that the average CPU high load of the SAP BI back end on System p is 79.29%, and SAP BI Accelerator on the cluster of blades is 71.69% for 5 TB and 27 blades.

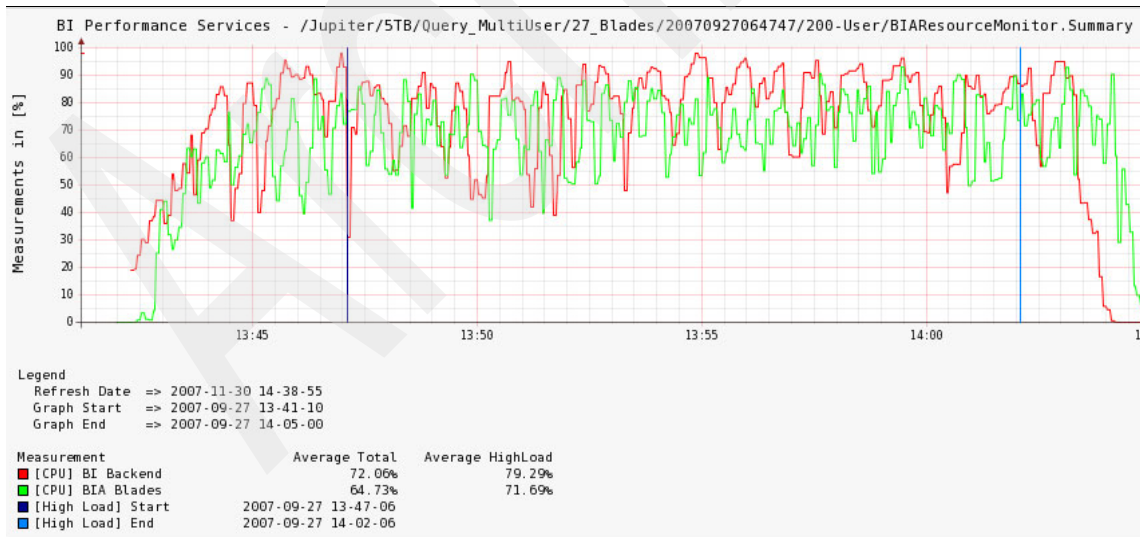


Figure 11-8 Multiuser reporting: average CPU load 5 TB, 27 blades

Figure 11-9 shows that the average CPU high load of the SAP BI back end on System p is 87.07% and the SAP BI Accelerator on cluster of blades is 66.85% for 15 TB and 81 blades.

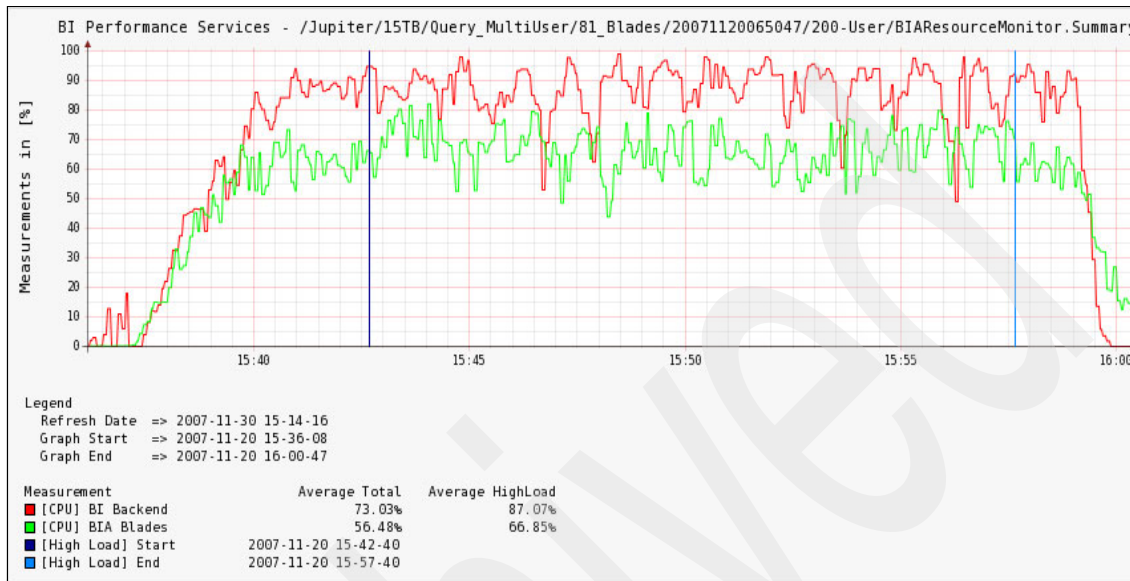


Figure 11-9 Multiuser reporting: average CPU load 15 TB, 81blades

Figure 11-10 shows that the average CPU high load of SAP BI back end on System p is 87.99% and the SAP BI Accelerator on the cluster of blades is 60.08% for 25 TB on 135 blades.

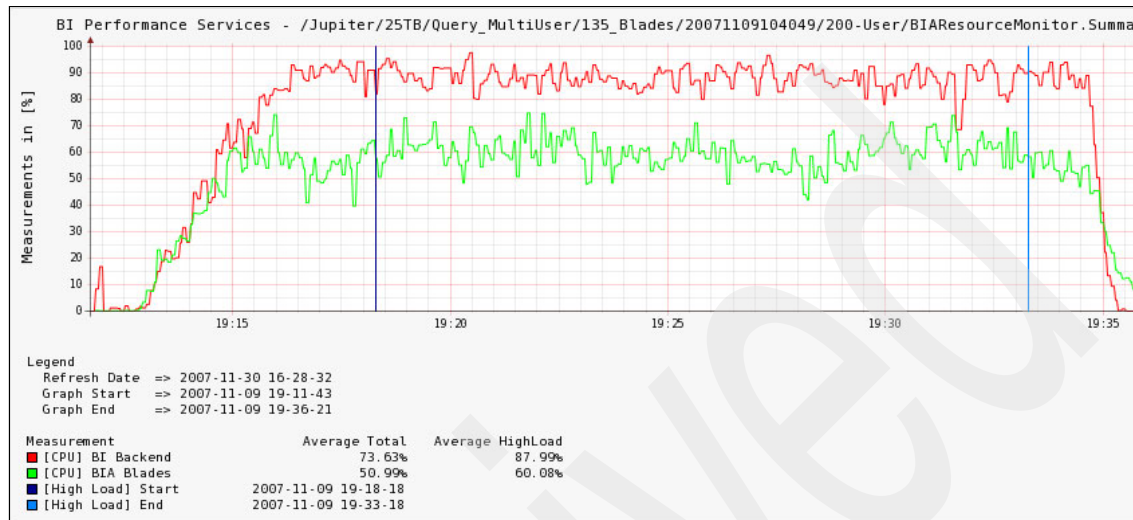


Figure 11-10 Multuser reporting: average CPU load 25 TB, 135 blades

11.5 Analysis and conclusions

Project Jupiter demonstrated that DB2 for z/OS coupled with the SAP BI Accelerator is an ideal solution for business information and data warehousing. DB2 for z/OS, as the premiere IBM relational database management system, offers industry-leading performance, scalability, reliability, and manageability.

The SAP BI Accelerator with DB2 for z/OS on IBM Server Infrastructure can deliver very fast query execution with consistent response times. This was achieved with the use of highly compressed BI Accelerator index data structures and efficient parallel in-memory query processing. During our project with 100% query off-load to BI Accelerator, the need for extensive aggregates and database indexes tuning was completely eliminated, while exploiting DB2's strength as the master data repository. Figure 11-11³ illustrates the key performance indicators (KPIs) of our project.

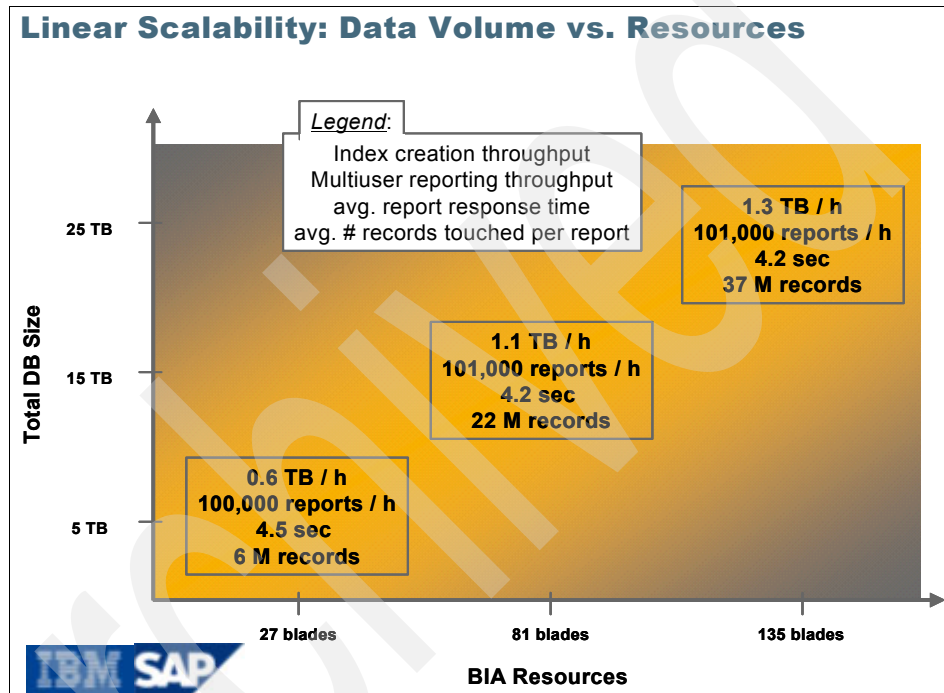


Figure 11-11 Linear scalability: data volume versus resources

Before our project, the largest supported SAP BI Accelerator configuration was the XLarge size that consists of up to 28 blades. Our results showed BI Accelerator query scalability up to 135 (plus 5 spare) blades with no observable scalability limits. Likewise, the BI Accelerator Index load rate can be sustained above 1 TB per hour. These impressive throughput results for BI Accelerator index load as well as for query data retrieval will readily meet the stringent requirements of the largest BI customers today.

³ This information was taken from a presentation by Thomas Becker of SAP, titled, "JUPITER: 25TB Showcase for SAP NetWeaver BI Accelerator and IBM Server Infrastructure," at the IBM Poughkeepsie Customer Briefing Center on December 13th, 2007.

11.5.1 Recommendations for configuration

In summary, the primary objective of project Jupiter was to evaluate BI Accelerator scalability. To ensure that no hardware resource constraints were encountered, we took a conservative approach with some of the hardware capacity planning. Since the scale of our test environment is several magnitudes larger than the 28 blades supported prior to our project, there was a high level of uncertainty about the scaling effects. As a result, a few hardware components were highly oversized. Therefore, the configuration used in our project should not be used as a direct guideline for customer environments. See 11.5.2, “Help getting started” on page 252, for guidance.

One exception was the BI Accelerator sizing used in our project. The general SAP rule-of-thumb for BI Accelerator sizing is to keep blade memory consumption by BI Accelerator indexes below 50% of the total memory on the blade servers. This is to ensure that sufficient memory is available on the blades to allow the query executions without causing performance issues. We adhered to this general guideline for our BI Accelerator blade configuration. The fast consistent query response times achieved during our project have further validated this guideline.

To minimize exposure to known problems, it is important to keep current with SAP support packs and the DB2 maintenance level as identified in SAP notes 364109 (DB2 Put Levels) and 81737 (APAR List).

It can be time-consuming to check that all required program temporary fixes (PTFs) listed in SAP Notes 364109 and 81737 have been applied to a z/OS system. SAP provides a tool called *automated PTF checker* to facilitate the checking. Read SAP note 183311 for details.

11.5.2 Help getting started

To facilitate the BI Accelerator sizing process, SAP has provided an ABAP report that can be used to estimate the memory consumption of the BI Accelerator indexes. Refer to SAP Note 917803 for how to use this tool on the target BI system. Though SAP BI 7.0 is required for the BI Accelerator, the tool can be run on the older SAP BW Releases 3.0, 3.1, and 3.5, as well as BI 7.0. Note that since the tool analyzes all cubes, it may run for a long time and therefore needs to be scheduled accordingly. Contact SAP if there is any issue with the report or information in note 917803.

After a successful run of the tool, you have two options for where to send the report output:

- ▶ E-mail your output to Sizings@us.ibm.com⁴. The Techline ISV sizing team will determine the appropriate size of blade configuration based on the memory requirement for the BI Accelerator index. An e-mail with the certified e1350 BI Accelerator configurations will be sent to the contact person.
- ▶ Contact an IBM Business Partner. Indicate the client's name and a contact name. The IBM Business Partner will work directly with you for sizing and delivery of the BI Accelerator configurations.

Note that for both options, there are standard SAP BI Accelerator configurations, where all options are fixed.

Besides the BI Accelerator blade configuration, it is also important to have a properly sized configuration for the rest of your SAP system landscape, including the database server, application servers, disk storage units, and network topology. For the proper sizing/configuration, contact the IBM technical sales specialist through your local IBM account team.

⁴ IBM Corp. 2006. How do I obtain a sizing for an IBM Systems Solution for SAP BI Accelerator (BI Accelerator) from SAP?

<http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/FQ117071>

Archived



Concepts, activities, and terminology

In this chapter we explain foundational concepts and describe activities that will help you gain the maximum benefit from this book. We also discuss the SAP BI definitions that were used throughout this publication.

Note that a complete overview of all SAP BI definitions is provided in Appendix A, “Glossary for SAP BI” on page 281. It is the most current glossary available at the time of publication.

12.1 Terminology: BI objects and processes

Here we provide you with some relevant terminology.

▶ InfoProvider

In SAP BI, objects that can be analyzed are called InfoProviders. There are two classes of InfoProviders: those that contain physical data (such as InfoObjects, InfoCubes, and DSO objects) and those that do not (such as InfoSets, RemoteCubes, and MultiProviders).

▶ MultiProvider

A MultiProvider provides access to data from several InfoProviders and makes it available for reporting and analysis.

▶ Aggregate

A materialized, summarized view of the data in an InfoCube. This data can be read directly from the database during query execution without having to be built at run time.

▶ Roll-up

This is the process of loading new data packets from an InfoCube into an existing aggregate.

▶ Condense

This refers to using a program to compress the contents of an InfoCube- fact table.

▶ SQL efficiency

This refers to a method to simplify the analysis of a query against an InfoCube, as described in 10.2, “SQL efficiency for InfoCube” on page 211.

▶ Realignment

This refers to the process of synchronizing (realigning) all related fields after a change is made to one data field.

▶ Tablespace partitioning

This refers to the process of reducing the physical size of a large tablespace into multiple smaller physical parts, according to a common key field — frequently time-based. Partitioned tablespaces allow parallel access across multiple partitions. Partitioned tablespaces also allow the accessing of individual partitions without affecting other partitions.

▶ DB2 compression

This process utilizes compression routines, supplied by IBM, to compress DB2 data at the hardware level.

- ▶ SAP compression

This refers to the consolidation and summarization of data from an InfoCube's F- fact table into its E- fact table. Compression results in the package ID being stripped off and all records summarized into the E- fact table.

12.2 SAP BI information model

The SAP BI information model is a basic structural element in the SAP BI architecture. It is designed to enable support to a range of decision makers. To do this, it supports the following conceptual layers of data warehousing:

- ▶ Multidimensional models, which enable views of data required for analytics
- ▶ DataStore (DSO), to hold current data updates from the operational transaction systems of the business
- ▶ Data warehouse, to hold transformed and accurate data that has been integrated from the business processes across the enterprise to enable business decision making

The information model is based on a fundamental building block called an InfoObject. InfoObjects may contain data about customers, sales orders, products, and so on. An InfoObject can be reused in other elements of the information model, such as an InfoCube, DSO, and InfoSource. These elements are discussed in 12.2.1, "Other key elements in the information model" on page 258.

Refer to Appendix A, "Glossary for SAP BI" on page 281, for a complete overview of SAP BI definitions.

InfoObjects also carry metadata that describes the data contained in the InfoObject, such as its origin, history, and technical properties. An InfoObject has three classes of metadata:

Technical metadata This describes technical properties, such as data type and field length.

User metadata This carries information about authorizations.

Business definitions These form the basis for a common understanding of business terms, such as *key performance indicators*.

Business definitions are particularly important in the SAP BI information model. They eliminate semantic discrepancies in data elements across systems and organizational units, and ensure that data is consistent and reliable. SAP BI contains several thousands of InfoObject templates that include business definitions.

Metadata plays a fundamental role in turning data into information. To clarify, information is typically defined as data-in-context. In that process, the metadata provides the context and an understanding of how various data elements are linked. Business rules are applied to the combination of data and metadata to create useful business information. The information model of SAP BI provides consistent and integrated metadata for all objects across the data warehousing process.

Figure 12-1 shows the elements of the SAP BI information model, all of which store metadata. In addition, the PSA, DSO, and InfoCube also store transactional or master data.

Master data is data that remains unchanged over long periods of time. Examples of master data are customer name and address, or an organizational structure. See Figure 12-1.

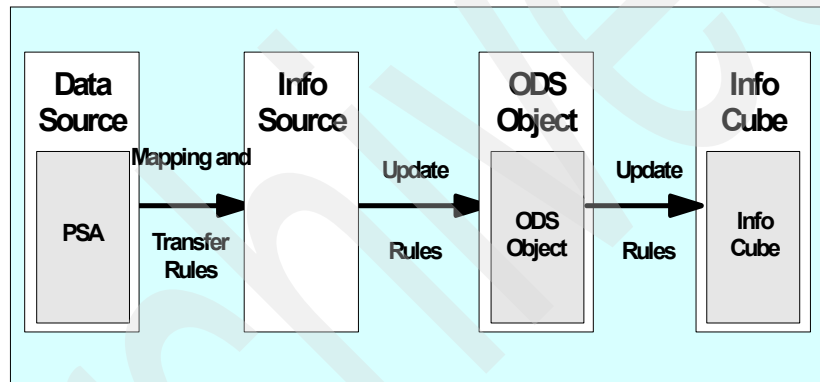


Figure 12-1 Elements of the information model

12.2.1 Other key elements in the information model

In addition to InfoObjects, here is a list of other key elements in the information model:

- ▶ DataSource

Data is transferred into SAP BI in a flat structure; that is, it is a table rather than a multidimensional data structure. DataSources contain the definitions of the source data.

- ▶ Persistent Staging Area (PSA)

In the SAP BI information model, data is physically stored in PSA objects. These are collections of flat tables holding extracted data that has not yet been cleaned or transformed. The PSA is the initial storage area of data

where requested data is saved unchanged from the source system according to the structure defined in the DataSource.

► InfoSource

InfoObjects that belong together logically, from a business point of view, are grouped into InfoSources. InfoSources (and their underlying InfoObjects) can be filled with any data from within the enterprise or from external sources. They can hold both transactional data and master data.

- *Transactional data* is generated from transactions in an Online Transaction Processing (OLTP) system, such as SAP R/3®. Transactional data is quantifiable, and it can be granular.
- *Master data*, such as a customer address or an organizational structure, typically remains unchanged over a long period of time. Master data in SAP BI includes attributes, texts, and hierarchies.

► DataStore (DSO) object

This describes a consolidated dataset from one or several InfoSources. In contrast to the multidimensional data models of InfoCubes, data in DSO objects is stored in flat, transparent, database tables.

DSO object data can be updated into InfoCubes or other DSO objects using a delta update. Data in an DSO object can be analyzed with the SAP BI Business Explorer (BEx) tool, which is included in the business intelligence suite of mySAP™ BI. It is typically used to integrate data that comes from different sources for delta update into InfoCubes and for day-to-day decision making.

► InfoCubes

These are containers that organize data around its multidimensionality, in terms of business dimensions. They are used to answer complex business questions on topics such as revenues per region, revenues per office within each region, year-to-date revenues, and for comparisons with previous periods. InfoCubes can be accessed by the SAP BI Business Explorer for reporting and Online Analytical Processing (OLAP) analysis.

12.3 Dataflow in SAP BI

Data flow is depicted at a very high level in Figure 12-1 on page 258 for an overall understanding. Figure 12-2 provides a more complete view of the possible flow of data in SAP BI. Data is normally loaded into a PSA first, and from there into DSO and InfoCubes. It is also possible to directly load data into the DSO and InfoCubes, and from DSO to InfoCube.

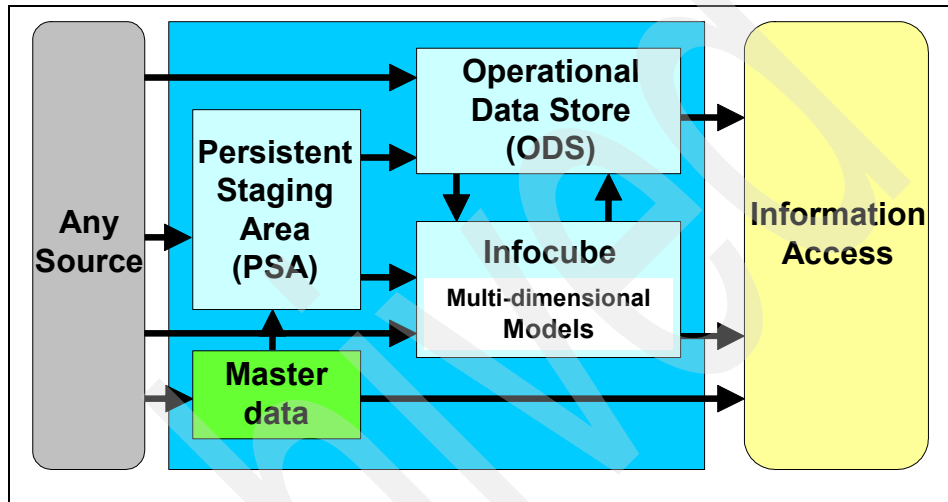


Figure 12-2 Dataflow in SAP BI

Data can be loaded from various heterogeneous data sources, such as:

- ▶ R/3 systems
- ▶ Non-R/3 systems
- ▶ Flat files
- ▶ XML files
- ▶ Other databases (via DB Connect)

ETL process

When data is loaded into SAP BI, it is integrated, standardized, synchronized, and enriched. This is performed through processes known as *extraction*, *transformation*, and *loading* (ETL).

You have to ensure that the ETL process captures and loads the full range of required data, while avoiding an overload of irrelevant data. Data is often stored in different formats, with different data types and different lengths. In addition, a data type may have the same name but a different meaning in the different systems where it is used.

All of these data differences require resolution to enable the data from the various sources to be combined and integrated in the data warehouse and to provide meaningful and accurate information. Data has to be collected and cleansed to eliminate duplication and incorrect values. It then must be enriched so that it is transformed into practical, business-descriptive information.

The transformation, cleansing, and enrichment of data is implemented as follows:

- ▶ A DataSource is assigned to an InfoSource through the *transfer rules* in SAP BI. Transfer rules map the fields of the DataSource to the InfoObjects that make up the InfoSource. A rich library of transformation functions, which represent business logic, can be applied at this point.
- ▶ SAP BI *update rules* handle the subsequent flow of data from InfoSources to DSO objects and InfoCubes.
- ▶ In many cases, the data that is stored in the PSA object (and described in the DataSource) has an incomplete set of metadata. Metadata is added during the creation and bundling of InfoObjects to create the InfoSource.

12.4 Information access

As previously mentioned, in SAP BI, objects that can be analyzed are called InfoProviders. As shown in Figure 12-3, there are two classes of InfoProviders:

- ▶ Those that contain physical data, such as InfoObjects, InfoCubes, and DSO objects
- ▶ Those that do not contain physical data, such as InfoSets, RemoteCubes, and MultiProviders

InfoCubes, DSO objects, and InfoObjects are discussed in 12.2.1, “Other key elements in the information model” on page 258.

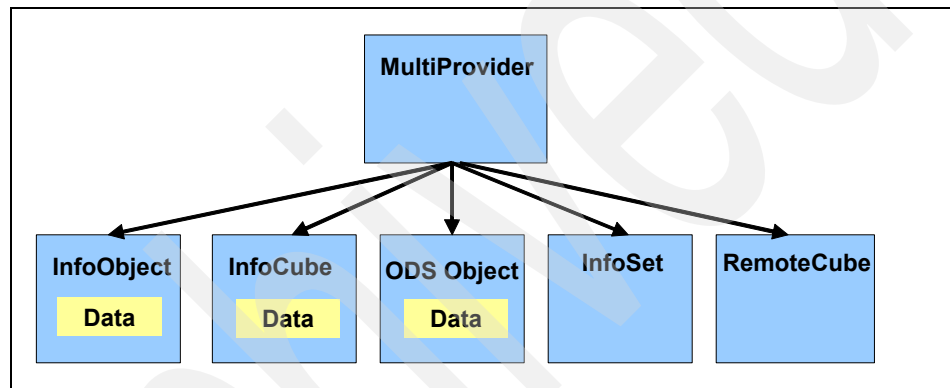


Figure 12-3 SAP BI InfoProviders

A MultiProvider provides access to data from several InfoProviders and makes it available for reporting and analysis. As shown in Figure 12-3, a MultiProvider itself does not contain any data, but can be assembled from different combinations of InfoProviders.

An InfoSet is a semantic layer using DSO objects and master data to create reports from these objects, in particular, joins between these objects. InfoSets are created and changed in the InfoSet Builder. You can define reports based on InfoSets using the BEx Query Designer.

A RemoteCube is an InfoCube whose transaction data is managed externally rather than in SAP BI. Only the structure of the RemoteCube is defined in SAP BI. Data for reporting is read from another system using a BAPI.

