International Technical Support Organization

Best Practices for SAP Business Information Warehouse on DB2 for z/OS V8

May 2005
First Edition (May 2005)

This edition applies to SAP NetWeaver 04 SR1, which includes Version 3.5 of SAP Business Information Warehouse and Version 3.53 of BI Content, for Version 8.1 of IBM DB2 UDB for OS/390 and z/OS

© Copyright International Business Machines Corporation 2005. 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
- Trademarks .......................................................... viii

## Preface
- The team that wrote this redbook ................................... ix
- Become a published author ........................................... xi
- Comments welcome ................................................... xii

## Chapter 1. Introduction
- 1.1 Product overview .................................................. 2
- 1.2 The audience for this redbook ................................... 2
- 1.3 Why run SAP BW on IBM zSeries ................................ 3
  - 1.3.1 Value provided by z/Series, z/OS, and DB2 .................. 3

## Chapter 2. Top ten SAP BW recommendations
- 2.1 Base recommendations - SAP BW implementations on DB2 z/OS ... 8

## Chapter 3. Best practices for PSA
- 3.1 Load PSA ............................................................ 12
  - 3.1.1 Load PSA ....................................................... 13
  - 3.1.2 Management of PSA .......................................... 20
  - 3.1.3 Load PSA DB2 compression .................................. 21
  - 3.1.4 PSA partitioning .............................................. 21

## Chapter 4. Best practices for InfoCubes
- 4.1 Types of InfoCubes .................................................. 24
  - 4.1.1 Physical data stores (data targets) ......................... 24
  - 4.1.2 Virtual data stores .......................................... 24
- 4.2 Partitioned E fact table .......................................... 25
  - 4.2.1 Advantages .................................................... 25
  - 4.2.2 Partitioning the E fact table ................................ 26
- 4.3 Parallel loading into InfoCube .................................... 29
- 4.4 Drop indexes (or not) ............................................. 31
- 4.5 Aggregates .......................................................... 33
- 4.6 SAP compression for loads ........................................ 41
  - 4.6.1 Overview ........................................................ 41
  - 4.6.2 Advantages .................................................... 41
  - 4.6.3 Data validation ................................................ 42
  - 4.6.4 Cumulative data versus non-cumulative data ............... 42
  - 4.6.5 Areas of impact ............................................... 43
  - 4.6.6 Implementing SAP compression ............................... 44
- 4.7 DB2 compression ................................................... 52

## Chapter 5. Best practices for Operational Data Store
- 5.1 Types of ODS ....................................................... 54
- 5.2 Secondary indexes on ODS objects ................................. 54
  - 5.2.1 Creating a secondary index on an ODS table ............... 55
- 5.3 Partitioning and clustering ........................................ 58
  - 5.3.1 Choosing the partitioning key ................................ 59
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3.2 Choosing and changing the clustering index</td>
<td>61</td>
</tr>
<tr>
<td>5.3.3 Data Partitioned Secondary Indexes (DPSIs)</td>
<td>61</td>
</tr>
<tr>
<td>5.3.4 Partitioning an ODS</td>
<td>62</td>
</tr>
<tr>
<td>5.4 DB2 compression</td>
<td>78</td>
</tr>
<tr>
<td>5.5 Materialized query tables</td>
<td>78</td>
</tr>
<tr>
<td>Chapter 6. Tips for SQL efficiency</td>
<td>79</td>
</tr>
<tr>
<td>6.1 SQL efficiency for ODS</td>
<td>80</td>
</tr>
<tr>
<td>6.2 SQL efficiency for InfoCube</td>
<td>84</td>
</tr>
<tr>
<td>6.3 Indexing of ODSs</td>
<td>90</td>
</tr>
<tr>
<td>6.3.1 Index design for post-ODS access</td>
<td>93</td>
</tr>
<tr>
<td>6.4 Indexing of InfoCubes</td>
<td>93</td>
</tr>
<tr>
<td>6.4.1 Fact table indexing</td>
<td>94</td>
</tr>
<tr>
<td>6.4.2 Index design for dimension/master data tables</td>
<td>94</td>
</tr>
<tr>
<td>Chapter 7. Troubleshooting poor performance</td>
<td>97</td>
</tr>
<tr>
<td>7.1 Troubleshooting general system performance</td>
<td>99</td>
</tr>
<tr>
<td>7.2 Analyzing load performance</td>
<td>101</td>
</tr>
<tr>
<td>7.3 Analyzing query performance</td>
<td>114</td>
</tr>
<tr>
<td>7.4 Starting the analysis from ST03N</td>
<td>114</td>
</tr>
<tr>
<td>7.5 Determining whether aggregates are required</td>
<td>116</td>
</tr>
<tr>
<td>7.5.1 QDBSEL/QDBTRANS</td>
<td>116</td>
</tr>
<tr>
<td>7.5.2 RSDU_ANALYZE_TOOL_DB2</td>
<td>117</td>
</tr>
<tr>
<td>7.5.3 The next steps</td>
<td>117</td>
</tr>
<tr>
<td>7.6 Front-end processing time</td>
<td>118</td>
</tr>
<tr>
<td>7.6.1 Further analysis</td>
<td>118</td>
</tr>
<tr>
<td>7.7 Database performance</td>
<td>119</td>
</tr>
<tr>
<td>7.7.1 RSRT Query Monitor</td>
<td>120</td>
</tr>
<tr>
<td>7.7.2 ST04 - database administration</td>
<td>127</td>
</tr>
<tr>
<td>7.8 DB2 Visual Explain</td>
<td>128</td>
</tr>
<tr>
<td>7.8.1 Stored procedures used by Visual Explain</td>
<td>128</td>
</tr>
<tr>
<td>7.9 Installation requirements</td>
<td>131</td>
</tr>
<tr>
<td>Chapter 8. SAP BW installation and checks under z/OS</td>
<td>135</td>
</tr>
<tr>
<td>8.1 Checks concerning SAP BW installation</td>
<td>136</td>
</tr>
<tr>
<td>8.2 SAP BW operations check</td>
<td>136</td>
</tr>
<tr>
<td>8.2.1 SAP BW administration roles</td>
<td>137</td>
</tr>
<tr>
<td>8.2.2 Data access design roles</td>
<td>137</td>
</tr>
<tr>
<td>8.2.3 Overview of typical SAP BW operations</td>
<td>137</td>
</tr>
<tr>
<td>8.3 SAP BW regular checks</td>
<td>138</td>
</tr>
<tr>
<td>8.4 Proactive performance measurement</td>
<td>139</td>
</tr>
<tr>
<td>8.5 SAP automated PTF checker</td>
<td>140</td>
</tr>
<tr>
<td>8.6 Workload Manager</td>
<td>143</td>
</tr>
<tr>
<td>8.6.1 WLM changes</td>
<td>143</td>
</tr>
<tr>
<td>8.6.2 Stored procedures changes</td>
<td>143</td>
</tr>
<tr>
<td>Chapter 9. DB2</td>
<td>145</td>
</tr>
<tr>
<td>9.1 General z/OS settings and ZPARMS</td>
<td>146</td>
</tr>
<tr>
<td>9.1.1 DB2 and ZPARMS settings</td>
<td>146</td>
</tr>
<tr>
<td>9.2 Buffer pool management</td>
<td>150</td>
</tr>
<tr>
<td>9.2.1 Changes</td>
<td>151</td>
</tr>
<tr>
<td>9.3 RUNSTATS</td>
<td>153</td>
</tr>
<tr>
<td>9.3.1 Highlights of RUNSTATS improvements</td>
<td>153</td>
</tr>
<tr>
<td>9.3.2 Collecting cardinality and distribution statistics</td>
<td>154</td>
</tr>
</tbody>
</table>
Chapter 11. Benefits of DB2 V8 and the zSeries platform

11.1 IBM DB2 V8 benefits for SAP .............................................. 190
11.2 The benefits of DB2 V8 for SAP BW ................................. 191
   11.2.1 Star join enhancements ........................................... 192
   11.2.2 In-memory workfiles ............................................. 195
   11.2.3 Sparse index during inside-out join phase ................... 195
   11.2.4 Snowflake handling: Controlled materialization .......... 196
11.3 Joins with up to 225 tables ............................................. 197
11.4 Common table expressions .............................................. 197
   11.4.1 Recursive SQL ..................................................... 199
11.5 Materialized query tables .............................................. 200
   11.5.1 Creating MQTs ................................................... 200
   11.5.2 Populating and maintaining an MQT ......................... 202
   11.5.3 Automatic query rewrite using MQTs ....................... 203
   11.5.4 Determining if query rewrite occurred ..................... 205
11.6 Partitioning ............................................................... 205
   11.6.1 Increases to table sizes ....................................... 206
   11.6.2 Adding new partitions and rotating partitions ......... 206
   11.6.3 Data partitioned secondary indexes ......................... 207
   11.6.4 Separating clustering from partitioning ................. 207
11.7 Deferred indexes do not prevent table access .................. 208
11.8 Multi-row operations .................................................... 209
   11.8.1 How SAP benefits from multi-row operations .......... 209
   11.8.2 Implementation details ........................................ 209
11.9 RUNSTATS enhancements ............................................... 210
11.10 2 GB bar ............................................................... 210
11.11 Automatic Space Management ....................................... 211
11.12 Flexible implementation with zSeries ......................... 215

Chapter 10. Concepts, activities, and terminology

10.1 Terminology: BW objects and processes ............................ 170
10.2 SAP BW information model ............................................ 171
   10.2.1 Other key elements in the information model .......... 172
10.3 Dataflow in SAP BW ..................................................... 173
10.4 Information access ...................................................... 174
10.5 Hierarchies ................................................................. 175
10.6 Extended star schema (InfoCubes) .................................. 176
10.7 Data load into InfoCube ................................................ 177
10.8 Aggregates ............................................................... 178
   10.8.1 Aggregate example .............................................. 179
   10.8.2 Maintaining aggregates ....................................... 180
10.9 Compression of requests ................................................. 182
10.10 Operational data store (ODS) ......................................... 182
10.11 SAP BW naming conventions ......................................... 184
10.12 The SAP BW functional components ............................... 185
10.13 SAP BW Business Content ............................................ 186

9.1.2  Flexible implementation with zSeries .......................... 215
9.1.1 Automatic Space Management .................................... 211
9.2  2 GB bar .................................................................... 210
9.3  Runstats enhancements ................................................ 209
9.4  Compression of requests ............................................... 209
9.5  Partition rotation and benefits for archiving .................... 207
9.6  Partition addition ........................................................ 207
9.7  Partition rotation and benefits for archiving .................... 207
9.8  Partitioning ............................................................... 205
9.9  SSD benefits for DB2 V8 ............................................... 204
9.10  Implicit use of partitioning .......................................... 205
9.11  How SAP benefits from multi-row operations .................. 203
9.12  Implementation details ................................................ 203
9.13  Flexible implementation with zSeries ............................ 202

Chapter 9. SAP BW data management

9.1 SAP BW naming conventions ........................................... 161
Notices

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

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

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

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

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

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

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

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

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

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

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

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

AIX®
DB2 Connect™
DB2 Universal Database™
DB2®
DFSMShsm™
DFS™
Distributed Relational Database Architecture™
Domino®
DRDA®
@server®

ETE™
GDPS®
HiperSockets™
ibm.com®
IBM®
MVS™
Notes®
Parallel Sysplex®
Perform™
RACF®

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

The following terms are trademarks of other companies:

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

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

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

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

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

This IBM Redbook describes the benefits of DB2® V8 for SAP Business Information Warehouse. It lists best practices and provides performance and tuning recommendations for SAP Business Information Warehouse.

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 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, ODS 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 IBM Redbook 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 redbook includes SAP and DB2 administrators. Knowledge of these products and of the z/OS® environment is assumed.

This redbook 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, ODS, and DB2 components. We give tips on how to gain efficiency when using compression and indexing. We also include a troubleshooting chapter that describes what to do when best practices do not address a problem.

The team that wrote this redbook

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

Mike Ebbers is a Consulting Certified IT Specialist at the International Technical Support Organization (ITSO), Poughkeepsie center. He has worked with IBM for 31 years and leads zSeries projects in ISV and Domino® solutions.

Brenda Beane is an Advisory Software Engineer at IBM in Poughkeepsie, NY. She has 22 years of experience in development and test at IBM, and her area of expertise is DB2. For the last nine years, she has been involved in testing the performance of SAP on zSeries. Brenda holds a Bachelor of Science degree in Computer Science and Mathematics from the University of Massachusetts and a Master’s degree in Computer Science from Marist College.

David Bellion is Technical Director of Oserve, an IBM Business Partner providing technical support for SAP on z/OS and OS/390®-DB2. He has 24 years of IT experience in the IBM environment, moving from mainframe systems programming in 1992, to work for the past 13
years as an SAP Basis consultant and technical architect. He is an SAP-certified technical consultant for SAP on the OS/390-DB2 platform. David has worked on many SAP z/OS OS/390-DB2 implementations at customer sites throughout Europe, including several years with IBM UK, supporting their global SAP implementation on OS/390-DB2. He also teaches several aspects of this subject and has co-authored a previous SAP IBM Redbook. He holds a B.A. Honors degree in History from Portsmouth University.

**Nick Dyke** is an SAP Performance Specialist in the UK. He has worked on large scale implementations with SAP on DB2 for z/OS, and on implementations on DB2 for AIX® and Oracle. He has nine years of experience in the SAP Basis field, covering all areas of the application development lifecycle, and has specialized in performance for the past three years. Prior to working on SAP, Nick worked in Application Development and Support. He holds an Honours degree in Computer Science and Accounting from UCW Aberystwyth.

**Andrew Hartman** is a DB2 Systems Programmer working for Standard Bank of South Africa. He has been involved in information technology for 24 years, specializing in DB2 for z/OS. He has also worked with SAP R/3, BW and SEM for the past several years. He co-authored the IBM Redbook *SAP on DB2 for z/OS and OS/390: DB2 System Cloning*, SG24-6287.

**Veng Ly** is an Advisory Software Engineer working at the IBM Poughkeepsie Development Lab, New York, USA. He has 15 years of experience in hardware and software development on the zSeries platform, and has worked for the last seven years on SAP (R/3) solution and performance testing for zSeries. He holds a Bachelor’s degree in Electrical Engineering from City College of New York, and a Master’s degree in Computing Management and Telecommunication from Polytechnic University.

**Mike Mardis** is the IBM Technical Advocate to SAP Consulting in the Americas. He has 30 years of experience in information technology, and his areas of expertise include SAP Technical Infrastructure. Mike holds a Master’s degree in Computer Information Systems from the University of Boston, and a Bachelor’s degree in Computer Science from the University of Texas, Austin. His areas of expertise include SAP Technical Infrastructure.

**Bala Prabahar** is an Infrastructure Specialist with a leading corporation in the beverage industry and is currently working in support of SAP on DB2 for z/OS. He has 17 years of experience in application programming, database administration, logical data modeling for both OLTP and data warehouse applications, shell scripting, performance tuning, physical data modeling and project management. Bala holds a BS degree in Engineering from the University of Madras, India.

**Michael Thelosen** is an SAP Technical Consultant for R3 on zSeries with SAP Germany. He studied Computer Science at the University of Dortmund, with a focus on databases and operating systems. Michael has 25 years of experience on IBM mainframe systems with SAP and non-SAP applications, and works on performance analysis.

**Theodor H. Kuebler** is a consultant and trainer for SAP and Data Management products. He is founder of Logos InformationsSysteme (LIS) AG in Switzerland, an IBM enabled Education Center for IBM Software (ECIS) and IBM Business Partner, which provides classroom and onsite courses in Data Management, Tivoli®, and WebSphere®. He has 12 years of experience in the implementation and tuning of SAP products on database management systems, such as DB2 UDB, Adabas-D, and Oracle on different platforms. Theodor teaches SAP Basis courses on z/OS and DB2 worldwide.

Special thanks to our consulting specialists for their assistance on this project:

**Mark Gordon** is a Senior Consulting I/T Specialist in the US. He has 10 years of experience with SAP software on IBM systems. He has written several papers on SAP-related topics.
Terry Purcell is a member of DB2 Optimizer Development for IBM Silicon Valley Lab. He has 13 years of experience with DB2, the last several with IBM. He holds a degree in Information Technology from Swinburne University, Australia.

Joachim Rese is a software engineer at IBM Boeblingen, Germany. His areas of expertise include database technology, data warehousing and SAP Netweaver platform. He is a member of the joint SAP/IBM on DB2 for z/OS platform team, and leads the SAP BW porting group. He holds Master's degrees in Computer Science and Mathematics.

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

Lydia Parziale is a Project Leader for the ITSO team in Poughkeepsie, NY with expertise in e-business development and database management technologies. She currently leads zSeries projects in ISV solutions.

Michael Zazulak
IBM Solutions Technical Sales

Rich Conway
Bob Haimowitz
International Technical Support Organization, Poughkeepsie Center

Figure 1 From left: Brenda Beane, Andrew Hartman, Veng Ly, Mike Mardis, Michael Thelosen, Bala Prabahar, Theodor Kuebler; David Bellion is not shown.

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'll team with IBM technical professionals, Business Partners and/or customers.
Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

► Use the online Contact us review redbook form found at:
  ibm.com/redbooks

► Send your comments in an e-mail to:
  redbook@us.ibm.com

► Mail your comments to:
  IBM® Corporation, International Technical Support Organization
  Dept. HYJF 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 some knowledge about this product. However, if you would like to read more about its concepts and terminology, see Chapter 10, “Concepts, activities, and terminology” on page 169.
1.1 Product overview

The business warehouse solution from IBM and SAP can help your company collect, analyze, and distribute critical data. mySAP 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 zSeries; see “Related publications” on page 259. The following subjects have been covered:

- Reliability, availability, and serviceability of SAP systems on IBM zSeries.
- Security and integrity.
- Scalability.
- Continuing compatibility.
- Changing workload.

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

Furthermore, the SAP BW 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 BW with DB2 V8 on zSeries, as this redbook does.

1.2 The audience for this redbook

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

The audience for this document includes both zSeries DB2 DBAs and SAP BW Basis Administrators responsible for administering SAP BW on zSeries DB2 production systems, and SAP BW functional architects responsible for designing the BW system's InfoCubes, Aggregates, and ODS objects.

**Note**: A key message is the importance of early cooperation between the functional designers and the technical implementation.
The SAP BW solution is not the only DB2 V8 solution within a company. Most companies have different legacy systems running on zSeries. For the IBM mainframe infrastructure, a support organization exists within the company. Most groups tend to work independently from each other.

With SAP BW on zSeries, 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 redbook 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 zSeries, as discussed in 11.12, “Flexible implementation with zSeries” on page 215. 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 BW 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 BW 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 BW 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 BW administrator may need to turn over some problems to the DB2 DBA.

### 1.3 Why run SAP BW on IBM zSeries

The “z” in zSeries signifies zero downtime. zSeries 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 z/Series, z/OS, and DB2

In this section we give an overview of the value provided by z/Series, 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 zSeries 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 upon demand of zSeries and z/OS.

Capacity and scalability
- Vertical and horizontal
  - 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 zSeries 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 V8 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 zSeries can use IBM unique hipersockets for application server-to-database server communications.
- Sysplex with Coupling Facility.
- IO bandwidth, data in memory.

Large database manageability
- Multi-terabyte databases with hardware compression.
- Online backup and reorganizations.
- Storage technology (flash copy, 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 the average, UNIX and Windows require 50% more disk than DB2 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.
  - MCOD - zSeries 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 (flash copy, 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 (PAV) with z/OS eliminates 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 zSeries/DB2 capabilities that provide total cost of ownership (TCO) and service level advantages:
  - Multiple LPARs and priority schemes.
  - z/OS and DB2 skills and tools.
  - MCOD - zSeries first - reduce complexity.
  - Proven automation and end-to-end systems management.
– Disaster Recovery (DR)-ready using a Geographically Dispersed Parallel Sysplex (GDPS), which is an S390 implementation that enables high availability across sites up to 40km 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).

Notes:
- For more information about product concepts, see Chapter 10, “Concepts, activities, and terminology” on page 169.
- For more information about the benefits of IBM DB2 V8 and zSeries, see Chapter 11, “Benefits of DB2 V8 and the zSeries platform” on page 189.
Top ten SAP BW recommendations

We have worked with many customers who implement SAP BW on zSeries, and we have noticed that there are recognizable patterns apparent when solving their performance problems. As a result, we have created a 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.

We’ve structured this redbook accordingly. The next several chapters cover the best practices necessary to avoid the most common performance problems. Chapter 7, “Troubleshooting poor performance” on page 97, covers troubleshooting activities.
2.1 Base recommendations - SAP BW implementations on DB2 z/OS

Following is a base set of recommendations for SAP BW implementations on DB2 z/OS. When these recommendations are followed, customers have demonstrated greater maintenance and query stability, resulting in better performance of the zSeries SAP BW installation.

1. Set proper DB2 ZPARMS.
   For more detailed information, refer to 9.1, “General z/OS settings and ZPARMS” on page 146.

2. Ensure the user selects the option to run RUNSTATS after data loads.
   For more detailed information, refer to 9.3, “RUNSTATS” on page 153.
   A cost-based optimizer relies on current and correct statistics to ensure optimal SQL access path selection. In an SAP BW environment, statistics do not require re-collection unless the object's data content has altered based upon a load, SAP compression, rollup, or data archive. Therefore, it is only these activities that require updating of DB2 statistics.

   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 RUNSTATS is triggered as part of the load. There are two ways to run RUNSTATS:
   a. To run it manually:
      RSA1 -> 'Manage' infocube -> tab strip 'performance' -> 'Create Statistics (Btch)'
   b. To use process chains to run it:
      If you are already using process chains to load data, then add RUNSTATS NODE to the process chain. For a further optimized RUNSTATS capability with process chains, see SAP Note 778437.

3. Keep current with SAP support packs.
   Each new SAP support pack delivers fixes and performance enhancements based upon prior customer requirements, or identified areas for improvement by joint DB2 and SAP development.

4. Utilize SAP compression (E fact table).
   For more detailed information, refer to 4.6.1, “Overview” on page 41.
   When F fact table packets are consolidated into the E fact table using a process known as SAP 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 down to a moderate size by compressing packets to E can improve overall RUNSTATS performance and cost for the entire infocube.

   SAP compression can reduce the impact of index drop and recreate during infocube data loads. SAP compression is a prerequisite for enabling partitioning on the E fact table.
And finally, the most important reason for SAP compression is for non-cumulative key figures whereby an additional row is inserted or updated in an E fact table that has the same combination of dimensions id but is infinite for time. Rather than requiring you to sum all delta values from the F fact table, the E fact table reference point can be accessed directly, thus improving query performance.

5. Partition the E fact table under SAP.

For more detailed information, refer to 4.2, “Partitioned E fact table” on page 25.

The E fact table (result of SAP compression) can be partitioned via SAP, which provides a 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.

An additional benefit for 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, it is recommended to partition the E fact table and use SAP compression.

6. Use DB2 compression.

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

a. Reduced disk space.
b. Improved elapsed time for sequential processing (for example, queries, copy utility).
c. Improved bufferpool hit ratio (more rows per page).
d. 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 the strong compression capabilities of zSeries.

7. Keep current with the SAP certified DB2 maintenance level.

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, it is recommended that you stay current with the SAP certified maintenance level and apply the additional PTFs identified in SAP notes 390016 and 81737. Refer to 8.5, “SAP automated PTF checker” on page 140 for more detailed information.

8. Utilize aggregates.

DB2 z/OS has a heritage founded on operational applications where high performance data updates can be as much a priority as data retrieval performance. Also, DB2 is not hindered by the update challenges of warehouse-specific index designs.

Both issues point to DB2 providing very strong aggregate build performance, which is beneficial given the SAP recommendation that this is the main tool for improving query performance in an SAP BW environment.
It is common for SAP customers to use the off-peak window, which is the quietest period that should be exploited fully to build as many aggregates as necessary to provide the required query response times for users. Customers with a CPU chargeback system may also benefit from pushing this work to off-peak hours, and thus off-peak CPU rates.

And for customers supporting multiple time zones, the sophisticated IRD/WLM capabilities can be utilized to ensure that the off-hours aggregate builds for one time zone do not impact the online workload for another time zone.

9. 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.

It is recommended that you purge unwanted data using the following methods:

- Trancode RSRV for dimension data
- Program to delete master data is RSDMD_DEL_BACKGROUND or functional Module RSDMD_DEL_MASTER_DATA.
- Program RSCDS_DEL_OLD_REQUESTS to remove obsolete rows from aggregate ‘F’ tables (SAP note 609164).

10. Use a multi-provider approach carefully.

If the multiprovider is made up of infocubes with the same data model, but the cubes cannot use structure-specific infoobject properties (that is, the cubes are made up of disjoint sets of data, where the partitioning characteristic has a range of values in each cube), then BW must query each cube in order to retrieve the result.

However, a homogenous multi-provider is useful if the underlying cubes have a constant value defined for a partitioning characteristic and queries restrict on that characteristic—as a result, only cubes that contain relevant data are accessed. Refer to SAP note 629541 for more detailed information.

Therefore, it is recommended that you examine the multi-provider approach for your installation to ensure the optimizer can adequately determine which cubes to access. The result will be improved query performance.
Best practices for PSA

One of the main areas of responsibility for the SAP BW system administrator is the loading of new data into the BW system. This is normally a scheduled overnight process that is managed by one or more process chains. The data must usually be available for reporting in the BW 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 BW data load performance for PSA on DB2 for z/OS.
3.1 Load PSA

In this section, we discuss the best practices for BW on z/OS DB2 V8 for loading data into the PSA as part of the regular data loads into the BW system. We discuss techniques that can be employed to ensure that the PSA is managed efficiently. We also discuss techniques which will optimize data loading into the PSA as part of the data load into the BW data targets.

The source systems store logically dependent data in the form of DataSource. Using the DataSource, the data is extracted and transferred to the BW System. In an SAP BW system, the PSA is used as the input storage, which stores the data in unchanged form. That means 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 BW from the follow-on processes like data analysis and queries. Based on this separation, the loading performance can be tuned independently from other BW activities.

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

The Data Update BW provides the following processing options:

1. Only PSA
   Data is just written to the PSA and is not transferred or updated to additional data targets (for example, InfoCube, ODS). Defining the maximum number of processes in the source system influences the target system processes as well and improves the load performance.

2. 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.

3. 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 3-1 on page 13 illustrates the Data Update types.
Independent of the DB2 functions and tuning options, the Data Update types impact the BW load performance and runtime.

**Tip:** 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)

### 3.1.1 Load PSA

The dataflow and the provided interfaces for loading data into an SAP BW system are described in 10.3, "Dataflow in SAP BW" on page 173.

To load data into the BW 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 as well.

Figure 3-2 on page 14 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.
Figure 3-2 Create a source system

Figure 3-3 shows the available options.

After the source system, the DataSource and the InfoPackage (see Figure 3-4 on page 15) are defined. The Infopackage may be scheduled.

Scheduling the InfoPackage is started by opening the context menu using the right mouse key on the ODS Infopackage, as depicted in Figure 3-5 on page 15.
Chapter 3. Best practices for PSA

Figure 3-4  Create InfoPackage

Figure 3-5  The location of external data
Note the available tabs (registers) in Figure 3-6 on page 16, such as:

- Data Selection (defines loading transaction data from the source system)
- External Data (defines the flat file location; see Figure 3-5 on page 15)
- The Processing options as shown in Figure 3-7 and described in “Processing options” on page 16
- The associated Data Targets such as ODS and Cubes
- The Update Mode such as full or delta update
- The Scheduling options

During the first test case, we defined the load into ODS using the Only PSA option. Because no target object is defined, the BW system visualizes this case using the processing activities as shown in Figure 3-7. First the PSA will be loaded. In a second step the data target, such as ODS or Infocub, can be populated.

Processing options

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

1. 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 BW system.
2. Start in Background, which starts a job assigned to a background (Batch) work process.
Figure 3-8 on page 17 shows the provided options.

![Scheduler (Maintain InfoPackage)](image)

**Figure 3-8  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 3-9.
The TAB header provides information about the defined and used BW objects. Additionally, it gives an overview about the runtime parameters. Selecting the Details TAB, the single steps and the associated time periods are provided.

Figure 3-10 shows the following:

- The phases of the load run, such as Extraction, Transfer and Load (ETL)
- The Data Package size
- The number of record, loaded into the PSA
- The time used for the single step
The package size depends on the BW customizing values in the rscustv6 transaction, as shown in Figure 3-11 on page 19.

The PSA Table Name is created automatically and is entered into the RSTSODS table. The values in Figure 3-12 include the Technical Name, which is /BIC/B00000349000. The Naming convention shows /BIO/B000—or /BIC/B000—or the ODS Tables.
Field names in table RSTSODS are misleading, because it actually contains information about PSA tables rather than ODS 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.

![Table RSTSODS Display](image)

**Figure 3-12  PSA newly created Table Name**

Now the PSA Table is created and loaded and can be used for further processing such as loading into targets like ODS or Infocubes.

### 3.1.2 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 anew load. 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 V8 partition rotation.

![Diagram](image)

Basics, 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/DELETED and the space is now required for reuse. For more information, refer to 8.9.6, “Partition rotation and benefits for archiving” on page 164.

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.
3.1.3 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 BW, the tablespaces for PSA tables are created with the option 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.

3.1.4 PSA partitioning

Partitioning of the PSA table is advisable due to its large size. With DB2 Version 8, the PSA table is always partitioned, so the parameter DB2_PSA_NUM_PARTITIONS in RSADMIN table becomes obsolete. Figure 3-13 shows related PSA parameters.

![Figure 3-13 RSADMIN parameters](image)

The number of partitions is not fixed; new partitions can be added if needed. SAP uses the DB2 V8 feature of adding partitions. Under V7, SAP could add partitions only up to the maximum limit, which was set at the time the PSA table was initially created. However, under DB2 V8, the maximum number of partitions is now 4096.