12.5 Hierarchies

SAP BI allows the definition of hierarchies on InfoObjects. Hierarchies can be unbalanced, unlevelled, and also time-dependent. Figure 12-4 shows an example of a sales hierarchy that is organized by region.

Hierarchies can be used for reporting. For example, with a structure such as that shown in Figure 12-4, you could retrieve the total sales. Or you could retrieve only the sales result of a particular branch of the sales organization, such as North America. With SAP BI, unlevelled or dynamic hierarchies are stored as parent-child lists.

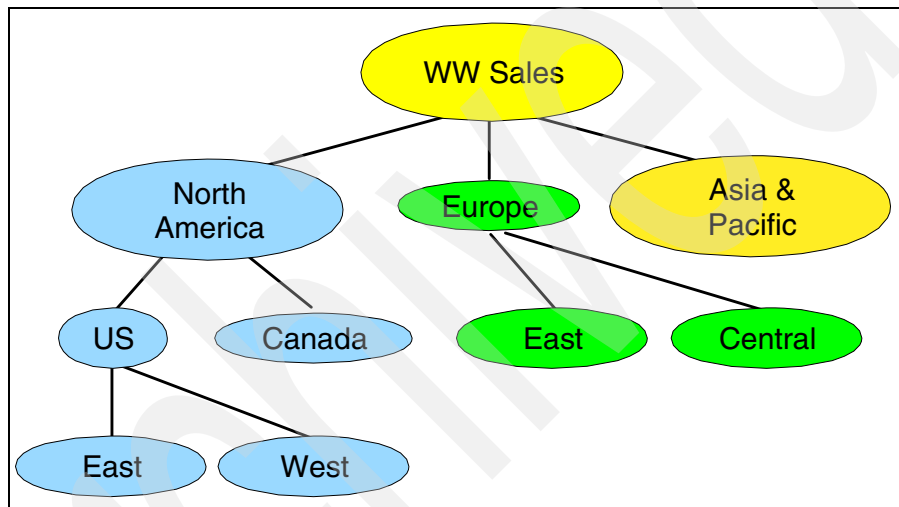


Figure 12-4 Sales hierarchy

The parent-child list that corresponds to Figure 12-4 is shown in Table 12-1.

Table 12-1 Parent-child list of example sales hierarchy

| Predecessor | Successor |
|---------------|------------------|
| WW Sales | Asia and Pacific |
| WW Sales | North America |
| North America | Canada |
| North America | US |
| US | West |
| US | East |

| Predecessor | Successor |
|-------------|-----------|
| WW Sales | Europe |
| Europe | East |
| Europe | Central |

If there are restrictions on hierarchies within a query, the results, that is, the corresponding leafs of the hierarchy nodes that you select, are stored in hierarchy result tables. The names of those tables consist of the prefix for SAP BI tables (/BIO or /BIC) followed by the digits 02. /BIO/0200000001 is an example of the name of a hierarchy result table.

12.6 Extended star schema (InfoCubes)

SAP uses an extended star schema in defining and creating InfoCubes. It contains two types of data used in analysis:

Key figures Sales revenue, fixed costs, sales quantity, or number of employees, and so on.

Characteristics Customer type, fiscal year, period, or region are examples. Characteristics are used to create evaluation groups for analysis.

The underlying InfoObjects that make up an InfoCube are categorized in terms of these two types of data. That is, a given InfoObject represents either key figures or characteristics. A third type of InfoObject, *attributes*, contains metadata describing other InfoObjects.

An InfoCube is represented in the database as a set of relational tables, as shown in Figure 12-5. These are arranged according to the star schema, a technique that organizes data in terms of data facts and business dimensions.

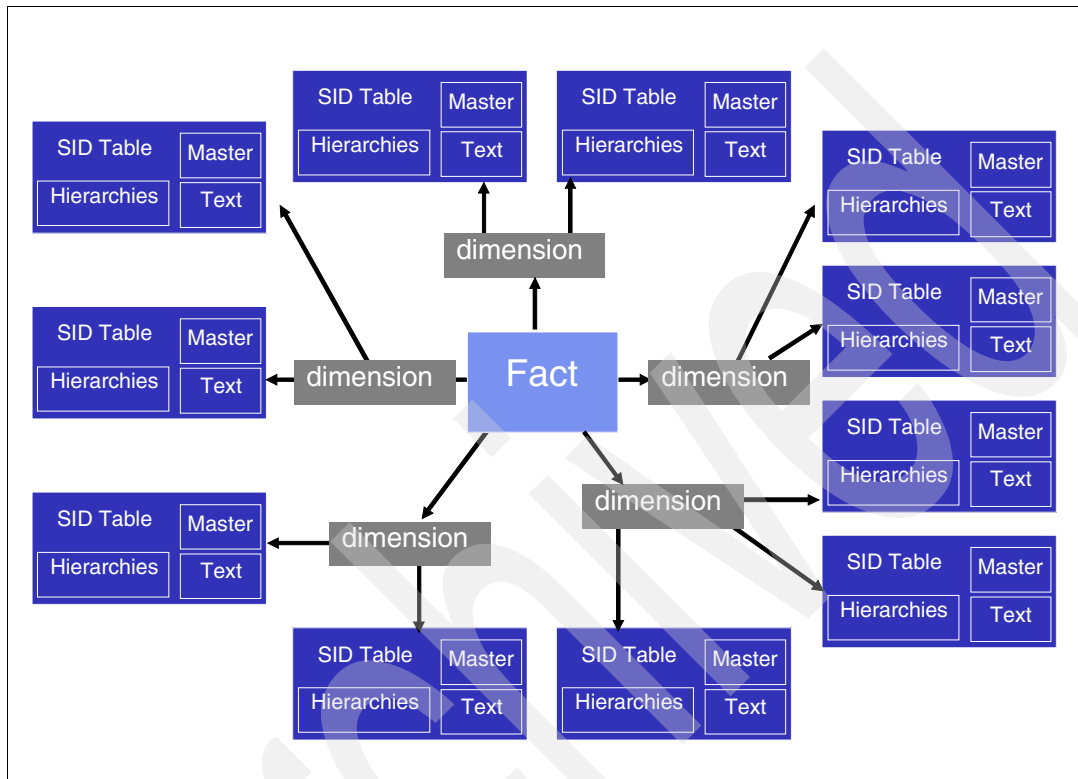


Figure 12-5 Extended star schema

The star schema places several *dimension tables* around a central *fact table*. The E- fact table stores key figures, while the surrounding dimension tables store the characteristics needed for evaluating and reporting on those key figures. fact tables are the largest tables in star schemas, and they may contain billions of entries.

Dimension tables are independent of each other. The E- fact table links the dimension tables and the key figures. To link these tables, dimension identifiers are used. A *dimension identifier* uniquely identifies a particular combination of characteristic values in a dimension table, for example, a certain product and the corresponding product group. Characteristics that are correlated, such as product and product group, are usually put in the same dimension.

An InfoCube in SAP BI can have up to 16 dimensions. By default, every InfoCube has the three standard dimensions *data package*, *time*, and *unit*.

SAP BI uses an *extended star schema*, which builds on the basic star schema by storing master data about attributes, hierarchies, and text, in separate tables that are shared between InfoCubes. This reduces data redundancies because master data is stored only once, and then used by various InfoCubes.

In Figure 12-6, actual characteristic values, such as the name of a region, are shown in the dimension tables. In reality, characteristic values are replaced by surrogate identifiers (SIDs). These are numeric key values (4-byte integers), which are more compact than the characteristic values.

The actual characteristic values are stored in the master table. Therefore, you have foreign key relationships between each characteristic in a dimension table and the corresponding attribute, hierarchy, and text tables. SIDs are used to keep dimension tables as small as possible, since they can also grow very large.

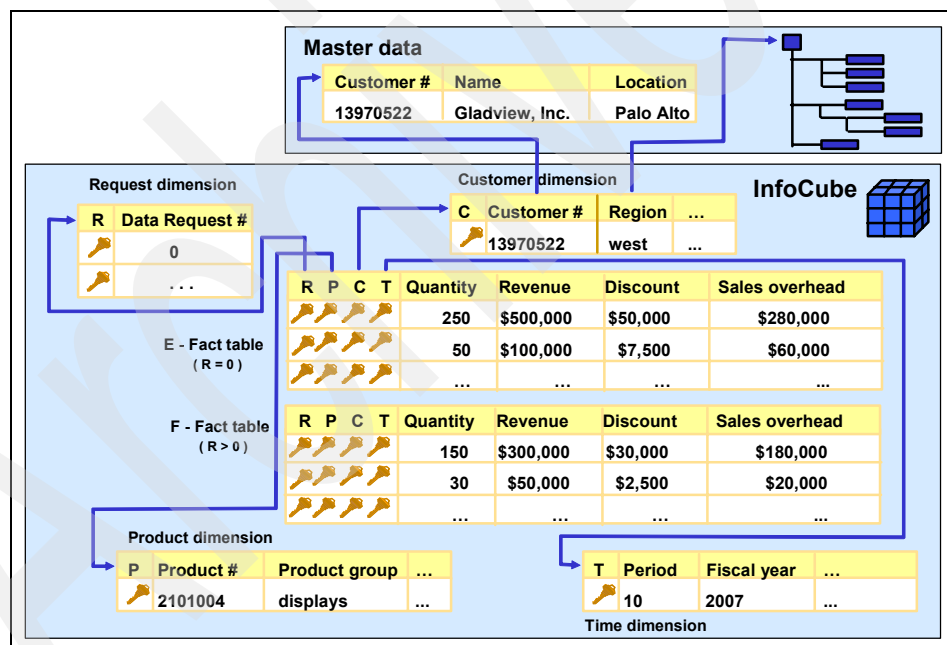


Figure 12-6 Foreign-key relationships

SAP BI provides the option to define a very large dimension as a *line item dimension*. In this case, the corresponding dimension is not stored as a separate table, but rather it is stored directly in the E- fact table. This eliminates the necessary join operation between dimension and fact table during SAP BI query processing, which can provide improved query performance.

12.7 Data load into InfoCube

When data is loaded from PSA or DSO into an InfoCube, it must be transformed from a flat table structure into the extended star schema. Figure 12-7 shows the steps that are required to transfer a new record from a PSA table into an InfoCube and the corresponding master data tables.

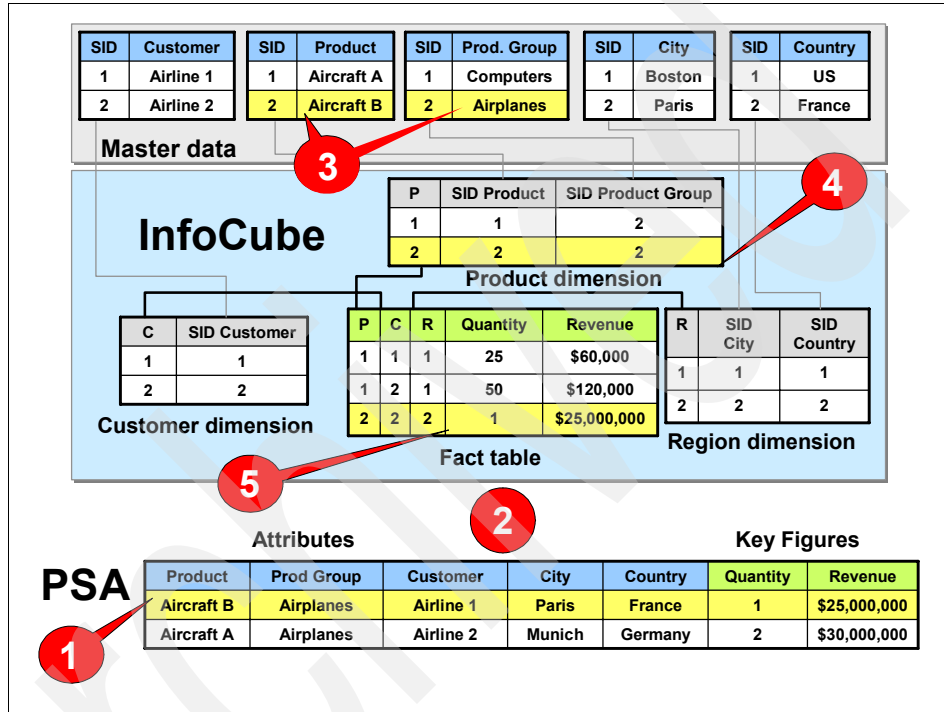


Figure 12-7 Steps during data load into InfoCube

As shown in the figure:

1. The record is selected from the PSA table.
2. Transfer and update rules are applied.
3. The master data tables are checked to see whether the attribute values from the PSA record already exist. If this is the case, the corresponding surrogate identifiers (SIDs) are retrieved from the master data tables. If the attribute values do not yet exist, they are inserted in the master data tables and corresponding SIDs are generated, or the record is skipped. In the given example, the attribute values Aircraft A and Airplanes are inserted in the master data tables.

4. The dimension tables are checked to see whether the combination of SID values that correspond to the attribute values from the PSA record exist in the dimension tables. For example, the product dimension table is checked to see if the SID combination 2/2 exists, which corresponds to the attribute combination Aircraft A/Airplanes.

If the SID combination exists, the corresponding dimension identifier is retrieved. If the SID combination does *not* exist, it is inserted in the dimension table and a corresponding dimension identifier is generated.

5. After the dimension identifiers that correspond to the given attribute values are retrieved/generated, a new record is inserted into the E- fact table. It contains the key figures from the PSA record and the corresponding dimension identifiers.

12.8 Aggregates

Aggregates represent another technique for improving query performance. They are materialized, summarized views of the data in an InfoCube, and store a subset of InfoCube data in a redundant form. When an appropriate aggregate for a query exists, the summarized data can be read directly from the database during query execution instead of having to perform this summarization during runtime. In SAP BI, the system generates suggestions for creating optimal aggregates. The system administrator can then decide whether to create those aggregates.

Aggregates reduce the volume of data to be read from the database, speed up query execution time, and reduce the overall load on the database. To use aggregates you must build and maintain them, and they require additional space in the database. Aggregates are very similar to the automatic summary tables defined in DB2.

An easy way to conceptualize an aggregate is to think of it as providing a benefit similar to adding another index to a database table. Aggregates require no special knowledge by the user, and in fact are completely transparent to the user. The only way an user might recognize the existence of an aggregate is by the performance gain that is observed.

12.8.1 Aggregate example

The contents of an example InfoCube are shown in Table 12-2. This, is an example of an aggregate and is based on a very simplified InfoCube that has the two characteristics country and customer, and the key figure sales.

Table 12-2 *Simplified InfoCube*

| Country | Customer | Sales |
|---------|-----------------|-------|
| USA | Blue Soft | 10 |
| Germany | Ocean Networks | 15 |
| USA | Funny Duds | 5 |
| Canada | Ocean Networks | 10 |
| Canada | Thor Industries | 10 |
| Germany | Funny Duds | 20 |
| USA | Blue Soft | 25 |

Table 12-3 shows another sample aggregate that only contains the country characteristic, and the key figure, sales.

Table 12-3 *Aggregate: country*

| Country | Sales |
|---------|-------|
| USA | 40 |
| Germany | 35 |
| Canada | 20 |

These example aggregates could be used by a query that reports the sales by country or total sales. The aggregates could also be used for reports according to a navigation attribute for the characteristic country or according to a hierarchy containing the countries.

If during query navigation there is a drilldown, or a selection is made for a particular customer, these aggregates would not be used. This is because the aggregates do not contain any detailed information about a particular customer, only summarized customer data.

The next example is an aggregate that is grouped according to the nodes of a hierarchy level. The hierarchy is shown in Figure 12-8.

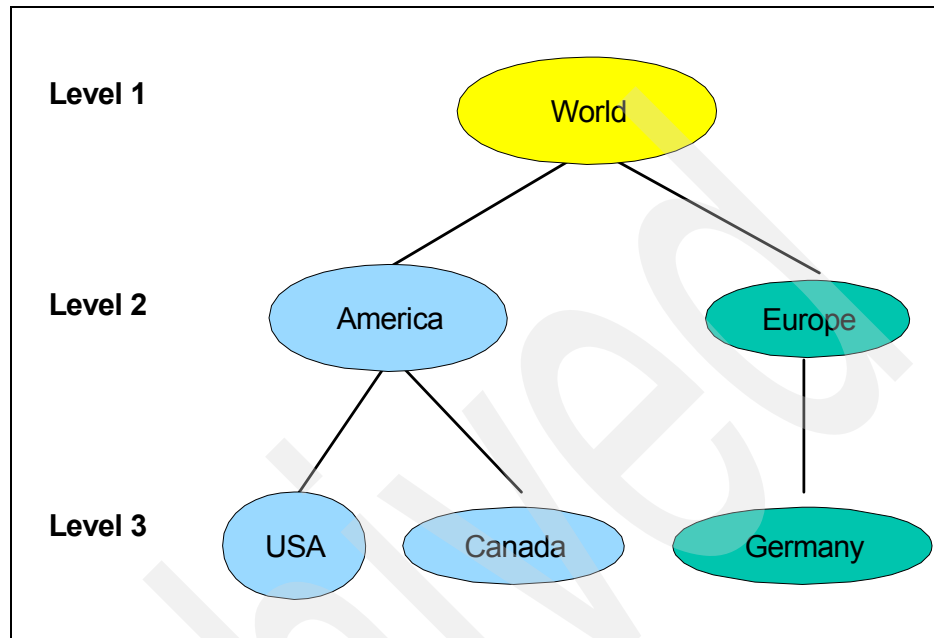


Figure 12-8 Hierarchy of country

The aggregate is shown in Table 12-4. That aggregate can be used by queries that report on sales for a hierarchy node on level 2 or higher. In the example shown in Figure 12-8, those are the nodes labeled Europe, America, and World.

It can also be used by queries that have this country hierarchy as a presentation hierarchy in the drilldown, but the drilldown goes no lower than the second level.

Table 12-4 Aggregate of Country hierarchy level 2

| Customer | Sales |
|----------|-------|
| America | 60 |
| Europe | 35 |

12.8.2 Maintaining aggregates

There are a number of activities required to maintain aggregates. For example, they must be loaded (filled with data), calculated, activated, and updated.

Activating and filling

In order to use an aggregate in the first place, it must be defined, activated, and filled. When activated, the required tables are created in the database from the aggregate definition. Technically speaking, an aggregate is actually a separate BasicCube with its own fact table and dimension tables. However, aggregates might share the dimension tables of the corresponding InfoCube if those tables have the appropriate structure.

Upon creation, every aggregate is assigned a technical name, which is a unique number consisting of six digits where the first digit is always set to 1. The table names that make up the logical object that is the aggregate are then derived in a similar manner, as are the table names of an InfoCube.

For example, if the aggregate has the technical name 100001, the E- fact tables have the following names: /BIC/E100001 and /BIC/F100001. Its dimensions, which are not the same as those in the InfoCube, have table names such as /BIC/D100001P, /BIC/D100001T, and so on.

When you *fill* an aggregate, you load it with data. This action can only be triggered from the aggregate maintenance. Also note that an aggregate can be filled from the data of a larger aggregate that is already filled. This means that very highly summarized aggregates, as a rule, can quickly obtain data from other aggregates. In contrast, it can take a long time to build aggregates from the InfoCube.

Because of this, all aggregates are filled in background processes. If there are several aggregates scheduled to be filled in one process, a hierarchy sequence for the aggregates is determined first, and it is then processed sequentially. This guarantees that very highly summarized aggregates are built from the more detailed aggregates.

Roll-up

If aggregates are defined, new data packets that are loaded into the InfoCube cannot be used for reporting until they are written to the aggregates by a roll-up. During this process you can continue to report using the data that existed prior to the recent data load.

There are three different options for rolling up data:

- ▶ You can set up the InfoCube so that every data packet is automatically rolled up into the aggregate if it is technically correct and the quality has been ensured.
- ▶ The roll-up can be also started manually from the Administrator Workbench. This is appropriate if the data of several packets form a logical unit and are only valid if they are released together.

For example, assume that different plants deliver their data at different times to be loaded into an InfoCube. It would not be valid to make any of the data visible until all plants have delivered their data to be loaded into the InfoCube as a consistent set.

- ▶ Another option to start the roll-up manually is by executing the program `RSDDK_AGGREGATES_ROLLUP`. This program can either be scheduled periodically or an event chain can be constructed to include aggregate roll-up.

Change-run: hierarchy-attribute realignment run

If you change master data, navigation attributes or hierarchies usually change as well. We therefore recommend that you adjust the data in the aggregates after loading the master data. To assure that reporting delivers consistent results, the master data and hierarchies are kept in two versions:

- ▶ The active version, where you can see the query
- ▶ The modified version, which at some point will become the active version

The change-run, also called hierarchy-attribute realignment run, adjusts the data in the aggregates and turns the modified version of the navigation attributes and hierarchies into an active version. In almost every phase of the change-run, you can continue reporting on the old master data and hierarchies.

You can either start the change-run manually in the Administrator Workbench or with the program `RSDDS_AGGREGATES_MAINTAIN`. You can give the program a list of characteristics and hierarchies that are to be taken into account for the change-run. By default, all those characteristics are taken into account whose master data you loaded or changed manually, and all the hierarchies that were marked for the realignment run.

The change-run takes places in different phases:

- ▶ A list is generated of all navigation attributes and hierarchy levels that have changed. Then the relevant aggregates that are to be adjusted are generated from this list.
- ▶ The system adjusts the aggregates and calculates the percentage of change per aggregate. If the percentage is small, the changes are calculated explicitly. If the percentage is large, the aggregate is completely rebuilt in a temporary table.
- ▶ Master data and hierarchies are activated.
- ▶ The temporary aggregate tables are renamed. When running on DB2, the indexes are also renamed at this stage.

12.9 Compression of requests

Data is loaded into an InfoCube by using smaller pieces, called *requests*. The data is first stored in the F- fact table of the InfoCube. Requests can be deleted from the F- fact table if, for example, the data is inconsistent. If the quality status of a request is appropriate, the request can be compressed, which significantly reduces the required storage space for the data.

This form of compression is called SAP compression and is a consolidation and summarization of data from the InfoCube's F-fact table into its E- fact table. Compression results in the package ID being stripped off and all records summarized into the E-fact table.

During compression, data is transferred from the F- fact table to the E- fact table of the InfoCube or aggregate, as shown in Figure 12-9. Compressed requests can only be deleted by deleting the entire contents of the InfoCube.

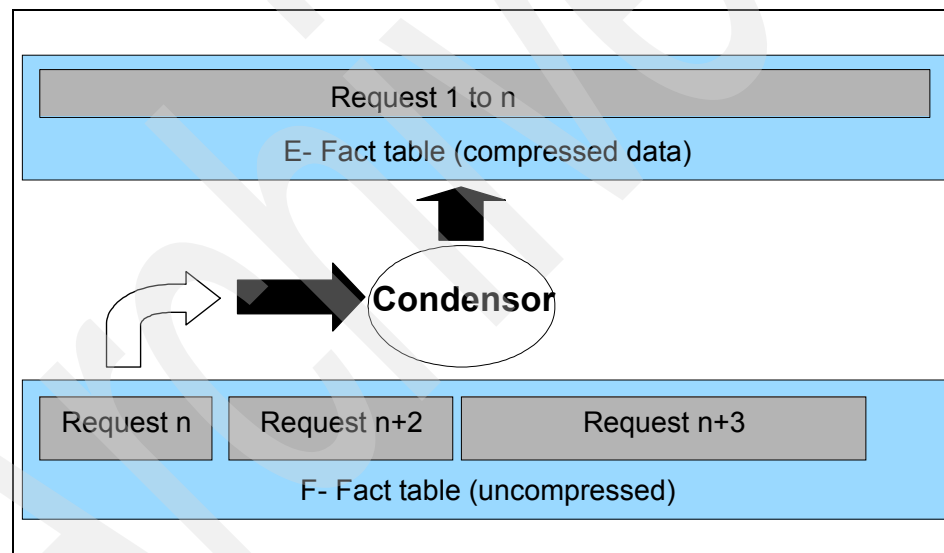


Figure 12-9 Compression of requests

12.10 DataStore (DSO)

A DSO object stores consolidated transaction data from one or several InfoSources. In contrast to the multidimensional data models of InfoCubes, data in DSO objects is stored in flat, transparent, database tables. DSO object data

can be updated into InfoCubes or other DSO objects using a *delta update*. Data in an DSO object can be analyzed with the SAP BI Business Explorer.

A DSO object contains a key (for example, order number) as well as data fields (key figures). As opposed to InfoCubes, fields in an DSO object can be overwritten. This is useful to process document structures, for example.

If you change documents in an OLTP system, the changes not only affect numeric fields such as order quantity, but also non-numeric fields such as status and delivery date. To reflect these changes in the DSO object, the relevant fields in the DSO objects must be overwritten and set to the current value.

Figure 12-10 shows an example of two layers of DSO objects that are used to update order and delivery information. These DSO objects allow you to determine, for example, which orders are open, partly delivered, or completely delivered.