If all 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):

```sql
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 3-14.

![DB2/390 Storage Attributes: Table](image)

To see if 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 1 more than LIMITKEY in SYSTABLEPART.
Best practices for InfoCubes

This chapter discusses various types of InfoCubes and describes how to query them.

InfoCubes are infoproviders but are not necessarily data targets (they are only data targets if data is physically stored in them). By this definition, Virtual Data Stores (Virtual InfoCubes) are infoproviders and not data targets.

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.

InfoCubes are supplied with data from one or more InfoSources or ODS objects (Basic InfoCube) or with data from a different system.

Optimizing the performance of loading data into the InfoCube object involves examining SQL efficiency and proper indexing.
4.1 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 virtual data stores.

Let’s look at these types in more detail.

4.1.1 Physical data stores (data targets)

- 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. SAP compression is discussed in 4.6, “SAP compression for loads” on page 41.
  We discuss Basic InfoCubes in this redbook.

- Transactional InfoCubes
  Transactional InfoCubes store data physically. These cubes are both readable and writable, whereas Basic InfoCubes are read only. Transactional InfoCubes are used in Strategic Enterprise Management (SEM) only.
  Transactional InfoCubes are beyond the scope of this redbook and are not discussed.

4.1.2 Virtual data stores

- 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 BW. 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.
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 Manager provides services for converting the parameters and data.

4.2 Partitioned E fact table

In this section we discuss the benefits of E fact table partitioning. This feature 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.

4.2.1 Advantages

In conjunction with SAP Compression (see 4.6, “SAP compression for loads” on page 41), partitioning the E fact table provides significant benefits for query performance:

- Partitioning enhances the possibility for queries to execute in parallel, thus reducing overall elapsed time.
- When users specify restriction based upon time, then SAP BW 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 fact table index used or whether a tablespace scan is used on the fact table. This has the benefit of allowing the 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 fact table.
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.

### 4.2.2 Partitioning the E fact table

Partitioning can be defined while creating InfoCubes. Select RSA1 → Create InfoCube. Figure 4-2 on page 26 appears.

![Figure 4-2 Editing the InfoCube](image)

Specify the name of the InfoCube you would like to create and press the Create button at the bottom of the screen. Figure 4-3 on page 27 appears. 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 -> Partitioning**.

Note the following points:

- Partitioning can only be done on infoobjects 0CALMONTH and 0FISCPER.
- The fact table gets an additional column (SID_0CALMONTH or SID_0FISCPER), which it is partitioned on. This is necessary because all key columns contain 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.

Figure 4-5  Full-screen view of selecting the partitioning option under "Extras"
4.3 Parallel loading into InfoCube

If you encounter performance problems while loading data into InfoCubes, we recommend the following actions.

**Source system: file**
1. Load from the application server instead of from the workstation.
2. Implement the most recent BW hot package and the most recent kernel patch in your system.
3. Before you begin loading data, make sure that all related master data has already been loaded into your system. If no master data has been loaded yet, the upload of transaction
data could take much longer as SIDs must be retrieved for the characteristics and new records must be inserted into the master data tables.

4. Keep the number of data IDOCs low. We recommend an IDOC size between 100,000 KB and 150,000 KB.

5. If possible, use a fixed record length when loading data from a file (ASCII file). For a CSV file, conversion to a fixed record length is only carried out by the system during the load process.

6. When loading large data quantities from a file, we recommend that you split the file into several parts. We recommend using as many files of the same size as there are CPUs. These files can then be loaded into the BW system in several requests at the same time.

7. When loading large quantities of data into an InfoCube, the number range buffer should be increased for the dimensions that are likely to have a high number of data sets. To do this, use Function module RSD_CUBE_GET to find the object name of the dimension that is likely to have a high number of data sets.

**Function module settings**

```plaintext
I_INFOCUBE = <INFOCUBE Name>
I_OBJVERS = 'A'
I_BYPASS_BUFFER = 'X'
```

The numbers for the dimensions are then contained in Table ‘E_T_DIME’, Column ‘NUMBRANR’. Specify ‘BID’ before this number, and the desired number range is given (for example, BID0000053).

Using Transaction SNRO (ABAP/4 Workbench → Development → Other tools → Number ranges), you can display all number ranges used in BW by entering BID*. By entering the object name determined beforehand, you can find the desired number range.

By double-clicking this line you get to the number range maintenance. Choose Edit → Set up buffering → Main memory to determine the ‘number of numbers in the buffers’. Set this value to 500, for example. The size depends on the expected data set of initial as well as future (delta) uploads.

**Important:** Never buffer the number range for the package dimension.

When loading large quantities of data, the number range buffer should be increased for the info objects that are likely to have a high number of data sets. Using function module RSD_IOBJ_GET, you find the number range name of the info object that is likely to have a high number of data sets.

**Function module settings**

```plaintext
I_IOBJNM = 'Info object name'
I_OBJVERS = 'A'
I_BYPASS_BUFFER = 'X'
```

The number for the info object is then in Table ‘E_S_VIOBJ’, column ‘NUMBRANR’. Specify BIM before this number, and the desired number range is given (for example, BIM0000053).

Using Transaction SNRO (ABAP/4 Workbench → Development → Other tools → Number ranges), you can display all number ranges used for the info objects in BW by entering BIM*. By entering the object name determined beforehand, you can find the desired number range.
By double-clicking this line, you get to the number range object maintenance. Choose Edit → Set up buffering → Main memory, to define the ‘number of numbers in the buffers’.

Set this value to 500, for example. The size depends on the expected data set of the initial as well as the future (delta) uploads.

**Important:** Never buffer the number range objects for the Characteristic OREQUEST.

**Source system: R/3 or another SAP system**

Carry out all steps except 1, 5, 6 as listed in “Source system: file” on page 29. Additionally, in R/3 or another SAP system, maintain this value in R/3 Customizing (SBIW → General settings → Control parameters for data transfer). In addition, set the value for Max. Perc. so that it matches the number of batch processes in the system.

### 4.4 Drop indexes (or not)

**Loading**

SAP creates indexes on F fact, E fact and dimension tables automatically while activating the InfoCube. The number of indexes created on F fact and E fact tables depends on the number of dimensions. Normally the presence of indexes would impact the loading activity into the F fact table (note our target table for loading is always the F fact table when dealing with InfoCubes).

In order to speed up loading activity, dropping indexes might sound like a good idea. However, if the number of records to be loaded is a small percentage of the size of the infocube, total time taken to load data with indexes many be less than the time to drop indexes, load data, and rebuild indexes.

**Rule of thumb:** If the number of records to be loaded is larger than 15%-20% of the target table, then drop indexes. Otherwise, do not drop.

The cutoff point for when it is beneficial to drop indexes varies from system to system. For instance, if DB2 is running in an LPAR with several processors, then parallel index rebuild will take less time than if only one processor is available. With one processor, the elapsed time penalty of index rebuild is larger than on a system with many processors. The CPU cost would be similar if the table size is the same.

One can determine the break-even point for dropping indexes more exactly by calculating the difference in time for loads with and without indexes, and comparing this time difference to the time to drop and recreate indexes. If keeping indexes during the load increases load time by less than the time to drop and recreate indexes, it is beneficial to keep indexes.

Normally customers load millions of records during initial load. So we recommend dropping of indexes during initial load. Likewise, if a cube is periodically refreshed, then the indexes should be dropped before the cube is loaded. Since delta loads are generally small compared to the size of the infocube, it is generally best to drop indexes for delta loads unless the described conditions are met.

**Note:** Some databases build bit-map indexes on F and E fact tables. DB2 builds b-tree indexes. In order to have parallelism in the load process, dropping and rebuilding indexes would be mandatory when dealing with bit-map indexes.
This can be very inefficient for small data loads into large tables. DB2 provides greater efficiency for data loads with large InfoCubes by providing the choice to drop or maintain indexes for the load.

**Compression**

After data is loaded and verified, you must compress requests regularly. As explained under SAP compression, this process would read data from F-Fact table, Insert/Update E-Fact table and delete data from F-Fact table. E-Fact tables normally grow in size on a ongoing basis. We recommend dropping and rebuilding of indexes (secondary only) during compression. You do not want to drop primary indexes. Whether dropping indexes improves performance depends on how large E-fact table is and how large the packet is.

As previously noted, if the rows being inserted during compression are more than 15%-20% of the size of the aggregates, then dropping and recreating indexes will probably help performance. Since you do not know ahead of time how many rows will be inserted on compression, you can also use the ratio of rows to be compressed to the total InfoCube size. If the rows to be compressed are more than 15-20% of the total InfoCube size, dropping and recreating indexes will probably help.

Depending on whether parallelism is used in index rebuild, the cutoff point for keeping the indexes may change.

We tested compressing with and without indexes on E fact. Our test configuration corresponded to a time early in the life of a project, where a relatively large amount of data is being added on each compression. In these tests, the number of rows in the F fact table was at least 25% of the total rows in the InfoCube. In this situation, dropping indexes for compression task reduces total time taken to compress. See Table 4-1 for the results.

<table>
<thead>
<tr>
<th>Job name</th>
<th>With or without index</th>
<th>Time taken to compress</th>
<th>Time taken to build index</th>
<th>Total time (compress time + build index)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BI_COMP__COMPRESS_X AA_WITH_IDX</td>
<td>With index</td>
<td>313 seconds</td>
<td>N/A</td>
<td>313 seconds</td>
</tr>
<tr>
<td>BI_COMP__COMPRESS_X ABWITHOUT_ID</td>
<td>Without index</td>
<td>165 seconds</td>
<td>71 seconds</td>
<td>236 seconds</td>
</tr>
<tr>
<td>BI_COMP__COMPRESS_X AC_WITH_IDX</td>
<td>With index</td>
<td>353 seconds</td>
<td>N/A</td>
<td>353 seconds</td>
</tr>
<tr>
<td>BI_COMP__COMPRESS_X AD_WOUT_IDX</td>
<td>Without index</td>
<td>147 seconds</td>
<td>69 seconds</td>
<td>216 seconds</td>
</tr>
<tr>
<td>BI_COMP__COMPRESS_X AE_WOUT_IDX</td>
<td>Without index</td>
<td>112 seconds</td>
<td>59 seconds</td>
<td>171 seconds</td>
</tr>
</tbody>
</table>

The layout of the InfoCube is the same as the benchmark InfoCube. Data was created using a homegrown korn shell script. We loaded five infopackages of 300,000 records each.
4.5 Aggregates

Aggregates are materialized, summarized views of the data in an InfoCube that has been pre-aggregated in anticipation of query requirements. The summarized data is stored in a physical table in the database. Queries to aggregates would return data quicker, since many rows in the InfoCube have been summarized into fewer rows in the aggregate.

If a query retrieves many rows which are then summarized to few rows in DB2, then database tuning is not generally going to be helpful in improving performance. When a query retrieves highly summarized data, then SAP tools (aggregates, query caching, and so on) should be used to optimize query performance.

The summarization is done in several different ways. One type of summarization is when the aggregate contains fewer characteristics than the InfoCube. Another type of summarization is to apply groupings (for example, navigation attributes or hierarchies) on characteristics in the InfoCube. Both are described in more detail in the following sections.

When an aggregate contains only characteristics, it does not need to be adjusted during master data change run. When an aggregate contains navigation attributes or hierarchies, it will need to be adjusted during change run. Thus, there is a trade-off between increased change run (which may be a weekly or relatively infrequent activity) with the query performance benefit of including navigation attributes and hierarchy levels in aggregates.

Aggregates would be useful in the following scenarios:

- Hierarchy levels aggregation
- Several users run a specific query which returns very small subset of very large infocubes
- Use of attributes in queries
- There is a high summarization rate required by queries

Types of aggregates

An aggregate is made up of characteristics and navigation attributes of an InfoCube. Data could be grouped in three levels, called aggregation levels.
**All characteristic values**
The following examples assume a simplified InfoCube (sales figures), containing the characteristics COUNTRY and CUSTOMER and the key figure SALES, as shown in Table 4-2.

**Table 4-2 InfoCube**

<table>
<thead>
<tr>
<th>Country</th>
<th>Customer</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>Buggy Soft</td>
<td>10</td>
</tr>
<tr>
<td>Germany</td>
<td>Ocean Networks</td>
<td>15</td>
</tr>
<tr>
<td>USA</td>
<td>Funny Duds</td>
<td>5</td>
</tr>
<tr>
<td>Austria</td>
<td>Ocean Networks</td>
<td>10</td>
</tr>
<tr>
<td>Austria</td>
<td>Thor Industries</td>
<td>10</td>
</tr>
<tr>
<td>Germany</td>
<td>Funny Duds</td>
<td>20</td>
</tr>
<tr>
<td>USA</td>
<td>Buggy Soft</td>
<td>25</td>
</tr>
</tbody>
</table>

**Example 1: Grouping a characteristic**
An aggregate, such as COUNTRY and SALES, can be created from this InfoCube. By removing the CUSTOMER characteristic, there will be fewer rows in the aggregate.

- Characteristics COUNTRY and SALES are used for grouping.
- Characteristic CUSTOMER is used for compressing. That is, rows in the InfoCube that have identical values in COUNTRY and SALES, but different values in CUSTOMER, are combined into a single row in the aggregate.

The data for key figure SALES is listed for the sum of the sales for each country and not for individual customers. See Table 4-3.

**Table 4-3 Aggregate data with fewer rows**

<table>
<thead>
<tr>
<th>Country</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>40</td>
</tr>
<tr>
<td>Germany</td>
<td>35</td>
</tr>
<tr>
<td>Austria</td>
<td>20</td>
</tr>
</tbody>
</table>

The aggregate can be used as follows:

- In a query that determines the sales for each country or the total sales
- For evaluations using a navigation attribute for characteristic COUNTRY, or using a hierarchy on the countries

The aggregate cannot be used if you want to expand or select in a query using characteristic CUSTOMER because it does not contain any information about the customers—the CUSTOMER characteristic has been removed.

**Example 2: Grouping a characteristic with a navigation attribute**
In the second example for the aggregation level All characteristic values, the industry is needed as a navigation attribute. See the master data in Table 4-4 on page 35.
Table 4-4  Master data table - Customer

<table>
<thead>
<tr>
<th>Customer</th>
<th>Industry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buggy Soft</td>
<td>Technology</td>
</tr>
<tr>
<td>Ocean Networks</td>
<td>Technology</td>
</tr>
<tr>
<td>Funny Duds</td>
<td>Consumption</td>
</tr>
<tr>
<td>Thor Industries</td>
<td>Chemicals</td>
</tr>
</tbody>
</table>

- Navigation attribute Industry is used for grouping CUSTOMER.
- Characteristic COUNTRY is used for compressing.

Table 4-5 shows that the aggregate can be used as follows:
- In a query that determines the sales for each industry or the total sales.
- For evaluations by node values, if there is a hierarchy for the industry.

Table 4-5  Industry aggregate

<table>
<thead>
<tr>
<th>Industry</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>60</td>
</tr>
<tr>
<td>Consumption</td>
<td>25</td>
</tr>
<tr>
<td>Chemicals</td>
<td>10</td>
</tr>
</tbody>
</table>

The aggregate cannot be used for queries referencing COUNTRY or CUSTOMER.

**Example 3: Time-dependent navigation attributes**

When a navigation attribute is time-dependent, it can have different values at different dates. If time-dependent navigation attributes are frequently used in queries, and the queries usually reference the same date (such as the close of the most recent period), then query performance may be improved by building aggregates using time-dependent navigation attributes.

In the third example of the aggregation level All characteristic values, the time-dependent navigation attribute SALESPERSON is needed. See Table 4-6.

Table 4-6  Salesperson table

<table>
<thead>
<tr>
<th>Country</th>
<th>From</th>
<th>To</th>
<th>Salesperson</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>January 1, 2000</td>
<td>December 31, 2000</td>
<td>Smith</td>
</tr>
<tr>
<td>USA</td>
<td>January 1, 2000</td>
<td>31.12.2001</td>
<td>Miller</td>
</tr>
<tr>
<td>Germany</td>
<td>January 1, 2000</td>
<td>30.03.2001</td>
<td>Meyer</td>
</tr>
<tr>
<td>Germany</td>
<td>January 4, 2000</td>
<td>31.12.2001</td>
<td>Huber</td>
</tr>
<tr>
<td>Austria</td>
<td>January 1, 2000</td>
<td>31.12.2001</td>
<td>Huber</td>
</tr>
</tbody>
</table>

- Navigation attribute SALESPERSON is used for grouping.
- Characteristic COUNTRY is used for compressing.
- Key date: Sept. 01, 2001

The aggregate can be used in a query that has the same key date as the aggregate. If a query were run for a different key date, such as August 01, 2001, then the aggregate could not be used; see Table 4-7 on page 36.
Table 4-7  Sales table

<table>
<thead>
<tr>
<th>SALESPERSON</th>
<th>SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Miller</td>
<td>40</td>
</tr>
<tr>
<td>Huber</td>
<td>55</td>
</tr>
</tbody>
</table>

Loading data into aggregates efficiently

Setting automatic compression

Aggregates like InfoCubes contain two fact tables, E and F. You can compress aggregates from F fact tables into E fact tables in one of two ways:

- Automatically when aggregates are filled with data, or
- Aggregates are compressed when InfoCubes are compressed

Navigation steps

1. Navigate to the Modeling function area of the Administrator Workbench; see Figure 4-10. In the InfoProvider tree, navigate to the required InfoCube.

![Figure 4-10 Modeling function area of the Administrator Workbench](image)

2. In the context menu of the InfoCube, select Manage. The Manage Data Targets dialog box appears.

3. Select the Rollup tab page in the lower part of the screen.
4) Under the Aggregates group area, set the corresponding indicator in the Compress After Rollup field; see Table 4-8.

Table 4-8 Compress after Rollup indicator defined

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>set (default): Automatic compression switched on</td>
<td>The aggregates of an InfoCube are compressed automatically when they are filled with data or after being rolled up from data packages (requests). If you want to delete a data package (request) from the InfoCube and the InfoCube was already rolled up to the aggregate, you have to deactivate the aggregate and build it again.</td>
</tr>
<tr>
<td>not set: Automatic compression switched off</td>
<td>The aggregates are only compressed together with the InfoCube. Use this setting if you have to frequently delete requests from the InfoCube. A specific request can be deleted from the aggregates when it has been deleted from the InfoCube. Keep an eye on the performance because aggregates can become quite large if they are not compressed automatically.</td>
</tr>
</tbody>
</table>

Reading the data in blocks

If the amount of data is very large during filling, the system reads the data in blocks and not all at one time. This avoids problems with the DB2 SORTWORK area on the database if you have very large sources (InfoCubes or aggregates).

The block size and Percentage change in the delta process could be changed to appropriate values to optimize this activity; see Figure 4-12 on page 38. You can change the values of these parameters using IMG → General BW settings → Parameters for Aggregates.
Limit with delta

In the change run, the aggregates are adjusted to match the new attributes and hierarchies. There are various strategies involved. You can completely rebuild the aggregate, or you can update the old records positively and the new records negatively (delta process).

The process used depends, among other things, on how much the data has changed. Specify the percentage at which aggregates should be rebuilt (0% means rebuild always).

Block size

The block size you choose depends on the amount of temporary tablespace available on the database. Specify a small number if you do not have a large temporary tablescape but expect long runtimes, because there would be more reads.

Defining aggregates

Defining appropriate aggregates involves these steps.

1. Gather information on how data is accessed:
   - Interview users and application team to understand the characteristics of reports they run. Get a list of poorly performing reports, if any.
   - Get a list of poorly performing reports, if any.
   - Use BW system transaction ST03 → Expert Mode → BW System Load to get a list of top 10 queries.
   - Get list of top queries from the BW statistics cubes.
   - Evaluate statistics for a query which as been reported to be slow.
   - In transaction SE16, check RSDDSTAT table entries either for a user or InfoCube or query.

2. Check two indicators of inefficient queries:
   - Rows selected per second (QDBSEL/QTIMEDB in RSDDSTAT table)
   - Ratio of rows selected to rows transferred (QDBSEL/QDBTRANS)

3. After finding the queries that have a high summarization ratio and long total DB time, the next step is to determine the characteristics used in the query. This can be done in several ways.
- Use RSA1 to propose an aggregate for a specific execution of the query, using the STATUID. The STATUID is found in RSDDSTAT.

- Use RSA1 to propose MIN and MAX aggregates for a query by query name. This will propose two aggregates, which contain the fewest and most characteristics which can be used in the query.

- Use SE16 to examine the InfoObjects used in a query - use the STATUID for an individual query to retrieve the InfoObjects from RSDDSTATAGGRDEF. The InfoObjects can be used in the aggregate definition in RSA1.

Figure 4-13 shows an example of the potential benefit of optimizing a query with an aggregate. In this screenshot, the query spent 2,465 seconds in DB(QTIMEDB field), selected 32+ million records (QDBSEL), but transported only 1188(QDBTRANS) records. This was a very good candidate for aggregation. An aggregate was created on DIVISION. After creation of the aggregate, the same query ran in 74 seconds.

![Figure 4-13 The RSDDSTAT entry for a query accessing the InfoCube](image)

This screen shows the RSDDSTAT entry for a query accessing the InfoCube. This query used 2,465+ seconds in the database (QTIMEDB). The number of records selected was 32+ million records (QDBSEL).
Since the number of records transported (QDBTRANS) was only 1,188, we decided to create an aggregate. The high ratio of QDBSEL/QDBTRANS is an indicator that an aggregate may help performance.

Figure 4-14 on page 40 shows the RSDDSTAT entry accessing the aggregate.

With aggregate, the QTIMEDB was only 74 seconds, number of records selected was 1,120,891.

Figure 4-15 on page 41 compares the RSDDSTAT table entries pre-aggregate and post-aggregate scenarios.
4.6 SAP compression for loads

4.6.1 Overview

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.

4.6.2 Advantages

- 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 values 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 (that is, Package 1 and 2 were loaded on the same day, for example). In situations where 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.

4.6.3 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 4.6.4, “Cumulative data versus non-cumulative data” on page 42), and all records summarized into the E fact table.

The replacement of the Package ID means 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.

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

4.6.4 Cumulative data versus non-cumulative data

There are two types of data in a fact table: cumulative key figures, and non-cumulative key figures. These data types are assigned special 99 or 00 package IDs in the E fact table.
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 of 99 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 is inserted or updated in the E fact table that has the same combination of dimensions id, but is infinite for time. Rather than requiring to sum 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 00 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.

### 4.6.5 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 of 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 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 will mean reduced RUNSTATS runtimes.
- Backups and restores will be 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 backup/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 E fact table.
Query impact

- Queries against the InfoCube will be faster in situations where 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 RUNSTATS is run to maintain accurate statistics, as the E fact table grows rapidly. The same applies as new partitions are populated.

4.6.6 Implementing SAP compression

Initially when you go live with new InfoCubes, it may be a good idea to wait for few weeks before deciding to compress regularly. As explained earlier, you could delete data from F fact tables based on the package id. So that 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 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.

Demonstration detail

Assume three approaches for the following: load/query into uncompressed F fact, load/query into unpartitioned E fact, and load/query into partitioned E fact.

For load we create two tests: one loading low granularity key figures into the F fact and one loading high granularity key figures. We do a series of loads looking to see if load times vary as more and more data exists in the table. Time dimension should be monthly with a daily load. Load enough records so the same data (compressed and non-compressed) can be used for the query tests and for the storage test. Four loads each followed by a query test. First cumulative load into F fact without compressing into E fact, secondly cumulative compressed into E fact, thirdly non-cumulative into F fact non-compressed, fourth non-cumulative compressed into E fact.

For storage we create two tests: one with cumulative key figures in the F Fact and one with non-cumulative. When we compress we count the total rows in both the F Fact and in the E fact. Expect to see a significant reduction with cumulative and a lesser reduction with non. Time dimension should be monthly with a daily load. Load enough records so the same data (compressed and non-compressed) can be used for the query tests.

For query tests, query against the F Fact table and against a E fact table with little compression (non cumulative) to show the query advantage the partitioned E fact table (covered in the partitioning section) has over the F fact.
**Chapter 4. Best practices for InfoCubes**

**Figure 4-16** Data browser initial screen showing an input field for the table name

This screen shows the table name (SAPC_6489) used to show the effects of SAP compression.

**Figure 4-17** This F fact table contains 400,000 records before compression

**Figure 4-18** Key figures and dimensions available in the demo
This figure shows the key figures and dimensions available in this demo table./BIC/FSAPC_6489. For the purpose of demonstration, we picked one time period and two available key figures, JO_QTY and JO_REV.

We ran RSRT transaction to display SQL query and Statistical data. We inactivated Cache use at the system level.
Since SAP compression has not been done yet, the query uses the F table.
Figure 4-21  The result of the query

Figure 4-22  The query took 18+ seconds (QTIMEDB)
Figure 4-23  Results of a request for an additional 400,000 records loaded

Here we see the results of another request of 400,000 records loaded, showing that the F table contains 800,000 records.

Figure 4-24  The same query with 800,000 records took 35+ seconds
Figure 4-25   Manage Data Targets

We used Manage Data Targets and selected the Collapse tab to compress two requests. As you can see, the third column has been checked to show that the requests are compressed. To navigate to Manage Data Targets, right-click InfoCube, then select Manage Data Targets.

Figure 4-26   The E fact table after compression
After compression, the E fact table contains 400,000 records. You may recall that F table contained 800,000 records. We received a 100% compression ratio.

The RSRT transaction is used to show the time taken to run the query we used before requests were compressed. We inactivated cache use at the system level and so “Do Not Use Cache” is not relevant.

The query which took 35+ seconds took 12+ seconds after compression. Before compression, the query selected 19,999 records. After compression, the query selected 9,999 records (QDBSEL).
4.7 DB2 compression

Because the size of SAP BW InfoCubes can grow to be extremely large, and thus use large amounts of disk space, we recommend the use of 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. 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. For a detailed discussion on DB2 Compression, see 9.4, “DB2 compression” on page 157.
Best practices for Operational Data Store

In this chapter we discuss the best practices for optimizing BW Operational Data Store (ODS) performance on the z/OS DB2 V8 platform. This chapter describes the following:

- Types of ODS
- Performance aspects of ODS query performance:
  - Indexing
  - Clustering
  - Partitioning

ODS basic functionality and terminology is described in 10.2, “SAP BW information model” on page 171 and in 10.10, “Operational data store (ODS)” on page 182.
5.1 Types of ODS

Basically, two ODS object types are available:

1. Standard ODS object
2. Transactional ODS object

The ODS is stored in transparent flat tables. Fact tables or dimensions are not created. The standard ODS creates three transparent tables, as shown in Table 5-1.

Table 5-1    ODS tables

<table>
<thead>
<tr>
<th>Position</th>
<th>PSA table</th>
<th>Naming convention</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Active ODS (A table)</td>
<td>/BIC/A&lt;ods_object&gt;00</td>
</tr>
<tr>
<td>1</td>
<td>Activation queue</td>
<td>/BIC/01C/B&lt;10-digits&gt;</td>
</tr>
<tr>
<td>1</td>
<td>Change log</td>
<td>/BIC/A&lt;ods_object&gt;50</td>
</tr>
</tbody>
</table>

The activation queue contains the data updated data records that are not activated yet. The Change log contains the change history. The active ODS table contains the active data. In the transactional ODS objects, 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.

5.2 Secondary indexes on ODS objects

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

5.2.1, “Creating a secondary index on an ODS table” on page 55, describes how to create a secondary index.
On the DEMO System we have created two indexes, as shown in Figure 5-1:
1. On MATERIAL, to provide a selective index for queries related on MM
2. On DIVISION, to build four partitions related on the four enterprise divisions B1 - B4

The additional secondary indexes Z01 in MATERIAL can impact the ODS and InfoCube Load performance. To reduce the load time, you could delete this index for the LOAD runs. Then recreate it after ODS or InfoCube Load, for maximum query performance.

5.2.1 Creating a secondary index on an ODS table

In earlier releases of SAP BW, transactions SE11 (ABAP Dictionary) and SE14 (ABAP Dictionary: Database Utility) must be used to create secondary indexes on ODS objects.

As of SAP BW 3.x, however, secondary indexes can be created from the Edit ODS Object screen, as follows:
1. Call transaction RSA1 (Administrator Workbench: Modeling → InfoProvider). Double-click the ODS object on which you want to create a secondary index.
2. On the Edit ODS Object screen, right-click Indexes and choose Create; see Figure 5-2 on page 56.
Figure 5-2  Create index screen

3. The Index Maintenance pop-up panel is displayed, with a name filled in for the index. Choose if you want this index to be a unique index or not. In general, secondary indexes are not unique (see Figure 5-3 on page 57). Press the green checkmark.
4. On the Edit ODS Object screen, expand Key Fields and expand Indexes. Drag and drop the columns you want in the index from Key Fields to your index name. In this example, we included CALDAY and DIVISION in the index; see Figure 5-4 on page 58.
5. Save and activate the ODS object. This creates the index in the database.

Run RUNSTATS to collect statistics for this new index. Use transaction DB13 (DBA Planning Calendar) to “update statistics for one SAP object”.

5.3 Partitioning and clustering

With table-controlled partitioning (introduced in DB2 V8), 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 V8 documentation or DB2 UDB for z/OS Version 8 Technical Preview, SG24-6871.

Partitioning provides many benefits in the areas of data management, data availability, and query performance, especially for large tables like ODS objects. The new support in DB2 V8, which allows addition and rotation of partitions, 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 new DB2 V8 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 V8, however, 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 ODS objects. ODS objects are typically large, containing many rows of data. Queries against an ODS 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.

### 5.3.1 Choosing the partitioning key

The partitioning key defines how a table is to be partitioned—it dictates which records are in which partitions. Before DB2 V8 and table-controlled partitioning, the partitioning key was the partitioning index. In DB2 V8, 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 column(s) 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 are looking at choosing the partitioning key to optimize query performance. Queries are executed against the active ODS table, so we are really discussing choosing the partitioning key for the /BIC/A<ODSname>00) table. Keep in mind that this table also is involved in the activation of ODS 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 ODS 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 an ODS query in order to analyze how the ODS is used by queries.

One common partitioning key for an ODS 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 ODS.

Let’s look again at the ODS query example discussed in 6.1, “SQL efficiency for ODS” on page 80. In addition to adding the indexes recommended in 6.3.1, “Index design for post-ODS access” on page 93 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 ODS.)

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.

### 5.3.2 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 ODS objects.

To optimize the query performance on ODS 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. In DB2 V8, 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 table space 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.

### 5.3.3 Data Partitioned Secondary Indexes (DPSIs)

Data Partitioned Secondary Indexes (DPSIs) are a new feature in DB2 V8. This feature allows physically partitioned secondary indexes. Prior to DB2 V8, secondary indexes on partitioned tables could not be partitioned. 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 ODS in order to make an informed decision about whether or not 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 ODS objects**

For ODS objects, DPSIs should only be considered if queries against the ODS 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's look again at the ODS query discussed in 6.1, “SQL efficiency for ODS” on page 80. Assume 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.

### 5.3.4 Partitioning an ODS

Ideally, an ODS 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 ODS from non-partitioned to partitioned. According to *SAP 6.40 DBA Guide*, DB2 utilities should be used to move data between tables if the table has more than 1 million rows or it is larger than 100 MB. In general, ODS tables will 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.

The following procedure describes how to convert an ODS 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 V8.
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 5-1 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 V8 Utility Guide and Reference.)

Example 5-1  Example of JCL for DB2 UNLOAD Utility

```jcl
//UNLOAD JOB (1,POK), 'ITSO', CLASS=A, MSGCLASS=T,
// TIME=NOLIMIT, REGION=OM
// JCLLIB ORDER=(DB8NU.PROCLIB)
/*JOBPARM S=SC04
//UNLOAD EXEC DSNUPROC,LIB=DB8N8.SDSNLOAD,
// SYSTEM='DB8N',
// UID='UNLOAD1', UTPROC=''
//SYSPRINT DD SYSOUT=* 
//SYSREC DD DSN=D8I0.OD60XSPU.AODSTES.T00.SYSREC.D041104,
// DISP=(NEW,CATLG,CATLG),
// UNIT=SYSDA, SPACE=(CYL,(200,50))
// The unloaded data is written to SYSREC.
//SYSPUNCH DD DSN=D8I0.OD60XSPU.AODSTES.T00.SYSPUNCH.D041104,
// DISP=(NEW,CATLG,CATLG),
// UNIT=SYSDA, SPACE=(TRK,(5,5))
// The control statement to be used to reload the data is written to SYSPUNCH.
//SYSPRINT DD SYSOUT=* 
//SYSIN DD *
UNLOAD TABLESPACE OD60XSPU.AODSTES
```

After the data has been unloaded, call transaction SE14 (ABAP Dictionary: Database Utility). Specify the name of the table to be partitioned and press Edit; see Figure 5-5.
From the ABAP Dictionary: Utility for Database Tables screen, choose Storage Parameters; see Figure 5-6.
From the screen DB2/390 Storage Attributes: Table, choose Attributes → Display↔Change. Under Target Tablespace, choose Partitioned; see Figure 5-7 on page 66.
At this point, you can choose whether or not 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 on using LIMITKEY data, see “Using the LimitKey function” on page 70.

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 5-8 on page 67.

**Note:** Keep in mind this is just a test ODS, and does not represent realistic data. The goal of this section is to show the mechanics of partitioning an ODS. 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 5-8   DB2/390 Storage Attributes: Table

Now return to the screen 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 5-9 on page 68.
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 72. 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 5-2 on page 69 for sample JCL for the DB2 LOAD utility, and Example 5-3 on page 69 for a sample control statement for the DB2 LOAD utility. For complete details on the DB2 utilities, refer to *DB2 V8 Utility Guide and Reference*. 
Example 5-2  Example of JCL for the DB2 Load Utility

//LOAD JOB (1,POK),'ITSO',CLASS=A,MSGCLASS=T,
// TIME=NOLIMIT,REGION=0M
// JCLLIB ORDER=(DBBNLU.PROCLIB)
/*JOBPARM S=SC04
//LOAD EXEC DSNUPROC,LIB=DB8N8.DSNLOAD,
// SYSTEM='DB8N',
// UID='LOAD1',UTPROC=''
//SORTOUT DD UNIT=SYSDA,SPACE=(CYL,(1500,500)),
// DISP=(NEW,DELETE)
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1500,500)),
// DISP=(NEW,DELETE)
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,(1500,500)),
// DISP=(NEW,DELETE)
//SORTWK02 DD UNIT=SYSDA,SPACE=(CYL,(1500,500)),
// DISP=(NEW,DELETE)
//SYSDISC DD UNIT=SYSDA,SPACE=(CYL,(100,100))
//SYSERR DD UNIT=SYSDA,SPACE=(CYL,(100,100))
//SYSMAP DD UNIT=SYSDA,SPACE=(CYL,(100,100)),,ROUND
//SYSPRINT DD SYSOUT=
//UTPRINT DD SYSOUT=
/* SYSREC contains the data to be loaded
//SYSREC DD DISP=(OLD,KEEP),
// DSN=DB100.DBDXSPU.ODSTES.T00.SYSREC.D041104,
// DCB=(BUFNO=20)
// SYSIN contains the control statement for the LOAD utility
//SYSIN DD DISP=(OLD,KEEP),
// DSN=DB100.IDD.ODSTEST

Example 5-3  Example of a control statement for the DB2 LOAD utility

LOAD DATA INDDN SYSREC   LOG NO REPLACE REUSE
UNICODE CCSID(00367,01208,01200)
INTO TABLE "SAPD8I"."/BIC/AODSTEST00"
WHEN(00001:00002 = X'0003')
  ( "VERSION"
    POSITION( 00003:00010) VARGRAPHIC
  "SOLD_TO"
    POSITION( 00011:00032) VARGRAPHIC
  "SALESORG"
    POSITION( 00033:00042) VARGRAPHIC
  "VTYPE"
    POSITION( 00043:00050) VARGRAPHIC
  "DISTR_CHAN"
    POSITION( 00051:00056) VARGRAPHIC
  "DIVISION"
    POSITION( 00057:00062)
  "MATERIAL"
    POSITION( 00063:00100) VARGRAPHIC
  "CALDAY"
    POSITION( 00101:00118) VARGRAPHIC
  "FISCVARNT"
    POSITION( 00119:00124) VARGRAPHIC
  "STAT_CURR"
    POSITION( 00125:00136) VARGRAPHIC
  "BASE_UOM"
    POSITION( 00137:00144) VARGRAPHIC
  "INCORDVAL"
    POSITION( 00145:00153) DECIMAL
  "INVCD_VAL"
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 and press **Edit**; see Figure 5-5 on page 64. Follow these steps:

1. From the screen ABAP Dictionary: Utility for Database Tables, choose **Storage Parameters**; see Figure 5-6 on page 65.

2. From the screen DB2/390 Storage Attributes: Table screen, choose **Goto → LIMITKEY Data**; see Figure 5-10 on page 71. 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.

Upon completion of the job, you will see the LIMITKEY data displayed, as shown in Figure 5-11.
Example of an SAP Object Log for table controlled partitioning

Example 5-4 shows the SAP Object Log created when a table was converted from non-partitioned to partitioned using table-controlled partitioning.

Example 5-4  SAP Object Log from partitioning of a table using table-controlled partitioning

```
DB-TABL-BIC-AODSTEST00

Request: Delete and recreate Table /BIC/AODSTEST00 (ITSO04/05.11.04/20:51)
Process: sap01_1
Enhancement category for table missing
Test activation of Table /BIC/AODSTEST00 successful
Activation and DDL statements for Table /BIC/AODSTEST00 required

Activate table /BIC/AODSTEST00 (ITSO04/05.11.04/20:51)
Enhancement category for table missing

Activate dependent table type /BIC/WAODSTEST00
Table type /BIC/WAODSTEST00 was activated
sql:
DROP TABLE "/BIC/AODSTEST00" +LOCATION
DDL time(__1): ........57 milliseconds
sql:
CREATE TABLESPACE AODSTESX IN OD60XHNT
USING STOGROUP REDODD PRIQTY 71
SECQTY 6827 FREEPAGE 16 PCTFREE 20
GBPCHACHE CHANGED COMPRESS NO
LOCKPART NO DSSIIZE 4 G NUMPARTS 6 (
PART 1 USING STOGROUP REDODD ,
PART 2 USING STOGROUP REDODD ,
PART 3 USING STOGROUP REDODD ,
PART 4 USING STOGROUP REDODD ,
PART 5 USING STOGROUP REDODD ,
PART 6 USING STOGROUP REDODD )
BUFFERPOOL BP2 LOCKSIZE ROW LOCKMAX 0
CCSID UNICODE
|CREATE TABLE "/BIC/AODSTEST00"
("VERSION" VARGRAPHIC (000003) NOT NULL
DEFAULT ' ' ,
"SOLD_TO" VARGRAPHIC (000010) NOT NULL
DEFAULT ' ' ,
"SALESORG" VARGRAPHIC (000004) NOT NULL
DEFAULT ' ' ,
"VTYPE" VARGRAPHIC (000003) NOT NULL
DEFAULT '000' ,
"DISTR_CHAN" VARGRAPHIC (000002) NOT NULL
DEFAULT ' ' ,
"DIVISION" VARGRAPHIC (000002) NOT NULL
DEFAULT ' ' ,
"MATERIAL" VARGRAPHIC (000018) NOT NULL
DEFAULT ' ' ,
"CALDAY" VARGRAPHIC (000008) NOT NULL
DEFAULT '00000000' ,
"FISCVARNT" VARGRAPHIC (000002) NOT NULL
DEFAULT ' ' ,
"STAT_CURR" VARGRAPHIC (000005) NOT NULL
DEFAULT ' ' ,
```

72  Best Practices for SAP Business Information Warehouse on DB2 for z/OS V8
"BASE_UOM" VARGRAPHIC (000003) NOT NULL DEFAULT '' ,
"INCORDVAL" DECIMAL (000017,000002) NOT NULL DEFAULT 0 ,
"INVCD_VAL" DECIMAL (000017,000002) NOT NULL DEFAULT 0 ,
"OPORDVALSC" DECIMAL (000017,000002) NOT NULL DEFAULT 0 ,
"RTNSVAL" DECIMAL (000017,000002) NOT NULL DEFAULT 0 ,
"CRMEM_VAL" DECIMAL (000017,000002) NOT NULL DEFAULT 0 ,
"INCORDQTY" DECIMAL (000017,000003) NOT NULL DEFAULT 0 ,
"INVCD_QTY" DECIMAL (000017,000003) NOT NULL DEFAULT 0 ,
"OPORDQTYBM" DECIMAL (000017,000003) NOT NULL DEFAULT 0 ,
"RTNSQTY" DECIMAL (000017,000003) NOT NULL DEFAULT 0 ,
"CRMEM_QTY" DECIMAL (000017,000003) NOT NULL DEFAULT 0 ,
"ORD_ITEMS" DECIMAL (000017,000003) NOT NULL DEFAULT 0 ,
"RTNS_ITEMS" DECIMAL (000017,000003) NOT NULL DEFAULT 0 ,
"INCORDCST" DECIMAL (000017,000002) NOT NULL DEFAULT 0 ,
"INVCD_CST" DECIMAL (000017,000002) NOT NULL DEFAULT 0 ,
"RTNSCST" DECIMAL (000017,000002) NOT NULL DEFAULT 0 ,
"CRMEM_CST" DECIMAL (000017,000002) NOT NULL DEFAULT 0 ,
"RECORDMODE" VARGRAPHIC (000001) NOT NULL DEFAULT ''
PARTITION BY ( "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 OD60XHNT.AODSTESX
DDL time(__1): .....3.282 milliseconds
sql:
CREATE TYPE 2 UNIQUE INDEX "/BIC/AODSTEST00" ON "/BIC/AODSTEST00"
("VERSION",  
"SOLD_TO",  
"SALESORG",  
"VTYPE",  
"DISTR_CHAN",  
"DIVISION",  
"MATERIAL",  
"CALDAY",  
"FISCVARNT",  
"STAT_CURR",  
"BASE_UOM" )
USING STOGROUP REDO1 PRIQTY 16
SECQTY 40960 FREEPAGE 10 PCTFREE 10
GBPcache CHANGED
BUFFERPOOL BP3 COPY NO
PIECESIZE 2097152 K CLUSTER
DDL time(__2): ........359 milliseconds
sql:
ALTER TABLE "/BIC/AODSTEST00"
ADD PRIMARY KEY ("VERSION",
"SOLD_TO",
"SALESORG",
"VTYPE",
"DISTR_CHAN",
"DIVISION",
"MATERIAL",
"CALDAY",
"FISCVARNT",
"STAT_CURR",
"BASE_UOM")
DDL time(__3): ........17 milliseconds
sql:
COMMIT WORK
DDL time(__4): ........7 milliseconds
sql:
DELETE FROM DDSTORAGE
WHERE DBSYSABBR = 'DB2'
AND TABNAME = '/BIC/AODSTEST00'
AND INDEXNAME = '0'
DDL time(__5): ........1 milliseconds
sql:
COMMIT WORK
DDL time(__6): ........5 milliseconds
sql:
DELETE FROM DDSTORAGE
WHERE DBSYSABBR = 'DB2'
AND TABNAME = '/BIC/AODSTEST00'
AND INDEXNAME = ''
DDL time(__7): ........2 milliseconds
sql:
CREATE TYPE 2 INDEX "/BIC/AODSTEST00~01" ON "/BIC/AODSTEST00"
("CALDAY")
USING STOGROUP REDODI PRIQTY 16
SECQTY 40960 FREEPAGE 10 PCTFREE 10
GBPCACHE CHANGED
BUFFERPOOL BP3 COPY NO
PIECESIZE 2097152 K
DDL time(__8): ........324 milliseconds
sql:
COMMIT WORK
DDL time(__9): ........7 milliseconds
sql:
DELETE FROM DDSTORAGE
WHERE DBSYSABBR = 'DB2'
AND TABNAME = '/BIC/AODSTEST00'
AND INDEXNAME = '010'
DDL time(__10): ........1 milliseconds
Request for /BIC/AODSTEST00 executed successfully

------------------------------------------------------------------------------------
Example of an 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 5-5.

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

- Table /BIC/AODSTEST00 is a partitioned table.
- Index /BIC/AODSTEST00--01 is a partitioned index—and it is also a partitioning index.
- Index /BIC/AODSTEST00--0 is a non-partitioned primary index.

We do not recommend using index-controlled partitioning. Table-controlled partitioning that was introduced in DB2 V8 is the recommended method, because it provides more support and flexibility for partitioning tables and indexes.

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

```
DB-TABL-BIC-AODSTEST00
```

Request: Delete and recreate Table /BIC/AODSTEST00 (ITS004/12.11.04/08:36)
Process: sap01_0
Enhancement category for table missing
Test activation of Table /BIC/AODSTEST00 successful
Activation and DDL statements for Table /BIC/AODSTEST00 required

Activate table /BIC/AODSTEST00 (ITS004/12.11.04/08:36)
Enhancement category for table missing

Activate dependent table type /BIC/WAODSTEST00
Table type /BIC/WAODSTEST00 was activated
sql:
DROP TABLE "/BIC/AODSTEST00" +LOCATION
DDL time(____): ......397 milliseconds
sql:
CREATE TABLESPACE AODSTESX IN OD60XHNT
USING STOGROUP REDO DD PRIQTY 72
SECKTY 6828 FREEPAGE 16 PCTFREE 20
GPBCACHE CHANGED COMPRESS NO
LOCKPART NO DSSIZE 4 G NUMPARTS 6 (PART 1 USING STOGROUP REDO DD,
PART 2 USING STOGROUP REDO DD,
PART 3 USING STOGROUP REDO DD,
PART 4 USING STOGROUP REDO DD,
PART 5 USING STOGROUP REDO DD,
PART 6 USING STOGROUP REDO DD)
BUFFERPOOL BP2 LOCKSIZE ROW LOCKMAX 0
CCSID UNICODE
|CREATE TABLE "/BIC/AODSTEST00"
("VERSION" VARCHAR (000003) NOT NULL DEFAULT ' ',
"SOLD_TO" VARCHAR (000010) NOT NULL DEFAULT ' ',
"SALESORG" VARCHAR (000004) NOT NULL DEFAULT ' ',
"VTYPE" VARCHAR (000004) NOT NULL DEFAULT '000',
"DISTR_CHAN" VARCHAR (000002) NOT NULL DEFAULT ' ',

Chapter 5. Best practices for Operational Data Store  75
"DIVISION" VARGRAPHIC (000002) NOT NULL DEFAULT ' ' ,
"MATERIAL" VARGRAPHIC (000018) NOT NULL DEFAULT ' ' ,
"CALDAY" VARGRAPHIC (000008) NOT NULL DEFAULT '00000000' ,
"FISCVARNT" VARGRAPHIC (000002) NOT NULL DEFAULT ' ' ,
"STAT_CURR" VARGRAPHIC (000005) NOT NULL DEFAULT ' ' ,
"BASE_UOM" VARGRAPHIC (000003) NOT NULL DEFAULT ' ' ,
"INCORDVAL" DECIMAL (000017,000002) NOT NULL DEFAULT 0 ,
"INVCD_VAL" DECIMAL (000017,000002) NOT NULL DEFAULT 0 ,
"OPORDVALSC" DECIMAL (000017,000002) NOT NULL DEFAULT 0 ,
"RTNSVAL" DECIMAL (000017,000002) NOT NULL DEFAULT 0 ,
"CRMEM_VAL" DECIMAL (000017,000002) NOT NULL DEFAULT 0 ,
"INCORDQTY" DECIMAL (000017,000003) NOT NULL DEFAULT 0 ,
"INVCD_QTY" DECIMAL (000017,000003) NOT NULL DEFAULT 0 ,
"OPORDQTYYM" DECIMAL (000017,000003) NOT NULL DEFAULT 0 ,
"RTNSQTY" DECIMAL (000017,000003) NOT NULL DEFAULT 0 ,
"CRMEM_QTY" DECIMAL (000017,000003) NOT NULL DEFAULT 0 ,
"ORD_ITEMS" DECIMAL (000017,000003) NOT NULL DEFAULT 0 ,
"RTNS_ITEMS" DECIMAL (000017,000003) NOT NULL DEFAULT 0 ,
"INCORDCST" DECIMAL (000017,000002) NOT NULL DEFAULT 0 ,
"INVCD_CST" DECIMAL (000017,000002) NOT NULL DEFAULT 0 ,
"RTNSCST" DECIMAL (000017,000002) NOT NULL DEFAULT 0 ,
"CRMEM_CST" DECIMAL (000017,000002) NOT NULL DEFAULT 0 ,
"RECORDMODE" VARGRAPHIC (000001) NOT NULL DEFAULT ' ' )
IN OD60XHNT.AODSTESX
DDL time([i]): 4.008 milliseconds
sql:
CREATE TYPE 2 INDEX "/BIC/AODSTEST00~01" ON "/BIC/AODSTEST00"
("CALDAY")
USING STOGROUP REDODI PRIQTY 12
SECQTY 6827 FREEPAGE 10 PCTFREE 10
GBPCACHE CHANGED CLUSTER ( PART 1 VALUES ( '19920423' )
USING STOGROUP REDODI , PART 2 VALUES ( '19940820' ) USING STOGROUP REDODI , PART 3 VALUES ( '19961222' ) USING STOGROUP REDODI ,
PART 4 VALUES ( '19990424' )
USING STOGROUP REDODI,
PART 5 VALUES ( '20010826' )
USING STOGROUP REDODI,
PART 6 VALUES ( '99999999' )
USING STOGROUP REDODI )
BUFFERPOOL BP3 COPY NO
DDL time(__2): ......2.109 milliseconds
sql:
COMMIT WORK
DDL time(__3): .........7 milliseconds
sql:
DELETE FROM DDSTORAGE
WHERE DBSYSABBR = 'DB2'
AND TABNAME = '/BIC/AODSTEST00'
AND INDEXNAME = '010'
DDL time(__4): ........6 milliseconds
sql:
CREATE TYPE 2 UNIQUE INDEX "/BIC/AODSTEST00" ON "/BIC/AODSTEST00"
("VERSION", 
"SOLD_TO", 
"SALESORG", 
"VTYPE", 
"DISTR_CHAN", 
"DIVISION", 
"MATERIAL", 
"CALDAY", 
"FISCVARNT", 
"STAT_CURR", 
"BASE_UOM")
USING STOGROUP REDODI PRIORITY 16
SEQCQTY 40960 FREEPAGE 10 PCTFREE 10
GBPCACHE CHANGED
BUFFERPOOL BP3 COPY NO
PIECESIZE 2097152 K
DDL time(__5): .......347 milliseconds
sql:
ALTER TABLE "/BIC/AODSTEST00"
ADD PRIMARY KEY ("VERSION", 
"SOLD_TO", 
"SALESORG", 
"VTYPE", 
"DISTR_CHAN", 
"DIVISION", 
"MATERIAL", 
"CALDAY", 
"FISCVARNT", 
"STAT_CURR", 
"BASE_UOM")
DDL time(__6): ........18 milliseconds
sql:
COMMIT WORK
DDL time(__7): ........9 milliseconds
sql:
DELETE FROM DDSTORAGE
WHERE DBSYSABBR = 'DB2'
AND TABNAME = '/BIC/AODSTEST00'
AND INDEXNAME = '0'
DDL time(__8): .......12 milliseconds
sql:
5.4 DB2 compression

Since ODS objects tend to be large (especially the active ODS data, the /BIC/A<ODSname>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 ODS 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. For a detailed discussion on DB2 Compression, see 9.4, “DB2 compression” on page 157.

5.5 Materialized query tables

Material query tables (MQTs), introduced in DB2 V8, can be used to improve ODS 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 ODS objects, so the DB2 V8 function of MQTs can be used to do this for ODS objects. For more detailed information about materialized query tables, refer to 11.5, “Materialized query tables” on page 200.
Tips for SQL efficiency

This chapter provides tips that will help you enhance the efficiency of your SQL processing in both ODSs and InfoCubes. It also describes how to set up your indexes for those components.
6.1 SQL efficiency for ODS

Queries against the ODS 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 ODS. Understanding and analyzing your queries is an important step in performance tuning.

In this section, we illustrate a method 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 6-1 shows the query that we will analyze in this section. It is a real-life customer query.

![Figure 6-1 Example customer query against an ODS](image)

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 will 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 6-2 on page 81. We will use the local predicates in a later step to determine where the filtering occurs.
Chapter 6. Tips for SQL efficiency

Step 1 - Join Analysis

WHERE "/BIC/AZOSASALE00"."CURRENCY" = "/BIO/SCURRENCY"."CURRENCY"
AND "/BIC/AZOSASALE00"."G_UVVUOM" = "A0001"."UNIT"
AND "/BIC/AZOSASALE00"."G_UABSMG" = "A0002"."UNIT"
AND "/BIC/AZOSASALE00"."REFER_DOC" = "/BIO/SREFER_DOC"."REFER_DOC"
AND "/BIC/AZOSASALE00"."REFER_ITM" = "/BIO/SREFER_ITM"."REFER_ITM"
AND "/BIC/AZOSASALE00"."CO_AREA" = "/BIO/SCO_AREA"."CO_AREA"
AND "/BIC/AZOSASALE00"."PROFIT_CTR" = "/BIO/SPROFIT_CTR"."PROFIT_CTR"
AND "/BIC/AZOSASALE00"."CO_AREA" = "/BIO/SPROFIT_CTR"."CO_AREA"
AND "/BIC/AZOSASALE00"."/BIC/ZA_SADIV" = "/BIO/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 "/BIO/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 ("/BIO/SCO_AREA"."SID" = 3))
AND (("/BIC/AZOSASALE00"."CURTYPE" = 'B0'))
AND (("/BIC/SREFER_ITM"."SID" = 31))
AND (("/BIC/SPROFIT_CTR"."SID" = '5X'))
AND ((("/BIC/XZMATLPLNT"."S__ZMATLONLY" = '360013'))
AND ((("/BIC/XZRLNSP"."S__ZA_SASGRP" = '80')))

Figure 6-2  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" = "/BIO/SCURRENCY"."CURRENCY", which maps to a join relationship between the tables "/BIC/AZOSASALE00" and "/BIO/SCURRENCY"; see Figure 6-3.

Step 1 - Building the Join Graph

WHERE "/BIC/AZOSASALE00"."CURRENCY" = "/BIO/SCURRENCY"."CURRENCY"
AND "/BIC/AZOSASALE00"."G_UVVUOM" = "A0001"."UNIT"
AND "/BIC/AZOSASALE00"."G_UABSMG" = "A0002"."UNIT"

Figure 6-3  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 6-4.

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, you map each local predicate in the query to the table that it refers to on the graph; refer to Figure 6-5.
What we have done is taken the original complex query and broken this 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.

3. Determine where the filtering is occurring for this SQL statement.

Now you 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 6-6 shows the SQL count statement and the returned count for the PDIVISION/AZOSASALE00 join relationship.

```
SELECT COUNT(*)
FROM "/BI0/PDIVISION" "/BIC/AZOSASALE00"
WHERE "/BIC/AZOSASALE00"."/BIC/ZA_SADIV" = "/BI0/PDIVISION"."DIVISION"
AND "/BI0/PDIVISION"."OBJVERS" = 'A'
AND "/BI0/PDIVISION"."/BIC/ZREPTDIV" = '5X'
AND "/BIC/AZOSASALE00"."ACCNT_ASGN" IN ('00', '01', '04', '06')
AND "/BIC/AZOSASALE00"."CALMONTH" BETWEEN '200311' AND '200404'
AND "/BIC/AZOSASALE00"."CURTYPE" = 'B0'
```

Figure 6-6 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 ODS table has 7 million rows.

Step 3 is repeated for each master data table. A count should also be returned for the ODS 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 6-7 on page 84 for these counts.
Step 4 - Compare with Actual Access Path

<table>
<thead>
<tr>
<th>Tables Accessed</th>
<th>Count with Filtering</th>
<th>Final Filtering</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDIVISION - AZOSASALE00</td>
<td>12432</td>
<td>4</td>
</tr>
<tr>
<td>SCO_AREA - AZOSASALE00</td>
<td>1042733</td>
<td>4</td>
</tr>
<tr>
<td>SPROFIT_CTR - AZOSASALE00</td>
<td>3164</td>
<td>4</td>
</tr>
<tr>
<td>XZMATLPLNT - AZOSASALE00</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>XZRLNSP - AZOSASALE00</td>
<td>2264</td>
<td>4</td>
</tr>
</tbody>
</table>

The original ODS 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 4 rows.

4. Compare the filtering provided by each master data table joined to the ODS.

   In this example, we can see that SCO_AREA provides no filtering (because the count equals the count of local ODS 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 ODS (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.

6.2 SQL efficiency for InfoCube

Now let's take a look at SQL efficiency for InfoCube. We again recommend a method, this time 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:** 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:** We do not 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:** 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.
   a. 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 fact table. Include the
      local predicates on the tables. This shows the effect of filtering of the
      dimension/snowflake against the fact table.

4. Compare the filtering of the each of the dimensions/snowflakes joined to the fact table.
   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's look at an example of using this process to analyze a real-life customer query.

Figure 6-8   Example InfoCube query

Using the InfoCube query example in Figure 6-8, we will use the described method to break
down the query and determine the requirements for indexing and statistics collection to
ensure the optimal access path is chosen. Without first breaking down and simplifying the
query, it is not possible to adequately understand these requirements.

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 will 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 will use the local predicates in a later step to determine where the filtering occurs; see Figure 6-9.

**Step 1 - Join Analysis**

```sql
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 200327 AND 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
```

*Figure 6-9  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.
Figure 6-10  Building the join graph

The first join predicate in Figure 6-10 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.

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.

Figure 6-11  The final join graph
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.

**Step 2 - Associate LocalPredicates**

```
AND "DP"."SID_0RECORDTP" = 0
AND "DP"."SID_0REQUID" <=76419
AND "Z1"."SID" => 200008999
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 "D5"."SID_Z_DOCTYP" = 0
AND "D5"."SID_Z_DOCTYP" <> 200008999
AND "D5"."SID_Z_DOCTYP" = 6
AND NOT "D5"."SID_Z_DOCTYP" = 2000008999
AND "D5"."SID_Z_DOCTYP" <> 6

AND "D6"."SID_Z_FACTWK" BETWEEN 200327 AND 200352
AND "D6"."SID_Z_FACTWK" => 200352
AND "D6"."SID_Z_FACTWK" <> 200327
AND "D6"."SID_Z_FACTWK" = 200327

AND "Z1"."SID" <> 2000008999
AND "D5"."SID_Z_DOCTYP" <> 2000008999
AND NOT "D5"."SID_Z_DOCTYP" = 6
AND "D5"."SID_Z_DOCTYP" = 0
AND "D5"."SID_Z_DOCTYP" <> 6
AND "D5"."SID_Z_DOCTYP" = 6
AND NOT "D5"."SID_Z_DOCTYP" = 6

AND "X1"."OBJVERS" = 'A'
AND "X1"."OBJVERS" <> 2000008999
AND NOT "X1"."OBJVERS" = 1
AND "X1"."OBJVERS" = 'A'
```

**Figure 6-12  Associate local predicates**

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.

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 fact table.
Chapter 6. Tips for SQL efficiency

Figure 6-13 Determining the filtering

The counts in Figure 6-14 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 fact table. These counts will be used to compare all filtering combinations.

Step 4 - Compare Filtering

<table>
<thead>
<tr>
<th>Join Sequence</th>
<th>Count with Filtering</th>
<th>Table CARDF</th>
<th>% Rows Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z1</td>
<td>13</td>
<td>13</td>
<td>100%</td>
</tr>
<tr>
<td>Z1-X2</td>
<td>26,514</td>
<td>353,413</td>
<td>7.50%</td>
</tr>
<tr>
<td>Z1-X2-D1</td>
<td>171,787</td>
<td>1,565,004</td>
<td>10.98%</td>
</tr>
<tr>
<td>Z1-X2-D1-F</td>
<td>1,702,101</td>
<td>10,743,216</td>
<td>15.84%</td>
</tr>
<tr>
<td>X1</td>
<td>258</td>
<td>43,511</td>
<td>0.59%</td>
</tr>
<tr>
<td>X1-D3</td>
<td>281</td>
<td>30,932</td>
<td>0.26%</td>
</tr>
<tr>
<td>X1-D3-F</td>
<td>511,179</td>
<td>10,743,216</td>
<td>4.76%</td>
</tr>
<tr>
<td>DP</td>
<td>432</td>
<td>432</td>
<td>100%</td>
</tr>
<tr>
<td>DP-F</td>
<td>10,743,216</td>
<td>10,743,216</td>
<td>100%</td>
</tr>
<tr>
<td>D5</td>
<td>598</td>
<td>598</td>
<td>100%</td>
</tr>
<tr>
<td>D5-F</td>
<td>10,743,216</td>
<td>10,743,216</td>
<td>100%</td>
</tr>
<tr>
<td>D6</td>
<td>37</td>
<td>423</td>
<td>8.75%</td>
</tr>
<tr>
<td>D6-F</td>
<td>1,317,164</td>
<td>10,743,216</td>
<td>12.26%</td>
</tr>
</tbody>
</table>

Figure 6-14 Comparing filtering

The original fact table contains 10,743,216 rows.
Comparing the filtering provided by each dimension/snowflake joined to the fact, we can see in Figure 6-14 on page 89 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 fact table), 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 fact table, then the most likely scenario is either:

- The dimension/snowflake chosen before the fact table has had its filtering overestimated (so the filtering looks better than it actually is).
- The good dimension/snowflake (not chosen before the fact table) 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 7.3, “Analyzing query performance” on page 114.

6.3 Indexing of ODSs

Query performance on ODS objects can be greatly influenced by having appropriate indexes on the ODS tables. When an ODS 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 ODS object.

This index most likely will not provide optimal performance for the queries being executed against the ODS. 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 ODS phase. Therefore, index design is important and it depends on the queries that will be executed against the ODS. The point is to create indexes that are worth the cost of maintenance.

The objective of indexing on the ODS is to minimize the number of ODS 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's look at the ODS query that is analyzed in 6.1, “SQL efficiency for ODS” on page 80 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:

```sql
WHERE "/BIC/AZOSASALE00"."/BIC/ZMATLPLNT" = "/BIC/XZMATLPLNT"."/BIC/ZMATLPLNT"
AND "/BIC/AZOSASALE00"."PLANT" = "/BIC/XZMATLPLNT"."PLANT"
AND "/BIC/AZOSASALE00"."ACCNTRG" 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:** 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 indexonly access is desired.

The index recommendation for table /BIC/XZMATLPLNT is shown in Table 6-1.

**Table 6-1 Index recommendation**

<table>
<thead>
<tr>
<th>Column</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>S__ZMATONLY</td>
<td>Required to support local filtering</td>
</tr>
<tr>
<td>/BIC/ZMATLPLNT</td>
<td>Optional to support additional joins (see 6.3.1, “Index design for post-ODS access” on page 93)</td>
</tr>
<tr>
<td>PLANT</td>
<td>Optional to support additional joins (see 6.3.1, “Index design for post-ODS access” on page 93)</td>
</tr>
</tbody>
</table>

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 ensures 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"."ACCT_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 BW ODS 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 6-2.

<table>
<thead>
<tr>
<th>Column</th>
<th>Local or Join</th>
<th>Operator</th>
<th>Partition candidate</th>
<th>Index Column Sequence</th>
<th>Mandatory/Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURTYPE</td>
<td>Local</td>
<td>=</td>
<td>Possibly</td>
<td>1</td>
<td>Optional</td>
</tr>
<tr>
<td>ACCT_ASGN</td>
<td>Local</td>
<td>IN</td>
<td>No</td>
<td>2</td>
<td>Optional</td>
</tr>
<tr>
<td>/BIC/ZMATLPLNT</td>
<td>Join</td>
<td>=</td>
<td>No</td>
<td>3</td>
<td>Mandatory</td>
</tr>
<tr>
<td>PLANT</td>
<td>Join</td>
<td>BETWEEN</td>
<td>No</td>
<td>4</td>
<td>Mandatory</td>
</tr>
<tr>
<td>CALMONTH</td>
<td>Local</td>
<td>BETWEEN</td>
<td>Yes</td>
<td>5</td>
<td>Optional</td>
</tr>
</tbody>
</table>

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 V8 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 ODS.

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 recommendation for table /BIC/AZOSASALE00 are shown in Table 6-3.

<table>
<thead>
<tr>
<th>Column</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURTYPE</td>
<td>Only if local predicate always exists in queries, and column is not chosen as a partitioning column</td>
</tr>
<tr>
<td>ACCT_ASGN</td>
<td>Only if local predicate always exists in queries, and index provides good clustering</td>
</tr>
</tbody>
</table>
6.3.1 Index design for post-ODS access

For each table not chosen to provide filtering before the ODS is joined, then 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 ODS example:

```
WHERE "/BIC/AZOSASALE00"."PROFIT_CTR" = "/BI0/SPROFIT_CTR"."PROFIT_CTR"
AND "/BI0/SPROFIT_CTR"."SID" = 31
```

The relevant columns from table /BI0/SPROFIT_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, then it may be sufficient to ensure an index exists on the join column(s) only. This provides the greatest degree of flexibility, especially since the local predicates are not going to limit the number of data pages accessed significantly.

As the master data table becomes larger, and the local predicates provide more filtering, then 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) it is recommended to have 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 /BI0/SPROFIT_CTR is shown Table 6-4.

<table>
<thead>
<tr>
<th>Column</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>/BIC/ZMATLPLNT</td>
<td>To support join filtering</td>
</tr>
<tr>
<td>PLANT</td>
<td>To support join filtering</td>
</tr>
<tr>
<td>CALMONTH</td>
<td>Only if not chosen as partitioning column</td>
</tr>
</tbody>
</table>

6.4 Indexing of InfoCubes

The default indexes for an InfoCube fact table 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 fact table.
The InfoCube query example shown in Figure 6-1 on page 80 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 fact table partitioning is by the time characteristic, then InfoCubes with custom time dimensions may not benefit from partition elimination. This makes custom indexing also a possible requirement.

6.4.1 Fact table indexing

From Figure 6-1 on page 80, the filtering dimensions (in order of filtering) are D3, D6 and D1. A fact table index may be created to support the combined filtering, although you must consider the Cartesian product size of before fact table dimensions.

After filtering, the dimension/snowflake sizes are:

- D3 - 281
- D6 - 37
- D1 - 171,787

A Cartesian product of D3 X D6 X D1 = 1.78 billion, which is not desirable. The size of the D1 snowflake is too large to include in a Cartesian product. Given that this dimension provides least filtering of the three listed, then this can be discarded in choosing the optimal post-fact table access.

A Cartesian product of D3 X D6 = 10,397, which is very desirable given the fact table size of 10 million rows. At most, the fact table will be probed 10,397 times from this Cartesian result (potentially less, if not all combinations of the two dimensions exist in the fact table).

This query may benefit from an index on KEY_ZIC_SFC3, KEY_ZIC_SFC6.

6.4.2 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 fact table. Thus a 10% error in estimation on a 100 row dimension is only 10 rows, but when joined to the fact table (10 million), the error factor becomes 1 million 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, then indexes that contain all filtering columns will 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 fact tables. This should be encouraged for the purposes of efficient data access/join performance, and also for effective statistics collection.
Troubleshooting poor performance

In this chapter we describe how to analyze poor performance in your BW DB2 V8 system on a z/OS platform.

Figure 7-1 on page 98 presents a roadmap for troubleshooting poor performance problems. Use it to determine what tools may be of use to you in determining the cause of your performance problem.
Figure 7-1  Analyzing performance - roadmap
7.1 Troubleshooting general system performance

Figure 7-2  First step: checking system-wide performance

In this section we provide a roadmap to take the reader through some general system-wide health checks. These ensure that there is no general problem that may be the root cause of poor performance in the data load process or the online queries. It is worth taking the time to perform this task before doing more specific in-depth analysis.

It is beyond the scope of this redbook 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 more detailed discussion about these subjects. For example, if you are running on a DB2 datasharing 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 poor performance, you need to follow a roadmap (such as Figure 7-3) or a procedure to capture the relevant information necessary to begin your analysis and follow it to a conclusion.