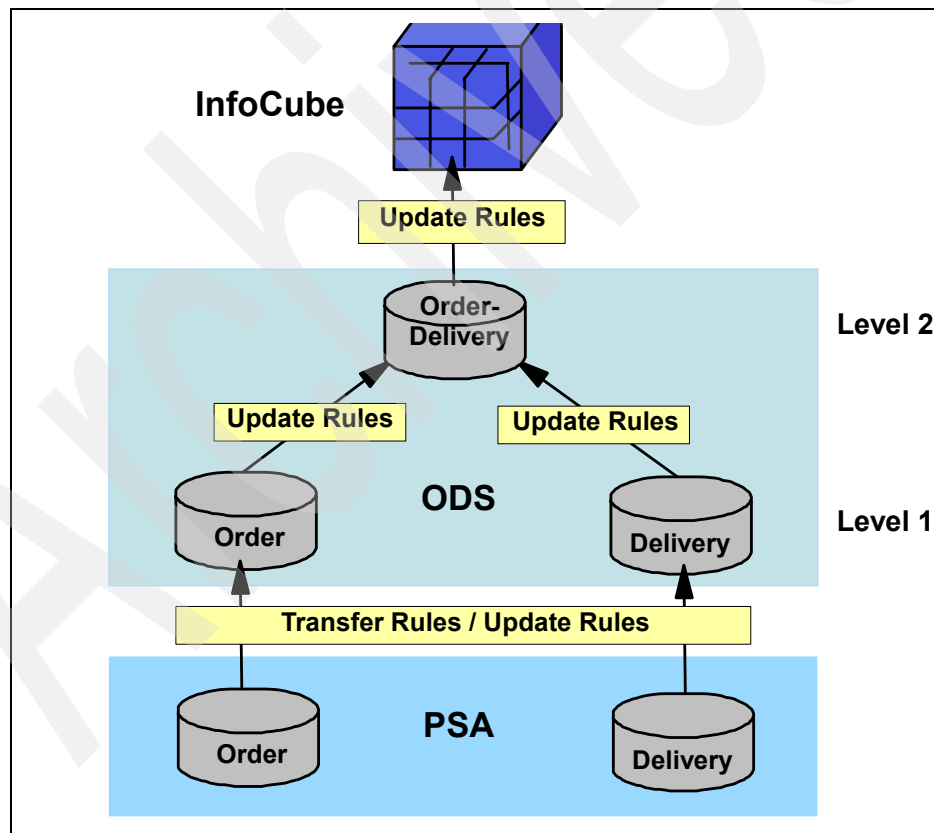


Figure 12-10 Using multiple layers of DSO objects to integrate data

The DSO objects can store the data at the same level of granularity as offered from the PSA (that is, the source system) because aggregation can be performed later during reporting.

The example shows how the DSO can be used to integrate data that describes the same processes but that potentially comes from different source systems (DataSources). The data can be loaded into SAP BI and stored in different PSA tables, each having their own transfer structure. Integration is achieved by applying transfer structure specific rules (that is, transfer rules) while transferring the data into the same consolidated structure (communication structure of an InfoSource) of an DSO object.

Thus the DSO objects of the inner level of the operational data store (level 2 in Figure 12-10 on page 274) offer data that is subject-oriented, consolidated, and integrated, with respect to the same process that is operated on different source systems.

The first-level DSO objects are part of the foundation of the data warehouse because it represents the transactional data archive. This data can be accessed to create new reporting scenarios based on integrated and consolidated data from the past.

On the database level, every DSO object consists of three transparent tables:

- ▶ Active table

The data in this table is used for reporting.

- ▶ Activation queue

This table contains data that is new or modified since the last activation.

- ▶ Change log table

The change log is used for the delta update from the DSO object into other DSO objects or InfoCubes.

Figure 12-11 shows the steps that occur during activation of DSO data. In this example we assume that document 123, with an amount of 100, is already loaded into the activation table of the DSO object.

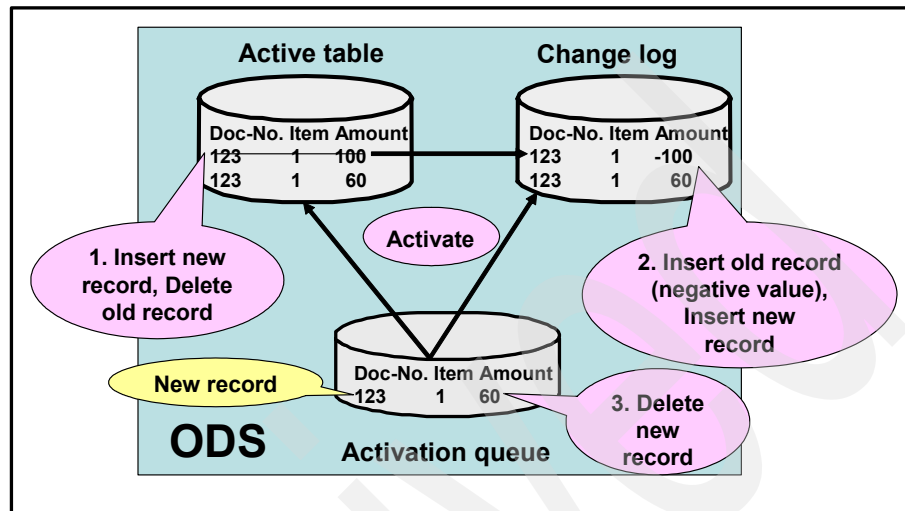


Figure 12-11 Activation of data in DSO

When data is activated in the DSO, the following steps occur:

1. The new/modified data is transferred into the activation queue. The corresponding old record is deleted from the active table.
2. The old record of the active table is saved as a negative (-100) value in the change log, and the new record is also stored in the change log (60).
3. The new/modified data is deleted from the active table.

If all the records are activated, you can propagate the changes from the change log to the datasets of the related DSO Objects or InfoCubes, in a separate step. The amount is therefore reduced by 40 in the related data targets in the example.

12.11 SAP BI naming conventions

Table 12-5 lists the SAP BI naming conventions.

Table 12-5 List of SAP BI naming conventions

| Name | Type | Description |
|------------------|-------|---------------|
| /Bix/F<InfoCube> | Table | F- fact table |

| Name | Type | Description |
|--------------------|------------|--|
| /Blx/E<InfoCube> | Table | E- fact table |
| /Blx/D<InfoCube>n | Table/view | Dimension table of dimension n
n= P,T,U,1...9,A,...D |
| /Blx/M<InfoObject> | View | Master data attribute (join of P + Q) |
| /Blx/P<InfoObject> | Table | Master data attributes (time-dependent) |
| /BlxQ<InfoObject> | Table | Master data attributes (time-dependent) |
| /Blx/T<InfoObject> | Table | Master data texts table |
| /Blx/X<InfoObject> | Table | Attributes SID table (time-dependent) |
| /Blx/Y<InfoObject> | Table | Attributes SID table (time-dependent) |
| /Blx/I<InfoObject> | Table | Hierarchy SID Table |
| /BI0/02<8-digits> | Table | Hierarchy reporting nodes (cached) |
| /BI0/03<8-digits> | View | View on hierarchy nodes
View on reporting query |
| /BI0/01<8-digits> | Table | Table on hierarchy nodes
Temporary SID table (2.x)
Query splitter intermediary results |
| /BI0/06<8-digits> | Table | Temporary SID table (3.x) |
| /BIC/B<10-digits> | Table | PSA/DSO Object table |
| /Blx/S,InfoObject> | Table | SID table |

12.12 The SAP BI functional components

So far we have discussed a number of key elements used in SAP BI. In this section we show how those elements are combined to form the functional components that comprise the SAP BI architecture.

Figure 12-12 on page 279 shows the primary functional components of SAP BI:

▶ Business Explorer

This tool is used to define and execute SAP BI queries. It comprises the following tools:

- BEx Analyzer and BEx Browser are two tools used to execute SAP BI queries. The BEx Analyzer is based on Microsoft Excel.
- Query Designer is used to define and design the queries.
- Web Application Designer is used to define and design the layout of Web-based reports.

▶ OLAP engine

When a user executes an SAP BI query, or query navigation takes place, the OLAP engine processes the query. It splits the request into several database queries. The system then looks for the best possible aggregate for each of the database queries. It generates SQL statements, which are then executed on the underlying database system.

If an SAP BI query contains restrictions on hierarchies, the OLAP engine resolves those restrictions and determines the corresponding leaf nodes of the hierarchy subtree. Finally, it consolidates the results of the executed SQL statements and sends this information back to the Business Explorer.

▶ Staging engine

This engine provides all processing that is collectively described as extraction, transformation, and loading (ETL). This is the process of accessing source data, cleaning, transforming, and integrating it, and then loading it into the data warehouse. The ETL processing is discussed in 12.3, “Dataflow in SAP BI” on page 260.

▶ Administrator Workbench

This allows the BW administrator to perform all data warehouse modeling and maintenance tasks within a single, unified environment. The workbench allows the administrator to back up and manage objects, define new objects and data flows, and to manage security, archiving, and other tasks.

The Administrator Workbench has the following components:

- Meta data maintenance, to allow the administrator to specify and maintain the InfoCubes, DSO definitions, and technical data
- Scheduler, to schedule the transfer of data from the source data systems at regular intervals
- Data load monitor, to supervise the load and staging processes
- Data access monitor, to obtain statistics on BW usage

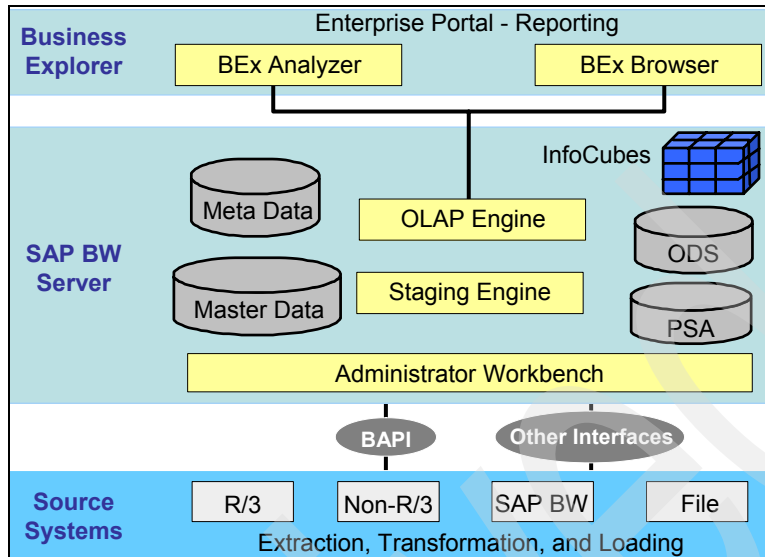


Figure 12-12 SAP BI functional components

The communication between the SAP systems is based on standard TCP/IP. In addition, SAP provides Remote Function Call (RFC) and Application Link Enabling (ALE) protocols. Using the ALE distribution models, selected business objects can be synchronized in different SAP Systems. This synchronization of the business objects is performed via the Business Application Programming Interface (BAPI). The business objects are transferred using the intermediate documents (Docs).

12.13 SAP BI business content

To accelerate the SAP BI implementation, SAP provides a component called *business content*. It incorporates best practices and the business process knowledge of SAP.

The content, depicted in Figure 12-13, includes preconfigured programs for extracting data, data models, and a wide range of predefined templates for reports and analysis. It supports both industry-specific and domain-specific models and templates in areas such as customer relationship management and supply-chain management. It includes predefined BW elements such as:

- ▶ + 14,000 InfoObjects
- ▶ + 530 DSO Objects
- ▶ + 710 InfoCubes
- ▶ + 140 MultiProviders
- ▶ + 3,600 queries
- ▶ + 1,900 workbooks
- ▶ + 800 roles

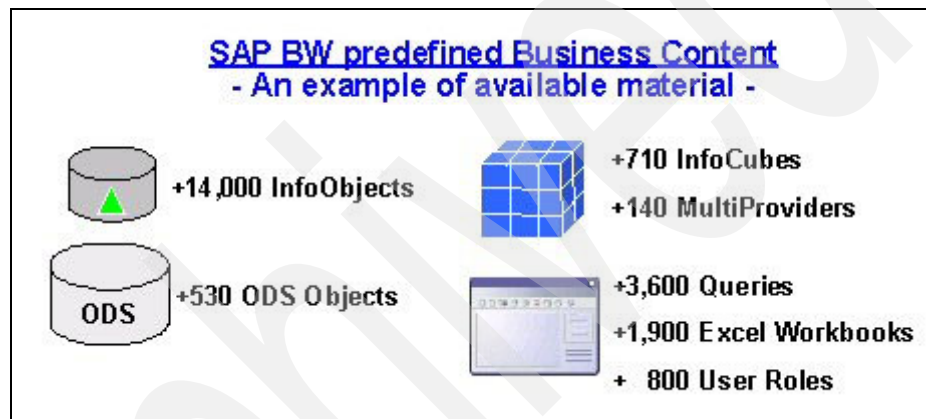


Figure 12-13 SAP BI business content

This content can be used as is or it can be modified to meet any particular requirements. It represents a wealth of supplied content that helps speed the implementation and shorten the time to realize benefits.

As can be seen, SAP BI is a robust solution for data warehousing and business intelligence. This has been a brief overview to supply sufficient information to enable a better understanding of the remaining contents of this book. There is more detailed information available concerning this offering that can be accessed from your IBM representative or directly from SAP.

Glossary for SAP BI

In this appendix we include the glossary from SAP BI. You can also find this SAP BI Glossary on the SAP BI home page (see “Related publications” on page 347).

ABC Classification

Web Item to classify objects (Customers, Products or Employees) based on a particular measure (Revenue or Profit) using certain classification rules.

account model

Type of data modeling in which every business key figure corresponds to a special characteristic (account or item characteristic) of the data basis. The key figure values themselves are saved in an individual, generically technical key figure.

Therefore, an individual key figure within an account model contains values, which for it alone reflects no business facts. The business meaning of a key figure value (for example sales, number of employees maintenance costs) can only be determined in conjunction with the attribute of the account characteristic.

A single data record in the account model is made up of the fields for classifying characteristic values (for example fiscal year, version, region), the account or item characteristic, and one field typically for two key figures (amount and quantity). In this way, such a data record is extremely compact. However,

considerably more data records are required for the same data volume as in the key figure model.

ActiveX Data Objects

A logical object model for programmatically accessing a variety of data sources through OLE DB interfaces. ADO, provided by Microsoft, is the foundation for Microsoft's ADO MD extension, upon which the BI ODBO Connector is based.

ActiveX Data Objects Multidimensional

A logical object model provided by Microsoft that facilitates easy access to multidimensional data by extending ADO with objects specific to multidimensional data, such as cubes and cellsets. Like ADO, ADO MD uses an underlying OLE DB provider to gain access to data. The BI ODBO Connector uses ADO MD to support connectivity to OLAP data sources.

Ad-hoc Query Designer

Web item that enables you to create and change ad hoc queries in a Web application. You can use the Web item Ad-hoc Query Designer in the Web Application Designer to design Web applications in which you can create or change queries.

after-import method

Method that is used in connection with the transport of an object into a different system. The after import method is called in the target system after the object has been imported. The after import method is object specific and, therefore, you have to rewrite it for every object type. You might typically use the after import method to activate the imported object, in order to integrate it in a new context in the target system.

aggregate

Stores the dataset of an InfoCube redundantly and persistently in a summarized form on the database.

When building an aggregate from the characteristics and navigation attributes from an InfoCube, you can group the data according to different aggregation levels. Remaining characteristics that are not used in the aggregate are summarized.

New data is loaded into an aggregate using logical data packages (requests). You differentiate between filling and rolling up with regard to loading data.

Aggregates enable you to access InfoCube data quickly for reporting. Thus, aggregates help to improve performance.

aggregation level

Selection of characteristics and navigation attributes of an InfoCube, from which aggregates are constructed.

You have the following aggregation options:

- ▶ All characteristic values (“*“): Data is grouped by all values of the characteristic or the navigation attribute.
- ▶ Hierarchy level (“H“): Data is grouped by hierarchy level nodes.
- ▶ Fixed value (“F“): Data is filtered according to a single value.

alert monitor

A monitoring tool for displaying exceptions whose threshold values have been exceeded or have not been reached. The critical exceptions are found in background processing with the help of the reporting agent. They are then displayed in the alert monitor as a follow-up action. Exceptions are displayed in the BEx Analyzer as well as in the reporting agent scheduler of the Administrator Workbench. Exceptions can also be displayed as an alert monitor in a Web application.

Significance in BEx Web applications:

Web item that displays the query views, which were found in background processing (with the help of the reporting agent) as a list or hierarchy in a Web application. You can jump between query views and see at a glance conspicuous key figure values that deviate from the defined threshold values. You can also set filters.

analysis process

Calculation of data transformations on mass data within an analytical application.

An analysis process enables you to:

- ▶ Read And Combine Data In BI From Different Data Sources
- ▶ Switch Transformations Sequentially
- ▶ Preview Calculated Data At A Specific Process Position
- ▶ Save The calculation results.

analysis process designer

Tool used to model an analysis process.

The analysis process designer provides a graphical interface to model analysis processes. Analysis process is built using nodes and data flow arrows.

The nodes stand for data sources, transformations and data targets. The dataflow arrows model the sequence in which the data is read and transformed.

and-process

Combined process of the process chain maintenance.

When you use an and-process in the process chain maintenance, the application process is started only when all events in the previous process, on which this process is waiting, have been triggered.

application process

A process that is automated in the process chain maintenance.

Example:

A data loading process or an attribute change run.

attribute

InfoObjects that are logically assigned or subordinated to a characteristic and that cannot be selected in a query.

Example:

For a cost center, you could assign the attributes 'Cost

- ▶ 'Cost Center Manager' (characteristic as attribute)
- ▶ 'Size of Cost Center in Square Meters' (key figure as attribute).

authorization

The authority to execute a particular action in the SAP System.

Each authorization references an authorization object and defines one or more permissible values for each authorization field contained in the authorization object. Authorizations are combined in profiles, which are entered in a user's master record.

axis data set

Combination of different values on an axis.

Data sets for elements (members) of an axis dimension are used when defining MDX queries.

basic characteristic

Characteristic, which appears as independent characteristic in a characteristic relationship. In a characteristic relationship, a relationship is defined between the

basic characteristic and the target characteristic, where individual values of the basic characteristic are assigned to certain valid values of the target characteristic.

BEx Broadcaster

Tool for precalculating and distributing queries, Web templates and workbooks. With the BEx Broadcaster you can create precalculated documents and online links and distribute them by e-mail or publish them to the Enterprise Portal.

BEx Download Scheduler

Assistant for downloading precalculated Web templates from the BIBI server to the PC as HTML pages.

BEx Information Broadcasting

Function that allows you to make Business Intelligence information available to a broad spectrum of users.

Information is distributed either by e-mail or in the Enterprise Portal. The Enterprise Portal serves as the central point of entry and allows you to use knowledge management and collaboration functions when working with content from SAP BI.

BEx Map

The Business Explorer's geographical information system (GIS). The BEx Map allows you to display data with a geographical connection (characteristics such as customer, sales region and country), together with key figures relevant for business management, on a map. You can use the map to evaluate this data.

BEx mobile intelligence

The process of using Web applications on mobile devices that have an online connection to a BI system.

BEx Portfolio

KM navigation iView that contains the layout "broadcasting". This is tailored to the specific needs of users who use Business Intelligence content in the Enterprise Portal.

The BEx Portfolio links to a generally accessible KM folder under /documents.

BEx Query Designer

Tool for defining queries that are based on a selection of characteristics and key figures (InfoObjects) or on reusable InfoProvider structures. In the BEx Query Designer, you can parameterize queries by defining variables for characteristic

values, hierarchies, hierarchy nodes, texts or formulas. You can specify the selection of InfoObjects more precisely by:

- ▶ Restricting characteristics and key figures by characteristic values, characteristic value intervals and hierarchy nodes
- ▶ Defining calculated and restricted key figures for reuse
- ▶ Defining structures for reuse
- ▶ Defining exceptions
- ▶ Defining conditions
- ▶ Defining exception cells

You can use all queries that you define in the BEx Query Designer for OLAP reporting, and also for flat reporting.

BEx Web Analyzer

Independent BEx Web Application for data analysis. The BEx Web Analyzer can be called using a URL or as an iView in the Enterprise Portal.

BEx Web application

A Web-based application of Business Explorer for data analysis, reporting, and analytical applications on the Web. You can format and display your data differently in the BEx Web Application Designer with a series of Web items (for example, tables, filters, charts, maps, and documents). In this way you can individually set Web applications like BI Cockpits and access them by using the intranet or by using an enterprise portal.

BEx Web Application Designer

Desktop application for creating Web sites with BI contents. With the BEx Web Application Designer you can create an HTML page that contains BI-specific contents, such as different tables, charts, or maps. This HTML page serves as the basis for Web applications with complex interactions like BI Cockpits. You can save the Web applications as a URL and access them by using the intranet or by using mobile terminals. Additionally, you can save the Web applications as an iView and integrate them into an enterprise portal.

BEx Web application wizard

Assistant that supports the creation of Web pages with BI-specific contents and enables a simplified design process by means of an automatic step-by-step sequence. The Web Application wizard is integrated in the Web Application Designer.

BI Java Connector

One of a set of four JCA (J2EE™ Connector Architecture)-compliant resource adapters that allow you to connect applications built with the BI Java™ SDK to heterogeneous data sources:

- ▶ BI JDBC™ Connector (for relational JDBC-compliant data sources)
- ▶ BI ODBO Connector (for ODBO-compliant OLAP data sources)
- ▶ BI SAP Query Connector (a component of the SAP Web Application Server Basis)
- ▶ BI XMLA Connector (for OLAP data sources such as SAP BW 3.x)

You can also use the Connectors to make external data sources available in BI, via BI's UD Connect. In the SDK, the term connector is synonymous with resource adapter.

BI JDBC Connector

A resource adapter for the Business Intelligence domain based on Sun's Java Database Connectivity (JDBC), which is the standard Java API for Relational Database Management Systems (RDBMS). The BI JDBC Connector may be deployed into SAP's Web Application Server, and allows you to connect applications built with the BI Java SDK to over 170 JDBC drivers, supporting data sources such as Teradata, Oracle®, Microsoft SQL Server®, Microsoft Access, DB2, Microsoft Excel, and text files such as CSV.

You can also use the BI JDBC Connector to make these data sources available in BI, via BI's UD Connect.

The JDBC Connector implements the BI Java SDK's IBIRelational interface.

BI ODBO Connector

A resource adapter for the Business Intelligence domain based on Microsoft's ODBO (OLE DB for OLAP), which is the established industry-standard OLAP API for the Windows platform. The BI ODBO Connector may be deployed into SAP's Web Application Server, and allows you to connect applications built with the BI Java SDK to ODBO-compliant OLAP data sources such as Microsoft Analysis Services, SAS, and Microsoft PivotTable® Services.

You can also use the BI ODBO Connector to make these data sources available in BI, via BI's UD Connect.

The ODBO Connector implements the BI Java SDK's IBIOlap interface.

BI SAP Query Connector

A resource adapter for the Business Intelligence domain based on SAP Query, which is a component of SAP's Web Application Server that allows you to create custom reports without any ABAP programming knowledge. The BI SAP Query Connector uses SAP Query to allow applications created with the BI Java SDK to access data from these SAP operational applications.

You can also use the BI SAP Query Connector to make these data sources available in BI, via BI's UD Connect.

The SAP Query Connector implements the BI Java SDK's IBIRelational interface.

BI XMLA Connector

A resource adapter for the Business Intelligence domain based on Microsoft's XMLA (XML for Analysis), which facilitates Web services-based, platform-independent access to OLAP providers. The BI XMLA Connector may be deployed into SAP's Web Application Server, and enables the exchange of analytical data between a client application and a data provider working over the Web, using a SOAP-based XML communication API.

The BI XMLA Connector allows you to connect applications built with the BI Java SDK to data sources such as Microsoft Analysis Services, Hyperion, MicroStrategy, MIS, and BW 3.x.

You can also use the BI XMLA Connector to make these data sources available in BI, via BI's UD Connect.

The BI XMLA Connector implements the BI Java SDK's IBIOlap interface.

broadcast setting

Collection of parameters that serve to distribute a query, a Web application or a workbook by e-mail or to the Enterprise Portal.

A broadcast setting can be identified uniquely by a technical name. You can determine whether the broadcast setting is to be executed immediately or whether it is to be scheduled for a particular point in time. You can also determine that the broadcast setting is scheduled to be executed when changes to data in the underlying InfoProvider are made.

broadcasting framework

Technical infrastructure for BEx Information Broadcasting.

broadcasting wizard

Assistant that supports you by giving step by step instructions on how to precalculate and distribute queries, Web templates or workbooks. With the broadcasting wizard you are able to create precalculated documents and online links and either distribute them by e-mail or publish them to the Enterprise Portal. You can also jump to the BEx Broadcaster in order to define further settings.

Business Content

Predefined role and task-related information models that can be suited to enterprise-specific requirements.

Business Content makes the supply of information available to roles in an enterprise, that require this to complete tasks.

Business Content consists of roles, workbooks, queries, InfoCubes, InfoObjects, InfoSources and update rules, as well as extractors, for SAP R/3, SAP New Dimension Applications and for additional selected applications.

Business Explorer

Component of the SAP Business Information Warehouse that makes flexible reporting and analysis tools available to enable strategic analysis and support the decision-making process in enterprises.

Business Explorer Analyzer

Analytical and reporting tool in the Business Explorer that is embedded in Microsoft Excel. In the Business Explorer Analyzer, you can analyze selected InfoProvider data by navigation to queries created in the BEx Query Designer and can create different query views for the data.

Business Explorer Browser

A tool for organizing and managing workbooks and documents.

The Business Explorer Browser (BEx Browser) enables you to access all documents in the Business Information Warehouse that are assigned to your role and that you have stored in your favorites.

You can work with the following document types in the BEx Browser:

- ▶ BI workbooks
- ▶ Documents that are stored in the Business Document Service (BDS)
- ▶ Links (references to the file system, short cuts)
- ▶ Links to Internet pages (URLs)
- ▶ SAP transaction calls
- ▶ Web applications and Web templates
- ▶ Crystal Reports

Business Intelligence Cockpit

Web-based control panel with Business Intelligence contents. Just like a cockpit in an airplane, it gives the management of a company an overview of all the relevant business data.