Figure 7-3  Performance strategy roadmap
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 on to or which server was the batch/ process chain executing on?
- Most importantly, which query or load process is causing problems?

After establishing the 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 7-4 illustrates a roadmap for analysis of server-wide and system-wide performance problems.

![Figure 7-4](image)

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

- 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.

- Check application server health.
  
  If there are no obvious errors on the SAP system logs, begin checking the health of the application server(s):

  - 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.
  - Check the health of the SAP buffers using transaction ST02.

- 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:
- Check the DRDA® network connection by using SAP transaction DB2, selecting the DB2 Performance Tuning tab, then 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 higher, then there is a problem with the link.

You can investigate this further by using the DB2 connect diagnostics button on the DB2 Performance Tuning tab.

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

- 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 joblog 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 on 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/atsmastr.nsf/WebIndex/WP100287

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

### 7.2 Analyzing load performance

In this section we discuss an approach for troubleshooting poor load performance. Figure 7-5 on page 102 shows the load portion of the performance analysis roadmap.
Figure 7-5   Analyzing performance roadmap for load

We assume a scenario where there is no system-wide performance problem, but only an individual load of data with an 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 source (R/3 system, legacy 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 screens 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 on which request ID and which component (source, target, transfer rules or update rules) take the most elapsed times. Note that source could be within the system, for example, PSA or ODS.

We use transaction **ST03N** → **Expert Mode** → **BW System Load** → **Select time window** → **Select Load Data under Analysis View pane** → **Select Aggregation** → **Select Request**. Figure 7-6 shows a sample screen.

![Figure 7-6 BW load data statistics from ST03N](image)

Note: In this sample screen, 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 7-7 on page 104 displays screen of a sampled load. Here you have access to header information for a load job.
Click the Details tab of the RSMO screen to display more information for each component of the load; see Figure 7-8 on page 105.

The load is broken into four smaller data packets for parallel processing. You can see detailed information for request, extraction, transfer, 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 7-9 on page 106.
Select the component of interests to drop to the SM37 joblog to check for more details of its activities. This can be done by right-clicking the Load Data icon to display all jobs. Figure 7-10 on page 107 shows dropping the index for a certain table.
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 7-11 on page 108 and Figure 7-12 on page 109.

![Figure 7-10  Detailed activities of job log from SM37](Image)
Figure 7-11  STAD screen
Figure 7-12  All Statistical Records from STAD

Double-click to go to the Single Record Statistic screen to determine which component takes most of the response time, for example, database request time to total response time. Also, check if there is 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 7-13  Single Statistical Record--work process time
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 specific user and specific job. If the job is not longer running, you can check which table is being processed in the SM37 job of the process chain graphical view.

Alternatively, if temporary trace is permitted for individual analysis, there is an SAP profile parameter stat/tabrec which 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 5 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 ST03N.
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 screen.

To look at table access statistics:
1. Start the workload monitor by calling transaction ST03N.
2. Choose the user mode.
3. In the Workload tree, choose the instance to be analyzed and the time period.
4. From **Analysis Views** → **Table Access Statistics**.

At this point, information from ST03N, RSMO, RSPC, and STAD transactions should provide you with general idea where 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 a legacy system, flat file, or R/3 system. Generally, these are beyond your control, especially a legacy 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 if 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 if indices of datasource tables are okay, and not missing or needed. Having indices on the source tables avoids full table scans.

For ST05 trace capture, make sure you filter only with user-running extraction and make sure no other parallel extraction is in progress. Next, check if runstats are up-to-date for tables 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 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; the memory resources of the R/3 system).

If custom enhancement exits are used for extraction, ABAP Runtime Analysis tool (SE30) can be used to help with debug performance of ABAP coding. It can help isolating 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 if there are enough work processes, especially for loading to PSA and Data Targets in parallel. Parallel processing can compete for resources. Use ST06 transaction to determine if the is bottleneck 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 and disk layout and striping configurations to see if 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, 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 9.6, “Partition rotation and benefits for archiving” on page 164).

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”.

112  Best Practices for SAP Business Information Warehouse on DB2 for z/OS V8
If transfer rules and update rules take the most elapsed time

For this condition, you can use the simulation tool to debug the problem. For example, in the RSMO detailed screen, 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.

If loading to data targets (ODS/InfoCube) takes the most elapsed time

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 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, and a server group for load balancing and so on.

For non-reporting ODS objects, if possible, turn off 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: rsa1 → InfoProvider, then double-check 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 number range requests. You can reduce application server access to the database by buffering 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 if there is index manipulation, and which tables are being processed.

Regarding index manipulation, you need to ask if this is necessary for your particular loading scenario. Generally, index manipulation is effective for the initial full load of large data because of 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, then 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. For more information on runstats, refer to 9.3, “RUNSTATS” on page 153.
7.3 Analyzing query performance

In this section we discuss an approach for troubleshooting poor query performance, following the roadmap laid out in Figure 7-3 on page 99.

If you are convinced that there are no system-wide reasons for the poor performance of a query, then the roadmap will help guide you through an analysis process to determine the cause of the performance problem.

**Important:** You must activate the saving of statistical data for each InfoCube by using the following menu path. If you do not activate statistics, then you will have no data recorded to allow you to analyze query performance.

**BW Admin Workbench → Tools → BW Statistics for Infoproviders → Mark field "OLAP" for the InfoCube.**

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

7.4 Starting the analysis from ST03N

The starting point we use for analyzing query performance is the ST03N Workload Monitor transaction for statistics on individual query executions. Alternatively, you can look at the raw data in table RSDDSTAT using transaction SE16.

From ST03N you can see an overview of all the queries that have been executed over a variety of time periods. The information displayed by ST03N 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 roadmap, we have broken the query response time into three major components:

- Database processing time
- Aggregates required?
- Front-end processing time
Figure 7-17 shows the data provided by ST03N for a number of queries. To reach this display, execute the following actions in ST03N:

1. Select **Expert Mode** from the change user mode tab in the top left corner of the screen.
2. Expand the submenu BW System Load.
3. Select the Relevant time period for your analysis; see Figure 7-17.

4. In the right-hand panel of the screen, change the aggregation mode to Query.
5. In the right-hand panel of the screen, select **All Data**.

You will now see the aggregated response time data for the query you want to analyze. This data will give 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 user(s) is the same, then this will give 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.

![Sample list of executions of a query](image)

You will then get a pop-up screen 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 ‘SDDSTAT Info.

You can identify the execution of the query you are interested in by the UNAME column and the STARTTIME columns; see Figure 7-20.

![Example STARTTIME time in RSDDSTAT](image)

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 YYYYMMDDhhmmssmmmuuun. 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

7.5 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 runtimes 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 ST03N.

7.5.1 QDBSEL/QDBTRANS

If the QDBTIME component of the response time is very high, then examine the fields QDBSEL/QDBTRANS to determine if 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 though 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 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 roadmap to determine the cause of the slow retrieval of records and high database time.

### 7.5.2 RSDU_ANALYZE_TOOL_DB2

Another tool to help you analyze the need for aggregates, or to determine if 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 have 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.

### 7.5.3 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 will help query performance, the updating of these aggregates with data will add time to your data load process, and require 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 redbook 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 → right-click the InfoCube that the query runs against → Maintain Aggregates.** At the top of the screen, Propose → propose (statistics, usually query)

Enter a time range in the pop-up screen. BW will analyze the queries run against this InfoCube, list existing aggregates, and propose new aggregates and the number of queries that would benefit from the proposed aggregate.

### 7.6 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 ST03N.

From the query response time breakdown in ST03N, the column that records the time taken in the network and the front end PC is entitled **Frontend**. In the table **RSDDSTAT**, the column that in 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.

#### 7.6.1 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 → 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 screen.

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.

### 7.7 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, then 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 7-21 on page 119.

![Figure 7-21 DB performance analysis](image)

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.

7.7.1 RSRT Query Monitor
The RSRT transaction can be used to run queries on InfoCubes or ODS.

![RSRT main screen](image)

**Figure 7-22   RSRT main screen**

Type in the query name that you would like to test, or use the F4 key to generate a query
value help box. Press Execute + Debug to display the options that can be selected; refer to
Figure 7-23.
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 7-24 for an example.
Best Practices for SAP Business Information Warehouse on DB2 for z/OS V8

Figure 7-24  RSRT query output

Pressing the green arrow will display the RSDDSTAT statistics; see Figure 7-25.

Figure 7-25  RSRT RSDDSTAT

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

122  Best Practices for SAP Business Information Warehouse on DB2 for z/OS V8
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 pop-up shown in Figure 7-26.

![Figure 7-26   ST04 - Cached Statements Statistics pop-up](image)

Select the appropriate button, enter the STATUID, and press Enter; refer to Figure 7-27.

![Figure 7-27   ST04 cache statistics selected by STATUID](image)

**Note:** If this functionality fails, check OSS note 633075.

Select a statement and press **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.
This example query used 29 seconds DB time, selected 880,091 rows that satisfied the predicates, and then transferred 5 rows to the output screen. 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 7.5, “Determining whether aggregates are required” on page 116 for more detailed explanations.

**Note:** In Figure 7-28, 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 screen, highlighting the line you selected and pressing **Details** → **Avg Execution Statistics**. Then check the Avg Parallel Groups field, as shown in Figure 7-29 on page 125.
To check the access path used by the SQL, from the statement cache statistics screen, select a statement and press Details; refer to Figure 7-30 on page 126.
Figure 7-30  Statement details

Press Explain. To the pop-up Turn on parallelism, answer YES; refer to Figure 7-31 on page 127.
Chapter 7. Troubleshooting poor performance

7.7.2 ST04 - database administration

Use table RSDDSTAT to identify the STATUID for the query and filter ST04 Cached Statement Statistics using this value. This will show 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 which is slow, and where this ratio is high (for example, 100 - 500, or more), then use the ST04 Explain tool.

7.8 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 which 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 if 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

7.8.1 Stored procedures used by Visual Explain

Visual Explain uses sample stored procedures, as listed and discussed here.

**DSNWZP**
This stored procedure is required to support the Browse Subsystems Parameters function. The stored procedure is also used by Service SQL for collection and transmission of subsystem parameter settings to IBM service.

**DSN8EXP (see APAR PQ90022)**
This stored procedure supports the explain with stored procedure option. The purpose of explain with stored procedure is to allow users that should be able to explain SQL against objects which users do not have authority to execute queries against.

For example, a developer may have responsibilities for SQL performance on a payroll application, but may not have SELECT, INSERT, UPDATE, DELETE access to the object. The developer can use the stored procedure to execute the explain against the object. The developer does not require access to the objects, but does require authority to execute the stored procedure.

**DSNUTILS**
Statistics Advisor generates RUNSTATS statements. These RUNSTATS statements can be executed dynamically if the customer has a license to execute RUNSTATS, the DSNUTILS stored procedure is installed, and the user has appropriate authorities to execute the RUNSTATS commands. (Of course, the RUNSTATS input can also be copied into a batch job and executed outside of Statistics Advisor.)
Java™ Universal Driver (see APAR PQ62695)
Installing the Java Universal Driver on DB2 for z/OS provides Visual Explain with stored procedures it uses. Additionally, stored procedure SYSIBM.SQLCAMEMESSAGE is installed, which provides error message descriptions for SQL errors in Visual Explain.

For more information, refer to:

Figure 7-32, Figure 7-33, and Figure 7-34 show examples of DB2 Visual Explain screens.

![Visual Explain Plan Table - all rows displayed](image)

Figure 7-32   Visual Explain Plan Table - all rows displayed
Figure 7-33   Visual Explain - SQL text displayed
7.9 Installation requirements

To install and use DB2 Visual Explain, you need the software product IBM DB2 Universal Database™ for z/OS Visual Explain Version 8 for specific Microsoft Windows products (NT, 2000, XP); refer to its README file for details. Basically, you need to install DB2 connect Personal Edition or Enterprise Edition, customize settings to get client connection to server on z/OS, and then install the DB2 Visual Explain product.

The following example lists the steps and parameter settings to connect to a database on z/OS:

1. Install DB2 Connect™.
   2. **IBM DB2** → **Setup tools** → **Configuration Assistant**.
      a. Add database now.
      b. Manually configure a connection to a database.
      c. Parameter settings
You will need information from z/OS for parameter settings. In order to obtain this, display the DDF. For example, to see this information, enter: `/db8n dis ddf`:

- **RESPONSE=SC04**
  - DSNL080I -DB8N DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
  - DSNL081I STATUS=STARTD
  - DSNL082I LOCATION LUNAME GENERICLU
  - DSNL083I DB8N USIBMSC.SCPDB8N -NONE
  - DSNL084I IPADDR TCPPORT RESPORT
  - DSNL085I 9.12.4.20 38150 38151
  - DSNL086I SQL DOMAIN=wtsc04.itso.ibm.com
  - DSNL086I RESYNC DOMAIN=wtsc04.itso.ibm.com

Additionally, you can get information by issuing the netstat command in an OMVS environment. To get this output, enter: `tso omvs > netstat`:

```
MVS TCP/IP onetstat CS V1R4 TCPIP Name: TCPIP 09:29:23
User Id  Conn     Local Socket           Foreign Socket         State
-------  ----     ------------           --------------         -----
DB8HDIST 00000034 0.0.0.0..38091         0.0.0.0..0             Listen
DB8HDIST 00000032 0.0.0.0..38090         0.0.0.0..0             Listen
DB8NDIST 00009FDE 0.0.0.0..38151         0.0.0.0..0             Listen
DB8NDIST 0000A254 9.12.4.20..38150       9.12.6.15..49816       Establsh
DB8NDIST 0000A23C 9.12.4.20..38150       9.12.6.15..49792       Establsh
DB8NDIST 0000A22A 9.12.4.20..38150       9.12.6.15..49775       Establsh
DB8NDIST 00009FFD 9.12.4.20..38150       9.12.6.15..49145       Establsh
DB8NDIST 00009FFF 9.12.4.20..38150       9.12.6.15..49146       Establsh
```

Note the following:

- Service name is DB8NDIST
- Port number: 38150
- Hostname is wtsc04.itso.ibm.com

At this point, there is sufficient information for parameter settings as provided here:

- **communication protocol:** tcpip
- Database physically resides on a host: Connect directly to the server
- Hostname: wtsc04.itso.ibm.com
- Service name: db8ndist
- Database name: db8n
- Database alias: db8n
- Comment: ....
- Default registration this database for ODBC
  - As system data source: selected
  - Data source name: db8n
  - Optimize for application: none

- **Node options:**
  - Operating system: os/390 or z/OS
  - Instance name: DB2 (issue in DOS prompt: `db2set db2instance`)

- **System options:**
  - System name: wtsc04.itso.ibm.com
  - Host name: wtsc04.itso.ibm.com

- **Security options:**
  - Server authentication
To finish, follow these steps.

1. Test connection:
   a. Default connection type: cli
   b. User ID: itso05 (TSO user ID with DBA authority)

2. DB2CONNECT is successfully connected.

3. Install DB2 Visual Explain.
SAP BW installation and checks under z/OS

Good system management of your SAP BW system starts with the planning for your installation and continues through the day-to-day maintenance tasks.

In this chapter, we provide a brief overview of important documentation concerning installation. For more details, refer to the documents listed in “Related publications” on page 259.

We also discuss organizational considerations to keep in mind when operating an SAP BW system on zSeries:

- Checks concerning SAP BW installation
- SAP BW operations check
- SAP BW regular checks
- Proactive performance measurement
- The SAP automated PTF checker
- The Workload Manager
8.1 Checks concerning SAP BW installation

Before starting the first SAP BW installation in your enterprise, consult the following SAP and IBM documentation:

- IBM DB2 UDB for z/OS: Planning Guide - SAP NetWeaver 6.40
- SAP R/3 on DB2 UDB for OS/390 and z/OS: Connectivity Guide
- SAP Database Administration Guide - SAP NetWeaver 6.40
- SAP Web Application Server ABAP/Java 6.40 on UNIX
- SAP Software on UNIX: OS Dependencies

The primary document to consult is the IBM/SAP Planning Guide. It describes which definitions are necessary in the z/OS subsystems (for example, the first DB2 setup, the first WLM definitions, and the RACF® or UNIX System Services (USS) definitions).

In addition, the IBM/SAP Connectivity Guide outlines important definitions such as the MTU size, which should be identical in all your SAP systems.

The SAP application server can be installed on different hardware; follow the hardware/system software documentation appropriate for your operating system. The installation of the most current SAP BW/R3 plug-ins and the other SAP support packages is very important. Furthermore, all delivered SAP modules, like the SAP kernel, should be current.

It is also important to read the SAP OSS notes in the SAP installation manual that are appropriate for the version of SAP BW you are using. An overview of the most important SAP BW 3.5 OSS notes is provided in Appendix A, “SAP BW 3.5 OSS Notes” on page 221.

You can obtain the SAP OSS notes (you will need an SAP OSS user ID) in the following ways:

- From this site:
  http://service.sap.com/
- By using SAP transaction OSS1, available in SAP R/3
- By using SAP Solution Manager

Restrict your search to SAP component BW-SYS-DB-DB2.

SAP OSS note 81737 is particularly important regarding installation and operation of the R3 z/OS environment because it lists the available SAP APARS that should be installed in your system. This SAP OSS note has expanded through the different z/OS/SAP versions, and there is an SAP PTF checker that can be processed within SAP BW to check the installed PTFs/APARs against SAP OSS note 81737. For more information, see 8.5, “SAP automated PTF checker” on page 140.

8.2 SAP BW operations check

Within the SAP BW environment, responsibility for operational support of the SAP BW IBM mainframe is distributed in different roles among several people. And often the same person fills multiple roles. Therefore, coordination of responsibilities is vital.

In the following sections we describe and explain these roles and their functions.
8.2.1 SAP BW administration roles

- **SAP BW Basis Administrator**
  Responsible for the operating system, the database system, and the SAP BW basis system, the tasks handled by this administrator can be delegated and coordinated.

- **SAP BW InfoCube Administrator**
  Responsible for one or more InfoCubes, the tasks of this administrator include extracting and updating the data and the subsequent processes: roll-up of aggregates, SAP BW statistics, and InfoCube compression. This administrator must coordinate tasks with the SAP BW Basis Administrator.

- **SAP BW InfoCube Designer**
  Responsible for the development of the InfoCubes, this designer can use the SAP BW Business Content or develop InfoCubes to conform to the demands of the different departments. This designer must coordinate with the SAP BW InfoCube Administrator. The SAP BW Basis Administrator should also be involved if, for example, there are performance problems with some InfoCubes.

8.2.2 Data access design roles

- **Master Data Designer**
  Responsible for development and configuration of the master data extractors (plug-in) on the different source systems and for correct processing of the master data, this designer should work closely with the SAP BW InfoCube Designer.

- **Transaction Data Designer**
  Responsible for development and configuration of the Transaction Data extractors (plug-in) on the different source systems and for correct processing of the Transaction Data, this designer should work closely with the SAP BW InfoCube Designer.

- **Query Designer**
  Responsible for developing SAP BW queries/reports, which this designer defines in consultation with staff from the different departments. If there are performance problems, all involved groups must be informed.

- **Standard Query User**
  These users the SAP BW queries/reports from the Query Designer.

- **Power User**
  These users develop their own ad hoc queries. If there are performance problems, all involved parties must be contacted.

As mentioned, coordination of all involved SAP BW groups is vital, including the SAP BW groups and SAP/IBM technical employees (for example, DB2 administrators, z/OS system programmers, AIX administrators, TCP administrators, and so on).

8.2.3 Overview of typical SAP BW operations

Following is a condensed overview of typical SAP BW operations, which must be processed on a regularly scheduled basis.

*Control of InfoPackages*

- Administration of the SAP BW workbench monitor
The SAP BW monitor is used for control of the LOAD processes and problem determination.

- **Status of LOAD jobs**
  For problem determination, the status of the LOAD jobs should be investigated with transaction SM37 in SAP BW and the source systems.

- **Status of LOAD processes**
  For problem determination, the status of the LOAD processes should be investigated with transaction SM50.

**Load of master data**
Prior to LOAD of the transaction data, the master data from all needed InfoObjects must be loaded in SAP BW. There should always be an actual overview from LOAD jobs.

- **Attribute change processing**
  Changes in the hierarchies and attributes should always take place after SAP BW LOAD processing, and no users should be active in the SAP BW system at that time.

- **Problem determination at SAP BW LOAD**
  - Yellow light - recommendations should be available as to which actions are necessary
  - Red light - recommendations should be available as to which actions are necessary

**SAP BW LOAD of transaction data**

- **Problem determination at SAP BW LOAD**
  - Yellow light - recommendations should be available as to which actions are necessary
  - Red light - recommendations should be available as to which actions are necessary

**InfoCube compression**

- After the requests are rolled p into the aggregates, InfoCubes with option partitioning can be compressed.

**Actual RUNSTATS**

- The DB2 RUNSTATS must always be actual.

**Overview of SAP BW process chains**

- There must be an actual overview of regular, scheduled process chains. There should be an actual overview of the processing time of the different process chains.

**Overview of existing InfoCubes and InfoSources**

- There must be an actual overview of all actual/production InfoCubes and InfoSources. There should also be an actual overview of the data volume of all SAP BW data objects.

**Processing of SAP BW queries**

- This should be processed at specified times; for example, SAP BW LOAD should not take place during heavy workloads.

### 8.3 SAP BW regular checks

As previously mentioned, it is very important to coordinate and communicate with the different groups that are needed to run the SAP BW system. Here is an overview of important sources that should be checked regularly.
8.4 Proactive performance measurement

Many customers use SAP CCMS functionality for performance management. Also keep the following points in mind:

- Create your SAP BW data model design as close to your actual data as possible. This influences the design of InfoCubes, PSAs, and ODSs. Discuss the data model design with all involved groups.
- An SAP BW hardware sizing must be done through the hardware vendor before the first installation. Additional hardware sizing checks should be done to handle additional SAP BW functionality or an increase in business intelligence users or data.
- You should have an overview of the processing times of the different SAP BW processes within your company, and you should react immediately if there are extreme differences. Monitor and correct the performance of the SAP BW processes regularly.
- The SAP BW administrator should use regular SAP BW and system-management transactions to analyze SAP BW behavior. Additionally, use DB2 PM or other performance tools to analyze and handle performance problems.
- Use the SAP Early Watch Reports or other SAP services for verification of your SAP BW environment.
- Do not define too many SAP processes. For example, SAP BW does not need an UP2 process, and needs, at most, only two UPD processes.
- Use HTTP Compression to reduce network traffic (refer to SAP note 550669).
- Use client (browser) caching of images to reduce unnecessary network traffic (refer to SAP note 561792).
- Increase the OLAP cache:
  - rsdb/esm/buffersize_KB: Size of exp/imp SHM buffer (default is 4096 K)
  - rsdb/esm/max_objects 2000: Maximum number of objects in the buffer
  - rsdb/esm/large_objects_size 8192 byte: Estimate of the size of the largest object
  - rsdb/esm/mutex_n 0: Number of mutexes in exp/imp SHM buffer
- Set ROIROCPRMS (which can be overwritten by selection criteria in the InfoPackage) according to recommendations in SAP OSS note 417307.
- Analyze your BW objects with transaction RSRV.

Refer to 4.6.1, “Overview” on page 41 for information about further SAP BW transactions.
8.5 SAP automated PTF checker

IBM provides for z/OS program corrections and recommendations with Program Temporary Fixes (PTFs). However, it can be time-consuming to check whether the recommended and needed PTFs are installed and applied.

SAP provides an automated PTF checker, which provides the following information:

- It checks the release and version of the needed software components, such as SAP BW System, SAP Kernel, Z/OS, and DB2 Subsystem.
- It checks the status of the required PTFs within z/OS.
- It provides a list of missing PTFs and functional module IDs (FMIDs).

Prerequisites

The following activities and resources are required before running the PTF checker:

- The customer administers all z/OS software components using System Modification Program Extended (SMP/E). SMP/E records the installed software and the software changes into the Consolidated Software Inventory (CSI). The PTF checker uses the CSI to check whether the components and the fixes are applied.

- SAP rfcoscol, an executable running under UNIX System Services (USS).

- Downloaded SAP note 81737, which includes the PTF and APAR list. This SAP note is formatted for direct use as the input file for the PTD check routine RSCDB2FIX.

This note can be obtained from the server sapservx, or it can be downloaded from the SAP service marketplace:

http://service.sap.com

Note: We recommend using Download Manager to download the required SAP notes 183311 (which describes the PTF Check routine) and 81737 from the SAP service marketplace; refer to Figure 8-1 on page 140.

Figure 8-1 The SAP Download Manager
Additionally, the file fix041019.txt is used to check the applied corrections.

Figure 8-2 illustrates the download of the “sapserv3” server. The newest RSDB2FIX is included in the ...CAR file, and the description of the needed fixes is in the fix041019.txt file.

![Figure 8-2 Download from sapserv3](image)

Whenever the application server is running on z/OS USS, the files must be converted to the z/OS code page. Example 8-1 shows conversion of 81737_notes and the fix041019.txt file.

**Example 8-1 Converting files to IBM-1047 code page**

```
sapuser @ SC04:/u/sapuser>chmod 777 *
sapuser @ SC04:/u/sapuser>mv NOTE_0000081737 NOTE_0000081737.ori
sapuser @ SC04:/u/sapuser>mv fix041019.txt fix041019.txt.ori
sapuser @ SC04:/u/sapuser>iconv -f ISO8859-1 -t IBM-1047 fix041019.txt.ori > fix041019.txt
sapuser @ SC04:/u/sapuser>iconv -f ISO8859-1 -t IBM-1047 NOTE_0000081737.ori > NOTE_0000081737
```

Additionally, the newest version of the RSDB2FIX should be used. If necessary, download and apply the newest version, which is available at this time on sapservx using the “~ftp/general/R3server/abap/note.0183311/SAPK34POSH.CAR” file.

The file SAPK34POSH.CAR must be adapted to the z/OS USS code page using the `iconv` command, and it must be unpacked by calling SAPCAR -xvf. The detailed syntax of this command is described in the Planning Guide and in the OSS notes.
After applying the prerequisites, ABAP RSDB2FIX can be started from transaction SA38; see Figure 8-3.

RSDB2FIX provides the PTF Check routines. As Figure 8-4 on page 142 shows, you must enter the file locations of SAP note 81737 and the Fix Level File. Additionally, you must specify the communication paths to the database server and the application server. Finally, you must define the SMP/E library and zone names.

RSDB2FIX stores and provides the associated logs for each run. The logs include a list of missing FMIDs and PTFs. The logs can be selected directly from the FIXCHECK screen, located in the file and log names pane using the Display Log button.
8.6 Workload Manager

To ensure good performance of your BW system and the correct functioning of DB2 stored procedures, you must correctly configure the Workload Manager (WLM) policy for your z/OS system. The following topics provide a cross-reference to the various sources of documentation on this subject.

8.6.1 WLM changes

With SAP WAS 6.40 and DB2 V8, the method of connecting the SAP application server to the database has changed from ICLI to DRDA. This means that the existing WLM service classes that you have for ICLI must be replaced by services classes for DRDA.

You can also now be much more granular in your workload classification. You have the ability to classify the workload down to the individual user and transaction or ABAP. (Note that we do not recommend this approach, due to the high level of maintenance required to keep this level of detailed classification up to date.)

The service classes for the SAP application server running under USS remain unchanged. For more information concerning WLM, refer to *IBM/SAP Planning Guide*.

8.6.2 Stored procedures changes

Prior to DB2 V8, JCL was submitted via FTP. With V8 this resulted in page conversion problems. To resolve this, the use of FTP has been replaced by Stored Procedures. This ensures that the code page conversion is done automatically by DB2.

The new Stored Procedures fall into several categories:

<table>
<thead>
<tr>
<th>Table 8-1  Categories of stored procedures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Administration</td>
</tr>
<tr>
<td>DSNACCMO Invoke DB2 utilities via DSNUTILS. SAP Transactions) DB13</td>
</tr>
<tr>
<td>DSNACCMD Executes DB2 commands. Also used by the DB Performance Monitor. ST ST04</td>
</tr>
<tr>
<td>DSNACCSS Used to query SSID of DB2 subsystem. ST ST04</td>
</tr>
<tr>
<td>Data Set Manipulation</td>
</tr>
<tr>
<td>DSNACCDS Creates data sets of type PS (physical sequential), PDS (partitioned), PDSE (partitioned extended) and GDS (generation data set)</td>
</tr>
<tr>
<td>DSNACCDE Checks whether non-VSAM data sets or members in a data set exist. ST DB13</td>
</tr>
<tr>
<td>JCL and UNIX Commands</td>
</tr>
<tr>
<td>DSNACCJS Submits JCL for batch execution.</td>
</tr>
<tr>
<td>DSNACCJQ Status of submitted JCL. Status shows if in input queue, executing, or in the output queue.</td>
</tr>
<tr>
<td>DSNACCJF Retrieves the JCL job output from the spool files.</td>
</tr>
<tr>
<td>DSNACCJF Purges or cancels a specific job.</td>
</tr>
</tbody>
</table>

There is now a requirement for several new WLM application environments to be created; refer to SAP note 717935.

For instructions on creating the application environments for DB2 stored procedures in WLM, refer to *SAP Database Administration Guide*. 
This chapter highlights important DB2 best practices and explains how they relate to your SAP BW system. This chapter covers the following:

- General DB2 parameters settings for execution on z/OS
- DB2 buffer pool management
- DB2 RUNSTATS
- DB2 compression
- DB2 partition addition
- DB2 partition rotation
9.1 General z/OS settings and ZPARMS

9.1.1 DB2 and ZPARMS settings

With SAP NetWeaver Version 6.40, there is no difference between the DB2 ZPARM parameters for non-BW systems and BW systems. The SAP publication *SAP on IBM DB2 UDB for z/OS: Database Administration Guide: SAP WEB Application Server* contains the full range of required, recommended, and optional DB2 installation parameters; you can refer to that publication for more detailed information.

To check the DB2 parameter settings from within SAP use transaction:

```
DB2 → Installation Parameters
```

The parameters listed in Table 9-1, Table 9-2 on page 147, and Table 9-3 on page 148 were extracted from *SAP on IBM DB2 UDB for z/OS: Database Administration Guide: SAP WEB Application Server*.

**Table 9-1  DB2 ZPARM parameters with required settings**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCSIDS</td>
<td>819</td>
</tr>
<tr>
<td>SCCSIDS</td>
<td>Can be set to any value where DB2 supports bijective translation between SCCSID and ASCSIDS and that supports the EBCDIC invariant character set (for example, 37 is such a CCSID). <em>Note:</em> A value of 500 for SCCSID is intended for Switzerland and Belgium only. The standard SCCSIDs that are not supported are 290, 420, 905, and 1026. Of course, SCCSID should not be changed once the SAP database is loaded. If you need to change SCCSID, contact IBM DB2 support for assistance.</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>Always use a period (.) and never a comma (,)</td>
</tr>
<tr>
<td>CTHREAD</td>
<td>The setting of CTHREAD is highly dependent on the overall environment size of the SAP systems. CTHREAD should be at least equal to the sum of TSO users and IDBACK.</td>
</tr>
<tr>
<td>IDBACK</td>
<td>The setting of IDBACK is highly dependent on the overall environment size of the SAP system. IDBACK should be at least 30% higher than the sum of the following values: Number of SAP work processes on all SAP Web application servers that run on z/OS and that are connected to this DB2 subsystem. The number of rfcoscol * 2. IDBACK should have a value of at least 20.</td>
</tr>
<tr>
<td>MAXDBAT</td>
<td>The setting of MAXDBAT is highly dependent on the overall environment size of the SAP system. MAXDBAT should be at least 30% higher than the sum of the number of work processes on all SAP application servers that are connected to this DB2 subsystem via DRDA. This calculation excludes SAP application servers on z/OS. During installation, MAXDBAT should have a value of at least 64.</td>
</tr>
<tr>
<td>CONDBAT</td>
<td>Set CONDBAT either to the same or a larger value than MAXDBAT.</td>
</tr>
<tr>
<td>IDTHTOIN</td>
<td>Set IDTHTOIN (idle thread timeout) to 0 to disable timeout processing.</td>
</tr>
</tbody>
</table>
### Table 9-2  DB2 ZPARM parameters with highly recommended settings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMLKUS</td>
<td>2097152</td>
<td>Sets a limit on the number of locks that any individual DB2 thread can hold. Once the limit is reached, the program that accumulated these locks will terminate with sqlcode n904. The maximum value for NUMLKUS is 2097152. This is recommended as an initial, first-cut value. Setting a lower value for NUMLKUS helps you to detect offending programs earlier and is especially recommended for test systems. In most production systems (except the Retail component), a lower value for NUMLKUS is acceptable, but it should not be lower than 500000. For some exceptions and more details, see Locking Considerations in Performance Tuning Considerations.</td>
</tr>
<tr>
<td>IRLMRWT</td>
<td>600</td>
<td>SAP applications are written so as to minimize database lock contentions. However, even when such a contention happens, SAP prefers a long wait rather than a quick conflict resolution that results in a potentially lengthy rollback.</td>
</tr>
<tr>
<td>DEADLOK</td>
<td>5,1</td>
<td>Belongs to the IRLM startup procedure. Lower values increase the likelihood of IRLM latch connections.</td>
</tr>
<tr>
<td>EVALUNC</td>
<td>YES</td>
<td>Setting this parameter to YES reduces lock contention caused by locking of nonqualifying rows.</td>
</tr>
<tr>
<td>CONTSTOR</td>
<td>YES</td>
<td>Caching dynamically prepared statements places a significant demand on the virtual storage in the DB2 address spaces. By turning this parameter on, you ensure that the unused storage is contracted on a regular basis, thus improving storage utilization.</td>
</tr>
<tr>
<td>MONFSIZE</td>
<td>500000</td>
<td>This parameter sets the OP buffer size (in bytes). The buffer is used by the statistics trace started for the RFCOSCOL alert router. By tuning this parameter, you ensure that all exception events can be held in the OP buffer until the RFCOSCOL alert router reads the buffer the next time. If the specified buffer size is not large enough, exception event loss occurs when the buffer fills before the RFCOSCOL alert router can obtain the exceptions.</td>
</tr>
<tr>
<td>NPGTHRSH</td>
<td>10</td>
<td>This parameter is inputted to the access path selection process. If NPAGES for a given table is less than the parameter value and not -1, it is better to have an index access path than a tablespace scan.</td>
</tr>
<tr>
<td>LOBVALA</td>
<td>1000000K</td>
<td>The size of the user storage for LOB values (in KB). The recommended value is 1 GB.</td>
</tr>
<tr>
<td>LOBVALS</td>
<td>50000M</td>
<td>The size of the system storage for LOB values (in MB). The recommended value is 50 GB.</td>
</tr>
<tr>
<td>STARJOIN</td>
<td>2</td>
<td>Enables a join type called star join for accessing data in a star schema.</td>
</tr>
</tbody>
</table>

Chapter 9. DB2  147
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SJTABLES</td>
<td>4</td>
<td>When star join is generally enabled by means of the STARJOIN parameter, it will be considered only if the number of tables is greater than or equal to the value specified in SJTABLES.</td>
</tr>
<tr>
<td>TABLES_JOINED_THRESHOLD</td>
<td>10</td>
<td>For SQL statements that access more than TABLES_JOINED_THRESHOLD tables, statement preparation resources are restricted as governed by the parameters MAX_OPT_CPU, MAX_OPT_ELAP and MAX_OPT_STOR. This is a hidden keyword ZPARM.</td>
</tr>
<tr>
<td>MAX_OPT_CPU</td>
<td>2</td>
<td>Controls statement preparation resources in terms of CPU consumption (specified in seconds). This is a hidden keyword ZPARM.</td>
</tr>
<tr>
<td>MAX_OPT_ELAP</td>
<td>4</td>
<td>Controls statement preparation resources in terms of elapsed time (specified in seconds). This is a hidden keyword ZPARM.</td>
</tr>
<tr>
<td>SJMXPOOL</td>
<td>128 MB</td>
<td>Size of the star join pool. If available real storage allows it, increase the size of the star join pool up to 256 MB.</td>
</tr>
<tr>
<td>MGEXTSZ</td>
<td>YES</td>
<td>Allows DB2 to optimize extent sizing.</td>
</tr>
</tbody>
</table>

**Table 9-3  DB2 ZPARM parameters with recommended settings**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDMSTMTCT</td>
<td>300000</td>
<td>Specifies the size (in KB) of the EDM statement cache.</td>
</tr>
<tr>
<td>MAXKEEPD</td>
<td>8000</td>
<td>Start at a value of 8000, and depending on the virtual storage usage and local cache hit ratio, adjust this value downwards.</td>
</tr>
<tr>
<td>SRTPOOL</td>
<td>28000</td>
<td>Amount of storage (in KB) needed for the sort pool. This value can be increased if there is enough real storage available.</td>
</tr>
<tr>
<td>MAXRBLK</td>
<td>100000</td>
<td>The maximum size for RID List processing should be 100 MB. This value can be increased if there is enough real storage available. This size is in kilobytes.</td>
</tr>
<tr>
<td>CHKFREQ</td>
<td>10</td>
<td>A value of 10 denotes that DB2 takes a checkpoint every 10 minutes.</td>
</tr>
<tr>
<td>RETLWAIT</td>
<td>1</td>
<td>Applies to data sharing only. It is recommended that you wait for retained locks rather than receive a resource unavailable message immediately.</td>
</tr>
<tr>
<td>UTIMOUT</td>
<td>3</td>
<td>The value is reduced from its default (6) due to the relatively large timeout (IRLMRWT) value.</td>
</tr>
</tbody>
</table>
### Parameter | Value | Remarks
--- | --- | ---
URCHKTH | 1 | Some SAP system processes commit very seldom, which can cause a number of problems. In some cases, there is not much that can be done about it because changing the application logic is too difficult. However, in the case of user-written programs, the appropriate changes are often feasible and the programs should be amended by inserting regular commits. This parameter enables you to identify such programs. As the frequency of messages identifying long-running units of recovery is directly proportional to the CHKFREQ value, adjust URCHKTH to avoid too-frequent occurrences.

URLGWTH | 100 | Additional threshold for identifying long-running, noncommitting transactions and reports. The value is in KB.

XLKUPDLT | YES | This parameter slightly reduces the overhead of acquiring locks for some statements.

SMFACCT | 1,2,3 | The Accounting Classes 2 and 3 provide valuable performance indicators. The overhead of Class 2 varies, it can be significant during major data imports (for example, SAP installation, migration or upgrade) and should therefore be deactivated at such times. However, after installation, especially during performance monitoring and tuning, it should be activated to facilitate efficient monitoring.

SYNCVAL | 30 | By specifying SEQ, DB2 prefetch reads data that is cached at the disk level.

SEQCACH | SEQ | By specifying SEQ, DB2 prefetch reads data that is cached at the disk level.

LBACKOUT | NO | Disables postponing backouts for long running units of recovery at restart. During the SAP application server startup, the application server accesses a large number of tables. Therefore, the overall DB2 subsystem should be in a consistent state once the application servers have been started. Setting LBACKOUT to NO ensures this, because it causes DB2 to completely process the backward log during DB2 restart.

If the objects that need to be recovered are known and are not crucial for the functioning of the SAP system, LBACKOUT can also be set to AUTO or YES, which postpones some backward log processing. This reduces DB2 restart time and particularly makes those DB2 objects available earlier that do not need to be recovered.

PARAMDEG | See explanation | Sets the limit to the maximum degree of parallelism with which a query can be executed. SAP systems can use query parallelism in a very controlled manner (for some components and some selected statements only) and can turn it on explicitly (on a statement level). We recommend limiting the maximum degree of parallelism to the number of available CPUs.
As a result of BW utilizing the star schema, a large number of tables are joined, which in turn results in the requirement for large temporary work areas. We recommend determining the total space required and evenly allocating each logical volume so that there is one workfile per volume.

### 9.2 Buffer pool management

With the very large and ever-cheaper main memories that are available on current and upcoming z/Architecture™ machines, it is becoming feasible for customers to configure very large buffer pools to gain significant performance advantages.

The use of 64-bit virtual addressing greatly increases the maximum buffer pool sizes. DB2 V8 is 64-bit exclusive, and therefore always allocates the buffer pools above the 2 GB bar. As of DB2 V8, the terms buffer pool and virtual pool become synonymous.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCLOSET</td>
<td>25</td>
<td>Indicates how many minutes will elapse after a page set or partition has been updated before DB2 converts the page set or partition from read-write to read-only state. This parameter is used in conjunction with PCLOSEN. If the condition for PCLOSEN or PCLOSET is met, the page set or partition is converted from read-write to read-only state. Having DB2 switch an infrequently updated page set from read-write to read-only state can result in performance benefits for recovery, logging, and data sharing processing.</td>
</tr>
<tr>
<td>PCLOSEN</td>
<td>15</td>
<td>Indicates how many checkpoints will be taken after a page set or partition has been updated before DB2 converts the page set or partition from read-write to read-only state. This parameter is used in conjunction with PCLOSET. If the condition for PCLOSEN or PCLOSESET is met, the page set or partition is converted from read-write to read-only state. Having DB2 switch an infrequently updated page set from read-write to read-only state can result in performance benefits for recovery, logging, and data sharing processing.</td>
</tr>
<tr>
<td>MLMT</td>
<td>4</td>
<td>Belongs to the IRLM startup procedure. Specifies the maximum amount of private storage available that IRLM uses for its locks. The unit is GB. Ensure that the IRLM private address space is backed by real storage.</td>
</tr>
<tr>
<td>DDF</td>
<td>AUTO</td>
<td>Automatically starts the DDF address space</td>
</tr>
<tr>
<td>ACCUMACC</td>
<td>5</td>
<td>Accumulates accounting records with identical values for the following identifiers before externalizing the accumulated record: Client user ID, Workstation name, Transaction name</td>
</tr>
<tr>
<td>LRDRTHLD</td>
<td>10</td>
<td>Threshold that controls the identification of long-running readers.</td>
</tr>
</tbody>
</table>
9.2.1 Changes

Buffer pools can now scale to extremely large sizes, constrained only by the physical memory limits of the machine (64-bit allows for 16 exabytes of addressability). The recommendation still stands that buffer pools should not be over-allocated relative to the amount of real storage that is available.

DB2 V8 issues the following warning messages when necessary:

- **DSNB536I**
  
  This message is issued as a warning that the total amount of allocated virtual buffer pool storage for this DB2 subsystem has exceeded the real storage capacity of the z/OS image. The amount of real storage should be at least as much as the total amount of virtual buffer pool storage, and should also be enough to accommodate the EDM pool and DB2 working storage in addition to the virtual buffer pool storage. Paging activity in the buffers is an indication of a performance problem.

- **DSNB610I**
  
  This indicates that a request to increase the size of the buffer pool will exceed twice the amount of real storage, or the normal allocation of a buffer pool not previously used will cause an aggregate size which exceeds the real storage. Either request will then be limited to 8 MB (2000 pages for 4 KB, 1000 pages for 8 KB, 500 pages for 16 KB, and 250 pages for 32 KB).

DB2 limits the total amount of storage that is allocated for buffer pools to twice the amount of real storage size in the system image. DB2 V8 increases the maximum buffer pool sizes to the limit of the architecture; however, the effective maximum is given by the real storage available:

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

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

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

DB2 V8 maintains the old V7 virtual pool and hiperpool definitions as they were at the time of migration to be used in case of fallback, and it adds new definitions of buffer pools for the catalog. For newly installed V8 subsystems, as in prior releases, DB2 initially uses the buffer pool sizes that were specified during the installation process. Thereafter, the buffer pool attributes can be changed through the ALTER BUFFERPOOL command, and they are stored in the BSDS.

**SAP BW recommendation**

DB2 assigns tablespaces and indexes to default buffer pools, as defined in the DB2 startup parameters; see Table 9-4 on page 152.
Table 9-4  Default buffer pool assignments

<table>
<thead>
<tr>
<th>Buffer pool</th>
<th>VPSIZE</th>
<th>VPSEQT</th>
<th>VPPSEQT</th>
<th>DWQT</th>
<th>VDWQT</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP0 (catalog)</td>
<td>2000</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>10</td>
</tr>
<tr>
<td>BP1 (work files)</td>
<td>40000</td>
<td>100</td>
<td>50</td>
<td>50</td>
<td>10</td>
</tr>
<tr>
<td>BP2 (4K tablespaces)</td>
<td>15000</td>
<td>50</td>
<td>50</td>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>BP3 (index spaces)</td>
<td>20000</td>
<td>40</td>
<td>50</td>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>BP6 dim tablespaces (DB2_DIM_BPOOL_DATA)</td>
<td>10000</td>
<td>30</td>
<td>50</td>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>BP7 dim indexespaces (DB2_DIM_BPOOL_INDEX)</td>
<td>15000</td>
<td>30</td>
<td>50</td>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>BP8 Fact tablespaces (DB2_FACT_BPOOL_DATA)</td>
<td>12000</td>
<td>50</td>
<td>50</td>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>BP9 Fact indexespaces (DB2_FACT_BPOOL_INDEX)</td>
<td>12000</td>
<td>50</td>
<td>50</td>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>BP10 Aggr tablespaces (DB2_AGGR_BPOOL_DATA)</td>
<td>20000</td>
<td>50</td>
<td>50</td>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>BP11 Aggr indexespaces (DB2_AGGR_BPOOL_INDEX)</td>
<td>20000</td>
<td>50</td>
<td>50</td>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>BP40 (LOB tablespaces)</td>
<td>4000</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>10</td>
</tr>
<tr>
<td>BP8KB0 (8K tablespaces)</td>
<td>3000</td>
<td>50</td>
<td>50</td>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>BP16KB0 (16K tablespaces)</td>
<td>3000</td>
<td>50</td>
<td>50</td>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>BP32K (32K tablespaces)</td>
<td>5500</td>
<td>50</td>
<td>50</td>
<td>30</td>
<td>5</td>
</tr>
</tbody>
</table>

The following parameters determine the buffer pools that are employed by the page sets of start schema fact and dimension tables:

- DB2_DIM_BPOOL_DATA
- DIM_BPOOL_INDEX
- DB2_FACT_BPOOL_DATA
- DB2_FACT_BPOOL_INDEX

They are maintained in table RSADMIN. Based on SAP OSS note, 536074, we recommend that “fact” and “dimension” tables be defined to their own bufferpools. Refer to note 536074 for recommendations on achieving this objective.

**Page fixing**

DB2 V8 adds the new optional keyword **PGFIX** to the `ALTER BUFFERPOOL` command. If it is set to YES, then the buffer pool is long-term fixed in real storage. Page buffers are fixed when they are first employed since the buffer pool is allocated or expanded. By saving CPU cycles, this results in a performance improvement.

In case PGFIX is set to NO, which is the default value, z/OS can page out buffer pool pages. Page buffer are fixed and unfixed in real storage across each I/O. To avoid situations where long-term fixed buffer pools exceed the real storage capacity of the z/OS image, DB2 ensures that these buffer pools do not exceed 80% of this capacity.
Following are recommendations for the use of PGFIX:

- All “FACT” table buffer pools (or any buffer pools with low hit ratios).
- Ensure 100% of the bufferpool in question is backed by real storage.
- Use PGFIX to help compensate for CPU performance regression.

The use of PGFIX reduces the CPU overhead of page fixing and page freeing. Understand that page fixing benefit is inversely proportional to the buffer hit ratio, thus the better the hit ratio, the worse the benefit.

Example 9-1  PGFIX command syntax

```
ALTER BUFFERPOOL(BP10) PGFIX(YES)
```

9.3 RUNSTATS

If appropriate RUNSTATS statistics are not gathered, the DB2 optimizer may not be able to choose a good access path. It is possible that better statistical data could allow DB2 to make a more informed decision.

One frequently occurring issue for the optimizer is when multiple columns in a table exhibit correlation, that is, the value in one column exactly or closely follows or determines the likely values in another column. If this correlation can be determined by analyzing the data content of a table, the optimizer can use this information in making intelligent choices of how to access data to service certain SQL requests. A good example of this phenomenon is where a ZIP code effectively determines the State, even though the State also exists as another discrete column.

As of DB2 V7, REAL-TIME STATISTICS has been available. This feature enables the interrogation of DB2 information about the status of its objects and includes hints on when to execute RUNSTATS, DB2 REORGs, and so on.

Stored Procedure DSNACCOR is a sample procedure on how to extract the information from the various DB2 tables. For further information on DSNACCOR and REAL-TIME STATISTICS, refer to the IBM publication for DB2 V8 Utility Guide and Reference.

**Note:** As previously mentioned, SAP administrative processes and the SAP support package level can greatly reduce the CPU and time impact of RUNSTATS.

9.3.1 Highlights of RUNSTATS improvements

When there is an asymmetrical distribution of data, not having distribution statistics on non-leading indexed columns and/or non-indexed columns can cause DB2 to make sub-optimal table join order and table join method decisions. With DB2 V8, equivalent and standard function is provided by RUNSTATS. It greatly improves the accuracy of the filter factors determined by DB2. More accurate filter factor computations should lead to better optimization choices. Thus the query performance improves with better filter factor information in the DB2 catalog. This enhancement is implemented only in RUNSTATS, and not in inline statistics. The RUNSTATS enhancement provides the following functionalities:

- Frequency value distributions for non-indexed columns or groups of columns
- Cardinality values for groups of non-indexed columns
- LEAST frequently occurring values, along with MOST for both index and non-indexed column distributions
9.3.2 Collecting cardinality and distribution statistics

To enable the collection of cardinality and distribution statistics on non-indexed columns, a new dist-spec block is introduced. New keywords COLGROUP, MOST, LEAST, and BOTH are introduced in this block. In addition, the existing keywords FREQVAL and COUNT are also used.

Cardinality and distribution statistics are collected only on the columns explicitly specified. Cardinality and distribution statistics are not collected if you specify COLUMN ALL.

For an SAP system, usually prior to proceeding down this path, you will have one or more specific problems you are addressing, and probably have some application knowledge that determines those columns about which you are attempting to influence the optimizer. Because of the huge number of objects in an SAP system, it is impractical to collect distribution statistics on all columns for table spaces and indexes, as the total number of columns is prohibitively large.

**Tip:** When searching for more clues in addressing specific performance problems around specific SAP tables, it may be possible to run RUNSTATS, select a number of columns in the COLGROUP specification (a reasonable value such as 20 for COUNT), and additionally specify UPDATE NO. The statistics produced in report form may lead to a better understanding of the problem and of the DB2 rationale for its choices.

**COLGROUP**
When the keyword COLGROUP is specified, the set of columns specified with the COLUMN keyword is treated as a group. The cardinality values are collected on the column group. When COLGROUP is not specified, the columns are treated as individual columns and cardinality values are collected on the columns specified in the list.

**FREQVAL**
This keyword controls the collection of frequent value statistics. These are collected either on the column group or on individual columns, depending on whether COLGROUP is specified or not. If FREQVAL is specified, then it must be followed by the keyword COUNT.

**COUNT integer**
COUNT indicates the number of frequent values to be collected. Specifying an integer value of 20 means to collect 20 frequent values for the specified columns. No default value is assumed for COUNT. Although the syntax might suggest that the default value is 10, you have to explicitly specify COUNT 10. This can be optionally followed by the keyword MOST (which is the default), LEAST, or BOTH.

We explain these keywords as follows:

**MOST**
The most frequent values are collected when the keyword MOST is specified.

**LEAST**
The least frequent values are collected when the keyword LEAST is specified.

**BOTH**
The most frequent values and the least frequent values are collected when the keyword BOTH is specified.
In Example 9-2, running RUNSTATS using the COLGROUP parameter results in the cardinality of the column group for columns (KEY_DB2C_PP, KEY_DB2C_PT, KEY_DB2C PU) for table /BIC/EDB2C_P to be collected.

**Example 9-2  RUNSTATS using COLGROUP parameter**

```
RUNSTATS TABLESPACE FA40X4X.EDB2CXU
   TABLE(SAPDB1:/BIC/EDB2C_P)
   COLGROUP(  
KEY_DB2C_PP, KEY_DB2C_PT, KEY_DB2C PU)
```

**REPORT NO UPDATE NONE option**

In certain situations, such as catalog statistics having been manually updated, or a new index created on a table, it is desired to allow SQL statements to use another access path. In general, most situations are the end product of investigations of performance problems, and quite often it will be desired to solve the problem as non-disruptively as possible.

As SAP exclusively uses dynamic SQL for application-related calls, the cached versions of the prepared statements will continue to take effect. Usage of the RUNSTATS utility in this way elegantly allows all dynamic statement cache (DSC) statements referencing the object in question to be invalidated. This can be an improvement to other techniques such as ALTER to AUDIT NONE.

Example 9-3 illustrates an example of invoking the RUNSTATS utility specifically to invalidate DSC statements that use table SAPDB1:/BIC/EDB2C_P.

**Example 9-3  RUNSTATS to invalidate the DSC statements**

```
RUNSTATS TABLESPACE FA40X4X.EDB2CXU
   TABLE(SAPDB1:/BIC/EDB2C_P)
   UPDATE NONE
```

### 9.3.3 SAP BW-specific considerations

We recommend running DB2 RUNSTATS for the BW specific tables as discussed here, and not outside of DB2. This will ensure that the statistics collected conform to SAP BW recommendations.

**Use of cache invalidation**

SAP exploits this new way to invalidate statements from DSC in transaction code ST04. In the cached statements statistics part of ST04, there is a button available that invalidates a selected SQL statement. When pressing the button, SAP calls the stored procedure DSNUTILS to invoke the RUNSTATS utility with options REPORT NO UPDATE NONE on the table space of the first table that is referenced in the selected statement.

**Note:** RUNSTATS will always invalidate dynamic cache, but the above option will not collect stats, will not update catalog tables, and will not generate a report.

```
ST04 → Cached Statement Statistics → “enter” → Invalidate SQL
```

**Frequency of RUNSTATS**

After loading of data to InfoCubes, updating of statistics is generally done on the INFOCUBE and all the related tables. The standard practice is to use the SAP process chain to update the various DB2 catalog tables.

For many scenarios, it is sufficient if the database statistics are only calculated for a subsection of these tables. For example, in general you just need to update the database
statistics for the F fact table and the dimension tables after loading an infopackage into an InfoCube. This may allow you to significantly improve performance. A method of achieving this objective is covered in OSS note 778437. Also refer to OSS notes 498218 and 688133, as well as to others as appropriate for additional information concerning RUNSTATS options.

Remember that SAP BW requires statistics to be updated on an event basis, and not as would normally occur (that is, on a time basis). This relates to BW-specific tables, whereas traditional SAP tables would still be done on a time basis. The SAPD8LT001 table is an example of non-BW table.

We recommend that you execute RUNSTATS after the following events have taken place:

- Load of F fact table
- Compression of F to E fact table
- DB2 compression
- Partitioning of BW objects

Setting up RUNSTATS as part of the process chain is illustrated in Figure 9-1.

![Figure 9-1 RUNSTATS as part of a process chain](image)

**Methods of executing RUNSTATS**

- As mentioned previously, the recommendation is to execute RUNSTATS as part of the process chain of the LOAD/COMPRESSION of BW objects.
If SAP note 795435 is applied to the BW system, the SAP transaction DB13 can be used to achieve the statistical information required by the DB2 optimizer. Using transaction SE38, you can find two DB2 collection programs: RSDB2_COLLECT_HOURLY and RSADB2_COLLECT_DAILY. These two programs interrogate DB2 and establish which DB2 object require RUNSTATS.

Using DB13, the option of running selective RUNSTATS is available, and will only execute against those objects listed by RSDB2_COLLECT routines. We recommend that this process be automated via the schedule for a time period when the system has spare capacity.

Figure 9-2 shows the DB2 program return via SAP transaction SE38.

### Figure 9-2  List of DB2 statistics collection programs

<table>
<thead>
<tr>
<th>Program Name</th>
<th>Report Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSDB2_COLLECT_DAILY</td>
<td>Program RSDB2_COLLECT_DAILY</td>
</tr>
<tr>
<td>RSDB2_COLLECT_HOURLY</td>
<td>Program RSDB2_COLLECT_HOURLY</td>
</tr>
</tbody>
</table>

Figure 9-3 shows the screen return from SAP transaction DB13. The option of RUNSTATS on obj. needing new statistics is selected. The sequence to obtain this screen is:

```
DB13 \rightarrow Select the current day \rightarrow Enter
```

### Figure 9-3  RUNSTATS selection for DB2 objects that require updating

#### 9.4  DB2 compression

DB2 compresses the data portion of the record one at a time, and does not compress the prefix. There is an 8-byte overhead on each record that is compressed. If the resultant row does not use less space than the original, then the row is question will not be compressed.
In addition, DB2 requires a compression dictionary per data set, which uses 64 KB of storage. However, as of DB2 V8, this can now reside above the 2 GB bar and it thus provides storage relief.

As this compression is hardware-assisted, the number of physical I/Os is reduced because DB2 data is copied into the buffer pools in compressed format. As a result, more data is returned to the buffer pools. Selected rows are decompressed before they are passed to the requesting application. Similarly, updated and inserted data rows are contained in pages that are compressed in the buffer pool.

DB2 compression provides the following advantages:

- Less space is used in the buffer pool.
- There is an improvement in the buffer pool hit ratio.
- There is reduced DB2 storage and its associated cost.
- Central storage is freed up for other tasks.
- Data transfer rates are improved for tablespace scans.
- With backup/recovery, log sizes and backup file sizes are potentially reduced (reduced tape and DASD requirements).

For example, a compression ratio of 50% for a compressed tablespace will double, on average, the number of rows in each page in the associated virtual buffer pool.

To determine whether DB2 compression would reduce the amount of disk space used, use the IBM utility DSN1COMP, which will show the compression ratio that will be achieved after compression has been activated. Example 9-4 shows sample JCL.

Example 9-4  JCL required for DSN1COMP

```
//jobname JOB acct information
//COMPEST EXEC PGM=DSN1COMP,PARM='REORG,ROWLIMIT(50000)' 
//STEPLIB DD DSN=prefix.SDSNLOAD,DISP=SHR  
//SYSPRINT DD SYSOUT=A
//SYSABEND DD SYSOUT=A
//SYSUT1 DD DSN=DSN=DB8I0.DSNDBD.OD40XNI4.B000027.I0001.A004,DISP=SHR
```

Example 9-5 shows the output after running the DSN1COMP utility using a sample of 50 000 records. The results show an estimated compression ratio of 86%.

Example 9-5  Output after running DSN1COMP

```
DSN1999I START OF DSN1COMP FOR JOB DB8NCOMP
DSN1981I INPUT DNAME = DBI0.DSNDBD.OD40XNI4.B000027.I0001.A004 , VSAM
DSN1944I DSN1COMP INPUT PARAMETERS
   4,096 DICTIONARY SIZE USED
   0 FREEPAGE VALUE USED
   5 PCTFREE VALUE USED
50,000 ROWLIMIT REQUESTED
   ESTIMATE BASED ON DB2 REORG METHOD
DSN1940I DSN1COMP COMPRESSION REPORT
   30,043 KB WITHOUT COMPRESSION
   4,154 KB WITH COMPRESSION
68 PERCENT OF THE BYTES WOULD BE SAVED
   494 ROWS SCANNED TO BUILD DICTIONARY
50,000 ROWS SCANNED TO PROVIDE COMPRESSION ESTIMATE
   4,096 DICTIONARY ENTRIES
618 BYTES FOR AVERAGE UNCOMPRESSED ROW LENGTH
```
The DB2 REORG utility needs to be executed against the tablespace to effect the compression. A compression dictionary is built during the REORG. Bearing in mind that this dictionary can only be built once there is data in the DB2 tablespace, we recommend that a small portion of the intended data be loaded.

This data should be representative of the full spectrum of the data. Then execute DSN1COMP to establish the compression ratio. Finally, execute the REORG to build the compression dictionary. After this is complete, the balance of the data can be loaded.

To effect the compression on a DB2 tablespace, use the DB2 syntax:

```
ALTER TABLESPACE dbname.tsname COMPRESS YES
```

For an E fact table, the PCTFREE parameter can be set to 10% if updates are frequent (overlapping packets compressing). Otherwise, set it to 0%.

In our case, a series of queries were executed, before and after the compression was done. The results are displayed in Table 9-5 and Table 9-6 on page 160.

The QUERY TIME shown is the value returned by QTIMEDB from SAP table RSDDSTAT, which shows the time that a SQL command took from leaving the application server, and on returning to the application server. This includes time taken to navigate across the network.

<table>
<thead>
<tr>
<th>Number of rows</th>
<th>Query time</th>
<th>% query time saved</th>
<th>Combined disk space used</th>
<th>% disk space saved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Results before SAP/DB2 compression has been done</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUERY 01 3 200 000 185.6758 0 4 172 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUERY 02 3 200 000 11.9023 0 4 172 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Results after DB2 RUNSTATS. No SAP/DB2 compression</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUERY 01 3 200 000 87.9023 + 53 4 172 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUERY 02 3 200 000 9.2188 + 22 4 172 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Results after SAP compression and DB2 RUNSTATS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUERY 01 2 000 000 101.0313 + 45 4 777 - 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUERY 02 2 000 000 4.9531 + 58 4 777 - 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Results after SAP compression, DB2 REORG and DB2 RUNSTATS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUERY 01 2 000 000 78.2109 + 57 4 777 - 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUERY 02 2 000 000 4.2226 + 64 4 777 - 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Results after SAP compression, DB2 compression, DB2 REORG and DB2 RUNSTATS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUERY 01 2 000 000 75.5976 + 60 2 047 + 51</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUERY 02 2 000 000 4.5429 + 61 2 047 + 51</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 9-6 Results from non-partitioned E fact table

<table>
<thead>
<tr>
<th>Query</th>
<th>Number of rows</th>
<th>Query time</th>
<th>% query time saved</th>
<th>Combined disk space used</th>
<th>% disk space saved</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUERY 01</td>
<td>3 200 000</td>
<td>88.8789</td>
<td>0</td>
<td>3 353</td>
<td>0</td>
</tr>
<tr>
<td>QUERY 02</td>
<td>3 200 000</td>
<td>1.9531</td>
<td>0</td>
<td>3 353</td>
<td>0</td>
</tr>
<tr>
<td>QUERY 01</td>
<td>3 200 000</td>
<td>88.4101</td>
<td>+ 0.5</td>
<td>3 353</td>
<td>0</td>
</tr>
<tr>
<td>QUERY 02</td>
<td>3 200 000</td>
<td>1.7461</td>
<td>+ 10</td>
<td>3 353</td>
<td>0</td>
</tr>
<tr>
<td>QUERY 01</td>
<td>2 000 000</td>
<td>88.5898</td>
<td>+ 0.3</td>
<td>2 849</td>
<td>+ 15</td>
</tr>
<tr>
<td>QUERY 02</td>
<td>2 000 000</td>
<td>2.7578</td>
<td>- 41</td>
<td>2 849</td>
<td>+ 15</td>
</tr>
<tr>
<td>QUERY 01</td>
<td>2 000 000</td>
<td>84.3046</td>
<td>+ 5</td>
<td>2 849</td>
<td>+ 15</td>
</tr>
<tr>
<td>QUERY 02</td>
<td>2 000 000</td>
<td>1.4492</td>
<td>+ 25</td>
<td>2 849</td>
<td>+ 15</td>
</tr>
<tr>
<td>QUERY 01</td>
<td>2 000 000</td>
<td>91.1445</td>
<td>- 2.5</td>
<td>1 533</td>
<td>+ 54</td>
</tr>
<tr>
<td>QUERY 02</td>
<td>2 000 000</td>
<td>1.4843</td>
<td>+ 24</td>
<td>1 533</td>
<td>+ 54</td>
</tr>
</tbody>
</table>

From these results, you can see that QUERY TIME and DISK USAGE savings can be achieved by implementing the various SAP/DB2 options that are available. The option of ensuring that DB2 RUNSTATS has been completed successfully already shows gains in QUERY TIME.

SAP has the ability to enable DB2 compression. In the RSADMIN table, there are two parameters:

- **DB2_FACT_COMPRESS**
- **DB2_AGGR_COMPRESS**

By setting these values to YES, the fact tables and/or the aggregate table are created with the DB2 'COMPRESS YES' option. Remember that the PSA tables are COMPRESSED, by default.

There is also the option of enabling DB2 compression at the table level.

- **DB2_COMPRESS_`table name`**

Example 9-6 on page 161 shows the SAP DB2 compression parameter.
Example 9-6 Using the SAP DB2 compression parameter

Set DB2_AGGR_COMPRESS to YES
Set DB2_AFAC'T_COMPRESS TO YES
SET DB2_COMPRESS_'BIC/F100321' to NO
All fact tables for all aggregates, excluding the F fact table for 100321, will be created with the COMPRESS YES option

Even though the DB2 compress parameter has been set to YES, a DB2 REORG is required to build the compression dictionary before compression can be utilized; see Figure 9-4 on page 158.

9.5 Partition addition

DB2 V8 has introduced the ability of adding extra partitions to an existing partitioned table. However, this does not mean that a non-partitioned table can be automatically converted to a partitioned table. Instead, it gives you the option of initially creating a few partitions and, as the need arises, to increase the number of partitions to the maximum as permitted by the original CREATE TABLESPACE statement. This means space savings until the actual disk is required. The existing PRIQTY and SECQTY values of the previous logical partition will be used.
**Note:** If the table uses index-controlled partitioning, it will be converted to use table-controlled partitioning.