You use the Business Explorer Web Application Designer to generate individual BI cockpits that display relevant data in tables, charts, or on maps. The alert monitor that is integrated into the BI cockpit tells you immediately if any critical data falls outside of the specified range of acceptable values. You also have the option of adding additional information to the business data in the form of documents, graphics, or hyperlinks.

BI cockpits have the following options:

- ▶ You can collect data from different data sources and display it in various ways (as a table or as a chart, for example)
- ▶ You can use structured (BI contents) and unstructured (documents) information to enhance each other
- ▶ Personalized initial window: Parameters are filled with user-specific values (for example, values regarding cost center or region) automatically
- ▶ Role-specific variants: Different BI cockpits for different roles

You can get a quick overview of business information in much the same way that you scan the front page of a newspaper. To access more detailed information, you can use user-friendly navigation elements such as hyperlinks, dropdown boxes, or pushbuttons. You can save BI cockpits as iViews. These are completely integrated into an Enterprise Portal.

Business Intelligence Java Software Development Kit

A Java software development kit with which you can build analytical applications that access, manipulate, and display both multidimensional (Online Analytical Processing, or OLAP) and tabular (relational) data. The BI Java SDK consists of:

- ▶ Java APIs for accessing, manipulating, and displaying data from diverse data sources
- ▶ Documentation
- ▶ Examples

BI Document Repository Manager

Tool in the Knowledge Management of SAP Enterprise Portal.

The BI Document Repository Manager regulates read and write access to documents that are stored in SAP Business Information Warehouse.

BI Metadata Repository Manager

Tool in the Knowledge Management of SAP Enterprise Portal.

The BI Metadata Repository Manager regulates read and write access to metadata (InfoCubes, queries, Web templates, workbooks etc.) and the documentation on this metadata in SAP Business Information Warehouse. The BI Metadata Repository Manager also allows access to online links generated using BEx Information Broadcasting.

Characteristic

A type of InfoObject in SAP BI systems that provides a classification such as company code, product, customer group, fiscal year, period, or region. Related to the OLAP-standard term dimension.

chart

Web item that retrieves data from a query view to create a diagram for a Web application. You can choose from a range of display options for the diagram. You can also navigate within interactive charts and analyze the displayed data.

checkboxes

Web item that puts characteristic values to be filtered into a group of checkboxes.

collation process

Allows you to gather together several chains into a single chain in process chain maintenance windows. This means that you no longer have to schedule application processes individually.

The process chains maintenance windows contain the following collation processes:

- ▶ **AND-process (last):** The application process starts when all the events for the preceding processes have been triggered successfully.
- ▶ **OR-process (each):** The application process starts each time an event in a preceding process is triggered successfully.
- ▶ **EXOR-process (first):** The application process starts when the first event in one of the preceding processes is triggered successfully.

command processor

Part of each of the BI Java SDK's query APIs, interfaces that make it easier to use the underlying query models by hiding the complexity of these models. With the command processors, you can create and manipulate complex queries with simple commands. You can think of the individual methods of the command

processors in terms of macros that consist of several method calls manipulating the structures of queries.

The BI Java SDK provides two command processors:

- ▶ OLAP Command Processor, for manipulating OLAP queries
- ▶ Relational Command Processor, for manipulating relational queries

Common Client Interface

An API defined by Sun's JCA specification that is common across heterogeneous EISs. It is designed to be "tool-able"—that is, it leverages the Java Beans architecture so that development tools can incorporate the CCI into their architecture.

Note that the BI Java Connectors implement only the connection interfaces defined by the CCI. The CCI's interaction interfaces, data interfaces, and metadata interfaces, however, are not implemented by the BI Java SDK. BI-specific client APIs that are tailored for OLAP interactions are provided by the BI Java Connectors.

common warehouse metamodel

A standard that is recognized by the Object Management Group (OMG) and that describes the exchange of metadata in the following areas:

- ▶ Data Warehousing
- ▶ Business Intelligence
- ▶ Knowledge Management
- ▶ Portal Technology

CWM uses

- ▶ UML to model metadata
- ▶ MOF to access metadata
- ▶ XMI to exchange metadata

You can find the current CWM specification on the OMG home page.

communication structure

The communication structure is independent of the source system and has the structure of an InfoSource. It contains all the InfoObjects that belong to an InfoSource.

All the data is updated into the InfoCube with this structure. The system always refers to the actively saved version of the communication structure.

Technically (that is, with respect to length, type) the fields of the communication structure correspond to the InfoObjects of the Business Information Warehouse.

comparison column

A write-protected data column in the planning layout.

These comparison columns serve to display data for specific characteristic value combinations for which no data is to be entered in the planning layout. Comparison columns can be used as a reference for the plan data that is being entered (for example, when planning for the next quarter, the comparison column could display the actual data from the previous quarter).

For planning layouts that contain key figures in the lead column, what is said about columns also applies to rows (comparison rows).

condenser

A program that compresses the contents of an InfoCubE- fact table.

Connector Gateway

An SAP Enterprise Portal service that provides instances of connections to Portal components.

control query

A help query that you execute before you execute queries in the Web template. You use the result of this query to parameterize the Web template.

Cube

A set of data organized as a multidimensional structure defined according to dimensions and measures.

Related SAP BI terms include InfoCube and query.

Crystal Enterprise

Server component that is required to publish formatted reports created with the Crystal Reports Designer on the basis of BI data.

Formatted reports that are stored on the Crystal Enterprise server can be called up from there and displayed on the Web. Content and user administration is carried out as part of the integration using the BI server.

Crystal Report

BI object type.

Report definition for a formatted report that has been created using the Crystal Reports Designer on the basis of a BI query. The report definition is saved in SAP BI and published to the Crystal Enterprise server.

Crystal Reports Designer

Design component for generating a Crystal Report, which contains the report definition (layout).

Data binding

A connection between two UI components (or between a Web service and a UI component) that channels identical data from the output port of one UI component to the input port of the other UI component.

Data flow

The means by which data is channeled between a data service and connected UI components, or between two UI components whose connection was changed from Data binding to Data flow.

Data mapping

Connection between two model elements, describing, for example, the data that is input to an element or the fields that are output from another element.

Data service

Any function call, business object or query imported into the model. At runtime, the data service is called and returns results.

DataStore object

Describes a consolidated dataset from one or more InfoSources. This dataset can be evaluated using a BEx query. Object that stores consolidated and cleaned up transaction data on the document level. A DataStore object consists of a key and data fields that, as key figures, can also contain character fields. You can use a delta update to update data from a DataStore object into InfoCubes or other DataStore objects in the same system or in a different system.

Data Warehousing Workbench

Tool for controlling, monitoring and maintaining all those processes involved in data procurement and processing within the Business Information Warehouse.

Data store

A central data container where data of a model can be temporarily stored for future use.

Dimension

In OLAP-standard systems:

A collection of similar data which, together with other such collections, forms the structure of a cube. Typical dimensions include time, product, and geography. Each dimension may be organized into a basic parent-child hierarchy or, if

supported by the data source, a hierarchy of levels. For example, a geography dimension might include levels for continent, country, state, and city.

The related term in SAP BI systems is characteristic.

In SAP BI systems:

A grouping of those evaluation groups (characteristics) that belong together under a common superordinate term.

With the definition of an InfoCube, characteristics are grouped together into dimensions in order to store them in a star schema table (dimension table).

Element

A general term indicating any item used to create a model, including: components, connectors and operators.

Enterprise service

A Web service defined to perform functions of an SAP system. Web services are published to and stored within a repository.

Field

An element of a table that contains a single piece of data. Fields are organized into rows, which contain all the data relevant for one specific entry in the table. In some databases, field is a synonym for column.

Filter

A set of criteria that restricts the set of records returned as the result of a query. With filters, you define which subset of data appears in the result set.

Hierarchy

A logical tree structure that organizes the members of a dimension into a parent-child relationship. If supported by the data source, the hierarchy consists of levels, where the top level is an aggregate of all members and each subsequent level has zero or more child members.

InfoArea

An element for grouping meta-objects in the Business Information Warehouse. Each InfoProvider is assigned to an InfoArea. The resulting hierarchy is displayed in the Administrator Workbench.

InfoCube

An SAP BI system that consists of a quantity of relational tables created according to the star schema: a large fact table in the center, with several dimension tables surrounding it. It provides a self-contained dataset which can be used for analysis and reporting.

Similar to the OLAP-standard term cube.

InfoObject

A business evaluation object (for example, customer or quantity) in SAP BI systems. Types of InfoObjects include characteristics, key figures, units, time characteristics, and technical characteristics (such as request numbers).

JDBC

Java Database Connectivity, which provides an API that lets you access relational databases using the Java programming language. This enables connectivity to a wide range of SQL databases, and also provides access to tabular data sources such as spreadsheets or flat files. The BI JDBC Connector accesses data from JDBD-compliant systems.

Join

A relationship between two tables that produces a result set that combines their contents. You create a join by indicating how selected fields in one table are related to selected fields in the other table.

Key figure

A value or quantity in SAP BI systems. Related to the OLAP-standard term measure. You may also define calculated key figures, which are derived using a formula.

Layer

A collection of UI elements that are all visible at the same time at runtime.

Level

A set of nodes (members) in a tree hierarchy in supporting data sources that are at the same distance from the root of the tree. For example, in a geography hierarchy, the top level might be all places, the second level might be continents, the third level might be countries, and the fourth level might be cities.

MDX

Multidimensional Expressions, a query language used to retrieve and manipulate multidimensional data.

Measure

One category of values - usually numeric - used to define a cube. These values are derived from one or more columns in the cube's fact table and are the basis for aggregation and analysis.

Related SAP BI terms include key figure and structure element.

Member

An element of a dimension that represents one or more occurrences of data. A

member can be unique (it occurs only once) or non-unique (it may occur more than once in its dimension). For example, in a geography dimension that includes cities in the US, the member Portland could be non-unique, since there is a city called Portland in the state of Oregon and in the state of Maine.

In SAP BI systems, members are referred to as instances of characteristics.

Model

An object created in Storyboard™. Models may contain packages, pages, iViews and any other model elements.

Multidimensional data

Data in dimensional models suitable for business analytics. In this documentation, we use the term multidimensional data synonymously with OLAP data.

Navigation line

A connection that provides event annotation, running between model layers. The source element raises the event that can be handled by the connected element. By default, a navigation line is curved.

ODBO

OLE DB for OLAP - Microsoft's set of objects and interfaces that extend the ability of OLE DB to provide access to multidimensional data sources on the Windows platform. Providers of OLAP data can implement the interfaces described with OLE DB for OLAP to allow all OLAP clients to access their data. The BI ODBO Connector accesses data from ODBO-compliant systems.

OLAP

Online analytical processing - a system of organizing data in a multidimensional model that is suitable for decision support. SAP BI systems are OLAP systems.

Operation

A functionality provided by a Web service.

Operator

A mechanism used to manipulate data returned from the data service before it is displayed in the iView.

or-process

Collation® process of the process chain maintenance.

When you use an or-process in the process chain maintenance, the application process starts each time an event in a previous process is triggered.

P table

Master data table for time-independent master data.

This table contains the following fields:

- ▶ The characteristic with the master data itself.
- ▶ The characteristic compounded to this characteristic (“super-ordinate characteristic”)
- ▶ All time-dependent attributes
- ▶ CHANGED (D: Delete record, I: Insert record, Blank space: No changes; changes evaluated only after activation)
- ▶ OBJEVERS (A: Active version, M: Modified and therefore, not active version)

These fields form the key.

PDA application

Web application on a PDA device with Pocket IE.

Persistent Staging Area

Transparent database table, in which request data is stored in the form of the transfer structure. A PSA is created per DataSource and source system. It represents an initial store in BI, in which the requested data is saved unchanged for the source system.

planning application

Generic term for complex planning functions, with whose help a specific business planning task is processed, and which are compiled from objects of cross-application planning, and delivered by SAP.

Examples of planning applications are balance sheet planning, investment planning, or the special function Capital Market Interpreter.

planning package

Selection of data in which planning functions operate.

A planning package is used to define a selection for those characteristics where no selection was already defined in the planning level. In the case of a complete data selection in the planning level, a planning package must also be available in order to execute planning functions.

planning profile

User-specific combination of planning objects, which are displayed during the planning session in the planning environment.

Planning profiles are used to only present those planning objects from the possibly very large quantity of planning objects, which are relevant for a specific user or for a certain planning task.

planning sequence

List of user-defined planning functions or parameter groups, which are processed sequentially in the order determined by the list.

Planning sequences are used to execute recurrent complex planning tasks as efficiently as possible. These can be defined both locally within a planning area and globally across several planning areas.

Port

A defined point of interface into and out of a component.

Portal Connection Framework

Part of SAP's Enterprise Portal, provides a set of APIs which extend the standard JCA interfaces and are used to build Portal-compliant connectors. The BI Java Connectors are compliant with the Portal Connection Framework.

primary source system

Source system from which recently created or changed objects need to be transported into another target source system.

A primary source system is, within the framework of a system landscape consisting of OLTP and BW systems, an OLTP development system. The respective target source system is the OLTP system that is connected with the BI target system.

In order to be able to transport objects that are specific to the source system (for example, transfer structures), the logical system names must be specified for the source systems before and after the transport in a mapping table in the BI target system.

process

Process with a definite beginning and end within or outside of an SAP system.

process chain

Sequence of processes that are scheduled in the background to wait for a particular event. Some of these processes trigger an additional event, which in turn can start other processes.

process instance

Value of the process:

The process instance contains the most important information that the process or the follow-on processes provide, for example, the name of the request during the loading process. The instance is determined by the process itself during the runtime. The logs for the process are stored under the process instance.

process type

Type of process, for example, a loading process.

The process type decides, among other things, which tasks the process has, and which properties the process has in the maintenance.

process variant

Name of the process.

A process can have various variants. For example, in a loading process the name of the InfoPackage tells you the variants of the process. A variant is defined by the user when he or she schedules the process.

Query

In SAP BI systems, a collection of selected characteristics and key figures (InfoObjects) used together to analyze the data of an InfoProvider. A query always refers exactly to one InfoProvider, whereas you can define as many queries as you like for each InfoProvider.

Query view

In SAP BI systems, a view of a query after navigation, saved in an InfoCube. You can use this saved query view as a basis for data analysis and reporting.

query API

Sets of interfaces provided by the BI Java SDK for creating queries against data sources. They are generated via JMI from the SDK's query models, providing methods to create and execute complex OLAP or relational queries based on the metadata in the SDK's CWM-based metadata models.

The SDK provides two query APIs:

- ▶ OLAP Query API, for defining queries against an OLAP server
- ▶ Relational Query API, for defining queries upon relational data sources

query model

An object-oriented abstraction layer, or model, in the BI Java SDK upon which to formulate queries on a variety of resources without being tied to a specific protocol or query language, such as MDX or SQL. The query models are the basis of their respective query APIs.

Two query models are provided by the SDK:

- ▶ OLAP Query Model
- ▶ Relational Query Model

query view - selection

Web item that makes navigation between different queries and query views possible.

radio button group

Web item that allows you to filter characteristic values using a group of radio buttons.

Relational Command Processor

Part of the BI Java SDK's Relational Query API, an interface that makes it easier to use the API by hiding the complexity of the underlying Relational Query Model. With this interface, you can create and manipulate complex relational queries with simple commands.

relational data

Data stored in tables, and hence often also referred to as tabular data.

relational data provider

A component that provides data in relational, or tabular, views and metadata compatible with the BI Java SDK's Relational Metadata Model.

Relational Metadata API

A set of interfaces provided by the BI Java SDK for accessing the metadata of a relational data source. Generated via JMI from the SDK's Relational Metadata Model.

Relational Metadata Model

A model provided by the BI Java SDK that describes data accessible through a relational interface such as JDBC. Based on the Common Warehouse Metamodel Relational Package.

Relational Query API

A set of interfaces provided by the BI Java SDK that let you define queries against a relational data source. The API is generated via JMI from the Relational Query Model, based on metadata provided by the Relational Metadata Model, and includes the simplified Relational Command Processor.

Relational Query Model

An abstraction layer, or model, in the BI Java SDK designed for formulating relational queries independently of data source-specific query APIs. The model is based on the CWM Expressions package and the CWM-compliant metadata provided by the Relational Metadata Model.

RemoteCube

InfoCube whose transaction data is not managed in the Business Information Warehouse, but rather externally. Only the structure of the RemoteCube is defined in BI. The data for reporting is read from another system using a BAPI.

Reporting Agent

Tool for scheduling reporting functions in the background. The following functions can be executed:

- ▶ Evaluating exceptions
- ▶ Printing queries
- ▶ Precalculating Web templates
- ▶ Precalculating characteristic variables of type precalculated value set
- ▶ Precalculating queries for Crystal reports
- ▶ Managing bookmarks

resource adapter

A system-level software driver component defined by the JCA specification and used to connect to an EIS. The BI Java SDK and UD Connect use resource adapters called BI Java Connectors.

resource adapter archive

Complete resource adapter modules, which as defined by the JCA specification consist of the required Java classes, documentation, native libraries, and deployment descriptors necessary to distribute a given resource adapter (connector). The BI Java Connectors are distributed in RAR files.

resource adapter module

A complete resource adapter which as specified by the JCA is represented physically by a RAR file.

results area

Part of a BEx Analyzer workbook that displays the results of a query.

The results area corresponds to the table Web item in Web applications.

ResultSet API

A set of interfaces that provide applications created with the BI Java SDK with access to the results of a query. The ResultSet API provides access to a relational result set from a relational data source, and an OLAP result set from an OLAP data source.

reusable structure

Part of a query that is saved so that it can be used again in an InfoCube.

Reusable structures enable you to use parts of a query definition again in other queries. These structures are freely definable reports consisting of combinations of characteristics and basic key figures (for example, calculated or restricted key figures from the InfoCube). A structure can be a plan/actual comparison or a contribution margin scheme, for example.

ROLAP

A store for multidimensional data in a relational database in tables that are organized in a star schema.

The opposite of ROLAP is MOLAP.

role

Combination of similar positions.

Example: The role “Purchasing Manager” covers the responsibility for orders in the framework of providing basic material, goods and business methods. The task area of the Purchasing Manager entails optimizing the relationship between price and value. Included in the task area of the Purchasing Manager are managing the order process, determining purchasing policies, and procurement market research (process tasks). The Purchasing Manager also plays the role of a superior, that is, she supervises the efficiency of the order process, controls the cost center data and is responsible for personnel administration in his/her area (administrative activity functions).

role menu

Web item that displays user favorites or roles in a tree structure.

roll up

Loads data packages (requests) for an InfoCube that are not yet available into all aggregates of the InfoCube. After it has been rolled up, the new data is used in queries.

SAP Business Information Warehouse

SAP's data warehouse and reporting interface.

SAP Business Information Warehouse (SAP BW) provides data warehousing functions, a business intelligence platform, and a suite of business intelligence tools. These tools enable businesses to integrate, transform, and consolidate relevant business information in SAP BW.

SAP BW facilitates the reporting, analysis and distribution of this information. On the basis of this analysis, businesses can make well-founded decisions and determine target-oriented activities. With the comprehensive predefined information models delivered for different roles in a business (BI Content), SAP BW increases the usability of analysis functions and facilitates a quick and cost-effective implementation.

SAP BW is a core component of SAP NetWeaver.

SAP exit

Processing type for variables that are delivered with SAP BW Business Content. Used for variables that are processed by automatic replacement using a predefined replacement path (SAP exit).

SAP Java Connector

A middleware component that facilitates the development of SAP-enabled components and applications in Java.

The SAP Java Connector (JCo) supports communication with the SAP server in two directions:

- ▶ Inbound (Java calls ABAP)
- ▶ Outbound calls (ABAP calls Java)

SAP JCo can be deployed with desktop and (Web) server applications.

scheduler

With the aid of the scheduler, you determine which data (transaction data, master data, texts, or hierarchies) is to be requested and updated from which InfoSource, DataSource, and source system and at which point in time.

scheduling package

Logical collection of several reporting agent settings for background processing.

selector

Element in a Web interface, which associates visible elements of the Web interface with planning objects (planning area, level, package, function).

The visible elements of a Web interface, with which the user interacts at runtime (tables on data entry, pushbuttons for execution of functions), have attributes

which contain a reference to planning objects. Instead of a direct reference to a planning object, an element can also contain a reference to a selector, which references the planning object on its part. When several visible elements of the Web interface refer to the same planning object, the maintenance effort is reduced by a unified reference to the same selector.

On their part, selectors can be included as visible elements in a Web interface and then enable the selection of the desired planning object by the user.

service

Parameterized service type.

service API

Technology package in SAP source systems of the SAP BW that facilitates a high level of integration for data transfer from the source systems to SAP BW.

The service API makes it possible to:

- ▶ Make SAP application extractors available as the basis for data transfer from source systems into SAP BW.
- ▶ Perform generic data extraction.
- ▶ Utilize delta processes.
- ▶ Access data in the source system directly from SAP BW (RemoteCube support).

Service Provider Interface

implements the application interfaces supplied by an engine class, as each class has a corresponding abstract Service Provider Interface (SPI) class that defines the SPI methods that the cryptographic service providers must implement.

single document

Web item that displays, in the Web application, single documents on master data that have been created in the Administrator Workbench or in master data maintenance.

SOAP

Simple Object Access Protocol.

You can find the current SOAP specification on the World Wide Web Consortium home page: <http://www.w3c.org>.

source Business Information Warehouse

Business Information Warehouse that is available to additional BW servers as a source system.

source object element

Component of the UD Connect source object, for example, a field in a table.

source system

System that makes the Business Information Warehouse available for data extraction.

staging

A process that prepares (stages) data in a Data Warehouse.

standard Web template

Web template that is used as the default template for the Web display of particular BEx functions.

In the Business Explorer the following standard Web templates are available:

- ▶ Standard Web template for ad-hoc analysis
- ▶ Standard Web template for broadcasting
- ▶ Standard Web template for query precalculation
- ▶ Standard Web template for the document browser

You can determine a user-defined Web template as the standard Web template for a particular function in SAP Reference IMG.

start process

Defines the start of a process chain.

stored query view

Stored record of a query, showing a particular navigation status.

subplan

Specific business subarea of business planning for example, sales planning, balance sheet planning, or cost center planning.

surrogate index

Special BW index above the key fields of a fact table.

The surrogate index is created on a fact table and not on the primary index. In contrast to the primary index, the surrogate index has no UNIQUE restriction.

table

Web item that retrieves data from a query view to create a table for a Web application. Links for navigation are also included with the table. Characteristics and structures can be displayed in rows and in columns.

tabular reporting

Reporting based on one-dimensional tables, meaning the analysis is restricted to one dimension and its attributes. Unlike OLAP reporting, with flat reporting you can arrange the columns any way you like when you are designing a query in the tabular editor mode of the BEx query designer.

For example, you can put a column for a characteristic between two columns for key figures. During the design of the query, you decide how you want the columns to be displayed. Once you have chosen a display type you are not able to change it. In flat reporting, the interactive options are restricted to filter, filter and drilldown according to, sort according to, and navigate on hierarchies. Navigation functions that alter the geometry of the flat list, meaning that they change the number and order of the columns, for example, swapping or adding a drilldown, are available with OLAP reporting but not with flat reporting.

target Business Information Warehouse

BW System to which another BW System is connected as a source system, and into which you can load data using export DataSources.

temporal join

Join containing at least one time-dependent characteristic.

The time-dependency is evaluated when the results set is determined. A time interval is assigned to each record in the results set. The record is valid for the interval to which it is assigned (valid time-interval). The results set, time dependencies are evaluated. Each record in the results set is assigned to a time interval, for which

test

Check of internal information about BW objects for consistency.

Gives a repair option, if necessary. A test consists of several elementary tests. You can select the required elementary tests individually so that you do not have to conduct unnecessary testing.

test package

Sequence of elementary tests as result of selecting specific tests or elementary tests.

You can save a test package and schedule it to run later.

text element

Web item that displays query information about which the query view, and consequently the Web application, are based. You can select individual text

elements to be displayed in the Web application. The 'text element' Web item can contain variables, static filter values, and general text elements (for example, technical name of the query, the query key date, or query author).

ticker

Web item that displays the content of a table as a ticker.

t-logo object

Logical transport object combining several table entries to be transported together.

Example:

The T-Logo object "InfoObject" consists of the table entries of the InfoObject table, the characteristic table, the text table, and the attribute table.

transfer rule

With the help of the transfer rules, you can determine how the fields for the transfer structure are assigned to the InfoObjects of the communication structure.

transfer structure

The transfer structure is the structure in which the data from the source system is transported into the Business Information Warehouse.

It displays a selection of fields for an extract structure of the source system.

UD Connect

Component of SAP BW that, together with the SAP Web AS J2EE server, provides connectivity to virtually all relational and multidimensional data sources.

UD Connect (Universal Data Connect) uses the BI Java Connectors as resource adapters for establishing connections to data sources. The data can either be loaded into BI, or accessed directly via a RemoteCube.

SAP Query

A component that allows you to create custom reports without any ABAP programming knowledge. The BI SAP Query Connector uses SAP Query to access data from such SAP operational applications.

Storyboard

The Visual Composer client from which you design models.

Table

A set of rows, also known as a relation. The table is the central object of the relational model.

Task panel

A work area of the Visual Composer Storyboard desktop that displays a specific set of tools for building a model.

Toolbar

The horizontal row of buttons under the main menu (main toolbar) or the vertical row of buttons in the task panel (task-panel toolbar).

Toolbox

A set of board-specific tools that assist in performing tasks in the Visual Composer workspace.