For SAP BW objects partitioned on date, the additional partitions can be added as further periods are added to the table until data is ARCHIVED off, DELETED, or the maximum numbers of partitions is reached. For the ARCHIVE/DELETE options, there is a method to reuse the partition that was cleared; refer to 9.6, “Partition rotation and benefits for archiving” on page 164 for more information.

The steps necessary to add a partition involve using the new DB2 syntax of `ALTER PARTITION...ENDING AT ...` (to establish a new end value for the high key), and then using the statement `ADD PARTITION END AT ...` (to create the boundary for the new partition that has been added).

The maximum partitions that can be created, as limited by the original CREATE TABLESPACE statement are listed below. If DSSIZE was originally specified, the partition limits are listed in Table 9-7.

**Table 9-7  Partition limits for DSSIZE**

<table>
<thead>
<tr>
<th>DSSIZE</th>
<th>Page size 4 KB</th>
<th>Page size 8 KB</th>
<th>Page size 16 KB</th>
<th>Page size 32 KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 GB-4GB</td>
<td>4 096</td>
<td>4 096</td>
<td>4 096</td>
<td>4 096</td>
</tr>
<tr>
<td>8 GB</td>
<td>2 048</td>
<td>4 096</td>
<td>4 096</td>
<td>4 096</td>
</tr>
<tr>
<td>16 GB</td>
<td>1 024</td>
<td>2 048</td>
<td>4 096</td>
<td>4 096</td>
</tr>
<tr>
<td>32 GB</td>
<td>512</td>
<td>1 024</td>
<td>2 048</td>
<td>4 096</td>
</tr>
<tr>
<td>64 GB</td>
<td>256</td>
<td>512</td>
<td>1 024</td>
<td>2 048</td>
</tr>
</tbody>
</table>

If LARGE was originally specified, the partition limits are listed in Table 9-8.

**Table 9-8  Partition limits for DSSIZE = 0**

<table>
<thead>
<tr>
<th>Type of tablespace</th>
<th>Number of existing partitions</th>
<th>Maximum partitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-LARGE</td>
<td>1 to 16</td>
<td>16</td>
</tr>
<tr>
<td>Non-LARGE</td>
<td>17 to 32</td>
<td>32</td>
</tr>
<tr>
<td>Non-LARGE</td>
<td>33 to 64</td>
<td>64</td>
</tr>
<tr>
<td>LARGE</td>
<td>N/A</td>
<td>4 096</td>
</tr>
</tbody>
</table>

We recommend that the ALTER PARTITION ‘part number’ ENDING AT ‘constant’ be issued prior to doing the ADD PARTITION ENDING AT ‘constant’. Refer to DB2 Universal Database for z/OS SQL Reference V8 for full details and options available.

Following is the output from the DISPLAY command before an ALTER TABLE ADD PARTITION command is issued. The number of VSAM datasets is also shown.

```
DSNT360I -DB8N ******************************************************
DSNT361I -DB8N * DISPLAY DATABASE SUMMARY
*       GLOBAL
DSNT360I -DB8N ******************************************************
DSNT362I -DB8N DATABASE = DSN8DB1A STATUS = RW
```
Chapter 9. DB2

DBD LENGTH = 20180

NAME TYPE PART STATUS PHYERRLO PHYERRHI CATALOG PIECE
-------- ---- ----- ----------------- -------- -------- -------- ----- 
DSN8S81E TS 0001 RW -THRU 0004
XEMP1 IX 0001 RW -THRU 0004

****** DISPLAY OF DATABASE DSN8D81A ENDED ***********************

Menu Options View Utilities Compilers Help

Command ==>  Scroll ==> CSR

Command - Enter "/" to select action  Message  Volume

<table>
<thead>
<tr>
<th>Command</th>
<th>Message</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB8NU.DSNDBC.DSN8D81A.DSN8S81E.I0001.A001</td>
<td><em>VSAM</em></td>
<td></td>
</tr>
<tr>
<td>DB8NU.DSNDBC.DSN8D81A.DSN8S81E.I0001.A002</td>
<td><em>VSAM</em></td>
<td></td>
</tr>
<tr>
<td>DB8NU.DSNDBC.DSN8D81A.DSN8S81E.I0001.A003</td>
<td><em>VSAM</em></td>
<td></td>
</tr>
<tr>
<td>DB8NU.DSNDBC.DSN8D81A.DSN8S81E.I0001.A004</td>
<td><em>VSAM</em></td>
<td></td>
</tr>
<tr>
<td>DB8NU.DSNDBC.DSN8D81A.DSN8S81E.I0001.A001</td>
<td>SAPBYX</td>
<td></td>
</tr>
<tr>
<td>DB8NU.DSNDBC.DSN8D81A.DSN8S81E.I0001.A002</td>
<td>SAPBYR</td>
<td></td>
</tr>
<tr>
<td>DB8NU.DSNDBC.DSN8D81A.DSN8S81E.I0001.A003</td>
<td>SAPBYU</td>
<td></td>
</tr>
<tr>
<td>DB8NU.DSNDBC.DSN8D81A.DSN8S81E.I0001.A004</td>
<td>SAPBYQ</td>
<td></td>
</tr>
<tr>
<td>DB8NU.DSNDBC.DSN8D81A.DSN8S81E.I0001.A005</td>
<td>SAPBYU</td>
<td></td>
</tr>
</tbody>
</table>

********************************************************************* End of Data Set list *********************************************************************

The below output shows the commands issued and subsequent output from the ALTER commands.

SDSF OUTPUT DISPLAY D88NTEJ1 J0825368 DSID 106 LINE 3       COLUMNS 02-81

Command Input ==>  Scroll ==> CSR

<table>
<thead>
<tr>
<th>Command</th>
<th>Message</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTER TABLE DSN8810.EMP ALTER PART 4 VALUES('499999')</td>
<td>SQL WARNING DURING EXECUTE IMMEDIATE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSNT404I SQLCODE = 20272, WARNING: THE LAST PARTITION'S LIMIT KEY VALUE OF '999 POSSIBLE VALUE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSNT418I SQLSTATE = 01666 SQLSTATE RETURN CODE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSNT415I SQLERRP = DSNXISB6 SQL PROCEDURE DETECTING ERROR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSNT416I SQLERRD = 42 0 0 -1 0 0 SQL DIAGNOSTIC INFORMATION</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSNT416I SQLERRD = X'0000002A' X'00000000' X'00000000' X'FFFFFFFF' X'0000 INFORMATION</td>
<td></td>
</tr>
<tr>
<td>READY</td>
<td>DSN SYSTEM(DB8N)</td>
<td></td>
</tr>
<tr>
<td>DSN</td>
<td>-STOP DB(DSN8D81A) SPACENAM(*)</td>
<td></td>
</tr>
<tr>
<td>DSN1022I - DB8N DSNTDIS 'STOP DATABASE' NORMAL COMPLETION</td>
<td>DSN RUN PROGRAM(DSNTIAD) PLAN(DSNTI81) LIB('DB8NU.RUNLIB.LOAD')</td>
<td></td>
</tr>
<tr>
<td>DSN</td>
<td>-START DB(DSN8D81A) SPACENAM(*)</td>
<td></td>
</tr>
<tr>
<td>DSN1022I - DB8N DSNTDIS 'START DATABASE' NORMAL COMPLETION</td>
<td>DSN DSNICTIAD - SAMPLE DYNAMIC SQL PROGRAM 2.0</td>
<td></td>
</tr>
<tr>
<td>END</td>
<td>DSN SYSTEM(DB8N)</td>
<td></td>
</tr>
</tbody>
</table>

ALTER TABLE DSN8810.EMP ADD PART VALUES('999999')

SQLCODE = 000, SUCCESSFUL EXECUTION

****************************************************************************** BOTTOM OF DATA ******************************************************************************
Here follows is the output after the ALTER, which indicates that PART 4 is in REORG PENDING status. Running the DB2 utility, REORG, will reset the REORP status. There are now 5 partitions as can be seen below the DISPLAY command output.

```
DSN9022I -DB8N DSNTDDIS 'DISPLAY DATABASE' NORMAL COMPLETION
***
```

```
DSNT360I -DB8N ***********************************
DSNT361I -DB8N * DISPLAY DATABASE SUMMARY
    * GLOBAL
DSNT360I -DB8N ***********************************
DSNT362I -DB8N DATABASE = DSNB81A STATUS = RW
    DBD LENGTH = 20180
DSNT397I -DB8N
NAME      TYPE PART STATUS     PHYERRLO PHYERRHI CATALOG PIECE
-------- ---- ----- ----------------- -------- -------- -------- -----  
 DSNB81E TS    0001 RW
    -THRU      0003
 DSNB81E TS    0004 RW,REORP
    -THRU      0005
XEMP1    IX    0001 RW
    -THRU      0005
******* DISPLAY OF DATABASE DSNB81A ENDED  ********************
DSN9022I -DB8N DSNTDDIS 'DISPLAY DATABASE' NORMAL COMPLETION
***
```

```
DSDLIST - Data Sets Matching DB8NU.DSNDBC.DSNB81A.DSNB81E
Command ===>  Scroll ===> CSR
Command - Enter "/" to select action                  Message           Volume
-------------------------------------------------------------------------------
 DB8NU.DSNDBC.DSNB81A.DSNB81E.I0001.A001                      *VSAM*
 DB8NU.DSNDBC.DSNB81A.DSNB81E.I0001.A002                      *VSAM*
 DB8NU.DSNDBC.DSNB81A.DSNB81E.I0001.A003                      *VSAM*
 DB8NU.DSNDBC.DSNB81A.DSNB81E.I0001.A004                      *VSAM*
 DB8NU.DSNDBC.DSNB81A.DSNB81E.I0001.A005                      *VSAM*
 DB8NU.DSNDBC.DSNB81A.DSNB81E.I0001.A006                      SAPBYX
 DB8NU.DSNDBC.DSNB81A.DSNB81E.I0001.A007                      SAPBDR
 DB8NU.DSNDBC.DSNB81A.DSNB81E.I0001.A008                      SAPBYU
 DB8NU.DSNDBC.DSNB81A.DSNB81E.I0001.A009                      SAPBYQ
 DB8NU.DSNDBC.DSNB81A.DSNB81E.I0001.A010                      SAPBYU
***************************** End of Data Set list ****************************
```

9.6 Partition rotation and benefits for archiving

In SAP BW, data that exceeds its time period is required to be deleted; for example, the data is required for 60 months and month 61 now needs to be loaded. Thus the data from month 1 needs to be deleted and the new data added in logical sequence after month 60. As of DB2 V8, this is possible to do without impact by using a new feature known as 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/DELETED and the space is now required for reuse. Because BW would normally use a date format for partitioning, it is not possible to utilize the originally partition with its partitioning key values.

A new value has to be set up for the partitioning key. Delete the data from the partition in question. Then use the ALTER command to ROTATE the partition. Next, use the parameter ENDING AT to set up the new high key value. The now-rotated partition is immediately available without the requirement of doing a DB2 REORG.

The following output demonstrates the steps required to rotate a partition. The DISPLAY command lists the status of the tablespace partitions in the logical partition. (Note that the DISPLAY command now lists the partitions using the THRU word instead of each partition.) The VSAM datasets are also listed to show that there is no change in the physical part number before and after the ALTER has been completed.

The addition of ADD PARTITION and ROTATE PARTITION to the ALTER TABLE commands allows for continuous availability, even when storage adjustments are required. The DISPLAY command now shows partitions 02 through 05 as one unit and partition 01 on its own, which

---

**Note:** Because data in the partition being rolled off is deleted, you may want to consider the concept of *SAP Archiving* before proceeding with partition rotation. This will ensure long-term retention of the data.
is the new logical sequence for this partitioned tablespace. The output showing the VSAM clusters shows no change in the partition numbers.

```
ALTER TABLE DSN8810.EMP
ROTATE PARTITION FIRST TO LAST ENDING AT('999999') RESET
```

```
DSNT400I SQLCODE = 000, SUCCESSFUL EXECUTION
*****************************************************************************
```

```
PART  DBNAME                    TSNAME                    LIMITKEY
--------+---------+---------+---------+---------+---------+--------
1  DSN8D81A                  DSN8S81E                  '999999'
2  DSN8D81A                  DSN8S81E                  '199999'
3  DSN8D81A                  DSN8S81E                  '299999'
4  DSN8D81A                  DSN8S81E                  '499999'
5  DSN8D81A                  DSN8S81E                  '799999'
```

The SPUFI output shows the partitions in PART sequence. Note the LIMITKEY for PART 1 is now higher than the other parts. Thus, the logical partitioning is in a different sequence.
9.7 Backup and recovery

DB2 V8 provides enhanced backup and recover capabilities at the DB2 subsystem or data sharing group level. The purpose is to provide an easier and less disruptive way to make fast volume-level backups of an entire DB2 subsystem or data sharing group, and to recover a subsystem or data sharing group to any point in time, regardless of whether you have uncommitted units of work. SAP exploits these new recovery functions.

Two new utilities provide the vehicle for system level point in time recovery (PITR):

- The BACKUP SYSTEM utility provides fast volume-level copies of DB2 databases and logs.
- The RESTORE SYSTEM utility recovers a DB2 system to an arbitrary point in time. RESTORE SYSTEM automatically handles any creates, drops, and LOG NO events that might have occurred between the backup and the recovery point in time.

As a further enhancement to taking system-level backups, the SET LOG SUSPEND command now quiesces 32 KB page writes (for page sets that are not defined with 32 KB CI size) and data set extensions, thus avoiding integrity exposures.

The BACKUP SYSTEM and RESTORE SYSTEM utilities rely on new DFSMShsm™ services and SMS constructs in z/OS V1R5 that automatically keep track of which volumes need to be copied.

9.7.1 Past, present, and future for backup and recovery

Prior to DB2 V8, it was necessary to back up an SAP system with a combination of the DB2 COPY utility and some form of volume dump methodology. The DB2 COPY is taken primarily as a means of correcting errors with one or a limited set of database objects, and most commonly this recovery takes place to ‘CURRENT’. Managing thousands of image copies is complex, but it is required in order to allow the ability to recover individual objects, as opposed to the entire database.

The volume dump methodology prior to DB2 V8 required use of the SET LOG SUSPEND command to stop all write activity while the copy relationship was being established, causing a disruption in service. After the copy was complete, the data had to be dumped off to tape so that it could be available at some point in the future in case recovery was required.

Recovering a system meant restoring the volume backups from tape to the DB2 source data volumes if a volume dump strategy was employed, identifying which objects had been changed with some kind of log scanning process, creating recovery JCL for all objects identified as requiring recovery, and recovering each object to the same point in time. At best this process was time-consuming, labor-intensive, and very prone to error.

DB2 V8 uses DFSMShsm functionality to simplify and improve the performance and reliability of the backup and recovery process. The BACKUP SYSTEM and RESTORE SYSTEM utilities encapsulate all tasks previously required to perform each function into one utility statement each. The new Copy Pool construct and Copy Pool backup storage group types in DFSMShsm V1R5 make use of fast replication support. A full system Copy Pool backup can also be used for cloning and disaster recovery.

For further information on backup and recovery, see IBM Redbook DB2 UDB for z/OS V8: Through the Looking Glass and What SAP Found There, SG24-7088.
Chapter 10. Concepts, activities, and terminology

In this chapter we explain foundational concepts and describe activities that will help you gain the maximum benefit from this redbook. We also discuss the SAP Business Warehouse definitions that we use throughout the publication.

Note: A complete overview of all SAP BW definitions is provided in Appendix B, “Glossary for SAP BW from September 2004” on page 223. It was the most current glossary available at the time of publication.
10.1 Terminology: BW objects and processes

**InfoProvider**
In SAP BW, objects that can be analyzed are called InfoProviders. There are two classes of InfoProviders: those that contain physical data (such as InfoObjects, InfoCubes, and ODS 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 runtime.

**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 6.2, “SQL efficiency for InfoCube” on page 84.

**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 table spaces allow parallel access across multiple partitions. Partitioned table spaces 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 are 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.
10.2 SAP BW information model

The SAP BW information model is a basic structural element in the SAP BW architecture. It is designed to enable support of a range of decisionmakers. To do this, it supports the following conceptual layers of data warehousing:

- Multidimensional models, which enable views of data required for analytics
- Operational data store (ODS), 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 decisionmaking

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, ODS, and InfoSource. These elements are discussed in 10.2.1, “Other key elements in the information model” on page 172.

Refer to Appendix B, “Glossary for SAP BW from September 2004” on page 223 for a complete overview of SAP BW 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 BW information model. They eliminate semantic discrepancies in data elements across systems and organizational units, and ensure that data is consistent and reliable. SAP BW 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 BW provides consistent and integrated metadata for all objects across the data warehousing process.

Figure 10-1 on page 172 shows the elements of the SAP BW information model, all of which store metadata. In addition, the PSA, ODS object, 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.
10.2.1 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 BW 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 BW 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 BW includes attributes, texts, and hierarchies.

- **Operational data store (ODS) object**
  
  This describes a consolidated dataset from one or several InfoSources. In contrast to the multidimensional data models of InfoCubes, data in ODS objects is stored in flat, transparent, database tables.

  ODS object data can be updated into InfoCubes or other ODS objects using a delta update. Data in an ODS object can be analyzed with the SAP BW Business Explorer (BEx) tool, which 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 decisionmaking.

- **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 BW Business Explorer for reporting and Online Analytical Processing (OLAP) analysis.

10.3 Dataflow in SAP BW

Data flow is depicted at a very high level in Figure 10-1 on page 172 for an overall understanding. Here, Figure 10-2 provides a more complete view of the possible flow of data in SAP BW. Data is normally loaded into a PSA first, and from there into ODS and InfoCubes. It is also possible to directly load data into the ODS and InfoCubes, and from ODS to InfoCube.

![Figure 10-2  Dataflow in SAP BW](image)

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 BW, 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. And it then has to 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 BW. 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 BW update rules handle the subsequent flow of data from InfoSources to ODS 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.

### 10.4 Information access

As previously mentioned, in SAP BW, objects that can be analyzed are called InfoProviders. As shown in Figure 10-3, there are two classes of InfoProviders:

- Those that contain physical data, such as InfoObjects, InfoCubes, and ODS objects.
- Those that do not contain physical data, such as InfoSets, RemoteCubes, and MultiProviders.

InfoCubes, ODS objects, and InfoObjects are discussed in 10.2.1, “Other key elements in the information model” on page 172.

![Figure 10-3 SAP BW InfoProviders](image)

A MultiProvider provides access to data from several InfoProviders and makes it available for reporting and analysis. As shown in Figure 10-3, a MultiProvider itself does not contain any data, but can be assembled from different combinations of InfoProviders.

An InfoSet is a semantic layer using ODS 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 BW. Only the structure of the RemoteCube is defined in SAP BW. Data for reporting is read from another system using a BAPI.
10.5 Hierarchies

SAP BW allows the definition of hierarchies on InfoObjects. Hierarchies can be unbalanced, unleveled, and also time-dependent. Figure 10-4 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 shown in Figure 10-4, 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 BW, unleveled or dynamic hierarchies are stored as parent-child lists.

Figure 10-4   Sales hierarchy

The parent-child list that corresponds to Figure 10-4 is shown in Table 10-1.

Table 10-1   Parent-child list of example sales hierarchy

<table>
<thead>
<tr>
<th>Predecessor</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>WW Sales</td>
<td>Asia &amp; Pacific</td>
</tr>
<tr>
<td>WW Sales</td>
<td>North America</td>
</tr>
<tr>
<td>North America</td>
<td>Canada</td>
</tr>
<tr>
<td>North America</td>
<td>US</td>
</tr>
<tr>
<td>US</td>
<td>West</td>
</tr>
<tr>
<td>US</td>
<td>East</td>
</tr>
<tr>
<td>WW Sales</td>
<td>Europe</td>
</tr>
<tr>
<td>Europe</td>
<td>East</td>
</tr>
<tr>
<td>Europe</td>
<td>Central</td>
</tr>
</tbody>
</table>

If there are restrictions on hierarchies within a query, the results, that is, the corresponding leaves of the hierarchy nodes that you select, are stored in hierarchy result tables. The names of those tables consist of the prefix for SAP BW tables (/BI0 or /BIC) followed by the digits 02; /BI0/0200000001 is an example of the name of a hierarchy result table.
10.6 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 10-5. These are arranged according to the star schema, a technique that organizes data in terms of data facts and business dimensions.

![Extended star schema](image)

**Figure 10-5  Extended star schema**

The star schema places several *dimension tables* around a central *fact table*. The 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 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 BW can have up to 16 dimensions. By default, every InfoCube has the three standard dimensions *Data Package, Time, and Unit*.
SAP BW 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 10-6, 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.

SAP BW 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 fact table. This eliminates the necessary join operation between dimension and fact table during SAP BW query processing, which can provide improved query performance.

### 10.7 Data load into InfoCube

When data is loaded from PSA or ODS into an InfoCube, it must be transformed from a flat table structure into the extended star schema. Figure 10-7 on page 178 shows the steps that are required to transfer a new record from a PSA table into an InfoCube and the corresponding master data tables.
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 if 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 if 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 fact table. It contains the key figures from the PSA record and the corresponding dimension identifiers.

### 10.8 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 BW, 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 similar benefit as adding another index to a database table. Aggregates require no special knowledge by the end user, and in fact are completely transparent to the end user. The only way an end user might recognize the existence of an aggregate is by the performance gain that is observed.

### 10.8.1 Aggregate example

The contents of an example InfoCube are shown in Table 10-2. It 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>
<thead>
<tr>
<th>Country</th>
<th>Customer</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>Blue Soft</td>
<td>10</td>
</tr>
<tr>
<td>Germany</td>
<td>Ocean Networks</td>
<td>15</td>
</tr>
<tr>
<td>USA</td>
<td>Funny Duds</td>
<td>5</td>
</tr>
<tr>
<td>Canada</td>
<td>Ocean Networks</td>
<td>10</td>
</tr>
<tr>
<td>Canada</td>
<td>Thor Industries</td>
<td>10</td>
</tr>
<tr>
<td>Germany</td>
<td>Funny Duds</td>
<td>20</td>
</tr>
<tr>
<td>USA</td>
<td>Blue Soft</td>
<td>25</td>
</tr>
</tbody>
</table>

Table 10-3 shows another sample aggregate that only contains the country characteristic, and the key figure, Sales.

<table>
<thead>
<tr>
<th>Country</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>40</td>
</tr>
<tr>
<td>Germany</td>
<td>35</td>
</tr>
<tr>
<td>Canada</td>
<td>20</td>
</tr>
</tbody>
</table>

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 10-8 on page 180.
The aggregate is shown in Table 10-4. That aggregate can be used by queries that report on sales for a hierarchy node on level 2 or higher. In this example shown in Figure 10-8, 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>
<thead>
<tr>
<th>Customer</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>America</td>
<td>60</td>
</tr>
<tr>
<td>Europe</td>
<td>35</td>
</tr>
</tbody>
</table>

### 10.8.2 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 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 RSDDKAgregarates_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. It is therefore recommended 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 a program RSDDSAgregarates_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 actually 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 UDB, the indices are also renamed at this stage.
10.9 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 10-9. Compressed requests can only be deleted by deleting the entire contents of the InfoCube.

![Figure 10-9 Compression of requests](image)

10.10 Operational data store (ODS)

An ODS object stores consolidated transaction data from one or several InfoSources. In contrast to the multidimensional data models of InfoCubes, data in ODS objects is stored in flat, transparent, database tables. ODS object data can be updated into InfoCubes or other ODS objects using a delta update. Data in an ODS object can be analyzed with the SAP BW Business Explorer.

An ODS object contains a key (for example, order number) as well as data fields (key figures). As opposed to InfoCubes, fields in an ODS 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 ODS object, the relevant fields in the ODS objects must be overwritten and set to the current value.

Figure 10-10 on page 183 shows an example of two layers of ODS objects that are used to update order and delivery information. These ODS objects allow you to determine, for example, which orders are open, partly delivered, or completely delivered.
The ODS 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 ODS 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 BW 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 ODS object.

Thus the ODS objects of the inner level of the operational data store (level 2 in Figure 10-10) 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 ODS 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 ODS 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 ODS object into other ODS objects or InfoCubes.

Figure 10-11 on page 184 shows the steps that occur during activation of ODS data. In this example, we assume that document 123, with an amount of 100, is already loaded into the activation table of the ODS object.
When data is activated in the ODS, 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 ODS Objects and/or InfoCubes, in a separate step. The amount is therefore reduced by 40 in the related data targets in the example.

Table 10-5 lists the SAP BW naming conventions.

### 10.11 SAP BW naming conventions

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/BIX/F&lt;InfoCube&gt;</td>
<td>Table</td>
<td>F fact table</td>
</tr>
<tr>
<td>/BIX/E&lt;InfoCube&gt;</td>
<td>Table</td>
<td>E fact table</td>
</tr>
<tr>
<td>/BIX/D&lt;InfoCube&gt;</td>
<td>Table/View</td>
<td>Dimension table of dimension n n = P,T,U,1...9,A,...D</td>
</tr>
<tr>
<td>/BIX/M&lt;InfoObject</td>
<td>View</td>
<td>Master data attribute (join of P + Q)</td>
</tr>
<tr>
<td>/BIX/P&lt;InfoObject</td>
<td>Table</td>
<td>Master data attributes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(time-dependent)</td>
</tr>
<tr>
<td>/BIX/Q&lt;InfoObject</td>
<td>Table</td>
<td>Master data attributes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(time-dependent)</td>
</tr>
<tr>
<td>/BIX/T&lt;InfoObject</td>
<td>Table</td>
<td>Master data texts table</td>
</tr>
</tbody>
</table>
10.12 The SAP BW functional components

So far, we have discussed a number of key elements used in SAP BW. In this section we show how those elements are combined to form the functional components that comprise the SAP BW architecture.

Figure 10-12 on page 186 shows the primary functional components of SAP BW:

- **Business Explorer**
  - This tool is used to define and execute SAP BW queries. It comprises the following tools:
    - BEx Analyzer and BEx Browser are two tools used to execute SAP BW 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 BW 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 BW 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 10.3, “Dataflow in SAP BW” on page 173.

- **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, ODS 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 10-12 SAP BW functional components](image)

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).

### 10.13 SAP BW Business Content

To accelerate the SAP BW implementation, SAP provides a component called *Business Content*. It incorporates best practices and the business process knowledge of SAP.
The content, depicted in Figure 10-13, includes preconfigured programs for extracting data, data models, and a wide range of predefined templates for reports and analyzes. 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 ODS Objects
- + 710 InfoCubes
- + 140 MultiProviders
- + 3,600 Queries
- + 1,900 Workbooks
- + 800 Roles

<table>
<thead>
<tr>
<th>SAP BW predefined Business Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>- An example of available material -</td>
</tr>
</tbody>
</table>

![SAP BW business content](image)

This content can be used as is, or 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 BW 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 redbook. There is more detailed information available concerning this offering that can be accessed from your IBM representative, or directly from SAP.
Benefits of DB2 V8 and the zSeries platform

SAP BW systems are often a critical tool for the company’s executive business decision-making process. Frequently the SAP BW database is very large, requiring careful management to ensure business information is available when needed and that access to that data is responsive. The zSeries platform and DB2 V8 provide an excellent infrastructure for these requirements. In this chapter we discuss the most important DB2 V8 features in SAP BW 3.5 and list some of the benefits offered by the IBM zSeries platform.

Note: For general information on choosing zSeries as a platform, see 1.3, “Why run SAP BW on IBM zSeries” on page 3. For more information concerning DB2 V8 as it relates to SAP, refer to “Related publications” on page 259.
11.1 IBM DB2 V8 benefits for SAP

The IBM zSeries is well known for its role as a business critical support platform. Many years of experience have gone into making the zSeries zOS and DB2 database the mature and reliable products they are today. A combination of legacy applications and new workload packaged solutions have stamped the zSeries platform as being the system of choice for mission-critical processing.

IBM DB2 V8 has been reengineered with a view toward meeting the requirements of upcoming high-end business environments, leading to many fundamental changes in its architecture.

The reengineering includes dozens of changes in SQL, improving consistency. Many barriers are now removed: using 64-bit memory, providing consistent table and column name lengths; allowing 2 MB SQL statements, 4096 partitions, and three times the log space. These improvements include:

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

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

For a more comprehensive overview of DB2 V8 functions, refer to the DB2 documentation listed in “Related publications” on page 259.

Following is an overview of the most important implementations under DB2 V8:

- Integrating zSeries data with DB2 Connect

  SAP NetWeaver requires support of Unicode and Java for the database and its connectivity between the SAP application server and the DB2 database server. DB2 Connect implements distributed relational database architecture as the standard protocol for remote database access. With SAP NetWeaver, DB2 Connect is used to access DB2
IBM DB2 V8 has been reengineered
This reengineering, with a view toward the requirements of upcoming high-end business environments, led to many fundamental changes in its architecture and structure, optimizing the SAP on zSeries solution.

Java and Unicode support for SAP solutions
DB2 V8 is ready to provide full Java and Unicode support. It exploits the 64-bit architecture and offers substantial improvements in the evolution of online schemas.

64-bit virtual addressing dramatically increases virtual storage
One of the major improvements from which SAP solutions can greatly benefit is the 64-bit virtual addressing capability. It relieves virtual memory constraints and supports significantly larger buffer pools of up to 1 TB and more concurrent threads. It also allows greater scalability of large DB2 subsystems or members of a data sharing group to meet the demands of increasing workloads in a dynamic business environment.

Highest availability for mission-critical applications
- zSeries has an expected meantime between failures over 40 years and is the ideal platform for mission-critical applications where high availability is important.

Based upon this hardware-platform, IBM continues to invest heavily in extending the technology leadership of the zSeries Parallel Sysplex cluster architecture. DB2 V8 exploits new Coupling Facility commands developed specifically for DB2 to write groups of data pages more efficiently. It improves the efficiency of DB2 data sharing, especially for SAP applications. In addition, DB2 V8 implements a new locking protocol for data sharing to reduce locking overheads in the coupling facility and to provide greater availability.

System-level, point-in-time recovery for faster recovery of entire SAP database
Enhancements to system-level, point-in-time recovery for DB2 provide improved usability, more flexibility, and faster recovery times. With DB2 V8, you can recover your data to any point in time, regardless of whether you have uncommitted units of work. As a result, data recovery time improves significantly for large SAP DB2 subsystems with thousands of tables, and the potential downtime for an SAP system decreases considerably.

SAP accounting and WLM
In order for SAP to be able to provide transaction-based DB2 accounting and workload management for its Java applications, it needs to be able set DB2 client identifiers using JDBC. The DB2 Java Universal Driver, which is the SAP JDBC driver of choice, implements the DRDA protocol like CLI.

Therefore, DB2 treats CLI clients and Java Universal Driver clients uniformly. The DB2 client identifiers that are the basis of accounting data rollup can be altered at transaction boundaries. The Universal Driver extends the Connection interface from the JDBC standard to also provide the following methods, which set client identifier values:
- DB2Connection.setDB2ClientUser(String user)
- DB2Connection.setDB2ClientWorkstation(String name)
- DB2Connection.setDB2ClientApplicationInformation(String info)
- DB2Connection.setDB2ClientAccountingInformation(String info)

11.2 The benefits of DB2 V8 for SAP BW

In the following sections we detail DB2 functions that are important for SAP BW and are available for DB2 V8.
11.2.1 Star join enhancements

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

While dimension tables and fact tables have a parent-child relationship, SAP BW does not define foreign keys to manifest this. If dimensions consist of single tables, the schema is called a star schema. If they are made up of multiple tables, the schema resembles a snowflake and is therefore called a snowflake schema (see Figure 11-1).

Unlike OLTP queries, where a large number of short duration queries execute, OLAP queries involve a large number of tables and an immensely large volume of data to perform decision-making tasks. Hence, OLAP queries are expected to run much longer than OLTP queries, but are less frequent.

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

- "Selectively" join from the outside-in

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

- Efficient “Cartesian join” from the outside-in

A physical Cartesian join generates a large number of resultant rows based on the cross-product of the unrelated dimensions. A more efficient Cartesian-type process is required as the number and size of the dimension tables increase, to avoid an exponential growth in storage requirements. Index key feedback technique is useful for making Cartesian joins efficient.

- Efficient “join back”, inside-out

The join back to dimension tables that are accessed after the fact table must also be efficient. Non-indexed or materialized dimensions present a challenge for excessive sort merge joins and workfile usage. To meet this challenge, DB2 V7 provided partial support with the introduction of sparse index on workfiles for the inside-out join phase only. DB2 V8 extends this further with controlled materialization and in-memory workfiles.