Value help

The offering, typically in a pop-up dialog box, of possible valid values for an input field. Also known as input help, selection help, or F4 help.

variables

Parameters of a query that are created in the BEx Query Designer and are not actually filled with values (processed) until the query is inserted into a workbook.

They function as a place holder for characteristic values, hierarchies, hierarchy nodes, texts, and formula elements. They can be processed in different ways.

Variables in the SAP Business Information Warehouse are global variables, meaning that they are uniquely defined and are available for the definition of all queries.

WAP application

Web application on a WAP device.

WAP device

Mobile phone with a WAP micro browser that displays WML content.

WAP gateway

Network component to connect the cellular phone network with the Internet.

WAP server

Server that provides WML contents. In BEx Mobile Intelligence, the BW server acts as a WAP server.

Web application object catalog

Web item with which information about Web template properties, details of data providers being used, and information about Web items used in Web templates can be generated in XML format.

The Web item is not visualized in the Web application.

Web interface

Quantity of components which represent planning objects (planning areas, levels, functions) and which can be combined to Web-enabled planning applications using the Web Interface Builder.

Web Interface Builder

Tool to create Web-enabled planning applications.

The Web Interface Builder is a graphic tool with which the planning objects created in the BWBPS planning environment (planning areas, levels, functions) can be combined in an application. The Web Interface Builder generates ABAP and JavaScript™ code from the graphic draft for an executable Business Server Page application.

Web item

An object that retrieves data from a data provider and presents it as HTML in a Web application.

Examples: Generic navigation block, tables, filter, text elements, alert monitor, chart, map.

Web item paging

Mechanism for dividing Web items in a Web template onto several pages, which are linked by an automatically generated overview page.

Web template

An HTML document that determines the structure of a Web application. It contains placeholders for Web items, data providers, and BW URLs.

Web template

Web item with which consistent sections of different Web templates can be managed centrally in one Web template.

Example:

You want to structure the header section of your Web application according to a certain pattern. Create a Web template with the company logo and heading and insert it into any other Web template using the “Web template” Web item.

WebDAV

World Wide Web Distributed Authoring and Versioning

An XML-based enhancement of the HTTP/1.1 protocol for asynchronous document management, which is the standard for accessing documents by using a Web browser application. Documents that are on a Web server are called resources and are combined into collections. WebDAV describes methods, headers, and content types in order to do the following:

- ▶ Prevent resources from being overwritten during distributed editing.
- ▶ Manage metadata using resources (properties).
- ▶ Create and manage collections.

According to the WebDAV Standard Specification RFC (Request for Comments) 2518 (February 1999), the IETF (Internet Engineering Task Force) is developing additional, special WebDAV specifications (for example, DASL).

Web service

An interface between two or more software applications that is implemented with the industry standards SOAP, WSDL and UDDI.

What If Prediction

Web item to perform online prediction for a single customer record on models defined under data mining methods such as Decision Tree, Scoring Clustering.

Wireless Application Protocol

Transfer protocol optimized for the compressed transfer of WML contents to the cellular phone network.

Wireless Bitmap

Black and white graphic format for WAP.

WML

Abbreviation of wireless markup language.

Internet-language standard for describing pages for mobile WAP devices.

workbook

A file containing several worksheets (an expression from Microsoft Excel terminology)

You insert one or more queries in the workbook in order to be able to display them in the Business Explorer Analyzer. You can save the workbook in your favorites or in your roles.

Workspace

The main grid area of Visual Composer that displays the model as it is built and modified. The workspace consists of boards.

X table

Table for assigning the SID of the characteristic to the SID of the navigation attribute for the characteristic (for time-independent masterdata).

This table contains the following fields:

- ▶ The characteristic SID (master data ID)
- ▶ OBJEVERS (object version)

These two fields form the key.

- ▶ The value of the super-ordinate characteristic
- ▶ The value of the master data-bearing characteristic itself
- ▶ CHANGED
- ▶ SIDs for time-independent attributes

OBJEVERS and CHANGED also appear in the P table

XMLA

XML for Analysis, an XML-messaging-based protocol specified by Microsoft for exchanging analytical data between client applications and servers (for example, OLAP providers) using HTTP and SOAP as a service on the Web. The BI XMLA Connector accesses data from XMLA-compliant systems.

XML for analysis

A protocol specified by Microsoft for exchanging analytical data between client-applications and servers using HTTP and SOAP as a service on the Web.

XML for analysis is not restricted to any particular platform, application, or development language.

Using XML for Analysis in the Business Information Warehouse allows a third-party reporting tool that is connected to the BW to communicate directly with the OLAP (Online Analytical Processing) processor.

XML metadata interchange

XML-based standard format for exchanging metadata between UML-based modeling tools and MOF-based metadata repositories in distributed, heterogeneous development environments. The exchange takes place in the form of data streams or files.

XMI, together with UML and MOF, forms the core of the metadata repository architecture of OMG (object management group).

You can find the current XMI specifications on the OMG home page.

Y table

Table for assigning the SID of the characteristic to the SID of the navigation attributes for the characteristic (for time-dependent master data).

The Y table has the same fields as the X table.

Archived

SAP BI installation and checks under DB2 and z/OS

This appendix describes our test scenario and the main installation activities and checks that we performed on DB2 and z/OS to set it up.

We also provide a brief description of the relevant documentation that gives information for the installation and the tuning of a BI System in a DB2 for z/OS installation.

A list of related SAP notes is provided in Appendix C, “SAP BW OSS Notes” on page 321.

IT scenario

The environment we used in our tests is shown in Figure B-1.

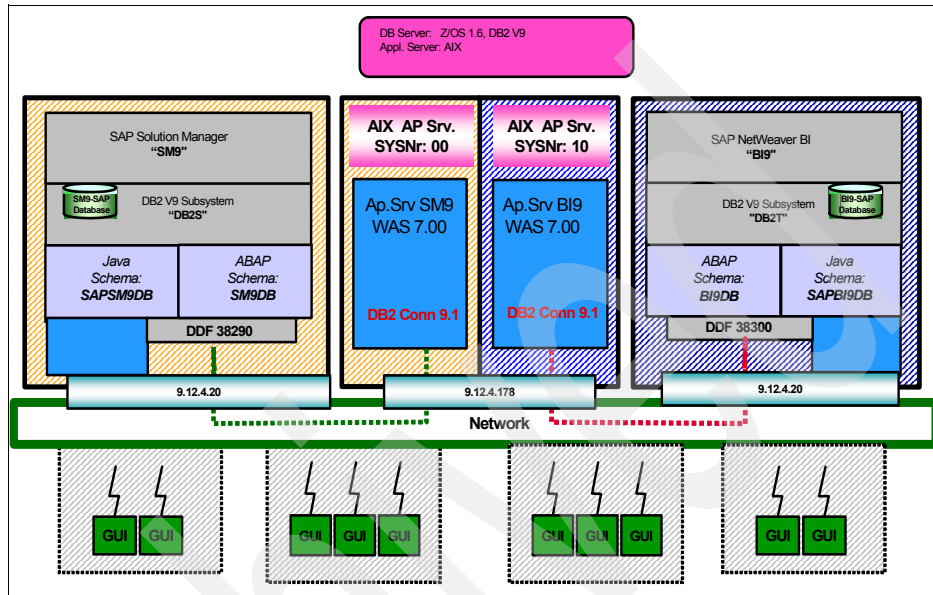


Figure B-1 Our test environment

We installed the following SAP products:

- ▶ SAP Solution Manager 4.0 SR1
- ▶ SAP NetWeaver 2004s SR1 ABAP and BI-Content 7.03

We updated the NetWeaver system with the latest available support packages (Table B-1).

Table B-1 Support packages

| Software component | Release | Level | Highest support package |
|--------------------|------------|-------|-------------------------|
| SAP_ABA | 700 | 0011 | SAPKA70011 |
| SAP_BASIS | 700 | 0011 | SAPKB70011 |
| PI_BASIS | 2005_1_700 | 0011 | SAPKIPYJ7B |
| SAP_BW | 700 | 0012 | SAPKW70012 |
| BI_CONT | 703 | 0004 | SAPKIBIIP4 |

SAP application servers ran on AIX 5.3 partitions of an IBM pSystem 550.

SAP database instances ran on IBM DB2 9 for z/OS and were installed on different LPARs of a z9 EC 2094-S18 IBM System z.

The DB2 Connect version was 9.1. FixPack 2 was used.

Useful SAP documentation

We used the following SAP installation guides:

- ▶ SAP Master Guide SAP NetWeaver 2004s, Document Version 1.10 - 11/08/2006
- ▶ SAP NetWeaver 2004s SR1 ABAP + Java on AIX: IBM DB2 UDB for z/OS, Version 1.0 - April 10, 2006
- ▶ SAP Planning Guide for SAP NetWeaver for IBM DB2 for z/OS, SAP NetWeaver 2004s SR1, Version 2.0 - 10/16/2006
- ▶ SAP Database Administration Guide for SAP NetWeaver on IBM DB2 UDB for z/OS, SAP NetWeaver 2004s SR1, Version 2.0 - 08/16/2006
- ▶ SAP Security Guide for IBM DB2 UDB for z/OS, SAP NetWeaver 2004s, Document Version 1.00 - October 24, 2005

The SAP Master Guide contains a description of the NetWeaver 2004s architectural components, application scenarios, and installation components sequence.

The Planning Guide covers planning, preparation, installation, and migration activity to be performed at z/OS level.

The Database Administration Guide provides information for planning, installing, monitoring, and tuning DB2 for z/OS.

The Security Guide is a reference material for security issues in an SAP on DB2 for z/OS environment.

SAP notes are also important because they integrate and sometimes correct official SAP documentation. A list of notes related to installation and DB2 and z/OS checks is provided in Appendix A, "Glossary for SAP BI" on page 281.

Checks under z/OS and DB2

We recommend the following setup and checks at z/OS and DB2 level:

- ▶ Workload monitor
- ▶ Stored procedures
- ▶ DB2 bufferpool

Installation of stored procedure SAPCL

Starting with NetWeaver 2004s, a new stored procedure, SAPCL, is used to collect DB2 performance data from the Instrumentation Facility Interface (IFI). In previous releases, this function was provided by rfcoscol.

Stored procedure SAPCL is available from the SAP Marketplace Web site as part of z/OS 64 bit kernel.

Also, since NetWeaver 2004s, CCMS corrections are delivered as part of SAP Basis Support Package and not with specific transports (see 427748).

It is important to keep the system updated with the latest support packages in order to have all IFI monitor functions work.

We downloaded SAPCAR and SAPDB2CL from the SAP Marketplace. To make SAPCAR work, we set the correct TAGs for automatic conversion with the following `chttag` command:

```
SC04:/SC04/sap/appo>chttag -t -c iso8859-1 -R SAPCAR
```

After this command, we verified the correct tag:

```
SC04:/SC04/sap/appo>1s -T  
t IS08859-1 T=on SAPCAR
```

So we could unpack the SAPDB2CL package that contained the following files:

- ▶ `sapdb2cl`, the stored procedure executable
- ▶ `DBRM.db2cldb`, the database request module

We then performed the installation of SAPCL from an SAP GUI.

We ensured that the SAP GUI user installing SAPCL had the authorizations and permissions described in the “Permissions, Privileges and Authorizations for SAPCL” section of the SAP Database Guide.

Figure B-2 shows how the GUI SAPCL window looked before installing SAPCL.

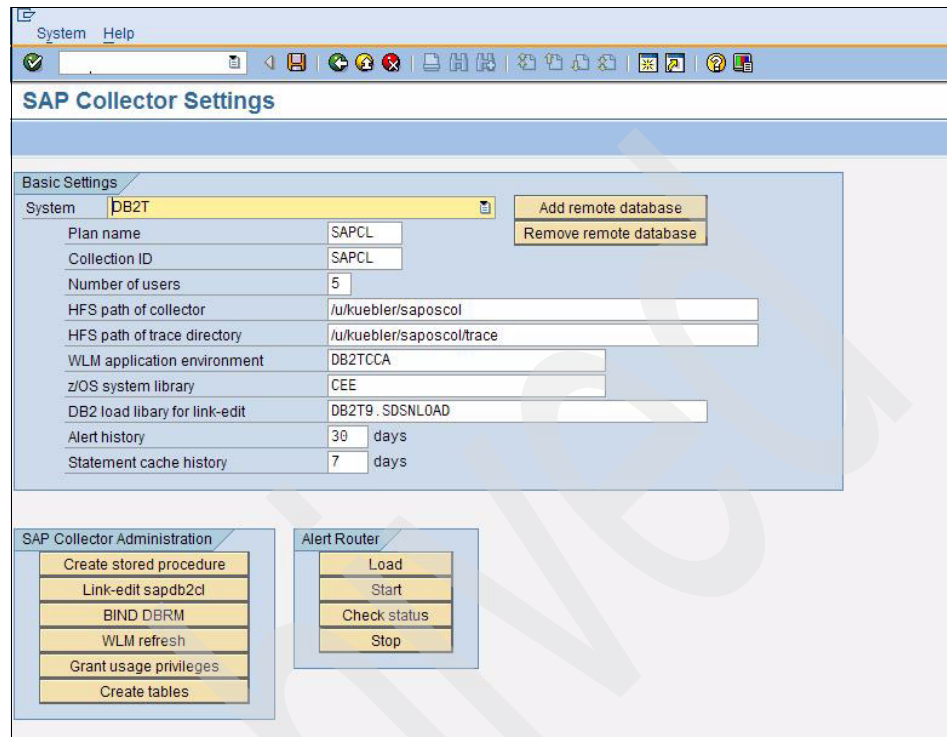


Figure B-2 Before installing SAPCL

We then sequentially performed the following actions:

1. Created stored procedure
2. Link-edit edsapdb2cl
3. BIND DBRM
4. WLM refresh
5. Granted usage privileges
6. Created tables

To check that the SAPCL procedure was working, we went back to the DB2 transaction and looked at the thread activity (see Figure B-3).

The screenshot shows the SAP Database Analysis Thread Activity window. The window title is "Thread Activity" and it displays data for subsystem DB2 at 08:37:10 on 07/03/2007. The DB system is DB2 and the DB release is 9.1.5. The window includes a "Thread analysis" section with buttons for Buffer pool act., Locking activity, Locked resources, DS locking, SQL activity, SQL statement, Lock waits, and Group BP. Below this is a table with columns: SAP ID, Server name, WP, End user, SAP tx, SAP program, DB conn, WP type, Status, Requests, Commits, Rollbacks, Max locks, Activity time, SAP no., WP ID, and Plan name. The table lists various threads for users like grita and sap02, with programs such as BIRDA000, BIRDA002, BIRDA001, BIRDA004, BIRDA005, BIRDA006, BIRDA007, BIRDA003, BIRDA009, BIRDA011, BIRDA015, BIRDA010, BIRDA012, BIRDA014, BIRDA013, BIRDA009, BIRDA019, BIRDA018, BIRDA017, BIRDA020, BIRDA022, BIRDA026, BIRDA035, BIRDA034, and BIRDA033. The table also shows statistics for Control threads DB2TAC and DB2TAC.

| SAP ID | Server name | WP | End user | SAP tx | SAP program | DB conn | WP type | Status | Requests | Commits | Rollbacks | Max locks | Activity time | SAP no. | WP ID | Plan name |
|--------|-----------------|----|----------|--------|-------------|---------------|---------|--------|----------|---------|-----------|-----------|----------------|---------|--------|-----------|
| 0 | Control: DB2TAC | 0 | | | | | | | 2,673 | 534 | 0 | 1 | 8:53:27.840313 | 0 | | TRRSJAF |
| 0 | Control: DB2TAC | 0 | | | | | | | 9 | 1 | 0 | 3 | 8:53:28.244775 | 0 | | TRRSJAF |
| B19 | sap02 | 0 | grita | sap02 | BIRDA000 | DB2T_on_w_DIA | In DB2 | | 24,351 | 10,444 | 1 | 6681 | 1:12:47.224107 | 10 | 629778 | DISTSERV |
| B19 | sap02 | 2 | grita | sap02 | BIRDA002 | DB2T_on_w_DIA | | | 13,567 | 267 | 0 | 4191 | 1:12:47.251481 | 10 | 331976 | DISTSERV |
| B19 | sap02 | 1 | grita | sap02 | BIRDA001 | DB2T_on_w_DIA | | | 15,590 | 1,090 | 0 | 2948 | 1:12:47.252070 | 10 | 573560 | DISTSERV |
| B19 | sap02 | 21 | grita | sap02 | BIBETC021 | DB2T_on_w_BTC | | | 11,063 | 147 | 0 | 504 | 1:12:44.581907 | 10 | 663716 | DISTSERV |
| B19 | sap02 | 4 | grita | sap02 | BIRDA004 | DB2T_on_w_DIA | | | 824 | 17 | 0 | 5 | 1:12:47.215762 | 10 | 385238 | DISTSERV |
| B19 | sap02 | 5 | grita | sap02 | BIRDA005 | DB2T_on_w_DIA | | | 720 | 4 | 0 | 9 | 1:12:47.224838 | 10 | 610494 | DISTSERV |
| B19 | sap02 | 8 | grita | sap02 | BIRDA006 | DB2T_on_w_DIA | | | 524 | 4 | 0 | 6 | 1:12:44.740226 | 10 | 602684 | DISTSERV |
| B19 | sap02 | 36 | grita | sap02 | BIRSP0036 | DB2T_on_w_SPO | | | 456 | 42 | 0 | 3 | 1:12:43.873188 | 10 | 655449 | DISTSERV |
| B19 | sap02 | 7 | grita | sap02 | BIRDA007 | DB2T_on_w_DIA | | | 431 | 4 | 0 | 5 | 1:12:44.868570 | 10 | 671872 | DISTSERV |
| B19 | sap02 | 3 | grita | sap02 | BIRDA003 | DB2T_on_w_DIA | | | 305 | 18 | 0 | 3029 | 1:12:47.218766 | 10 | 827416 | DISTSERV |
| B19 | sap02 | 22 | grita | sap02 | BIBETC022 | DB2T_on_w_BTC | | | 234 | 20 | 0 | 5 | 1:12:44.555274 | 10 | 561194 | DISTSERV |
| B19 | sap02 | 16 | grita | sap02 | BIRUPD016 | DB2T_on_w_UPD | | | 194 | 4 | 0 | 3 | 1:12:44.814260 | 10 | 651350 | DISTSERV |
| B19 | sap02 | 6 | grita | sap02 | BIRDA006 | DB2T_on_w_DIA | | | 164 | 3 | 0 | 1 | 1:12:47.212401 | 10 | 454696 | DISTSERV |
| B19 | sap02 | 11 | grita | sap02 | BIRDA011 | DB2T_on_w_DIA | | | 151 | 9 | 0 | 1 | 1:12:44.839150 | 10 | 405730 | DISTSERV |
| B19 | sap02 | 15 | grita | sap02 | BIRDA015 | DB2T_on_w_DIA | | | 148 | 3 | 0 | 1 | 1:12:44.831240 | 10 | 368848 | DISTSERV |
| B19 | sap02 | 10 | grita | sap02 | BIRDA010 | DB2T_on_w_DIA | | | 118 | 3 | 0 | 1 | 1:12:44.832198 | 10 | 336160 | DISTSERV |
| B19 | sap02 | 12 | grita | sap02 | BIRDA012 | DB2T_on_w_DIA | | | 118 | 3 | 0 | 1 | 1:12:44.731571 | 10 | 616568 | DISTSERV |
| B19 | sap02 | 14 | grita | sap02 | BIRDA014 | DB2T_on_w_DIA | | | 109 | 6 | 0 | 1 | 1:12:44.739611 | 10 | 615150 | DISTSERV |
| B19 | sap02 | 13 | grita | sap02 | BIRDA013 | DB2T_on_w_DIA | | | 100 | 3 | 0 | 1 | 1:12:44.810284 | 10 | 675954 | DISTSERV |
| B19 | sap02 | 9 | grita | sap02 | BIRDA009 | DB2T_on_w_DIA | | | 78 | 3 | 0 | 1 | 1:12:44.823148 | 10 | 507942 | DISTSERV |
| B19 | sap02 | 19 | grita | sap02 | BIRUPD019 | DB2T_on_w_UPD | | | 58 | 2 | 0 | 1 | 1:12:44.646353 | 10 | 516326 | DISTSERV |
| B19 | sap02 | 18 | grita | sap02 | BIRUPD018 | DB2T_on_w_UPD | | | 56 | 2 | 0 | 1 | 1:12:44.649150 | 10 | 487430 | DISTSERV |
| B19 | sap02 | 17 | grita | sap02 | BIRUPD017 | DB2T_on_w_UPD | | | 56 | 2 | 0 | 1 | 1:12:44.663636 | 10 | 577568 | DISTSERV |
| B19 | sap02 | 20 | grita | sap02 | BIBENW020 | DB2T_on_w_BVQ | | | 54 | 2 | 0 | 1 | 1:12:44.594039 | 10 | 884800 | DISTSERV |
| B19 | sap02 | 32 | grita | sap02 | BIBETC032 | DB2T_on_w_BTC | | | 33 | 2 | 0 | 1 | 1:12:44.668972 | 10 | 340712 | DISTSERV |
| B19 | sap02 | 26 | grita | sap02 | BIBETC026 | DB2T_on_w_BTC | | | 33 | 2 | 0 | 1 | 1:12:44.123787 | 10 | 934134 | DISTSERV |
| B19 | sap02 | 35 | grita | sap02 | BIBETC035 | DB2T_on_w_BTC | | | 33 | 2 | 0 | 1 | 1:12:43.802938 | 10 | 659548 | DISTSERV |
| B19 | sap02 | 34 | grita | sap02 | BIBETC034 | DB2T_on_w_BTC | | | 33 | 2 | 0 | 1 | 1:12:43.887938 | 10 | 839812 | DISTSERV |
| B19 | sap02 | 33 | grita | sap02 | BIBETC033 | DB2T_on_w_BTC | | | 33 | 2 | 0 | 1 | 1:12:43.835639 | 10 | 418030 | DISTSERV |

Figure B-3 Thread activity

SAP BW OSS Notes

This appendix lists useful SAP BW OSS Notes. This is current only up to about a month before the publishing date of this document. You may find that you will need to apply additional OSS Notes to your system. Please check with on the SAP support portal at:

<http://service.sap.com/notes>

You must have SAP support to access this information.

The notes are grouped into the following areas:

- ▶ Installation
- ▶ Tuning and performance analysis for SAP
- ▶ CCMS for DB2 z/OS
- ▶ General SAP BI notes
- ▶ SAP BI on DB2 z/OS specific notes
 - Partitioning
 - RUNSTATS
 - Other

List of SAP BW OSS Notes

Table C-1 Notes on installation

| | Installation |
|--------|---|
| 858969 | SAP NetWeaver 2004s Installation: IBM DB2 UDB for z/OS |
| 916834 | BI_CONT 7.03: Installation and upgrade information |
| 717935 | FTP replacement by Stored Procedures |
| 958252 | SAPCL FAQ |
| 81737 | APAR List |
| 913109 | SAPCL Patch Collection |
| 569996 | High availability and automation solution for DB2 on z/OS |

Table C-2 Notes on tuning and performance analysis

| | Tuning and performance analysis |
|---------|---|
| 994670 | DB2-z/OS: Display of RMF Monitor III Data from within SAP |
| 192658 | Setting basis parameters for BW Systems |
| 1032273 | Configuring DB2 9: recommended parameter settings |

Table C-3 Notes on CCMS

| | CCMS for DB2 on z/OS |
|---------|---|
| 427748 | CCMS corrections (6.10, 6.20, 6.40, 7.00) |
| 955047 | History of statement cache statistics in ST04 |
| 1028657 | Configure statement cache statistics history |

Table C-4 Notes on BI general

| | General notes for SAP BI |
|---------|---|
| | Repartitioning and remodelling |
| 946641 | BW Repartitioning Composite SAP Note |
| 1008833 | Supplementary SAP Note about repartitioning |
| | SAP Query Analysis Tool |

| | |
|--------|---|
| | General notes for SAP BI |
| 935815 | Query analysis tool |
| | General notes |
| 514907 | Processing complex queries (DataMart, and so on) |
| 567747 | Composite note BW 3.x performance: Extraction and loading |
| 323151 | Several DB connections with Native SQL |
| 912367 | BW general RSADMIN Parameters |
| 89188 | R/3 System copy |

Table C-5 BI on DB2 for z/OS

| | |
|---------|---|
| | SAP BI on DB2 for z/OS |
| | Partitioning |
| 917568 | E- fact tables with DPSIs |
| 860830 | Partitioning an F- fact table |
| 848462 | Aggregate setup after physical partitioning |
| 485878 | Partitioning the PSA tables |
| 894045 | Deleting a partition quickly |
| 926378 | Only one request per partition in PSA load |
| 1004889 | Numerous ADD PARTITION in succession |
| | |
| | RUNSTATS |
| 915398 | Neuer Algorithmus für Aufruf von RUNSTATS |
| 884548 | Additional option for RUNSTATS using DB13 |
| 688133 | Customer-specific RUNSTATS options |
| 775318 | Reduced database statistics |
| 748159 | Parallel statistics calculation |
| | |
| | Other |

| | SAP BI on DB2 for z/OS |
|---------|---|
| 390016 | BW Database settings and performance |
| 633832 | BW Performance analysis for DB2 database |
| 842792 | DD card RNPRIN01 or RNPRINT is missing |
| 859583 | Report SAP_BWTOOLPGM_DB2 |
| 523552 | DB-Connect: Background information for BI on DB2 z/OS |
| 986202 | Migrating PSA tables to the new schema |
| 884340 | Long runtime for realignment |
| 756933 | fact tables with DB2 data compression |
| 732917 | Deadlocks during hierarchy processing |
| 828290 | Long run time to fill XY table |
| 536074 | Buffer pools for fact and dimension tables |
| 1008295 | Dedicated WLM qualifier for DSNUTILS |
| 594263 | RSADMIN parameters for DB2 z/OS |



D

Project Jupiter: evaluation details

Archived

Network configuration: static multipath

Project Jupiter is a joint project between IBM and SAP that demonstrates the performance capabilities of SAP BIA in a very large IBM configuration. It is described in Chapter 11, “Project Jupiter: large BI Accelerator scalability evaluation” on page 227. To illustrate that chapter, this appendix contains definitions and detailed results.

Examples D-1 through D-3 show our static multipath connection setup between System z and System p. Example D-1 shows the TCP/IP profile setup for System z. It defines eight GbE connections and one VIPA connection.

Example D-1 System z TCP/IP profile

```
; Hardware definitions:
;
; CU Ethernet (10GBE) Performance subnet
  DEVICE CU041010 MPCIPA
  LINK P17E11 IPAQENET CU041010

; CU Ethernet (10GBE) Performance subnet
  DEVICE CU051020 MPCIPA
  LINK P17E12 IPAQENET CU051020

; CU Ethernet (10GBE) Performance subnet
  DEVICE CU061030 MPCIPA
  LINK P17E13 IPAQENET CU061030

; CU Ethernet (10GBE) Performance subnet
  DEVICE CU071040 MPCIPA
  LINK P17E14 IPAQENET CU071040

; CU Ethernet (10GBE) Performance subnet
  DEVICE CU081050 MPCIPA
  LINK P17E15 IPAQENET CU081050

; CU Ethernet (10GBE) Performance subnet
  DEVICE CU091060 MPCIPA
  LINK P17E16 IPAQENET CU091060

; CU Ethernet (10GBE) Performance subnet
  DEVICE CU0A1070 MPCIPA
  LINK P17E17 IPAQENET CU0A1070

; CU Ethernet (10GBE) Performance subnet
```



```

DEVICE CU0B1080 MPCIPA
LINK P17E18    IPAQENET    CU0B1080

; VIPA interface
DEVICE VDEV1 VIRTUAL 0
LINK VIPA1    VIRTUAL 0 VDEV1

HOME
    129.40.178.17    PELP17
    129.40.241.11    P17E11
    129.40.242.12    P17E12
    129.40.243.13    P17E13
    129.40.244.14    P17E14
    129.40.245.15    P17E15
    129.40.246.16    P17E16
    129.40.247.17    P17E17
    129.40.248.18    P17E18

    129.40.184.1     VIPA1

GATEWAY
;
; Direct Routes - Routes that are directly connected to my interfaces.
;
; Network    First Hop    Link Name    PacketSize    Subnet Mask    Subnet Value
;
    129.40    =            P17E11      9000          0.0.255.0     0.0.241.0
    129.40    =            P17E12      9000          0.0.255.0     0.0.242.0
    129.40    =            P17E13      9000          0.0.255.0     0.0.243.0
    129.40    =            P17E14      9000          0.0.255.0     0.0.244.0
    129.40    =            P17E15      9000          0.0.255.0     0.0.245.0
    129.40    =            P17E16      9000          0.0.255.0     0.0.246.0
    129.40    =            P17E17      9000          0.0.255.0     0.0.247.0
    129.40    =            P17E18      9000          0.0.255.0     0.0.248.0

; Start all the defined devices.

START CU041010
START CU051020
START CU061030
START CU071040
START CU081050

```

```

START CU091060
START CU0A1070
START CU0B1080

```

Example D-2 shows **netstat -in** information for System p. It also shows eight GbE connections using an MTU size of 9000 for our network interface.

Example D-2 System p network interface information

```

r60t01:napadm 13> netstat -in
Name Mtu Network Address Ipkts Ierrs Opkts Oerrs Coll
en0 9000 link#2 0.11.25.7b.5.9f 326810878 0 141344687 1 0
en0 9000 129.40.243 129.40.243.3 326810878 0 141344687 1 0
en1 9000 link#3 0.11.25.7b.4.f3 357858123 0 153794074 2 0
en1 9000 129.40.241 129.40.241.1 357858123 0 153794074 2 0
en2 9000 link#4 0.11.25.7b.7.45 293587854 0 126895182 2 0
en2 9000 129.40.242 129.40.242.2 293587854 0 126895182 2 0
en3 9000 link#5 0.11.25.7b.5.71 227762617 0 98547230 2 0
en3 9000 129.40.244 129.40.244.4 227762617 0 98547230 2 0
en7 9000 link#6 0.11.25.7b.5.47 296920097 0 127977123 2 0
en7 9000 129.40.247 129.40.247.7 296920097 0 127977123 2 0
en8 9000 link#7 0.11.25.7b.a.42 325638428 0 140604715 2 0
en8 9000 129.40.245 129.40.245.5 325638428 0 140604715 2 0
en9 9000 link#8 0.11.25.7b.4.d9 379675884 0 162570375 2 0
en9 9000 129.40.248 129.40.248.8 379675884 0 162570375 2 0
en10 9000 link#9 0.11.25.7b.5.5c 407537314 0 173067762 2 0
en10 9000 129.40.246 129.40.246.6 407537314 0 173067762 2 0
en12 1500 link#10 0.14.5e.b7.73.92 10745001 0 8899014 2 0
en12 1500 129.40.28 129.40.28.10 10745001 0 8899014 2 0
en11 1500 link#11 0.11.25.7b.4.d2 980950395 0 172190608 0 0
en11 1500 129.40.49 129.40.49.200 980950395 0 172190608 0 0
lo0 16896 link#1 2509044 0 2512091 0 0
lo0 16896 127 127.0.0.1 2509044 0 2512091 0 0
lo0 16896 ::1 2509044 0 2512091 0 0

```

Example D-3 shows static routing of the GbE connections to a VIPA connection. This is defined through smitty with the multipath option and policy=weighed round-robin (RR) algorithm and weight=1. Thus, network traffic is sprayed to all eight GbE connections evenly for load balancing.

Example D-3 System p route with multipath option

```

Destination Gateway Flags Refs Use If Exp Groups
Route Tree for Protocol Family 2 (Internet):
default 129.40.49.254 UG 6 64967 en11 - -
127/8 127.0.0.1 U 26 2499406 lo0 - -

```

| | | | | | | | | |
|------------------------|----------------------|-----------|-----------|------------------|-------------|---|---|--------------|
| 129.40.28.0 | 129.40.28.10 | UHSb | 0 | 0 | en12 | - | - | => |
| 129.40.28/27 | 129.40.28.10 | U | 2 | 8898501 | en12 | - | - | |
| 129.40.28.10 | 127.0.0.1 | UGHS | 0 | 7 | 1o0 | - | - | |
| 129.40.28.31 | 129.40.28.10 | UHSb | 0 | 4 | en12 | - | - | |
| 129.40.49.0 | 129.40.49.200 | UHSb | 0 | 0 | en11 | - | - | => |
| 129.40.49/24 | 129.40.49.200 | U | 835 | 172156947 | en11 | - | - | |
| | | | | | | | | |
| 129.40.184.1/32 | 129.40.243.13 | UG | 57 | 141346068 | en0 | - | - | => |
| 129.40.184.1/32 | 129.40.241.11 | UG | 57 | 153795620 | en1 | - | - | => |
| 129.40.184.1/32 | 129.40.242.12 | UG | 58 | 126899865 | en2 | - | - | => |
| 129.40.184.1/32 | 129.40.244.14 | UG | 58 | 98547899 | en3 | - | - | => |
| 129.40.184.1/32 | 129.40.245.15 | UG | 58 | 140606360 | en8 | - | - | => |
| 129.40.184.1/32 | 129.40.247.17 | UG | 58 | 127978194 | en7 | - | - | => |
| 129.40.184.1/32 | 129.40.246.16 | UG | 57 | 173077376 | en10 | - | - | => |
| 129.40.184.1/32 | 129.40.248.18 | UG | 58 | 162571986 | en9 | - | - | |
| | | | | | | | | |
| 129.40.248/24 | 129.40.248.8 | U | 1 | 1 | en9 | - | - | |
| 129.40.248.8 | 127.0.0.1 | UGHS | 0 | 7 | 1o0 | - | - | |
| 129.40.248.255 | 129.40.248.8 | UHSb | 0 | 4 | en9 | - | - | |

SAP BI database

This section contains information about our InfoCube specifications and our bufferpool parameters. Table D-1 shows the row length and total rows for each database size.

Table D-1 Database summary of InfoCubes

| Table name | Average row length | Total rows 5 TB | Total rows 15.3 TB | Total rows 25 TB |
|----------------|--------------------|-----------------|--------------------|------------------|
| /BIC/FJSDBLC01 | 590 | 60,000,000 | 270,000,000 | 495,000,000 |
| /BIC/FJSDBLC02 | 590 | 60,000,000 | 275,000,000 | 500,000,000 |
| /BIC/FJSDBLC03 | 590 | 60,000,000 | 275,000,000 | 499,600,000 |
| /BIC/FJSDBLC04 | 590 | 60,000,000 | 282,800,000 | 507,800,000 |
| /BIC/FJSDBLC05 | 590 | 60,000,000 | 272,000,000 | 497,000,000 |
| /BIC/FJSDBLC06 | 590 | 60,000,000 | 282,200,000 | 507,000,000 |
| /BIC/FJSDBLC07 | 590 | 60,000,000 | 275,000,000 | 500,000,000 |
| /BIC/FJSDBLC08 | 590 | 60,000,000 | 283,000,000 | 508,000,000 |

| Table name | Average row length | Total rows 5 TB | Total rows 15.3 TB | Total rows 25 TB |
|-----------------|--------------------|-----------------|--------------------|------------------|
| /BIC/FJSDBLC09 | 590 | 60,000,000 | 270,000,000 | 493,200,000 |
| /BIC/FJSDBLC10 | 590 | 60,000,000 | 283,000,000 | 506,800,000 |
| | | | | |
| /BIC/FJSDDL01 | 760 | 60,000,000 | 270,000,000 | 499,000,000 |
| /BIC/FJSDDL02 | 760 | 60,000,000 | 280,000,000 | 509,000,000 |
| /BIC/FJSDDL03 | 760 | 60,000,000 | 270,086,933 | 499,200,000 |
| /BIC/FJSDDL04 | 760 | 60,000,000 | 280,000,000 | 509,000,000 |
| /BIC/FJSDDL05 | 760 | 60,000,000 | 275,000,000 | 504,000,000 |
| /BIC/FJSDDL06 | 760 | 60,000,000 | 280,013,811 | 509,000,000 |
| /BIC/FJSDDL07 | 760 | 60,000,000 | 270,000,000 | 499,200,000 |
| /BIC/FJSDDL08 | 760 | 60,000,000 | 275,000,000 | 504,000,000 |
| /BIC/FJSDDL09 | 760 | 60,000,000 | 275,000,054 | 504,200,000 |
| /BIC/FJSDDL10 | 760 | 60,000,000 | 270,000,000 | 499,000,000 |
| | | | | |
| /BIC/FJSDORC01 | 910 | 55,000,000 | 265,000,000 | 481,200,000 |
| /BIC/FJSDORC02 | 910 | 60,000,000 | 275,000,000 | 491,400,000 |
| /BIC/FJSDORC03 | 910 | 60,000,000 | 270,000,000 | 486,400,000 |
| /BIC/FJSDORC04 | 910 | 60,000,000 | 270,000,000 | 486,400,000 |
| /BIC/FJSDORC05 | 910 | 60,000,000 | 270,000,000 | 486,200,000 |
| /BIC/FJSDORC06 | 910 | 60,000,000 | 275,000,000 | 491,400,000 |
| /BIC/FJSDORC07 | 910 | 60,000,000 | 265,000,000 | 486,400,000 |
| /BIC/FJSDORC08 | 910 | 60,000,000 | 275,000,000 | 491,400,000 |
| /BIC/FJSDORC09 | 910 | 60,000,000 | 269,999,999 | 486,400,000 |
| /BIC/FJSDORC10 | 910 | 60,000,000 | 275,000,000 | 491,400,000 |
| | | | | |
| /BIC/FZJB_07_01 | 653 | 55,000,000 | 179,999,998 | 280,000,000 |

| Table name | Average row length | Total rows 5 TB | Total rows 15.3 TB | Total rows 25 TB |
|-----------------|--------------------|-----------------|--------------------|------------------|
| /BIC/FZJB_07_02 | 653 | 60,000,000 | 185,000,000 | 285,000,000 |
| /BIC/FZJB_07_03 | 653 | 60,000,000 | 185,000,000 | 285,000,000 |
| /BIC/FZJB_07_04 | 653 | 55,000,000 | 180,000,000 | 280,000,000 |
| /BIC/FZJB_07_05 | 653 | 60,000,000 | 185,004,657 | 285,000,000 |
| /BIC/FZJB_07_06 | 653 | 60,000,000 | 185,000,000 | 285,000,000 |
| /BIC/FZJB_07_07 | 653 | 60,000,000 | 185,000,000 | 285,000,000 |
| /BIC/FZJB_07_08 | 653 | 60,000,000 | 185,000,000 | 285,000,000 |
| /BIC/FZJB_07_09 | 653 | 60,000,000 | 185,000,000 | 285,000,000 |
| /BIC/FZJB_07_10 | 653 | 55,000,000 | 180,000,000 | 280,000,000 |
| /BIC/FZJB_07_11 | 653 | 60,000,000 | 185,000,000 | 285,000,000 |
| /BIC/FZJB_07_12 | 653 | 60,000,000 | 185,000,000 | 285,000,000 |
| | | | | |
| /BIC/FZJB_08_01 | 653 | 55,000,000 | 180,000,000 | 278,400,000 |
| /BIC/FZJB_08_02 | 653 | 60,000,000 | 185,000,000 | 283,400,000 |
| /BIC/FZJB_08_03 | 653 | 60,000,000 | 185,000,000 | 283,800,000 |
| /BIC/FZJB_08_04 | 653 | 55,000,000 | 180,000,000 | 278,400,000 |
| /BIC/FZJB_08_05 | 653 | 60,000,000 | 187,553,848 | 284,200,000 |
| /BIC/FZJB_08_06 | 653 | 60,000,000 | 185,000,000 | 283,800,000 |
| /BIC/FZJB_08_07 | 653 | 60,000,000 | 185,000,000 | 283,200,000 |
| /BIC/FZJB_08_08 | 653 | 60,000,000 | 185,000,000 | 283,400,000 |
| /BIC/FZJB_08_09 | 653 | 60,000,000 | 185,000,000 | 283,600,000 |
| /BIC/FZJB_08_10 | 653 | 55,000,000 | 175,000,000 | 274,000,000 |
| /BIC/FZJB_08_11 | 653 | 60,000,000 | 185,000,000 | 283,600,000 |
| /BIC/FZJB_08_12 | 653 | 60,000,000 | 185,000,000 | 283,600,000 |
| | | | | |
| /BIC/FZJB_09_01 | 653 | 55,000,000 | 180,000,000 | 280,000,000 |

| Table name | Average row length | Total rows 5 TB | Total rows 15.3 TB | Total rows 25 TB |
|-----------------|--------------------|-----------------|--------------------|------------------|
| /BIC/FZJB_09_02 | 653 | 60,000,000 | 185,000,000 | 280,000,000 |
| /BIC/FZJB_09_03 | 653 | 60,000,000 | 185,000,000 | 285,000,000 |
| /BIC/FZJB_09_04 | 653 | 55,000,000 | 180,000,000 | 280,000,000 |
| /BIC/FZJB_09_05 | 653 | 60,000,000 | 185,000,000 | 285,000,000 |
| /BIC/FZJB_09_06 | 653 | 60,000,000 | 185,000,000 | 285,000,000 |
| /BIC/FZJB_09_07 | 653 | 60,000,000 | 185,000,000 | 285,000,000 |
| /BIC/FZJB_09_08 | 653 | 60,000,000 | 185,000,000 | 285,000,000 |
| /BIC/FZJB_09_09 | 653 | 60,000,000 | 185,000,000 | 285,000,000 |
| /BIC/FZJB_09_10 | 653 | 55,000,000 | 180,000,000 | 280,000,000 |
| /BIC/FZJB_09_11 | 653 | 60,000,000 | 185,000,000 | 285,000,000 |
| /BIC/FZJB_09_12 | 653 | 60,000,000 | 185,000,000 | 285,000,000 |
| | | | | |
| /BIC/FZJB_10_01 | 653 | 55,000,000 | 180,000,000 | 280,000,000 |
| /BIC/FZJB_10_02 | 653 | 60,000,000 | 185,000,000 | 280,000,000 |
| /BIC/FZJB_10_03 | 653 | 60,000,000 | 185,000,000 | 285,000,000 |
| /BIC/FZJB_10_04 | 653 | 50,000,000 | 175,000,000 | 275,000,000 |
| /BIC/FZJB_10_05 | 653 | 60,000,000 | 185,000,000 | 285,000,000 |
| /BIC/FZJB_10_06 | 653 | 60,000,000 | 185,000,000 | 285,000,000 |
| /BIC/FZJB_10_07 | 653 | 60,000,000 | 185,000,000 | 285,000,000 |
| /BIC/FZJB_10_08 | 653 | 60,000,000 | 186,631,315 | 285,000,000 |
| /BIC/FZJB_10_09 | 653 | 60,000,000 | 180,000,000 | 280,000,000 |
| /BIC/FZJB_10_10 | 653 | 55,000,000 | 180,000,000 | 280,000,000 |
| /BIC/FZJB_10_11 | 653 | 60,000,000 | 185,000,000 | 285,000,000 |
| /BIC/FZJB_10_12 | 653 | 60,000,000 | 185,000,000 | 285,000,000 |

Tables D-2 through D-5 show the settings of our bufferpools for each page size.

Table D-2 DB2 bufferpool settings - 4 K pages

| Parameters | BP0 | BP1 | BP2 | BP3 | BP40 |
|---|-----|-----|-----|-----|------|
| VIRTUAL POOL SIZE (in 1000) | 4 | 320 | 160 | 160 | 20 |
| VIRTUAL POOL SEQUENTIAL THRESHOLD | 50 | 100 | 50 | 40 | 50 |
| HORIZONTAL DEFERRED WRITE THRESHOLD | 50 | 50 | 50 | 30 | 50 |
| VERTICAL DEFERRED WRITE THRESHOLD (%) | 10 | 10 | 5 | 5 | 10 |
| VERTICAL DEFERRED WRITE THRESHOLD (BUFFERS) | 0 | 0 | 0 | 0 | 0 |
| VIRTUAL POOL PARALLEL SEQUENTIAL THRESHOLD | 50 | 50 | 50 | 50 | 50 |
| PGFIX ATTRIBUTE | YES | YES | YES | YES | YES |
| PAGE STEAL METHOD | LRU | LRU | LRU | LRU | LRU |

Table D-3 DB2 bufferpool settings - 8 k pages

| Parameters | BP8K0 | BP8K1 |
|---|-------|-------|
| VIRTUAL POOL SIZE (in thousands) | 160 | 10 |
| VIRTUAL POOL SEQUENTIAL THRESHOLD | 50 | 80 |
| HORIZONTAL DEFERRED WRITE THRESHOLD | 30 | 30 |
| VERTICAL DEFERRED WRITE THRESHOLD (%) | 5 | 5 |
| VERTICAL DEFERRED WRITE THRESHOLD (BUFFERS) | 0 | 0 |
| VIRTUAL POOL PARALLEL SEQUENTIAL THRESHOLD | 50 | 50 |
| PGFIX ATTRIBUTE | YES | NO |
| PAGE STEAL METHOD | LRU | LRU |

Table D-4 DB2 bufferpool settings - 16 k pages

| Parameters | BPK16K0 | BPK16K1 |
|---|---------|---------|
| VIRTUAL POOL SIZE (in thousands) | 3 | 10 |
| VIRTUAL POOL SEQUENTIAL THRESHOLD | 50 | 80 |
| HORIZONTAL DEFERRED WRITE THRESHOLD | 30 | 30 |
| VERTICAL DEFERRED WRITE THRESHOLD (%) | 5 | 5 |
| VERTICAL DEFERRED WRITE THRESHOLD (BUFFERS) | 0 | 0 |

| Parameters | BPK16K0 | BPK16K1 |
|--|---------|---------|
| VIRTUAL POOL PARALLEL SEQUENTIAL THRESHOLD | 50 | 50 |
| PGFIX ATTRIBUTE | YES | NO |
| PAGE STEAL METHOD | LRU | LRU |

Table D-5 DB2 bufferpool settings - 32 k pages

| Parameters | BP32K | BP32K1 | BP32K2 | BP32K3 | BP32K4 |
|---|-------|--------|--------|--------|--------|
| VIRTUAL POOL SIZE (in thousands) | 320 | 320 | 320 | 320 | 80 |
| VIRTUAL POOL SEQUENTIAL THRESHOLD | 50 | 50 | 50 | 80 | 80 |
| HORIZONTAL DEFERRED WRITE THRESHOLD | 30 | 30 | 30 | 30 | 30 |
| VERTICAL DEFERRED WRITE THRESHOLD (%) | 5 | 5 | 5 | 5 | 5 |
| VERTICAL DEFERRED WRITE THRESHOLD (BUFFERS) | 0 | 0 | 0 | 0 | 0 |
| VIRTUAL POOL PARALLEL SEQUENTIAL THRESHOLD | 50 | 50 | 50 | 50 | 50 |
| PGFIX ATTRIBUTE | YES | YES | YES | YES | YES |
| PAGE STEAL METHOD | LRU | LRU | LRU | LRU | LRU |

BIA index creation

Examples D-4 through D-8 show cubes that have been indexed to GPFS. They display examples of horizontal partitioning, vertical decomposition, and smart compression of BIA index creation. Example D-4 shows a list of cubes that were indexed to BIA blade GPFS.

Example D-4 Cubes indexed to GPFS

```
t11adm@jup001:/gpfs/fs1/T11/TRX00/index> ls -rtl
total 454
drwxr-xr-x   3 t11adm sapsys   2048 2007-08-10 11:52 nap_jsdblc00
drwxr-xr-x  16 t11adm sapsys   2048 2007-08-10 11:53 nap_b49
drwxr-xr-x   3 t11adm sapsys   2048 2007-08-10 11:53 nap_zjb_00_00
drwxr-xr-x   3 t11adm sapsys   2048 2007-08-10 11:58 nap_jsddl00
drwxr-xr-x 139 t11adm sapsys  8192 2007-08-10 12:01 _trex
```


| | | | | | | | |
|------------|-----|--------|--------|------|------------|-------|----------------------|
| drwxr-xr-x | 108 | tlladm | sapsys | 4096 | 2007-08-10 | 12:02 | nap_bi0 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:02 | nap_jsdorc00 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:18 | nap_jsdblc01 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:20 | nap_jsddl01 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:22 | nap_jsdorc01 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:24 | nap_zjb_07_01 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:26 | nap_zjb_07_02 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:29 | nap_zjb_07_03 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:30 | nap_zjb_07_04 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:31 | nap_zjb_07_05 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:32 | nap_zjb_07_06 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:33 | nap_zjb_07_07 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:35 | nap_zjb_07_08 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:35 | nap_zjb_07_09 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:37 | nap_zjb_07_10 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:37 | nap_zjb_07_11 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:39 | nap_zjb_07_12 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:40 | nap_jsdblc02 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:40 | nap_jsddl02 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:41 | nap_zjb_08_01 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:41 | nap_jsdorc02 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:41 | nap_zjb_08_02 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:44 | nap_zjb_08_03 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:45 | nap_zjb_08_04 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:45 | nap_zjb_08_05 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:46 | nap_zjb_08_06 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:48 | nap_zjb_08_07 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:48 | nap_zjb_08_08 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:50 | nap_zjb_08_09 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:51 | nap_zjb_08_10 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:51 | nap_zjb_08_11 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:51 | nap_zjb_08_12 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:54 | nap_jsdblc03 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:58 | nap_jsddl03 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 12:59 | nap_jsdorc03 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 13:00 | nap_zjb_09_01 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 13:01 | nap_zjb_09_02 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 13:02 | nap_zjb_09_03 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 13:05 | nap_zjb_09_04 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 13:07 | nap_zjb_09_05 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 13:11 | nap_zjb_09_06 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 13:11 | nap_zjb_09_07 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 13:13 | nap_zjb_09_08 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 13:13 | nap_zjb_09_09 |
| drwxr-xr-x | 3 | tlladm | sapsys | 2048 | 2007-08-10 | 13:14 | nap_zjb_09_10 |

```

drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:19 nap_zjb_09_11
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:19 nap_zjb_09_12
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:20 nap_jsdblc04
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:21 nap_jsddl04
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:22 nap_jsdorc04
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:22 nap_zjb_10_02
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:22 nap_zjb_10_01
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:27 nap_zjb_10_03
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:29 nap_zjb_10_04
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:30 nap_zjb_10_05
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:34 nap_zjb_10_06
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:35 nap_zjb_10_07
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:37 nap_zjb_10_08
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:40 nap_zjb_10_09
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:42 nap_zjb_10_11
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:43 nap_zjb_10_10
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:43 nap_zjb_10_12
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:43 nap_jsdblc05
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:44 nap_jsddl05
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:47 nap_jsdblc06
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:48 nap_jsdorc05
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:50 nap_jsdorc06
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:50 nap_jsddl06
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:50 nap_jsdblc07
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:51 nap_jsddl07
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:53 nap_jsdorc07
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:58 nap_jsdblc08
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 13:59 nap_jsddl08
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 14:00 nap_jsdblc09
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 14:00 nap_jsdorc08
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 14:01 nap_jsddl09
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 14:03 nap_jsdorc09
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 14:09 nap_jsdblc10
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 14:11 nap_jsddl10
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 14:11 nap_phsdblc01
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 14:11 nap_phsdblc02
drwxr-xr-x      3 t11adm sapsys  2048 2007-08-10 14:14 nap_jsdorc10
drwxr-xr-x 4585 t11adm sapsys 262144 2007-08-10 15:28 nap_bic
t11adm@jup001:/gpfs/fs1/T11/TRX00/index>

```

Figure D-1 shows TREX Admin settings for horizontal partitioning and the threshold. The threshold was set to max_size=250,000,000 cells to be considered for horizontal partitioning and max_split_parts=40 for number of parts or blade parallelism.