- Efficient access of the fact table: Due to the generation of arbitrary (or unrelated) key ranges from the Cartesian process, the fact table must minimize unnecessary probes and provide the greatest level of matching index columns based on the pre-joined dimensions.

Sparse index in outside-in join phase

For an efficient Cartesian process, DB2 employs a logical, rather than physical, Cartesian join of the dimension tables. Each dimension that the optimizer chooses to access before the fact table has all local predicates applied, with the result sorted into join column order and is materialized into its own separate workfile.

DB2 simulates a Cartesian join by repositioning itself within each workfile during nested loop join processing, to potentially join all possible combinations to the central fact table. Notice that the nested loop join is pushed down to Data Manager (stage 1) in this phase of star join processing. The sequence of this simulated Cartesian join respects the column order of the selected fact table index.

The sparseness of data within the fact table implies that a significant number of values of the Cartesian product do not match any fact table records. To optimize execution time, DB2 avoids joining unnecessarily derived rows to the fact table.

It accomplishes this by introducing an index key feedback loop to return the next highest key value whenever a not-found condition is encountered. A matching record in the fact table returns the record. A miss returns the next valid fact table key so that DB2 can reposition itself within the dimension workfiles, thus skipping composite rows with no possibility of obtaining a fact table match. Figure 11-2 on page 194 illustrates this process.
This approach allows significant skipping in the Cartesian join, but skipping the index keys is not free. As there are no indexes on workfiles used in the outside-in phase of star join processing, these workfiles are scanned sequentially. If a dimension table is not at a leading position within the Cartesian product, it is likely that the workfile containing it is scanned repetitively. In this case, the cost of skipping workfile entries may grow significantly.

DB2 V8 introduces sparse indexes on snowflake workfiles that are used during outside-in processing to overcome this problem. These sparse indexes reside in memory and consume up to 240 KB. If the total number of workfile entries is small enough, then all entries can be represented in the index. This provides a one-to-one relationship between index entries and workfile records.

The index only becomes sparse if the number of index entries cannot be contained within the space allocated. The structure of a sparse index is flat rather than a b-tree structure of standard table indexes (see Figure 11-3 on page 195). The index is probed through an equi-join predicate, and a binary search of the index is utilized to find the target portion of the workfile. Then a sequential search of the identified workfile portion is performed to find the corresponding record.

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

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

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

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

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

11.2.3 Sparse index during inside-out join phase

After joining the logical Cartesian product of selective dimension tables to the fact table, the intermediate result (composite table) may still be large. The lack of indexes on workfiles used during the inside-out phase of star join processing makes the optimizer choose sort merge join to subsequently join the remaining dimension tables to the composite table. This may involve sorting the large composite table, with excessive CPU and I/O consumption and increased parallelism overhead. Therefore, DB2 V7 introduced the support of sparse indexes on the snowflake workfiles of the inside-out phase. This allows the optimizer to consider nested loop join, which is more efficient in many cases.
DB2 V8 provides further enhancements in the inside-out join phase by being able to select supporting nested loop join (rather than merge scan join) and in-memory workfiles in this phase (see Figure 11-4). These in-memory workfiles behave the same as the in-memory workfiles from the outside-in phase, and also populate the star join pool. Moreover, DB2 V8 accomplishes controlled materialization of snowflakes in this phase.

**Figure 11-4   Star join: Inside-out join phase**

11.2.4 Snowflake handling: Controlled materialization

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

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

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

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

The choice not to materialize can overcome the sort and workfile allocation overhead, and rather than requiring an index to be built on the workfile, the indexes on the underlying snowflake tables can be used for efficient joins after the fact table.
In addition to the star join enhancements already described, DB2 V8 provides an improved
cost estimation algorithm that better estimates the filtering effect of dimensions. This results
in a better table join sequence and can yield a significant performance improvement.

11.3 Joins with up to 225 tables

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

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

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

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

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

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

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

11.4 Common table expressions

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

Figure 11-5 contains a sample SQL statement that makes use of a CTE. The first CTE in the statement is the query for table expression E, which determines employee number, last name, salary, and hiring decade for all employees of the EMPLOYEE table. The columns of the associated result table are those named in the SELECT statement. For all available decades, the CTE M determines the minimum salary paid to the employees that were hired during the appropriate decade.

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

Figure 11-5  Common table expression

For comparison, as shown in Figure 11-6, an NTE can be used to produce the same result set.

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

Figure 11-6  Nested table expression
As the example shows, the column names of the CTE are specified in parentheses and follow the name of the CTE. This is the same technique as used in naming the columns of a view. The SELECT follows the common table expressions. Since it can refer to the CTEs, the SQL statement is more comprehensible compared to the use of NTEs.

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

11.4.1 Recursive SQL

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

Figure 11-7 shows a hierarchy that groups countries at different granularities. Hierarchy definitions are stored in single tables. To determine the countries in EMEA, for example, in a first run, the table first needs to be accessed to retrieve the subnodes Europe and Africa. Recursively, in a second run, the countries that are assigned to the nodes Europe and Africa are retrieved to establish the final result set.

The query described could be satisfied by using the recursive statement from Figure 11-7.

![Figure 11-7 SAP InfoCubes: external hierarchy](image)

Recursive SQL involves defining a CTE that references itself. The initialization SELECT is executed only once. It controls the starting point of the recursion. The iterative SELECT is executed for all qualifying records and then repetitively for all further records that qualify.

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

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

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

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

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

The following steps are necessary to exploit MQTs:

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

11.5.1 Creating MQTs

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

You can either:

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

Creating an MQT from scratch

The CREATE TABLE statement syntax has been enhanced to allow you to create an MQT. The new syntax is shown in Figure 11-8 on page 201.
Figure 11-8 CREATE MQT: Syntax

Creating an MQT is similar to creating a view. In both cases you specify a fullselect to define its contents. The difference is that a view is only a logical definition, while a materialized query table contains materialized data of the query result on disk. For this reason, an MQT is also called a materialized view.

Example 11-1 shows a CREATE TABLE statement to create an MQT. The fullselect, together with the DATA INITIALLY DEFERRED and REFRESH DEFERRED clauses, defines the table as a materialized query table.

Example 11-1 Sample create of a materialized query table

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

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

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

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

The statement in Example 11-2 registers a table TRANSCOUNT as a materialized query table with the associated subselect to DB2. The data in the table will remain the same as indicated by DATA INITIALLY DEFERRED, and will still be maintained by the user, as specified by the MAINTAINED BY USER clause. The user can continue to LOAD, INSERT, UPDATE, or DELETE data in the table TRANSCOUNT. ALTER TABLE can also change a materialized query table into a base table.

Example 11-2   Converting a base table into an MQT

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

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

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

11.5.2 Populating and maintaining an MQT

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

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

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

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

The MAINTAINED BY option indicates how the data in the MQT is to be refreshed:
MAINTAINED BY SYSTEM, which is the default, indicates that the MQT is system-maintained. The only way to refresh the data in a system-maintained MQT is by using the REFRESH TABLE statement. A system-maintained MQT cannot be updated by using the LOAD utility, INSERT, UPDATE or DELETE SQL statements. Therefore, a system-maintained MQT is read-only. If a view or a cursor is defined on a system-maintained MQT, it becomes read-only.

Alternatively, MAINTAINED BY USER can be specified to define a user-maintained MQT. A user-maintained MQT can be updated by the LOAD utility, INSERT, UPDATE or DELETE SQL statements, as well as the REFRESH TABLE statement. Therefore, a user-maintained MQT is updateable.

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

REFRESH TABLE
The REFRESH TABLE statement can be used to populate an MQT. See Example 11-3 for a sample REFRESH TABLE statement.

Example 11-3 Sample REFRESH TABLE statement

REFRESH TABLE mq_table;

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

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

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

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

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

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

The automatic query rewrite process searches for matched MQTs that result in an access path with the lowest cost after rewrite. The costs of the rewritten query and the original query are compared and the one with the lowest cost is chosen. If the final query plan comes from a rewritten query, the PLAN_TABLE shows the name of the matched MQTs and the access path using them.

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

**DDL options**

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

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

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

**Special registers**

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

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

  In DB2 V8, only CURRENT REFRESH AGE of 0 or ANY is supported:
  - 0 means only MQTs that are kept current with the source tables are considered by automatic query rewrite. Since DB2 V8 does not support immediately refreshed MQTs, specifying 0 means that AQR will not consider any MQTs.
  - ANY represents the maximum duration, meaning all MQTs are considered by automatic query rewrite.

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

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

**ALL**
This indicates that all MQTs are considered by automatic query rewrite.

**NONE**
This indicates that no MQTs are considered.

**NONSEYSTEM**
This indicates that (under the assumption that CURRENT REFRESH AGE is set to ANY), only system-maintained MQTs are considered.

**USER**
This indicates that (under the assumption that CURRENT REFRESH AGE is set to ANY), only user-maintained MQTs are considered.

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

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

### 11.5.4 Determining if query rewrite occurred

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

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

### 11.6 Partitioning

In this section we describe enhancements that are introduced in DB2 V8 in the area of table partitioning. This will be of interest to many SAP users, where the high volume of data in particular tables requires the use of partitioning functionality.

Most commonly, partitioning is considered as tables start to approach the size of 64 GB for the table space. Partitioning can improve the manageability of the data, through more effective REORG, data management, and backup/recovery strategies, as well as potentially improving performance of programs using this data. For more information, see 9.5, “Partition addition” on page 161.

DB2 V8 also introduces the concept of Data Partitioned Secondary Indexes, where secondary indexes on tables can also be partitioned into separate partitions, in this case corresponding to the set of rows in the corresponding table space partition. Within this IBM Redbook there are different examples concerning partitioning on different places. You can find some of these in the “Index” on page 263.
11.6.1 Increases to table sizes

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

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

11.6.2 Adding new partitions and rotating partitions

With DB2 for z/OS Version 7, it was possible to redistribute data among partitions by altering the limit keys, and conducting reorganization utilities on the tables in question. However, it was not possible to change the number of partitions.

DB2 V8 implements the ability to add a partition with the `ALTER TABLE ADD PARTITION` statement. Issuing this command results in DB2 creating a new partition at the end of the current partitions.

Rotating partitions allows old data to “roll off” while reusing the partition for new data with the `ALTER TABLE ALTER PARTITION ROTATE FIRST TO LAST` statement. A common case in an SAP system is where (n+1) partitions are used to continuously keep the last n periods of data. When rotating, one can specify that all the data rows in the oldest (or logically first) partition is to be deleted, probably after archiving activity has removed the data. Then you specify a new table space high boundary so that the partition essentially becomes the last logical partition in sequence ready to hold the data which is added.

Because the data of the partition being rolled off is deleted, if long-term retention of the data is required, you would certainly want to consider running an SAP Archiving process before rotating the partition, as any remaining rows will be deleted by the process. Confirmation of successful SAP archiving would be done by checking that no rows remain in the partition.

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

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

The availability of these two additional ALTER statements allows for continuous availability, even where the growth of data requires adjustments to the storage layout used. For many SAP tables, the nature of growth of data may not be able to be determined in advance, or anticipated by the project team implementing the software. The enhancements to partition controls described here allow a more dynamic approach to data management, and can mean that unanticipated data growth in an SAP system is able to be addressed with minimal impact to users, greatly increasing availability.

Previously, if a chosen partitioning scheme did not have the ability to handle the data volumes in a system, a significant amount of “downtime” may be needed to unload the data, and
increase the number of partitions. Now, it is possible to increase the theoretical data capacity “on the fly” without any significant impact to availability.

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

11.6.3 Data partitioned secondary indexes

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

You create a DPSI by specifying the new PARTITIONED keyword in the CREATE INDEX statement, shown in Figure 11-9, and defining keys on columns that do not coincide with the partitioning columns of the table. When defining a DPSI, the index cannot be created as UNIQUE or UNIQUE WHERE NOT NULL. This is to avoid having to search each part of the DPSI to make sure the key is unique.

![Figure 11-9 Syntax of CREATE INDEX statement](image)

11.6.4 Separating clustering from partitioning

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

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

In the case where performance or throughput may be enhanced through additional indexes being created, tools are supplied as part of the package to facilitate this action. This action is often performed where additional functions are coded and implemented by a customer, or the system is used in a manner not expected in the default setup of the supplied application.

In many cases, the SAP application is supplied with definition data in the SAP Data Dictionary for indexes which are not created by default. These may be then be created easily in the database, for cases where specific functionality that is not universally used by all SAP implementations takes place in an SAP system.

The result of these two design constraints of the DBMS and the application package (prior to DB2 V8) previously meant that the clustering and partitioning indexes were usually one and the same, and—as the primary clustering index is predominantly used as the access path for SQL statements—in most customer cases, this works well.

The one remaining consideration, in this case, is where the key range of the primary clustering index does not lend itself to an efficient scheme of partitioning. For example, the primary key distribution may be very “lumpy”, and change activity is not evenly distributed, or is concentrated unevenly across the key ranges. This results in added overhead in maintaining good data distribution, for example, with REORG PART utilities. Also, it can result in inefficient usage of the available underlying disk if the chosen partitioning scheme is less than ideal.

**11.7 Deferred indexes do not prevent table access**

Prior to V8, DB2 already allowed you to create deferred indexes. Deferred indexes are those created using the DEFER YES clause. When creating a deferred index, the description of the index and its index space is added to the catalog, but it is not built. It is placed in Rebuild Pending status. The only exception to this is if the table on which the index is defined is empty.

The advantage of deferring the creation of the index tree is that multiple indexes, which are defined on the same table, can be built in a single run rather than scanning the table multiple times. SAP BW makes considerable use of deferring index creation.

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

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

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

In this section we describe the usage of multi-row fetch and multi-row insert in an SAP environment. Prior to DB2 V8, SAP already used multi-row operations, but only through the network, in order to save network trips. On the server side, SAP had to process row by row.

With this new feature, SAP can reduce the number of DB2 calls dramatically. DB2 V8 itself benefits a lot. It is no longer necessary to go through the whole software stack for each row. The benefit is obvious. The usage of multi-row operations saves cross-memory trips and significantly reduces elapsed time and CPU cost on the server.

Multi-row inserts have the capability of inserting up to 32 737 rows in a single API call. Depending on the number of indexes and columns, a reduction of up to 30% of CPU overhead can be achieved. This is a result of avoiding cross-address space trips.

11.8.1 How SAP benefits from multi-row operations

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

11.8.2 Implementation details

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

The old code, needed prior to DB2 V8, is shown in Example 11-4. The DB2 call sequence of an ABAP INSERT operation is:

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

The DB2 call sequence of an ABAP READ operation is:

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

Example 11-4  Processing each record in a loop

```java
for ( i = 0; i < row_count; i++ )
{
    /* process single row */
    /* insert case */
    EXEC SQL EXECUTE $_0000 USING DESCRIPTOR :*db2da_p;
    /* fetch case */
    EXEC SQL FETCH C_0000 USING DESCRIPTOR :*db2da_p;
    if ( SQLERROR and NOT SQLCODE == DUPLICATE_RECORD)
        exit
}
```

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

In Example 11-5, we show the new code for DB2 V8.

Example 11-5  The need for a loop is eliminated

```java
/* process multiple rows; nr is set to row_count*/
/* insert case */
```
11.9 RUNSTATS enhancements

Following is a list of several enhancements related to query optimization.

- RUNSTATS improvements
  These improvements fall into three categories:
  - SAP and distribution statistics
    These statistics can be critical to the DB2 optimizer by choosing the most efficient method of servicing an SQL request in case of non-uniform distributions. The DSTATS tool has been widely available for some time now, and works with DB2 Version 5 and above. DB2 V8 effectively provides equivalent functions in the normal RUNSTATS utility.
  - REPORT NO UPDATE NONE option
    The parameter combination of REPORT NO and UPDATE NONE has been added to the RUNSTATS utility, and the usage of these two options will invoke the invalidation of DSC statements involving the object(s) in question.
  - Host variables impact on SAP access paths
    We describe a new feature of DB2 V8 that eliminates the disadvantages of REOPT(VARS) and still takes the host variables into account at access path selection.

- Index-only access path for varying-length columns
  We describe the new option to create indexes that do not pad varying-length columns of data type VARCHAR and VARGRAPHIC. This allows the DB2 optimizer to consider index-only access for indexes that contain VARCHAR or VARGRAPHIC columns, which accelerates query execution time due to saved I/O operations against the corresponding table space. As SAP creates all character columns of tables using data type VARCHAR or VARGRAPHIC, this new feature potentially speeds up query execution in all areas.

- Multiple value IN lists and SAP
  We describe a DB2 V8 enhancement to the handling of INLIST predicates involving multiple values in the INLIST. The SAP application makes extensive use of INLIST predicates in dynamic SQL.

11.10 2 GB bar

DB2 V8 utilizes 64-bit addressing, which relieves storage constraints. The following DB2 components now reside above the 2 GB bar, and thus are able to be allocated using far larger sizes than with DB2 V7.

- EDM pool (DBDs and cached dynamic statements only)
- RID pool
 Chapter 11. Benefits of DB2 V8 and the zSeries platform

Following is a brief discussion of the DB2 components which still use storage below the 2 GB bar.

**EDM pool**
In DB2 V8, the EDM pool consists of three separate pools. A new dynamic statement cache is created above the 2 GB bar. The DB2 ZPARM EDMDBDC, with a default of 102400 KB, defines the size required. A new EDM DBD cache pool is also created above the 2 GB bar. These two features relieve contention with other objects in the EDM pool. Plans and packages remain in the EDM pool below the 2 GB bar.

**RID pool**
The RID is now split into two portions. Roughly 25% still resides below the 2 GB bar, and is used for RID maps and the other 75%, above the 2 GB bar is used for RID lists.

**SORT pool**
Two kinds of storage pools are used by DB2 sort to store control structures and data records. One pool is an agent-related local storage pool, and the other is a global sort pool. To take advantage of the 64-bit addressability for a larger storage pool, some high level sort control structures remain in agent-related storage below the 2 GB bar, but these structures contain 64-bit pointers to areas in the global sort pool above the 2 GB bar. The sort pool above the 2 GB bar contains sort tree nodes and data buffers. Therefore, it consumes less storage below the 2 GB bar and contributes to virtual storage constraint relief in this critical storage area.

**11.11 Automatic Space Management**

In this section we describe the new space management features introduced in DB2 V8. For an SAP implementation, these features are extremely valuable for continuous availability, as they provide adaptability when data growth patterns are not predictable, or do not follow those expected.

The new Automatic Space Management feature introduced in DB2 V8 is considered highly valuable for all SAP implementations. It will potentially eliminate one of the main causes of failures where growth has not been completely anticipated. As discussed in this section, administrative tasks still remain, but reducing or eliminating a cause of failure is a major step forward.

**Impact on database administration (DBA) function**
If a database object in the form of a table space, table space part, or index reaches the maximum number of extents permitted, usually the following message is issued on the system console:

```
DSNP007I csect - EXTEND FAILED FOR data-set-name. RC=00D70014
CONNECTION-ID=xxxxxxxxxx,
CORRELATION-ID=yyyyyyyyyyyyy,
LUW-ID=logical-unit-of-work-id =token
```
The unit of work issuing the SQL statement resulting in a space request will abend. Subsequently, repeating any work trying to add data to this object will similarly fail until the issue is resolved.

This situation can potentially be avoided through the correct “pre-sizing” of the objects to be used in an SAP system; however, it is not always possible to anticipate future space requirements. If these failures do occur, they are generally seen either during SAP system implementation, or after the system goes live.

### Issues arising during SAP system implementation

Typically, at some point in the process of implementing a SAP component, there is a phase, or a number of phases, where a customer will perform activities normally referred to as “data loads”. This is where data from other systems is placed into the new SAP system, using a number of means available to perform the transfer.

The various techniques for loading data vary in their time to implement and runtime efficiency, but eventually all lead to rows being inserted and/or modified in a table or series of tables. Due to the complex and highly integrated nature of SAP, it is sometime difficult to know ahead of time what database objects will carry the new data.

In the situation where many different types of data loads are performed over a period of time, it can be very frustrating to iteratively deal with out-of-space conditions. Where the out-of-space conditions occur due to the maximum number of extents being reached (255), this could have been avoided had suitable values been chosen for PRIQTY and SECQTY of the database object. It may be possible to anticipate the size of the major tables which will have data inserted into them.

However, in an SAP system, it is often not the well known large tables in the system that cause the most issues during data loads, but rather the small-to-medium-size tables. This can happen due to the size estimates provided in the SAP system not being adequate to handle the data expected in every customer.

The provision of the new Automatic Space Management function fundamentally addresses this issue by allowing a more adaptive scheme to take place when table spaces and indexes have more data placed in them than was anticipated. It should be noted that this solution addresses the problem of failures associated with reaching the maximum permitted number of extents, particularly where this results in a data set size at the time of failure that is less than the theoretical maximum of the value DSSIZE for table spaces or PIECESIZE for indexes.

As such, here are a few issues that are not within the scope of this enhancement:

- There may be a situation in which the database object will have sufficient data inserted into it that it would also receive a failure when it reaches the DSSIZE or PIECESIZE limit.
- This is probably very rare in the case of an SAP system. When loading data into an empty system, in most cases the major tables that will have data loaded into them are well known to those programmers and teams performing the loads. In these cases, as part of their normal practice, the teams should have alerted those responsible for the DBA function to those tables in question, and the anticipated number of records involved. In many cases, these tables may be partitioned to address size limitations.
- The database object in question may, after the data loading activity, still have a large number of extents, while not having reached the maximum.
- There may be some performance and administrative benefits that result in the subsequent resizing of the object, through altering the PRIQTY and SECQTY and subsequently REORG on the table space or index. This is considered to be much less of an issue than
actually suffering space-related failures, as they can be addressed in a timely manner while allowing the implementation process to continue.

- No space is available on the disk where DFSMS routines require the data set extents to be placed.
- If a growing data set is extending, and the VSAM Access Method Services routines are not able to find sufficient space on the disk volumes allowed, it is possible that an extend failure will occur before 255 extents are reached. Also, if the remaining disk space is in small, fragmented pieces, it is possible that 255 extents will be reached earlier due to numerous small extents being taken to fulfill space requirements.

**Automatic Space Management functions**

Automatic Space Management is enabled with the new DSNZPARM of MGEXTSZ, which can be set to either YES or NO, and the default is NO. That is, after moving to DB2 V8, the new functionality is disabled initially, and space management occurs as it previously did in DB2 Version 7.

When enabled by setting MGEXTSZ, in certain situations DB2 will automatically adapt the size of each subsequent secondary allocation requested. It does this in such a way that it follows a sliding scale that guarantees the object will reach its theoretical maximum data set size (DSSIZE) prior to reaching the maximum number of extents allowed for a VSAM data set of 255.

**New defaults TSQTY and IXQTY**

APAR PQ53067 for DB2 V6 and V7 introduced the new DSNZPARM parameters TSQTY and IXQTY, allowing the administrator to override the default allocation values for primary and secondary quantities for the cases where these are not specified in the object creation DDL.

- The actual value applied by DB2 for default PRIQTY will be determined by the applicable ZPARMs, TSQTY or IXQTY, which were introduced with APAR PQ53067. The ZPARMs TSQTY and IXQTY will now have global scope.
- TSQTY will apply to non-LOB table spaces. For LOB table spaces, a 10x multiplier will be applied to TSQTY to provide the default value for PRIQTY.
- IXQTY will apply to indexes. ZPARMs TSQTY and IXQTY will continue to have a default value of 0 (zero), but this value will indicate a new default value of 720 KB (1 cylinder) is to be applied.
- If TSQTY is set to 0, then 1 cylinder will be the default PRIQTY space allocation for non-LOB table spaces and 10 cylinders will be the default PRIQTY space allocation for LOB table spaces.
- If IXQTY is set to 0, then 1 cylinder will be the default PRIQTY space allocation for indexes.

The user can provide override values for TSQTY and IXQTY ZPARMs to avoid wasting excessive disk space. For example on a development subsystem, TSQTY and IXQTY may be set to 48 KB for track allocation. The use of the default for PRIQTY will be recorded in the associated PQTY column as -1 in the SYSIBM.SYSTABLEPART or SYSIBM.SYSINDEXPART catalog table.

**Implementation details**

The main objective of Automatic Space Management is to avoid the situation where a DB2 managed page set reaches the VSAM maximum extent limit of 255 before it can reach the maximum data set size, which may be less than 1 GB, 2 GB, 4 GB, 8 GB, 16 GB, 32 GB, or...
64 GB. The actual secondary allocation quantity applied will not be reflected in the Catalog, and will not exceed DSSIZE or PIECESIZE.

For an index, the user specified PIECESIZE, which limits data set size and can start as low as 256 KB. It allows for a managed DB2 page set to reach the maximum page set size before running out of extents. This will help to reduce the number of out-of-space conditions, improve user productivity, and additionally avoid the performance penalty associated with small extent sizes.

Two sliding scales can be used, one for 32 GB and 64 GB data sets, and one for the rest (less than 1 GB, 1 GB, 2 GB, 4 GB, 8 GB, 16 GB). Maximum data set size can be determined based on DSSIZE, LARGE and PIECESIZE specification for the object. Both sliding scales will allocate an increasing secondary quantity size up to 127 extents and a constant number thereafter. The constant is 559 cylinders for the 32 GB and 64 GB data sets, and 127 cylinders for the rest.

This approach of sliding the secondary quantity minimizes the potential for wasted space by increasing the extents size slowly at first, and it also avoids very large secondary allocations from extents 128-255, which will most likely cause fragmentation where multiple extents have to be used to satisfy a data set extension. The solution will address new data sets that will be allocated, and also existing data sets requiring additional extents. Therefore, in the case of an already installed SAP system, after upgrading to DB2 V8 and enabling the new ZPARM, the full benefits of this enhancement can be realized immediately.

When MGEXTSZ is enabled, if the SECQTY value specified by the user is greater than 0, the actual secondary allocation quantity will be the maximum of the calculated quantity size using sliding scale methodology, and the SECQTY value specified by the user. When a page set spills onto a secondary data set, the actual secondary allocation quantity will be determined as described and applied, and the progression will continue. Prior to DB2 V8, the PRIQTY would have been used.

**Scenarios and benefits of Automatic Space Management**

One of the major benefits of Automatic Space Management is preventing disruptions to all users, based on the unexpected activity against one or more database tables in an SAP system. Most often this will occur in situations such as data loads prior to the productive usage of the SAP system, or implementation of additional functions within the SAP application.

It is possible, but much more unlikely, that out-of-space conditions will occur during normal operations. In this case, the growth would be expected to have some steady rate. Normal monitoring of data set extents across all objects would normally catch this growth, and corrective action taken prior to objects reaching 255 extents.

Because the MGEXTSZ ZPARM can be enabled and disabled, it is possible that it could be set to YES only during known periods of growth. This would be particularly useful before the first execution of loading data into the SAP system for a new functional area. In this way it would prevent what commonly occurs in this situation, where the loads run until one object reaches 255 extents.

The load process abends, possibly requiring a non-trivial time to back out, depending on how it is written in regard to commit frequency. The database object then needs to be resized through ALTER commands and subsequent REORG utility execution. The data loads then continue, and often this is just long enough until the next space-related error happens, and the preceding steps are repeated. Typically in an SAP system, this can occur with indexes on tables, as there are many indexes, and frequently their default allocation results in growth to 255 extents very quickly.
In this situation, where the application knowledgeable personnel are unable to comprehensively specify which database objects are expected to receive what volume of data, this new DB2 functionality will remove this frustration. It should be stressed again that subsequent action will most likely need to take place after the data load phase completion, with table spaces and indexes that are in a non-trivial number of extents and/or disorganized being REORGed. But this can happen in a much more controlled and less time critical manner. Finally, while it may be possible to deactivate the new MGEXTSZ parameter at times where no exceptional activity is expected, it is highly recommended that the parameter is active (that is set to YES) at all times. The benefit of reducing user impact due to out-of-space conditions greatly outweighs the slight possibility of using additional space unnecessarily. The algorithm used ensures that while objects are in small numbers of extents, the potential wasted space is small. And with good practices in place to detect objects growing into larger numbers of extents, and subsequently addressing them, the best of all worlds is achieved.

**Tips:**
- If SECQTY is specified by the user as 0 to indicate do not extend, this will always be honored. So for certain circumstances, especially DSNDB07 workfiles, where users deliberately set SECQTY=0 to prevent data set extension, this will remain the case even with the MGEXTSZ parameter enabled.
- While the activation of the new MGEXTSZ DSNZPARM has to be considered in terms of the benefits and drawbacks discussed in this section, we feel that all SAP implementations would benefit from turning it on, and leaving it on.
  
  This is in contrast with what would happen if MGEXTSZ was set to NO. In that case, the data set size would reach 2565 cylinders, at which point subsequent activity requiring more space would fail, as the object is at the maximum VSAM limit of 255 extents. These failures occur despite the fact the DSSIZE parameter of the object indicates it should be able to grow to 4 GB or 22,000 cylinders.

### 11.12 Flexible implementation with zSeries

The flexibility of the SAP on zSeries solution gives customers the freedom of choice to pick and choose among the available implementation options. The SAP database server always runs on zSeries in an z/OS environment, but the SAP application servers can run on any certified platform, preferably Linux for zSeries, AIX or Windows. Customers can connect multiple application servers on different platforms to one database server on zSeries, depending on their requirements.

Figure 11-10 on page 216 illustrates a structural view of the implementation of the three-tier SAP architecture with zSeries.
The implementation is described as follows:

- The SAP database server runs on DB2 UDB for z/OS.
- The SAP Web Application Server can run on different servers and operating systems.
- An SAP Web Application Server on Linux for zSeries can run on the same zSeries machine in a separate LPAR natively, under z/VM®, or even on a separate, physical machine.
- Connectivity from SAP Web Application Server to the database server occurs with TCP/IP over the network. To accelerate communication, a cross-memory or HiperSocket connection can be used if the application and database server are located on the same zSeries machine.
- The presentation layer is implemented as an SAP GUI running on Windows, as a Java GUI, or on a Web browser.

To offer some practical examples of how customers can set up their environment, we chose three scenarios with typical implementations driven by the following major motivators:

- Scenario 1 - High-end database environment, with only the database server on zSeries
Scenario 2 - Ultimate availability of the SAP solution
Scenario 3 - Reduce TCO and increase qualities of service with server consolidation

Let's discuss these scenarios in more detail now.

**Scenario 1: Database server on zSeries**

The database is the most critical component of every SAP system. Accordingly, many companies decide to run the SAP database server on zSeries because they have special requirements that call for the strengths of DB2 UDB for z/OS.

If customer cannot or does not want the additional benefit of consolidation capabilities and qualities of service of the SAP application servers on zSeries, the entire workload of the SAP application server is typically run on other platforms, such as AIX or Windows.

In case only the database server runs on zSeries, the implementation of this scenario could look like Figure 11-11.

![Figure 11-11 Database Server DB2 on zSeries](image)

Typically, companies run the SAP database server on zSeries for the following or similar reasons:

- High availability of the database is considered critical
- Data volumes call for a very large database
- Need for high volume throughput
- Non/disruptive operation
- High scalability of database server is required
- Saving physical disk space with data compression
- Benefit of database server consolidation capabilities
The SAP database server on zSeries can be implemented with DB2 either on a single system or on Parallel Sysplex, with multiple data sharing members for high availability and scalability.

**Scenario 2: Continuous operation of business applications**

The business success of many companies depends on the continuous availability of their mission-critical IT applications. Complex system landscapes require automation of IT operations. This scenario describes a solution that incorporates centralized, automatic operations and the highest availability of the whole SAP environment.

Several elements of an SAP system must be protected to achieve high availability:

- Database server
- Central services
- File systems
- Network

A system landscape of several SAP systems becomes increasingly complex. To keep the entire landscape up and running, automation is needed to avoid expensive and slow manual interaction.

![Architecture for an IBM/SAP zSeries high availability solution](image)

*Figure 11-12  Architecture for an IBM/SAP zSeries high availability solution*

In z/OS, the implementation of System Automation is very sophisticated. It exploits Parallel Sysplex features, allows for DB2 data sharing with sysplex failover, and enables automatic
disaster recovery based upon geographically dispersed Parallel Sysplex (GDPS). Because of the high availability features of the file system services (shared hierarchical file system, for example), z/OS Parallel Sysplex becomes a highly available server platform.

System Automation not only provides the means for the implementation of a high-availability system, but also includes all features needed to streamline daily operations, including for example, features of automated operation and monitoring of the SAP components. A failure to include these features affects the system's availability.

System Automation also provides for continuous availability while reducing or even eliminating operating or handling problems.

The high-availability solution with zSeries is available with System Automation for OS/390 and z/OS, as well as with System Automation for Linux. For detailed information concerning high availability with SAP systems under zSeries, refer to “Related publications” on page 259.