The screenshot shows the SAP TREX Administration interface. The 'Tree' view on the left shows a hierarchy of folders: topology, int.all, cruise, defaults, alertserver, cruise, index, landscape_reorganize, nameserver, queue, globals, host, index, queue, reorg, and timing. The 'index' folder is selected. The main pane displays a list of settings with columns for 'Name' and 'Value'. The following table represents the data shown in the screenshot:

| Name | Value |
|--------------------------|---|
| backup_hosts_jup118 | jup140 |
| backup_hosts_jup119 | jup140 |
| backup_hosts_jup120 | jup140 |
| backup_hosts_jup121 | jup140 |
| backup_hosts_jup122 | jup140 |
| backup_hosts_jup123 | jup140 |
| backup_hosts_jup124 | jup140 |
| backup_hosts_jup125 | jup140 |
| backup_hosts_jup126 | jup140 |
| backup_hosts_jup127 | jup140 |
| backup_hosts_jup128 | jup140 |
| backup_hosts_jup129 | jup140 |
| backup_hosts_jup130 | jup140 |
| backup_hosts_jup131 | jup140 |
| backup_hosts_jup132 | jup140 |
| backup_hosts_jup133 | jup140 |
| backup_hosts_jup134 | jup140 |
| backup_hosts_jup135 | jup140 |
| backup_hosts_jup136 | jup140 |
| backup_hosts_jup137 | jup140 |
| backup_hosts_jup138 | jup140 |
| backup_hosts_jup139 | jup140 |
| createbackup@createindex | yes |
| createbackup@setactive | yes |
| createslaves@createindex | yes |
| createslaves@setactive | yes |
| master_hosts | jup001 jup002 jup003 jup004 jup005 jup006 jup007 jup008 jup009 jup010 jup011 jup012 |
| max_size | 250000000 |
| max_split_parts | 40 |
| preload | no |
| replica_threads | 2 |

Figure D-1 TREX Admin settings for horizontal partitioning and threshold

Example D-6 shows a cube index creation with horizontal partitioning of 40 parts.

Example D-5 Horizontal partitioning with 40 parts

```
t11adm@jup001:/gpfs/fs1/T11/TRX00/index/nap_bic>ls -rtl
...
drwxr-xr-x  3 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~01
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~02
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~03
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~04
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~05
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~06
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~07
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~08
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~09
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~10
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~11
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~12
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~13
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~14
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~15
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~16
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~17
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~18
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~19
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~20
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~21
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~22
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~23
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~24
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~25
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:13 fzjb_07_11~26
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:14 fzjb_07_11~27
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:14 fzjb_07_11~28
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:14 fzjb_07_11~29
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:14 fzjb_07_11~30
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:14 fzjb_07_11~31
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:14 fzjb_07_11~32
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:14 fzjb_07_11~33
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:14 fzjb_07_11~34
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:14 fzjb_07_11~35
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:14 fzjb_07_11~36
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:14 fzjb_07_11~37
drwxr-xr-x  4 t11adm sapsys 2048 2007-12-05 17:14 fzjb_07_11~38
```

```

drwxr-xr-x 4 t1ladm sapsys 2048 2007-12-05 17:14 fzjb_07_11~39
drwxr-xr-x 4 t1ladm sapsys 2048 2007-12-05 17:14 fzjb_07_11~40
drwxr-xr-x 4 t1ladm sapsys 2048 2007-12-05 17:14 dzjb_07_11p
drwxr-xr-x 4 t1ladm sapsys 2048 2007-12-05 17:14 dzjb_07_11u
drwxr-xr-x 4 t1ladm sapsys 2048 2007-12-05 17:14 dzjb_07_11i
drwxr-xr-x 4 t1ladm sapsys 2048 2007-12-05 17:14 dzjb_07_11t
drwxr-xr-x 4 t1ladm sapsys 2048 2007-12-05 17:14 dzjb_07_113
drwxr-xr-x 4 t1ladm sapsys 2048 2007-12-05 17:14 dzjb_07_115
drwxr-xr-x 4 t1ladm sapsys 2048 2007-12-05 17:14 dzjb_07_114
drwxr-xr-x 4 t1ladm sapsys 2048 2007-12-05 17:14 dzjb_07

```

Example D-5 displays all the columns in Table SAPR3./BIC/FZJB_07_11 using the DB2 Admin tool. It is helpful to observe the column names and match them with the list of temporary and persistent objects in subsequent examples.

Example D-6 Displaying the columns of a fact table of an index cube

```

DB2 Admin -- DSN9 Columns in Table SAPR3./BIC/FZJB_07_11 > PAGE

```

| Select | Column Name | Col No | Col Type | Length | Scale | Null | Def | FP | Col Card |
|--------|------------------|--------|----------|--------|-------|------|-----|----|----------|
| * | | * | * | * | * | * | * | * | * |
| | KEY_ZJB_07_11P | 1 | INTEGER | 4 | 0 | N | 4 | N | 35 |
| | KEY_ZJB_07_11T | 2 | INTEGER | 4 | 0 | N | 4 | N | 30 |
| | KEY_ZJB_07_11U | 3 | INTEGER | 4 | 0 | N | 4 | N | 1 |
| | KEY_ZJB_07_11I | 4 | INTEGER | 4 | 0 | N | 4 | N | 0 |
| | KEY_ZJB_07_112 | 5 | INTEGER | 4 | 0 | N | 4 | N | 91136 |
| | KEY_ZJB_07_113 | 6 | INTEGER | 4 | 0 | N | 4 | N | 81920 |
| | KEY_ZJB_07_114 | 7 | INTEGER | 4 | 0 | N | 4 | N | 99 |
| | KEY_ZJB_07_115 | 8 | INTEGER | 4 | 0 | N | 4 | N | 99 |
| | SID_OCALMONTH | 9 | INTEGER | 4 | 0 | N | 4 | N | -1 |
| | /B49/S_CRMEM_CST | 10 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| | /B49/S_CRMEM_QTY | 11 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| | /B49/S_CRMEM_VAL | 12 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| | /B49/S_INCORDCST | 13 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| | /B49/S_INCORDQTY | 14 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| | /B49/S_INCORDVAL | 15 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| | /B49/S_INVCD_CST | 16 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| | /B49/S_INVCD_QTY | 17 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| | /B49/S_INVCD_VAL | 18 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| | /B49/S_OPORDQTYB | 19 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| | /B49/S_OPORDVALS | 20 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| | /B49/S_ORD_ITEMS | 21 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| | /B49/S_RTNSCST | 22 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| | /B49/S_RTNSQTY | 23 | DECIMAL | 17 | 3 | N | 4 | N | -1 |

| | | | | | | | | |
|------------------|----|---------|----|---|---|---|---|----|
| /B49/S_RTNSVAL | 24 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| /B49/S_RTNS_ITEM | 25 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/AMOUNT01 | 26 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| /BIC/AMOUNT02 | 27 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| /BIC/AMOUNT03 | 28 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| /BIC/AMOUNT04 | 29 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| /BIC/AMOUNT05 | 30 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| /BIC/AMOUNT06 | 31 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| /BIC/AMOUNT07 | 32 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| /BIC/AMOUNT08 | 33 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| /BIC/AMOUNT09 | 34 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| /BIC/AMOUNT10 | 35 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| /BIC/AMOUNT11 | 36 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| /BIC/AMOUNT12 | 37 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| /BIC/AMOUNT13 | 38 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| /BIC/AMOUNT14 | 39 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| /BIC/AMOUNT15 | 40 | DECIMAL | 17 | 2 | N | 4 | N | -1 |
| /BIC/COUNTER01 | 41 | INTEGER | 4 | 0 | N | 4 | N | -1 |
| /BIC/COUNTER02 | 42 | INTEGER | 4 | 0 | N | 4 | N | -1 |
| /BIC/COUNTER03 | 43 | INTEGER | 4 | 0 | N | 4 | N | -1 |
| /BIC/COUNTER04 | 44 | INTEGER | 4 | 0 | N | 4 | N | -1 |
| /BIC/COUNTER05 | 45 | INTEGER | 4 | 0 | N | 4 | N | -1 |
| /BIC/COUNTER06 | 46 | INTEGER | 4 | 0 | N | 4 | N | -1 |
| /BIC/COUNTER07 | 47 | INTEGER | 4 | 0 | N | 4 | N | -1 |
| /BIC/COUNTER08 | 48 | INTEGER | 4 | 0 | N | 4 | N | -1 |
| /BIC/COUNTER09 | 49 | INTEGER | 4 | 0 | N | 4 | N | -1 |
| /BIC/COUNTER10 | 50 | INTEGER | 4 | 0 | N | 4 | N | -1 |
| /BIC/COUNTER11 | 51 | INTEGER | 4 | 0 | N | 4 | N | -1 |
| /BIC/COUNTER12 | 52 | INTEGER | 4 | 0 | N | 4 | N | -1 |
| /BIC/COUNTER13 | 53 | INTEGER | 4 | 0 | N | 4 | N | -1 |
| /BIC/COUNTER14 | 54 | INTEGER | 4 | 0 | N | 4 | N | -1 |
| /BIC/COUNTER15 | 55 | INTEGER | 4 | 0 | N | 4 | N | -1 |
| /BIC/NUMBER01 | 56 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/NUMBER02 | 57 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/NUMBER03 | 58 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/NUMBER04 | 59 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/NUMBER05 | 60 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/NUMBER06 | 61 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/NUMBER07 | 62 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/NUMBER08 | 63 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/NUMBER09 | 64 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/NUMBER10 | 65 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/NUMBER11 | 66 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/NUMBER12 | 67 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/NUMBER13 | 68 | DECIMAL | 17 | 3 | N | 4 | N | -1 |

| | | | | | | | | |
|---------------|----|---------|----|---|---|---|---|----|
| /BIC/NUMBER14 | 69 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/NUMBER15 | 70 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/QUAN01 | 71 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/QUAN02 | 72 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/QUAN03 | 73 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/QUAN04 | 74 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/QUAN05 | 75 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/QUAN06 | 76 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/QUAN07 | 77 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/QUAN08 | 78 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/QUAN09 | 79 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/QUAN10 | 80 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/QUAN11 | 81 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/QUAN12 | 82 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/QUAN13 | 83 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/QUAN14 | 84 | DECIMAL | 17 | 3 | N | 4 | N | -1 |
| /BIC/QUAN15 | 85 | DECIMAL | 17 | 3 | N | 4 | N | -1 |

Example D-7 displays temporary objects of an InfoCube during the first phase of vertical decomposition. The temporary objects will be removed after persistent objects are created. Note that each *.tmpx file corresponds to a column listed earlier in Example D-5

Example D-7 Temporary objects of an indexed cube

```
t11adm@jup001:/home/t11adm/nap_bic/fzjb_07_11~23/en> ls -al
(when it is using local disk)
total 193876
drwxr-xr-x  4 t11adm sapsys  4096 2007-08-02 09:33 .
drwxr-xr-x  4 t11adm sapsys  4096 2007-08-02 09:32 ..
drwxr-xr-x  2 t11adm sapsys  4096 2007-08-02 09:32 attributes
-rw-r--r--  1 t11adm sapsys 2007641 2007-08-02 09:44 $b49$s_crmem_cst.tmpx
-rw-r--r--  1 t11adm sapsys 2084482 2007-08-02 09:44 $b49$s_crmem_qty.tmpx
-rw-r--r--  1 t11adm sapsys 2122285 2007-08-02 09:44 $b49$s_crmem_val.tmpx
-rw-r--r--  1 t11adm sapsys 3573234 2007-08-02 09:44 $b49$s_incordcst.tmpx
-rw-r--r--  1 t11adm sapsys 2716452 2007-08-02 09:44 $b49$s_incordqty.tmpx
-rw-r--r--  1 t11adm sapsys 3573234 2007-08-02 09:44 $b49$s_incordval.tmpx
-rw-r--r--  1 t11adm sapsys 3573234 2007-08-02 09:44 $b49$s_invcd_cst.tmpx
-rw-r--r--  1 t11adm sapsys 2714842 2007-08-02 09:44 $b49$s_invcd_qty.tmpx
-rw-r--r--  1 t11adm sapsys 3555548 2007-08-02 09:44 $b49$s_invcd_val.tmpx
-rw-r--r--  1 t11adm sapsys 2057188 2007-08-02 09:44 $b49$s_opordqtyb.tmpx
-rw-r--r--  1 t11adm sapsys 2149884 2007-08-02 09:44 $b49$s_opordvals.tmpx
-rw-r--r--  1 t11adm sapsys 2799927 2007-08-02 09:44 $b49$s_ord_items.tmpx
-rw-r--r--  1 t11adm sapsys 2007641 2007-08-02 09:44 $b49$s_rtncst.tmpx
-rw-r--r--  1 t11adm sapsys 2007689 2007-08-02 09:44 $b49$s_rtncst_item.tmpx
```

```

-rw-r--r-- 1 t1ladm sapsys 2084482 2007-08-02 09:44 $b49$s_rtnsqty.tmpx
-rw-r--r-- 1 t1ladm sapsys 2122285 2007-08-02 09:44 $b49$s_rtnsval.tmpx
-rw-r--r-- 1 t1ladm sapsys 2091822 2007-08-02 09:44 $bic$amount01.tmpx
-rw-r--r-- 1 t1ladm sapsys 2119557 2007-08-02 09:44 $bic$amount02.tmpx
-rw-r--r-- 1 t1ladm sapsys 2131350 2007-08-02 09:44 $bic$amount03.tmpx
-rw-r--r-- 1 t1ladm sapsys 2156915 2007-08-02 09:44 $bic$amount04.tmpx
-rw-r--r-- 1 t1ladm sapsys 2154358 2007-08-02 09:44 $bic$amount05.tmpx
-rw-r--r-- 1 t1ladm sapsys 2156747 2007-08-02 09:44 $bic$amount06.tmpx
-rw-r--r-- 1 t1ladm sapsys 2155379 2007-08-02 09:44 $bic$amount07.tmpx
-rw-r--r-- 1 t1ladm sapsys 2157579 2007-08-02 09:44 $bic$amount08.tmpx
-rw-r--r-- 1 t1ladm sapsys 2156158 2007-08-02 09:44 $bic$amount09.tmpx
-rw-r--r-- 1 t1ladm sapsys 2157088 2007-08-02 09:44 $bic$amount10.tmpx
-rw-r--r-- 1 t1ladm sapsys 2183358 2007-08-02 09:44 $bic$amount11.tmpx
-rw-r--r-- 1 t1ladm sapsys 2181383 2007-08-02 09:44 $bic$amount12.tmpx
-rw-r--r-- 1 t1ladm sapsys 2179923 2007-08-02 09:44 $bic$amount13.tmpx
-rw-r--r-- 1 t1ladm sapsys 2179145 2007-08-02 09:44 $bic$amount14.tmpx
-rw-r--r-- 1 t1ladm sapsys 2176888 2007-08-02 09:44 $bic$amount15.tmpx
-rw-r--r-- 1 t1ladm sapsys 1995904 2007-08-02 09:44 $bic$counter01.tmpx
-rw-r--r-- 1 t1ladm sapsys 1995970 2007-08-02 09:44 $bic$counter02.tmpx
-rw-r--r-- 1 t1ladm sapsys 1995594 2007-08-02 09:44 $bic$counter03.tmpx
-rw-r--r-- 1 t1ladm sapsys 1995658 2007-08-02 09:44 $bic$counter04.tmpx
-rw-r--r-- 1 t1ladm sapsys 1995793 2007-08-02 09:44 $bic$counter05.tmpx
-rw-r--r-- 1 t1ladm sapsys 1996173 2007-08-02 09:44 $bic$counter06.tmpx
-rw-r--r-- 1 t1ladm sapsys 1995748 2007-08-02 09:44 $bic$counter07.tmpx
-rw-r--r-- 1 t1ladm sapsys 1996012 2007-08-02 09:44 $bic$counter08.tmpx
-rw-r--r-- 1 t1ladm sapsys 1996284 2007-08-02 09:44 $bic$counter09.tmpx
-rw-r--r-- 1 t1ladm sapsys 1996173 2007-08-02 09:44 $bic$counter10.tmpx
-rw-r--r-- 1 t1ladm sapsys 1995566 2007-08-02 09:44 $bic$counter11.tmpx
-rw-r--r-- 1 t1ladm sapsys 1996104 2007-08-02 09:44 $bic$counter12.tmpx
-rw-r--r-- 1 t1ladm sapsys 1995707 2007-08-02 09:44 $bic$counter13.tmpx
-rw-r--r-- 1 t1ladm sapsys 1996117 2007-08-02 09:44 $bic$counter14.tmpx
-rw-r--r-- 1 t1ladm sapsys 1995722 2007-08-02 09:44 $bic$counter15.tmpx
-rw-r--r-- 1 t1ladm sapsys 2094322 2007-08-02 09:44 $bic$number01.tmpx
-rw-r--r-- 1 t1ladm sapsys 2107627 2007-08-02 09:44 $bic$number02.tmpx
-rw-r--r-- 1 t1ladm sapsys 2116150 2007-08-02 09:44 $bic$number03.tmpx
-rw-r--r-- 1 t1ladm sapsys 2153452 2007-08-02 09:44 $bic$number04.tmpx
-rw-r--r-- 1 t1ladm sapsys 2156289 2007-08-02 09:44 $bic$number05.tmpx
-rw-r--r-- 1 t1ladm sapsys 2155899 2007-08-02 09:44 $bic$number06.tmpx
-rw-r--r-- 1 t1ladm sapsys 2156776 2007-08-02 09:44 $bic$number07.tmpx
-rw-r--r-- 1 t1ladm sapsys 2154834 2007-08-02 09:44 $bic$number08.tmpx
-rw-r--r-- 1 t1ladm sapsys 2154793 2007-08-02 09:44 $bic$number09.tmpx
-rw-r--r-- 1 t1ladm sapsys 2154130 2007-08-02 09:44 $bic$number10.tmpx
-rw-r--r-- 1 t1ladm sapsys 2178435 2007-08-02 09:44 $bic$number11.tmpx
-rw-r--r-- 1 t1ladm sapsys 2179602 2007-08-02 09:44 $bic$number12.tmpx
-rw-r--r-- 1 t1ladm sapsys 2179118 2007-08-02 09:44 $bic$number13.tmpx

```



```

-rw-r--r-- 1 t1ladm sapsys 2181036 2007-08-02 09:44 $bic$number14.tmpx
-rw-r--r-- 1 t1ladm sapsys 2181499 2007-08-02 09:44 $bic$number15.tmpx
-rw-r--r-- 1 t1ladm sapsys 2084057 2007-08-02 09:44 $bic$quan01.tmpx
-rw-r--r-- 1 t1ladm sapsys 2097352 2007-08-02 09:44 $bic$quan02.tmpx
-rw-r--r-- 1 t1ladm sapsys 2104614 2007-08-02 09:44 $bic$quan03.tmpx
-rw-r--r-- 1 t1ladm sapsys 2109508 2007-08-02 09:44 $bic$quan04.tmpx
-rw-r--r-- 1 t1ladm sapsys 2114724 2007-08-02 09:44 $bic$quan05.tmpx
-rw-r--r-- 1 t1ladm sapsys 2155726 2007-08-02 09:44 $bic$quan06.tmpx
-rw-r--r-- 1 t1ladm sapsys 2158725 2007-08-02 09:44 $bic$quan07.tmpx
-rw-r--r-- 1 t1ladm sapsys 2157700 2007-08-02 09:44 $bic$quan08.tmpx
-rw-r--r-- 1 t1ladm sapsys 2156676 2007-08-02 09:44 $bic$quan09.tmpx
-rw-r--r-- 1 t1ladm sapsys 2157039 2007-08-02 09:44 $bic$quan10.tmpx
-rw-r--r-- 1 t1ladm sapsys 2155510 2007-08-02 09:44 $bic$quan11.tmpx
-rw-r--r-- 1 t1ladm sapsys 2154730 2007-08-02 09:44 $bic$quan12.tmpx
-rw-r--r-- 1 t1ladm sapsys 2155985 2007-08-02 09:44 $bic$quan13.tmpx
-rw-r--r-- 1 t1ladm sapsys 2155565 2007-08-02 09:44 $bic$quan14.tmpx
-rw-r--r-- 1 t1ladm sapsys 2156150 2007-08-02 09:44 $bic$quan15.tmpx
-rw-r--r-- 1 t1ladm sapsys 5425676 2007-08-02 09:44 key_zjb_07_111.tmpx
-rw-r--r-- 1 t1ladm sapsys 6091352 2007-08-02 09:44 key_zjb_07_112.tmpx
-rw-r--r-- 1 t1ladm sapsys 5935488 2007-08-02 09:44 key_zjb_07_113.tmpx
-rw-r--r-- 1 t1ladm sapsys 3040371 2007-08-02 09:44 key_zjb_07_114.tmpx
-rw-r--r-- 1 t1ladm sapsys 3555863 2007-08-02 09:44 key_zjb_07_115.tmpx
-rw-r--r-- 1 t1ladm sapsys 1950018 2007-08-02 09:44 key_zjb_07_11p.tmpx
-rw-r--r-- 1 t1ladm sapsys 2182508 2007-08-02 09:44 key_zjb_07_11t.tmpx
-rw-r--r-- 1 t1ladm sapsys 1949503 2007-08-02 09:44 key_zjb_07_11u.tmpx
drwxr-xr-x 2 t1ladm sapsys 4096 2007-08-02 09:32 trex

```

Figure D-8 shows permanent or persistent objects with files *.bin as final vertical decomposition. Again, each *.bin file corresponds to a column listed in Example D-5. Also notice the file size differences between *.bin and *.tmpx files, after compression.

Example D-8 Permanent objects of an indexed cube

```

t1ladm@jup001:/gpfs/fs1/T11/TRX00/index/nap_bic/fzjb_07_11~23/en/attributes> ls -rtl
total 13874
-rw-r--r-- 1 t1ladm sapsys 12 2007-08-10 12:37 iattribute_$termfrequencies$.bin
-rw-r--r-- 1 t1ladm sapsys 196000 2007-08-10 12:45 attribute_key_zjb_07_11p.bin
-rw-r--r-- 1 t1ladm sapsys 390756 2007-08-10 12:45 attribute_key_zjb_07_11t.bin
-rw-r--r-- 1 t1ladm sapsys 62 2007-08-10 12:45 attribute_key_zjb_07_11u.bin
-rw-r--r-- 1 t1ladm sapsys 310897 2007-08-10 12:45 attribute_$b49$s_ord_items.bin
-rw-r--r-- 1 t1ladm sapsys 306584 2007-08-10 12:45 attribute_$b49$s_invcd_qty.bin
-rw-r--r-- 1 t1ladm sapsys 52287 2007-08-10 12:45 attribute_$b49$s_rtns_item.bin
-rw-r--r-- 1 t1ladm sapsys 536801 2007-08-10 12:45 attribute_$b49$s_invcd_val.bin
-rw-r--r-- 1 t1ladm sapsys 84121 2007-08-10 12:45 attribute_$b49$s_opordvals.bin

```

```

-rw-r--r-- 1 t1ladm sapsys 547449 2007-08-10 12:45 attribute_$b49$s_invcd_cst.bin
-rw-r--r-- 1 t1ladm sapsys 64610 2007-08-10 12:45 attribute_$bic$number01.bin
-rw-r--r-- 1 t1ladm sapsys 53914 2007-08-10 12:45 attribute_$bic$number02.bin
-rw-r--r-- 1 t1ladm sapsys 85588 2007-08-10 12:45 attribute_$bic$number03.bin
-rw-r--r-- 1 t1ladm sapsys 61248 2007-08-10 12:45 attribute_$bic$number04.bin
-rw-r--r-- 1 t1ladm sapsys 64187 2007-08-10 12:45 attribute_$b49$s_crmmem_qty.bin
-rw-r--r-- 1 t1ladm sapsys 72452 2007-08-10 12:45 attribute_$bic$number10.bin
-rw-r--r-- 1 t1ladm sapsys 104971 2007-08-10 12:45 attribute_$bic$number05.bin
-rw-r--r-- 1 t1ladm sapsys 120335 2007-08-10 12:45 attribute_$bic$number11.bin
-rw-r--r-- 1 t1ladm sapsys 104392 2007-08-10 12:45 attribute_$bic$number06.bin
-rw-r--r-- 1 t1ladm sapsys 34538 2007-08-10 12:45 attribute_$bic$amount01.bin
-rw-r--r-- 1 t1ladm sapsys 112041 2007-08-10 12:45 attribute_$bic$number12.bin
-rw-r--r-- 1 t1ladm sapsys 89547 2007-08-10 12:45 attribute_$bic$number07.bin
-rw-r--r-- 1 t1ladm sapsys 81451 2007-08-10 12:45 attribute_$bic$amount02.bin
-rw-r--r-- 1 t1ladm sapsys 85326 2007-08-10 12:45 attribute_$bic$number13.bin
-rw-r--r-- 1 t1ladm sapsys 95457 2007-08-10 12:45 attribute_$bic$number08.bin
-rw-r--r-- 1 t1ladm sapsys 75548 2007-08-10 12:45 attribute_$bic$amount03.bin
-rw-r--r-- 1 t1ladm sapsys 117510 2007-08-10 12:45 attribute_$bic$number14.bin
-rw-r--r-- 1 t1ladm sapsys 108196 2007-08-10 12:45 attribute_$bic$number09.bin
-rw-r--r-- 1 t1ladm sapsys 91261 2007-08-10 12:45 attribute_$bic$amount04.bin
-rw-r--r-- 1 t1ladm sapsys 111757 2007-08-10 12:45 attribute_$bic$number15.bin
-rw-r--r-- 1 t1ladm sapsys 85718 2007-08-10 12:45 attribute_$bic$amount10.bin
-rw-r--r-- 1 t1ladm sapsys 57490 2007-08-10 12:45 attribute_$bic$amount05.bin
-rw-r--r-- 1 t1ladm sapsys 111879 2007-08-10 12:45 attribute_$bic$amount11.bin
-rw-r--r-- 1 t1ladm sapsys 84593 2007-08-10 12:45 attribute_$bic$amount06.bin
-rw-r--r-- 1 t1ladm sapsys 78207 2007-08-10 12:45 attribute_$bic$amount12.bin
-rw-r--r-- 1 t1ladm sapsys 60472 2007-08-10 12:45 attribute_$bic$amount07.bin
-rw-r--r-- 1 t1ladm sapsys 102033 2007-08-10 12:45 attribute_$bic$amount13.bin
-rw-r--r-- 1 t1ladm sapsys 64900 2007-08-10 12:45 attribute_$bic$amount08.bin
-rw-r--r-- 1 t1ladm sapsys 126182 2007-08-10 12:45 attribute_$bic$amount14.bin
-rw-r--r-- 1 t1ladm sapsys 101342 2007-08-10 12:45 attribute_$bic$amount09.bin
-rw-r--r-- 1 t1ladm sapsys 114999 2007-08-10 12:45 attribute_$bic$amount15.bin
-rw-r--r-- 1 t1ladm sapsys 78057 2007-08-10 12:45 attribute_$b49$s_crmmem_val.bin
-rw-r--r-- 1 t1ladm sapsys 52276 2007-08-10 12:45 attribute_$b49$s_crmmem_cst.bin
-rw-r--r-- 1 t1ladm sapsys 1517722 2007-08-10 12:45 attribute_key_zjb_07_111.bin
-rw-r--r-- 1 t1ladm sapsys 1517776 2007-08-10 12:45 attribute_key_zjb_07_112.bin
-rw-r--r-- 1 t1ladm sapsys 1483329 2007-08-10 12:45 attribute_key_zjb_07_113.bin
-rw-r--r-- 1 t1ladm sapsys 547149 2007-08-10 12:45 attribute_key_zjb_07_114.bin
-rw-r--r-- 1 t1ladm sapsys 547149 2007-08-10 12:45 attribute_key_zjb_07_115.bin
-rw-r--r-- 1 t1ladm sapsys 70536 2007-08-10 12:45 attribute_$b49$s_opordqtyb.bin
-rw-r--r-- 1 t1ladm sapsys 32770 2007-08-10 12:45 attribute_$bic$counter01.bin
-rw-r--r-- 1 t1ladm sapsys 10218 2007-08-10 12:45 attribute_$bic$counter02.bin
-rw-r--r-- 1 t1ladm sapsys 56330 2007-08-10 12:45 attribute_$bic$counter03.bin
-rw-r--r-- 1 t1ladm sapsys 12266 2007-08-10 12:45 attribute_$bic$counter04.bin
-rw-r--r-- 1 t1ladm sapsys 29442 2007-08-10 12:45 attribute_$bic$counter10.bin

```

```
-rw-r--r-- 1 t1ladm sapsys 36098 2007-08-10 12:45 attribute_$bic$counter05.bin
-rw-r--r-- 1 t1ladm sapsys 64187 2007-08-10 12:45 attribute_$b49$s_rtnsqty.bin
-rw-r--r-- 1 t1ladm sapsys 312464 2007-08-10 12:45 attribute_$b49$s_incordqty.bin
-rw-r--r-- 1 t1ladm sapsys 24322 2007-08-10 12:45 attribute_$bic$counter11.bin
-rw-r--r-- 1 t1ladm sapsys 56610 2007-08-10 12:45 attribute_$bic$counter06.bin
-rw-r--r-- 1 t1ladm sapsys 58074 2007-08-10 12:45 attribute_$bic$counter12.bin
-rw-r--r-- 1 t1ladm sapsys 6618 2007-08-10 12:45 attribute_$bic$counter07.bin
-rw-r--r-- 1 t1ladm sapsys 13290 2007-08-10 12:45 attribute_$bic$counter13.bin
-rw-r--r-- 1 t1ladm sapsys 52746 2007-08-10 12:45 attribute_$bic$counter08.bin
-rw-r--r-- 1 t1ladm sapsys 15858 2007-08-10 12:45 attribute_$bic$counter14.bin
-rw-r--r-- 1 t1ladm sapsys 57114 2007-08-10 12:45 attribute_$bic$counter09.bin
-rw-r--r-- 1 t1ladm sapsys 4818 2007-08-10 12:45 attribute_$bic$counter15.bin
-rw-r--r-- 1 t1ladm sapsys 56035 2007-08-10 12:45 attribute_$bic$quan01.bin
-rw-r--r-- 1 t1ladm sapsys 33886 2007-08-10 12:45 attribute_$bic$quan02.bin
-rw-r--r-- 1 t1ladm sapsys 67441 2007-08-10 12:45 attribute_$bic$quan03.bin
-rw-r--r-- 1 t1ladm sapsys 57880 2007-08-10 12:45 attribute_$bic$quan04.bin
-rw-r--r-- 1 t1ladm sapsys 81661 2007-08-10 12:45 attribute_$bic$quan10.bin
-rw-r--r-- 1 t1ladm sapsys 35785 2007-08-10 12:45 attribute_$bic$quan05.bin
-rw-r--r-- 1 t1ladm sapsys 77588 2007-08-10 12:45 attribute_$bic$quan11.bin
-rw-r--r-- 1 t1ladm sapsys 101134 2007-08-10 12:45 attribute_$bic$quan06.bin
-rw-r--r-- 1 t1ladm sapsys 102855 2007-08-10 12:45 attribute_$bic$quan12.bin
-rw-r--r-- 1 t1ladm sapsys 87969 2007-08-10 12:45 attribute_$bic$quan07.bin
-rw-r--r-- 1 t1ladm sapsys 63103 2007-08-10 12:45 attribute_$bic$quan13.bin
-rw-r--r-- 1 t1ladm sapsys 74679 2007-08-10 12:45 attribute_$bic$quan08.bin
-rw-r--r-- 1 t1ladm sapsys 98341 2007-08-10 12:45 attribute_$bic$quan14.bin
-rw-r--r-- 1 t1ladm sapsys 51910 2007-08-10 12:45 attribute_$bic$quan09.bin
-rw-r--r-- 1 t1ladm sapsys 87323 2007-08-10 12:45 attribute_$bic$quan15.bin
-rw-r--r-- 1 t1ladm sapsys 78057 2007-08-10 12:45 attribute_$b49$s_rtnsval.bin
-rw-r--r-- 1 t1ladm sapsys 547449 2007-08-10 12:45 attribute_$b49$s_incordval.bin
-rw-r--r-- 1 t1ladm sapsys 52276 2007-08-10 12:45 attribute_$b49$s_rtncst.bin
-rw-r--r-- 1 t1ladm sapsys 547449 2007-08-10 12:45 attribute_$b49$s_incordcst.bin
-rw-r--r-- 1 t1ladm sapsys 12991 2007-08-10 12:45 AttributeStore.ini
```

t1ladm@jup00

1:/gpfs/fs1/T11/TRX00/index/nap_bic/fzjb_07_11~23/en/attributes>

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 about ordering these publications, see “How to get IBM Redbooks” on page 348. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *DB2 Version 9.1 for z/OS What's New?* GC18-9856-00
<http://publib.boulder.ibm.com/epubs/pdf/dsnwnk10.pdf>
- ▶ *SAP on DB2 for z/OS and OS/390: DB2 System Cloning*, SG24-6287
<http://www.redbooks.ibm.com/abstracts/sg246287.html?open>

Other publications

These publications are also relevant as further information sources. Go to:

<http://www.sap.com>

- ▶ SAP Master Guide SAP NetWeaver 2004s, Document Version: 1.10 - 11/08/2006
- ▶ SAP NetWeaver 2004s SR1 ABAP+Java on AIX: IBM DB2 UDB for z/OS, Version 1.0 - April 10, 2006
- ▶ SAP Planning Guide for SAP NetWeaver for IBM DB2 for z/OS, SAP NetWeaver 2004s SR1, Version 2.0 - 10/16/2006
- ▶ SAP Database Administration Guide for SAP NetWeaver on IBM DB2 UDB for z/OS, SAP NetWeaver 2004s SR1, Version 2.0 - 08/16/2006
- ▶ SAP Security Guide for IBM DB2 UDB for z/OS, SAP NetWeaver 2004s, Document Version 1.00 - October 24, 2005

Online resources

These Web sites are also relevant as further information sources:

- ▶ SAP BI Glossary
<https://websmp109.sap-ag.de/~sapidb/011000358700000106922002E>
- ▶ SAP official information about BI 7.0, news, document enhancements, and so on - especially examine the performance folder
<http://service.sap.com/bi>
- ▶ SAP Checklist, Installation Guides of BI and the BI-CONT
<http://service.sap.com/instguides>
- ▶ SAP search for OSS notes, BI patches, and so on
<http://service.sap.com/notes>
- ▶ SAP Procedure for implementing SAP Notes
<http://service.sap.com/noteassistant>
- ▶ SAP Basis and BW Support Packages, BI-Cont, Frontend, and so on
<http://service.sap.com/patches>
- ▶ SAP R/3 Plug-In for R/3 Enterprise components
<http://service.sap.com/r3-plug-in>
- ▶ Information about supported operating systems and databases
<http://service.sap.com/platforms>
- ▶ Help from SAP (concerning documentation)
<http://help.sap.com>

How to get IBM Redbooks

You can search for, view, or download redbooks, Redpapers, hints and tips, draft publications and additional materials, as well as order hardcopy redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services/

Archived

Help from SAP

SAP Support and downloads

<http://service.sap.com>

SAP Consultancy and other services

<http://www.sap.com/services/index.aspx>

Archived

Index

A

- ABAP Dictionary 115
- access path 4, 163, 174, 211
- Activate the saving of statistical data 158
- Activation table 275
- active ODS
 - data 132
 - table 104, 110
- Active table 275
- administrative activity (AREA) 292
- Administrator Workbench 65, 107, 271, 278, 282
 - Data Access Monitor 278
 - Data Load Monitor 278
 - Meta Data Maintenance 278
 - monitoring function 65
 - reporting agent scheduler 283
 - Scheduler 278
- advanced business application programming (ABAP) 154
- Aggregate 268–269
 - also see InfoCube
 - defined 27
- aggregate roll-ups 162
- aggregate table 162
- Analyzing load performance 142
- API 287
- Application Link
 - Enabling 279
- Application Link Enabling 279
- application server 6, 139, 154, 162
 - especially very large flat file 154
 - IBM unique hipsockets 6
- Audience for this document 3

B

- Background, which (BW) 56
- BAPI 279
- Base recommendations 30
- BEx Analyzer 278, 283
 - also see Business Explorer
- BEx Browser
 - also see Business Explorer
- BEx Portfolio 285

- BEx Query
 - Designer 285
- BEx Query Designer 262
- BEx Web
 - Analyzer 286
 - application 283
 - application wizard Assistant 285
- BEx Web Application
 - Designer 286
 - Wizard 286
- BI Architecture 10
- BI Cockpit 286
- BI data and DB2 Partitioning 14
- BI Java SDK 287
- BI Overview
 - Advanced analytics 2
 - Mobile business intelligence 2
- buffer pool 69
- Business Application
 - Programming Interface 76, 302
- Business Application Programming Interface (BAPI) 279
- Business Content 279
- Business Document Service (BDS) 289
- Business Explorer 259
 - BEx Browser 278
 - Query Designer 278
- Business Information Warehouse 294
- business intelligence (BI) 259, 285
- BW 3 156, 287
- BW administrator 4, 278
- BW data
 - load performance 55
 - target 56
- BW increases (BI) 282
- BW information model 257
- BW LOAD 57
- BW load
 - data statistic 144
- BW system 55–56, 144, 159, 285
 - new data 55
 - regular data loads 56

C

- Candidates for partitioning
 - Data Warehouse and Data Store Objects 14
 - Multi-Dimensional Models 14
 - PSA
 - Persistent staging area (PSA) 14
- Change log table 275
- Change-run 272
- characteristic value 265, 281
 - key figures 286
 - particular combination 265
 - place holder 309
- characteristics 264
- clustering index 108
 - CLUSTER option 112
- Compression 273
- Compression result 273
- Condense
 - defined 27
- Continuous operations 5
- corresponding InfoCube
 - dimension tables 271

D

- Data Access Monitor 278
- Data granularity
 - defined 28
- data load 56, 100, 155
- Data Load Monitor 278
- Data Partitioned Secondary Index (DPSI) 109
- data set 284
- data source 77, 282
- data target 56, 73, 155, 284
- Data targets
 - defined 76
- Data warehouse 221, 257, 275
- data warehousing
 - e-business operations 2
 - robust solution 280
- database management
 - system 5, 287
- dataset 282
- DataStore tables 104
- DataSource 58, 258, 261, 275
- DataStore object types
 - DataStore object for direct update 104
 - Standard ODS 104
 - Write-Optimized DataStore object 104

- DB2 155
- DB2 compression 28, 69, 132, 256
 - defined 28
- DB2 connect diagnostics 139
- DB2 subsystem 114
- DB2 utility 109, 114
 - complete details 114
- DB2 V8 3, 56, 133
 - new support 109
 - recommended method 114
- DBA 99, 166
- DBA Guide 114
- DBA responsibility 4
- dimension table 74, 99, 265–266
- dimension/snowflake 211
- disaster recovery (DR) 7
- distributed relational database architecture (DRDA) 174
- DPSI
 - defined 21
 - Difference between DPSI and NPSI 22
 - Usage of Data Partitioned Secondary Indexes 21
- DSO
 - defined 11

E

- E-Fact table 273
 - partitioning 18
- END 2, 7, 268
- Enhanced Star Join access method 36
- Enterprise Portal 285
 - Business Intelligence content 285
- ETL - see extraction, transformation, and loading
- example sale 281
- Extract, Transform, and Load 260
- Extraction, Transfer and Load (ETL) 66
- extraction, transformation, and loading (ETL) 260

F

- F fact table 100, 273
- fact table 27–28, 74, 88, 96, 104, 162, 211, 224, 256–257, 265, 293
 - compressed data 28
 - compression results 99
 - consolidated data 98
 - initial compression 100
 - many records 28

- same record 98
- Shortens RUNSTATS processing time 96
- Front End
 - Processing Time 164
- Front-end processing time 158
- function module
 - RSDU_ANALYZE_TOOL_DB2 163

G

- generally available (GA) 6
- geographical information system (GIS) 285
- geographically dispersed Parallel Sysplex (GDPS) 7
- given SQL statement
 - access path 174
- Global Work Process Overview
 - SM37 152
 - SM66 152
- Goals of this book 3
- granularity 275

H

- Health checker
 - application server
 - SM21 139
 - ST06 139
 - database connection. 139
 - DRDA network connection
 - DB2 139
 - z/OS database server health 139
- hierarchy 263, 269, 272
- hierarchy node 264, 286
 - corresponding leafs 264
- High granularity
 - defined 28
- Histogram
 - what it does 46
- HTML page 285

I

- IBIOLap interface 287
- IBM zSeries 2
- IDocs - see Intermediate Documents
- index availability 109, 113
- Index compression 147
- Index manipulation 147
- index-controlled partitioning 108

- Index-sequential access 112
- InfoCube 27–28, 77, 96, 158, 211, 256–257, 259, 261–262, 269, 271, 282
 - Aggregate - see Aggregate
 - Characteristics - see characteristics
 - defined 10, 74
- InfoCubes
 - Basic InfoCubes
 - defined 76
 - Real-Time InfoCubes
 - defined 76
- InfoObject 27, 256–257, 261–262
 - classes of metadata 257
 - See information model 257
 - templates 257
- InfoPackage 60, 155, 273
- InfoProvider 27, 256, 262, 285
 - defined 27
- information model 257
 - business definitions 257
 - conceptual layers 257
 - InfoCube - see InfoCube
 - InfoSource - see InfoSource
 - master data - see master data
 - Persistent Staging Area - see Persistent Staging Area
- InfoSource 257, 259, 261, 292
- Intermediate Documents 279
- Internet Engineering Task Force (IETF) 311

J

- J2EE Connector Architecture (JCA) 287
- JCA specification 292
- JDBC driver 287

K

- Key figure 2, 77, 97, 264, 281
- key figures 264

L

- line item dimension 266
- Load Data 148
- local predicate 78, 111, 204, 211, 220, 225
 - master data tables 223
 - Only fields 111
- Low granularity
 - defined 28

M

- master data 155, 211, 258, 277
 - single documents 305
- master data table 267
- maximum number 56, 153
- Meta Object Facility (MOF) 292
- MTF 5
- Multidimensional models 257
- Multiple LPARs 6
- MultiProvider
 - defined 27
- multiprovider 27, 256, 262

N

- navigation attribute 269, 283
 - modified version 272
- New features and functions of SAP BI 7.x 26

O

- Object Log 119
- Object Management Group (OMG) 292
- object version (OBJEVERS) 298
- ODS - see Operational Data Store
- ODS object 74, 109, 132–133, 219, 258–259, 274
 - multiple layers 274
 - query performance 112
 - relevant fields 274
 - typical queries 132
- ODS query
 - performance 133
- OLAP data
 - source 282
- OLAP Engine 162, 278
- OLAP reporting 286
 - BEx Query Designer 286
- OLE DB
 - interface 282
- OMG homepage 292
 - current CWM specification 292
 - current XMI specifications 313
- Online Analytical Processing (OLAP) 259, 290
- Online Transaction Processing (OLTP) 259
- only, the decisions should (ODS) 143, 204
- Operational Data Store 273
 - ODS object 256, 259, 262, 273
- Operational data store 257
- Operational data store (ODS) 56, 259
- OSS note

- 130696 160
- 633075 170
- 633832 163

P

- Package id 28, 97, 257, 273
 - Compression results 28, 97, 257
- Parallel Sysplex 5
- Parameter/tabrec
 - changing online
 - ST03 153
- Partition elimination
 - definition 88
- partition elimination 88, 110, 224
- partition rotation 69
- partitioned E fact table
 - time restriction 224
- Partitioning
 - why 15
- Partitioning Key 110
 - leading column 113
- Partitioning of PSA Tables 19
- Partitioning the F fact table 18
- Peer-to-peer remote copy (PPRC) 7
- Performance
 - extraction takes the most elapsed time 153
 - Loading from a flat file 154
 - Loading from an R/3 system 154
 - partitioning the F fact table. 18
 - Transfer time takes the most elapsed time 155
 - upload (update) to a PSA takes the most elapsed time 155
- performance problem 29, 136, 141–142, 153, 174
 - reliable indicator 174
- Persistent Staging Area (PSA) 56, 258, 261, 298
- Physical data stores 76
- planning level 298
 - complete data selection 298
- planning package 298
- pre-summarized data 162
- Principles of processing dynamic index ANDING for pre-fact table access 38
- process chain 55, 138, 143, 284
 - RSPC graphical view 156
- Processing RUNSTATS for BI tables 40
- PSA
 - defined 10
- PSA - see Persistent Staging Area

PSA record 267
 attribute values 267
 key figures 268
PSA Table
 Name 68, 71
 partition 155

Q

Query Designer 278
Query performance
 indicators
 RSDDSTAT 4
query performance 3, 88, 96, 109, 132, 157, 219,
266
 degradation 113
 main component 166
 significant benefits 88
query time 99
query view 283

R

R/3 system 143, 260
 general performance 154
 memory resources 154
 slow network 154
Range partitioning 78
Range partitioning for E-Fact tables 17
Range partitioning of Fact tables 17
Realignment
 defined 28
Redbooks focus
 decisions concerning SAP on System z 3
 performance problem 3
Redbooks Web site 348
 Contact us xv
Remote Function Call (RFC) 279
RemoteCube 262, 302
response time 139, 151, 158
 detailed breakdown 162
 major component 160
 major components 160
 QDBTIME component 162
road map 137
Roadmap for analyzing performance issues 141
Roll-up
 defined 27
roll-up 271
RS_BW_POST_MIGRATION 20

L_PSAVER (PSA version) 20
RSRT transaction 166
RTF33363533353a204361707469 242
RTF35333235393a204361707469 232
RTF39393332333a204361707469 230
rule of thumb (ROT) 162

S

sample JCL 114
SAP BI Product overview 2
 Advanced Analytics 2
 Advanced analytics 2
 Business content 2
 Mobile business intelligence 2
SAP BI snowflake 37
SAP Business Information Warehouse
 Administrator Workbench 278
 business content 279
 Business Explorer 259, 278
 data flow 260
 OLAP engine 278
 Staging Engine 278
SAP compression
 defined 28
SAP system logs
 SM21 138
SAP_BWTOOLPGM_DB2 13
Secondary Index 105, 107, 112, 156, 219
self-contained dataset 74
Service Provider Interface (SPI) 305
SID combination 268
Simple Object Access Protocol (SOAP) 288
SM37 job
 log 156
 source system 56, 153, 259, 292
 business client 58
 data transfer 305
 extract structure 308
 resource problem 154
 work processes 154
SPOF 5
SQL access path (SAP) 29, 173
SQL efficiency 27, 74, 256
 defined 27
SQL statement 154, 174, 205, 212, 278
Stabilizing data warehouse query performance 36
Staging Engine 278
Standard ODS 104

- star schema 265–266
 - also see InfoCube
 - extended star schema 264, 266
- Starting the analysis from ST03 158
- Statistical information
 - STAD 149
- STATUID 169
- surrogate identifier 266
- System workload statistics
 - checking 143
 - RSMO 145
 - ST03 144
- system-wide health checks 135–136

T

- Table access statistics
 - activate for specific transactions 153
 - ST03 153
- Table screen 117
- table space 112
- table-controlled partitioning 108
- Tablespace partitioning
 - defined 28
- TCP/IP 279
- Terminology
 - BI objects and processes 27
- Terminology changes
 - Analysis Authorizations 12
 - Analytic Engine 12
 - BI Query Runtime Statistics 12
 - Data Warehouseing Workbench 11
 - Data Warehousing Workbench 15
 - DateStore Object (DSO) 11
 - DateStore object for direct update 11
 - Extraction Monitor 11
 - Real-time InfoCube 11
 - Virtual Provider 11
- Terminology changes for SAP NetWeaver 2004 11
- time period 97, 153, 158
- Top SAP BI recommendations 29
 - Ensure user selects option to run RUNSTATS after data loads 30
 - Keep current with SAP support packs and DB2 maintenance level 31
 - Partition E and F fact tables 32
 - Reviewing your Data Model with respect to query performance 30
 - Set DB2 ZPARMS 30

- Use DB2 HW compression 32
- Use fast deletion of table partitions 32
- Use the SAP transaction/programs to remove unused data 33
- Utilize SAP compression 31
- Utilize SAP options to improve performance 30
- transaction data 62, 76, 262
- transaction ST03N 144
- transfer rule 143, 261, 308
- Transfer Structure 275

U

- Unified Modeling Language (UML) 292

V

- Value
 - 7 x 24 application availability 5
 - Capacity and scalability 5
 - Large database manageability 6
 - One-stop shopping 4
- Value provided by System z, z/OS, and DB2 4
- VirtualProviders 76
 - RemoteCube 76
 - SAP RemoteCube 77
 - Virtual InfoCube with services 77

W

- Web Application 282
 - Designer 278, 282
 - Server 287
- Web application
 - alert monitor 283
 - table Web item 302
- Web Interface
 - Builder 310
 - Builder Tool 294
- Web item 281
 - Ad-hoc Query Designer 282
 - paging Mechanism 310
- Web item paging 310
- Web template 285
 - consistent sections 310
- WML content 309
 - compressed transfer 311
- World Wide Web
 - Consortium homepage 305



Redbooks

Best Practices for SAP BI using DB2 9 for z/OS

Archived



Best Practices for SAP BI using DB2 9 for z/OS



Redbooks

Benefits of SAP Business Information Warehouse on z/OS

Best practices and troubleshooting

Scalability of the BI Accelerator

Clients want to know how to exploit the new features of both DB2 V9 and SAP in their SAP BI environments. This IBM Redbooks publication describes the benefits of DB2 V9 for SAP Business Information Warehouse. It covers best practices and provides performance and tuning recommendations.

SAP Business Information Warehouse is the central reporting tool for almost all SAP business solutions. It is based on building blocks called InfoObjects that contain data about customers, sales, and business information. InfoObjects include InfoSources, DataSore objects, and InfoCubes. The business intelligence solution from IBM and SAP can help you aggregate and leverage this information, giving you a system-wide view of your business data and delivering it across your enterprise to support sound business decisions. This book describes best practices for this product on a zSeries platform, and provides performance and tuning recommendations for loading and querying data. It also addresses general system administration and troubleshooting.

The audience for this book includes SAP and DB2 administrators. Knowledge of these products and of the z/OS environment is assumed.

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