**Scenario 3: Server consolidation with zSeries**

Server consolidation is the appropriate solution when customers want to increase manageability and lower the TCO of an SAP implementation.

With zSeries, customers have excellent opportunities to consolidate SAP systems by logical partitioning on zSeries, z/VM virtualization technology, and intelligent workload management. Companies decide to put several SAP systems on zSeries because they want to benefit from the consolidation capabilities and leverage the qualities of service of the zSeries solution.

The server consolidation capabilities of zSeries, including z/VM, allow many SAP systems to run on a single zSeries machine, which leads to significant savings and operational advantages because of the following:

- High level of capacity and resource utilization
- Easy handling of unpredictable workload peaks
- Reduced complexity and better manageability
- Easy setup and cloning of new SAP instances almost instantly (SAP application server provisioning)

The server consolidation scenario on zSeries could be that the SAP database server runs on z/OS in one LPAR, but the application servers run on Linux for zSeries in one or more separate LPARs, most often using z/VM.
With the cloning mechanisms of z/VM, new Linux servers can be created very quickly. New SAP instances can be added for testing and development within minutes. When they are no longer used, resources can be returned to the pool for redeployment, providing more efficient use of system resources.

With all server partitions running on the same machine, HiperSockets™ can be used for internal connections between the application servers and the database server. Doing so improves overall system performance, significantly reducing the batch window or database update times.
SAP BW 3.5 OSS Notes

This appendix contains an overview from the most important SAP BW OSS notes.

Table A-1  SAP BW 3.5 OSS Notes

<table>
<thead>
<tr>
<th>Note Number</th>
<th>Title</th>
<th>Further description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runstats</td>
<td></td>
<td></td>
</tr>
<tr>
<td>775318</td>
<td>Reduce Database Statistics</td>
<td>DB2_REDUCED_DB_STATS=X</td>
</tr>
<tr>
<td>732078</td>
<td>Non-optimal access to time dimension</td>
<td>Better distribution after support package installation</td>
</tr>
<tr>
<td>748159</td>
<td>Parallel statistics calculation</td>
<td>DB2_ANALYSE_PARALLEL_DEGREE=3</td>
</tr>
<tr>
<td>778437</td>
<td>DB statistics compilation in process chain</td>
<td>Add ABAP program in process chain for BW statistics</td>
</tr>
<tr>
<td>Partitioning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>752725</td>
<td>Subsequent partitioning of InfoCubes</td>
<td>Z_SAP_BW_CONT_TO_PART_CUBE_DB2</td>
</tr>
<tr>
<td>485878</td>
<td>Partitioning the PSA tables</td>
<td>Outdated with DB2 V8</td>
</tr>
<tr>
<td>DB2 z/OS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>390016</td>
<td>DB settings and performance</td>
<td>Star Join, Query Parallelism, Additional DB2 Parameters, PSA Partitioning/Compression, and so on</td>
</tr>
<tr>
<td>717935</td>
<td>FTP replacement by Stored Procedures</td>
<td>Description of Stored Procedures Implementation for SAP BWi</td>
</tr>
<tr>
<td>SAP BW additional</td>
<td></td>
<td></td>
</tr>
<tr>
<td>567745</td>
<td>Composite note BW 3.x performance: DB specific settings</td>
<td>Reference to z/OS notes 192658, 568998</td>
</tr>
<tr>
<td>Note Number</td>
<td>Title</td>
<td>Further description</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>192658</td>
<td>Setting basis parameters for SAP BW</td>
<td>R3 Parameters for SAP BW</td>
</tr>
</tbody>
</table>
| 568998      | Size of partition: Definition of DSIZE in RSADMIN                    | DB2_FACT_DSSIZE  
DB2_PSA_DSSIZE                                                                 |
| 567746      | Composite note BW 3.x performance: Query & Web Applications         | This composite note deals with different subjects (reference to other notes)        |
| 567747      | Composite note BW 3.x performance: Extraction & loading              | This composite note deals with different subjects (reference to other notes)        |
| 409611      | Examples of packet size dependency ROIDOCPRMS                        | General formula for packet size = MAXSIZE * 1000/transfer structure size, but not more than MAXLINES |
|             | **AIX**                                                              |                                                                                     |
| 146289      | Parameter recommendations for 64-bit kernel                          | Different recommendations concerning AIX, R3, and so on.                             |
| 323816      | User limits on AIX                                                  | For example, Soft CPU=-1                                                            |
|             | **General**                                                          |                                                                                     |
| 703480      | Do not partition aggregates (small aggregates)                       | DB2_AGGR_NOT_PARTITIONED=1                                                          |
| 688133      | Customer specific runstats options                                   | Changing CLUSTERRATIO in RSADMIN                                                    |
| 130696      | Performance trace in BW                                             | Analysis of RSDDSTAT                                                                |
| 633832      | Performance analysis for DB2 database                                | RSDU_ANALYZE_TOOL_DB2                                                               |
| 536074      | Buffer pool for fact table and dimension table                       | RSADMIN                                                                             |
| 757329      | Reporting and DB2 accounting                                         | With DB2 V8 user name, SAP transaction name and SAP program name are available      |
Glossary for SAP BW from September 2004

In this appendix, we include the Glossary from SAP BW. You can also find this SAP BW Glossary on the SAP BW homepage (see "Related publications" on page 259).

**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.

**Administrator Workbench**
Tool for controlling, monitoring and maintaining all those processes involved in data procurement and processing within the Business Information Warehouse.

**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 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 BW 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 on-line links and distribute them by email or publish them to the Enterprise Portal.

**BEx Download Scheduler**  
Assistant for downloading precalculated Web templates from the BW 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 BW.

**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 BW 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
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 Websites with BW contents. With the BEx Web Application Designer you can create an HTML page that contains BW-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 BW-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 BW, via BW'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 BW, via BW'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 BW, via BW'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 BW, via BW'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 BW, via BW'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 email 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:

- BW 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 screen: 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

**BW Document Repository Manager**

Tool in the Knowledge Management of SAP Enterprise Portal.

The BW Document Repository Manager regulates read and write access to documents that are stored in SAP Business Information Warehouse.

**BW Metadata Repository Manager**

Tool in the Knowledge Management of SAP Enterprise Portal.

The BW 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 BW Metadata Repository Manager also allows access to online links generated using BEx Information Broadcasting.

**characteristic**

Type of InfoObject.

An evaluation group such as company code, product, customer group, fiscal year, period, or region.

Characteristics provide classification possibilities for the dataset. An InfoCube generally contains only a partial quantity of the characteristic values from the master data table at a time. The master data includes the permitted values for a characteristic, also called characteristic values. Characteristic values are discrete names.

The characteristic "Region" has the following specifications, for example:

- North
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 screens. This means that you no longer have to schedule application processes individually.

The process chains maintenance screens 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 "toolable" -- 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 homepage.

**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 (i.e. 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.

**Crystal Enterprise**
Server component that is required to publish formatted reports created with the Crystal Reports Designer on the basis of BW 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 BW server.

**Crystal Report**
BW object type.

Report definition for a formatted report that has been created using the Crystal Reports Designer on the basis of a BW query. The report definition is saved in SAP BW and published to the Crystal Enterprise server.
**Crystal Reports Designer**
Design component for generating a Crystal Report, which contains the report definition (layout).

**data manager**
Part of the OLAP processor that controls the data bank access that arises from the definition of a query.
Part of Warehouse Management that writes data into the data bank.

**data mart interface**
Enables the user to update data from one data target to another.

The data mart interface allows the user to update data within a BW system (Myself system) and also between several other systems. If several BW systems are used, the system delivering the data is called the source BW and the system that is receiving data is called the target BW. The individual Business Information Warehouses in this type of setup are called data marts.

**data provider**
Object that delivers data for one or more Web items. A data provider has the same navigation status as a query at a particular point in time. The start view of a data provider corresponds to the query view. By navigating through the data or parameterizing the call, you change the status of the data provider.

**data provider - information**
Web item that can be used for generating the result data or navigational state of a data provider in XML format.

The Web item is not visualized in the Web application.

**data request**
Denotes
1. The request that is sent to the source system by the scheduler
2. The quantity of data and information that results in BW and in the source system through this request
3. The loading procedure

**data set**
Record of values that belong together in a relational database table. A data set is saved in the relational database management system (DBMS) as a line.

**data target**
Data load-relevant view of a BW object with a physical data store.

InfoCubes, ODS objects, and InfoObjects (characteristics with attributes, texts, or hierarchies) are among these types of objects. You distinguish between normal data targets for which no queries can be created or executed, and data targets for which queries can be defined. The latter type of data targets are also called InfoProviders.

**data warehouse**
A data collection or database that is created by integrating various datasets and data from external sources.
Data warehouses provide users with a global view of the data that has many useful applications.

**DataSource**
Object that makes data for a business unit available to SAP BW.

From a technical perspective the DataSource contains a number of logically-related fields that are provided in a flat structure for data transfer to SAP BW.

**DB connect**
Enables connections to different relational database management systems and the transfer of data from tables or views from these database management systems into the SAP Business Information Warehouse.

**delta process**
Extractor property. It specifies how the data is to be transferred. As a DataSource attribute, it specifies how the DataSource data is to be forwarded to the data target. The user can determine, for example, with which data targets a DataSource is compatible, how the data is to be updated, and how serialization is to take place.

**delta queue**
Data storage in a BW source system.

The data records are either written automatically into the delta queue by using an updating process in the source system or by using an extraction by means of a function module after a data request from BW.

The data is transferred into BW with a delta request from the BW Scheduler.

**device recognition**
Recognition by the server of mobile devices for device-specific adjustment for displaying Web applications. Using device recognition, the system decides whether a Web application or a mobile application (WAP or PDA report) is to be sent back to the client.

**dimension**
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).

**dropdown box**
Web item that places characteristic values for filtering in a Web application in a dropdown box.

**elementary test**
Part of a test that can not be split further into sub-tests. An elementary test checks related logical objects for consistency.

**event**
Signal to the background processing system that a certain status has been reached in the SAP system. The background processing system then starts all processes that were waiting for this event.
**event collector**
A number of events that have been successfully completed independently of each other, and to which the background processing is to respond.

The event collector corresponds to the and-process and the process chain maintenance. If an application process is scheduled using an event collector, it starts when all events for the preceding process are successfully triggered.

**exception**
A deviation from a defined threshold value.

Exceptions are defined in the Query Designer and can be evaluated either online (in BEx Web Applications, BEx Mobile Applications, or in the BEx Analyzer), or in the background, using the Reporting Agent.

Exceptions are defined by setting the threshold values or intervals, which are given a priority (bad, critical, good), and by determining the areas of the query for which the exception is valid. In the evaluation, the areas of the query lying above or below the threshold are marked in different colors. Up to nine shades of the red, yellow, and green traffic light colors are used here.

**exception reporting**
Highlighting of deviating and critical objects for a query.

Deviations from defined threshold values (exceptions) are highlighted in a different color so that unusual deviations from the expected results can be seen at a glance.

**exor-process**
Collation process in process chain maintenance.

When you use an exor-process in the process chain maintenance, the application process starts when the first event of the previous processes has been triggered successfully.

**external system**
SAP-external data source for a BW system through which the transfer of data and metadata occurs using staging BAPIs.

**fact table**
Table in the center of an InfoCube star schema.

The data part contains all key figures of the InfoCube and the key is formed by links to the entries of the dimensions of the InfoCube.

**fill**
An aggregate is filled when data is loaded to it for the first time after activation. The active aggregate, filled with data is then used in reporting and can have data added to it in a roll up.

**filter**
Web item that displays those filter values for a query view in a Web application that you created when navigating. Also allows you to select single values.

**foreign key relationship**
The relationship that exists between particular fields of a table A, and the key fields of a table B, when all the foreign key values that occur in A correspond to the primary key values in B.
The BW analysis and repair environment checks this as the criteria for referential integrity.

**formatted reporting**
Structure for reports that use master data, ODS objects, and multi-dimensional InfoProviders. Formatted reporting has all of the options for formatting reports (for example, font, font size, colors, graphics, and styles), and allows you to place report elements anywhere on the screen to the accuracy of one pixel (no grid positioning). The focus is on formula-based reports and the printing of reports.

There are no analytical functions: Interaction options are introduced at the time of designing the report.

**generation template**
Template, from which a program is generated.

A generation template is used when the desired program can not be generically written and must be suitably written for each special situation anew.

**generic navigation block**
Web item that retrieves data from a query view and displays it in the form of a table. All characteristics and structures of the query view are listed in the table and their filter values are displayed. You can change the navigation status in the block by:

- Placing characteristics and structures in an axis (rows or columns) or removing them from an axis.
- Filtering by single values or removing filters.

**hierarchical filter selection**
Web item that generates a hierarchical filter selection from the hierarchy or structure of a characteristic.

Hierarchy nodes appear as menu entries and can be set as filters.

**hierarchy**
The organization of the characteristic values of a characteristic into a tree structure.

Example: A hierarchy for cost centers that are combined in cost center groups.

In reporting, hierarchies for characteristics can be set in the following ways:

- As presentation hierarchies for a characteristic, if it is to be displayed as a hierarchy
- As a way of selecting particular characteristic values if a characteristic is to be restricted to a hierarchy or a hierarchy node

Hierarchies can be loaded into the BW system or created in the BW system for hierarchy basic characteristics. They can be used across InfoProviders.

**hierarchy attribute**
Attribute that describes the properties of an entire hierarchy (for example, level table type: specifies the form that a level table takes).

**IBIOlap**
An interface provided by the BI Java SDK and implemented by all OLAP connectors which serves as an entry point to interfaces that support access to multidimensional metadata and queries.
**IBIRelational**
An provided by the BI Java SDK and implemented by all relational connectors which serves as a point of entry to a set of interfaces that provide access to relational metadata and queries.

**impact analysis**
Group of methods that assess the effect of changes made to any dependent objects when an object is checked and activated.

The impact analysis ensures that the objects are consistent.

Examples:
- A navigation attribute is deleted from a characteristic. When the characteristic is activated, all the InfoCubes that use this characteristic are set to inactive. The InfoCubes have to be reactivated (without the navigation attribute) to make them consistent.
- In other cases, dependent objects are adjusted immediately if this is not possible without manual changes.

**INative**
An optional interface defined in the Portal Connection Framework API which can be implemented by a connector. INative enables you to access the connected EIS via an API that is tailored specifically for that underlying EIS. The interface returned depends on the connected EIS.

**InfoArea**
Groups meta-objects together in the Business Information Warehouse:

Every data target is assigned to an InfoArea. The resulting hierarchy is then displayed in the Administrator Workbench.

In addition to their property as a data target, InfoObjects can also be assigned to different InfoAreas using InfoObject catalogs.

**InfoCatalog**
This is a tree-like structure in the Administrator Workbench that displays Business Information Warehouse workbooks. The various InfoCatalog trees contain:

- Workbooks that SAP delivers
- Workbooks that can be used in an enterprise
- Workbooks that are used by certain user groups
- Workbooks that an individual user is allowed to use
- Workbooks (favorites) that a user has put together

The structure of the subtrees can be freely defined by the administrator.

A user accesses his or her InfoCatalog workbooks using the Business Explorer Browser.

**InfoCube**
Objects that can function both as data targets as well as InfoProviders.

An InfoCube 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.
An InfoCube is a quantity of relational tables that are created according to the star schema. A large fact table in the center, with several dimension tables surrounding it.

**InfoCube tree**
A proposal list of existing InfoCubes.

**InfoObject**
Business evaluation objects (for example, customers or sales) are called InfoObjects in BW.

InfoObjects are subdivided into characteristics, key figures, units, time characteristics, and technical characteristics (such as request numbers).

**InfoPackage**
This describes which data in a DataSource should be requested from a source system. The data can be precisely selected using selection parameters (for example, only controlling area 001 in period 10.1997).

An InfoPackage can request the following types of data:
- Transaction data
- Attributes for master data
- Hierarchies for master data
- Master data texts

**InfoPackage group**
Combines InfoPackages that are logically related.

**InfoProvider**
Analysis-relevant view of a BW object for which queries in SAP BW can be created or executed.

There are two types of InfoProviders. One type includes objects that contain physical data. These are known as data targets, such as InfoCubes, ODS objects, and InfoObjects (characteristics with attributes, texts, or hierarchies). The other type includes objects that display no physical data storage, such as InfoSets, RemoteCubes, SAP RemoteCubes, and MultiProviders.

**Information Consumer Pattern**
A user interface pattern developed for end users without special knowledge of the SAP Business Information Warehouse to simplify analysis of displayed data.

**InfoSet**
A semantic view of ODS objects and InfoObjects (characteristics with master data) that allows you to create reports on these objects, particularly on the joins between these objects.

Unlike the classic InfoSet, this view of data is BW-specific. In the InfoSet builder, InfoSets are created and changed. InfoSets allow you to use the query designer to define reports.

**InfoSet Builder**
Tool for creating and changing InfoSets using repository objects from BW (InfoObjects with master data and ODS objects).
**InfoSource**  
A quantity of all available data for a business event, or type of business event (for example, Cost Center Accounting).

An InfoSource is a quantity of information that has been grouped together from information that logically belongs together. InfoSources can contain transaction data or master data (attributes, texts, and hierarchies).

An InfoSource is always a quantity of InfoObjects that belong together logically. The structure where they are stored is called a communication structure.

**InfoSpoke**  
Object for the export of data within the open hub service.

Defined in the InfoSpoke are the following:
- from which open hub data source the data is extracted,
- in which extraction mode the data is delivered,
- and into which open hub destination the data is delivered.

**J2EE Connector Architecture**  
A standard architecture from Sun designed for connecting J2EE servers with EISs. The architecture defines a set of contracts, such as transactions, security, and connection management, that a connector has to support to plug in to an application server.

JCA provides an API for connecting to heterogeneous data sources in a consistent manner. The BI Java Connectors are JCA-compliant.

**Java Metadata Interface**  
An extensible metadata service for the Java platform that provides a common Java programming model for accessing metadata. JMI defines a Java mapping for the Meta Object Facility (MOF) specification from the Object Management Group (OMG). The BI Java SDK uses JMI mapping to render its query and metadata models into Java APIs.

For more information, see [http://jcp.org/jsr/detail/40.jsp](http://jcp.org/jsr/detail/40.jsp).

**Java Metadata Interface Service**  
Any system that provides a JMI-compliant API to access its public metadata. The BI Java Connectors expose metadata of the underlying EIS via JMI services.

**key figure**  
Values or quantities.

In addition to the key figures saved on the database, you have the option of defining derived (calculated) key figures in the query definition in the Business Explorer. Such key figures are calculated using a formula from the key figures of the InfoCube.

Examples of key figures include the following:

Sales revenue, fixed costs, sales quantity, or number of employees.

Examples of derived key figures include the following:

Sales revenue per employee, variance as a percentage, or contribution margin.
**key figure model**
Type of data modeling in which every business key figure corresponds to a technical key figure of the data basis.

Therefore, an individual key figure within a key figure model contains values, which reflect a unique business situation (for example sales, number of employees, maintenance costs).

A single data record in the key figure model is made up of the fields for the classifying characteristic (for example fiscal year, version, region) and one field for every key figure contained in the model. In this way, such a data record is more extensive than a data record within an account model. However, less data records are required for the same data volume.

**key figure overview**
Web item that displays a tabular configuration of important business key figures from the Strategic Enterprise Management (SAP SEM) component.

**label**
Web item with which you display characteristic, attribute, and structure component descriptions, and with which you can set a link to the context menu for the characteristic, attribute, or structure component.

**list of conditions**
Web item that lists the available conditions and their respective status (active/not active/can be used/not used) for a query view in a Web application.

**list of documents**
Web item that displays context-sensitive documents on transaction data used in the Web application, in the form of a list.

It is also possible to create new documents in the Web application itself.

**list of exceptions**
Web item that lists the available exceptions and their status (active/not active) for a query view in a Web application.

**map**
Web item containing all the information needed to display a geographical map in a Web application.

**map layer**
Object of the map Web item that contains all the information about a particular layer of a map.

You use this information to determine the various display formats (color shading, bar charts, pie charts) and their settings for the map layer.

**master data ID**
Internal key of type INT4 that you use for master data for master data-bearing characteristics, especially for hierarchy nodes and for characteristic names.

Master data IDs (SIDs) and characteristic values are stored in master data tables (SID tables).

Information about time-independent or time-dependent master data, which is stored in a P table or a Q table, is saved again in an X or Y table, using SIDs instead of characteristic values.
**master web item**
Template for a Web item.

The master Web item determines the type of the Web item (for example, table, filter, chart, map, and so on) and includes default values for the attributes of each Web item. The various master Web items are available on the "Standard" tab page in the BEx Web Application Designer, or in the BEx Web Application Wizard. You choose a master Web item from this list, assign a Data Provider to the Web item, and process the attributes. Thus, you create your own Web item, which you add to your Web template or store in the library so that it can be reused in the future. Web items that are stored in the library can take on the character of master Web items and be used as a template for additional Web items.

Example:

You take the 'Chart' master Web item and create various master Web items for your library: Bar charts, column charts, pie charts, tachometers, and so on.

**master Web template**
A Web template that is copied and used as a template for a new Web template.

**MDX**
Multidimensional Expressions

Query language for queries about data that is saved in multidimensional cubes.

**metadata**
Data about data.

Metadata describes data models. According to the MOF (Meta Object Facility) standard for the OMG (Object Management Group) there are the following layers:

M3 Meta-meta-meta data = meta-meta model
M2 Meta-meta-data = meta-model
M1 Meta-data = model
M0 Data

**metadata API**
A set of interfaces provided by the BI Java SDK which expose the metadata of a given data source. The SDK includes two metadata APIs, both generated via JMI from their respective metadata models:

- OLAP Metadata API, for exposing metadata in an OLAP data source
- Relational Metadata API, for exposing metadata in a relational data source

**metadata model**
An abstract language for expressing metadata. The BI Java SDK leverages CWM metadata models (metamodels), and the following two CWM packages in particular:

- org.omg.cwm.analysis.olap --> basis of the SDK's OLAP Metadata Model, for expressing the metadata of a multidimensional data source
- org.omg.cwm.resource.relational --> basis of the SDK's Relational Metadata Model, for expressing the metadata of a relational data source
The SDK's metadata models also rely upon reference classes from CWM's Foundation and Objectmodel layers.

**metadata repository**
Provides central access to information about metadata objects in the Business Information Warehouse:
- Information about active objects in the system (activated objects)
- Information about SAP delivery objects in the system (Business Content)

The metadata repository provides the following functions:
- Metadata search
- HTML page exports
- Graphical object display

Further functions for metadata:
- Exchange metadata between different systems (transport connection)
- Create documents for metadata objects and select them to be displayed as on-line documentation (document management)

**Metamodel Repository**
SAP-specific implementation of a repository for metadata models and metadata.

The Metamodel Repository is named after the metamodel layer (the meta- metadata layer or M2layer) of the MOF specification of the Object Management Group (OMG). This is because the main focus of SAP-specific implementation rests on this layer.

**Metamodel Repository Designer**
Graphical user interface for the Metamodel Repository. It is a plug-in for the Eclipse Java development environment.

**mobile application**
Web application on a mobile device with an online connection to a BW system.

**MOF**
Meta-Object Facility
- One of the OMG-recognized (Object Management Group) standards which
  - determines guidelines for the definition of metadata models and provides
  - programming tools for saving and accessing metadata in repositories.

The MOF standard is integrated in XMI.

You can find the current MOF specifications on the OMG homepage.
- A metadata service which abides by the MOF specifications.

**MOLAP**
Multidimensional Online Analytical Processing

Multidimensional data storage in special data structures that are based on arrays or cubes.

MOLAP is used mostly in comparison with or as an alternative to ROLAP.
**MOLAP aggregate**
Aggregate of a MOLAP cube.

The aggregate is stored in a MOLAP repository, as is the MOLAP cube itself.

**MOLAP cube**
Basic cube, whose data is stored physically in a MOLAP store.

**monitor**
The monitoring tool of the Administrator Workbench.

Using the monitor, you can oversee the data request and processing in the Business Information Warehouse.

**multidimensional data**
Data in dimensional models suitable for business analytics. In the BI Java SDK, we use the term "multidimensional data" synonymously with "OLAP data."

**multi-planning area**
Planning area that combines the characteristics and key figures of several planning areas.

A multi-planning area does not contain its own data, instead it is completely based on the data of the standard planning area, which is defined in the same system. The planning areas combined in a multi-planning area can be identified by a characteristic that is generated automatically within the multi-planning area and can be referred to for selection purposes.

**MultiProvider**
Type of InfoProvider that combines data from several InfoProviders and makes it available for reporting.

The MultiProvider itself contains no data; its data comes exclusively from the InfoProviders on which it is based (collated using a union operation). You can assemble a MultiProvider from different combinations of InfoProviders.

MultiProviders, like InfoProviders, are objects or views that are relevant for reporting.

**my portfolio**
KM navigation iView that contains the "broadcasting" layout. This is tailored to the specific needs of users who use Business Intelligence content in the Enterprise Portal.

The "my portfolio" iView links to the user-specific KM folder in which users store their personal documents.

**Myself system**
System that is connected to itself for data extraction by using the data mart interface.

Thus, the user can update data from one data target into other data targets.

**navigation**
Analysis of the InfoProvider data by displaying different views on the data of a query or a Web application.

With the aid of the various navigational functions, such as:
you can generate different views of the data (query views) that are presented in the results area of the query or Web application. Changing views is considered to be navigation.

**navigation attribute**
Attributes that you can select in the query.

**node**
Objects that create a hierarchy.

A node can have subnodes. We differentiate between two types of nodes:

- Postable nodes
- Unpostable nodes

**node attribute**
Attribute that every node in the hierarchy possesses (for example, the date fields DATETO and DATETIME, if the hierarchy structure is time-dependent).

**Object Linking and Embedding Database**
A set of Component Object Model (COM) interfaces from Microsoft that provide applications with uniform access to data stored in diverse information sources. OLE DB also provides the ability to implement additional database services.

**Object Management Group**
An open membership, not-for-profit consortium that produces and maintains computer industry specifications for interoperable enterprise applications.

For more information, see http://www.omg.org/.

**object tag**
Placeholders in a Web template that start with `<object>` and end with `</object>`. You use object tags to generate data providers and Web items in a Web template.

**ODS object**
Object that stores consolidated and cleaned transaction data on a document level.

An ODS object describes a consolidated dataset from one or several InfoSources. This dataset can be evaluated using a BEx query. An ODS object contains a key (for example, document number, position) as well as data fields that, as key figures, can also contain character fields (for example, customer). You can update ODS object data into InfoCubes or other ODS objects in the same system or a different system using a delta update. In contrast to multi-dimensional data stores for InfoCubes, data in ODS objects is stored in transparent, flat database tables.

**OLAP**
A system of organizing data in a multidimensional model that is suitable for decision support. OLAP is the analytical counterpart of OLTP, or Online Transactional Processing. SAP BW is an OLAP system.
**OLAP Command Processor**
Part of the BI Java SDK's OLAP Query API, an interface that makes it easier to use the API by hiding the complexity of the underlying OLAP Query Model. With this interface, you can create and manipulate complex multidimensional queries with simple commands.

**OLAP data provider**
A component that provides data in multidimensional views and metadata compatible with the BI Java SDK's OLAP Metadata Model.

**OLAP Metadata API**
A set of interfaces provided by the BI Java SDK for accessing the metadata of an OLAP data source. Generated via JMI from the SDK's OLAP Metadata Model.

**OLAP Metadata Model**
A model provided by the BI Java SDK that exposes business data in a multidimensional format which specifically supports data analysis. Based on the Common Warehouse Metamodel OLAP package.

**OLAP Query API**
A set of interfaces provided by the BI Java SDK that let you define queries against an OLAP server. The API is generated via JMI from the OLAP Query Model, based on metadata provided by the OLAP Metadata Model, and includes the simplified OLAP Command Processor.

**OLAP Query Model**
An abstraction layer, or model, in the BI Java SDK designed for formulating OLAP queries independently of data source-specific query APIs. The model is based on the CWM-compliant metadata provided by the OLAP Metadata Model.

**OLAP reporting**
Reporting based on multidimensional data sources (InfoProvider).

OLAP reporting allows you to analyze several dimensions at the same time (for example, time, place, or product). The aim of OLAP reporting is to analyze key figures, such as for an analysis of the sales figures for a certain product in a particular time period. The business questions that you have about this product in this period are formulated in a query. The query includes the key figures and characteristics that contain the data that is necessary for analyzing or answering your questions. The data is displayed in a table and is the starting point for a more detailed analysis to answer a number of different questions.

A range of interaction options, such as sorting, filtering, swapping characteristics, and recalculating values, allows you to navigate flexibly through the data during the runtime.

In the Business Information Warehouse you can analyze the data in the following areas of the Business Explorer:
- In the BEx Analyzer in the form of queries
- In BEx Web applications

**OLAP Table Model**
A companion to the BI Java SDK's ResultSet API that facilitates the rendering of a multidimensional dataset into a two-dimensional matrix.
**OLAP trace**
Record of system activities in a log.

**open hub data source**
An object that delivers data to the open hub service.

The BW objects InfoCube, ODS object and InfoObject (attribute or text) can be used as open hub data sources.

**open hub destination**
An object within the open hub service that contains all information about a source system for data in an InfoProvider.

**open hub service**
A service that enables you to share data from a SAP BW system with non-SAP data marts, analytical applications and other applications.

The open hub service ensures controlled distribution and the consistency of data across several systems.

**Operational Data Store**
A data administration layer that saves data in flat, transparent tables.

This layer can make data available in real time and enables operational reporting.

**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 BW, 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.

**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 BW 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 BW 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.

**Q table**
Master data table for time-dependent master data. The Q table has the same fields as the P table.

**query**
Collection of a selection of characteristics and key figures (InfoObjects) for the analysis of 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.

You define a query in the BEx Query Designer, in which you select InfoObjects or reusable structures for an InfoProvider and specify the view of the data (query view) by distributing them to filters, rows, columns, and free characteristics. You used this saved query view as a basis for data analysis and reporting in the BEx Analyzer, in BEx Web applications, in BEx Mobile Intelligence, or in formatted 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 BW. 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, he/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**

You can find the current SOAP specification on the World Wide Web Consortium homepage 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 on 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 BW, or accessed directly via a RemoteCube.
**UD Connect source**
Instance that can be addressed as a data source of SAP BW using the BI Java Connectors.

**UD Connect source object**
Multidimensional or relational data store in the UD Connect source for example, table or cube.

**Unified Modeling Language**
UML is the OMG-recognized (object management group) standard language for the semantic analysis of objects and for the design of object-oriented models with the help of graphic tools.

The UML standard is integrated in XMI.


**user interface**
Visual presentation in which a software program is presented to the users on the screen.

The user interface is made up of display and operation elements which give the users information about the program status and the currently available interaction possibilities. An ergonomically successful (this means above all self-explanatory and conforming to expectations) user interface contributes considerably to a better understanding of software programs, to the ability to learn them easily and quickly, and to the reduction of help desk activities.

**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 on Web template properties, details of data providers being used, and information on 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 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).
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.

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.

XML
Extensible Markup Language
XML is a developed subset of the Standard Generalized Markup Language (SGML) for applications in the World Wide Web.
XML documents consist of entities which include either analyzed (parsed) or not analyzed (unparsed) data. An analyzed entity includes text, which is a sequence of characters. There are the following types of characters:
▶ Character data
▶ Markup (start tags, end tags, tags for empty elements, entity references, character references, comments, limitations for CDTA sections, document type declarations, and processing instructions)

Numerous standards for special tasks were and are being developed (for example, XLink, XPointer; XSL, XSLT; DOM) based on XML.

**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 homepage.

**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.
Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 260. Note that some of the documents referenced here may be available in softcopy only.

- SAP Business Information Warehouse on OS/390, SG24-5681
- DB2 UDB for z/OS Version 8: Everything You Wanted to Know, ... and More, SG24-6079
- SAP on DB2 UDB for OS/390 and z/OS: High Availability Solution Using System Automation, SG24-6836
- SAP on DB2 UDB for OS/390 and z/OS: Implementing Application Servers on Linux for zSeries, SG24-6847
- DB2 UDB for z/S Version 8 Technical Preview, SG24-6871
- SAP on DB2 for z/OS and OS/390: High Availability and Performance Monitoring with Data Sharing, SG24-6950
- DB2 UDB for z/OS V8: Through the Looking Glass and What SAP Found There, SG24-7088
- SAP R/3 on DB2 for OS/390: DB2 Features That Benefit SAP, REDP-0131

Other publications

These publications are also relevant as further information sources; contact:

http://www.sap.com

- IBM DB2 UDB for z/OS: Planning Guide - SAP NetWeaver 6.40
- SAP R/3 on DB2 UDB for OS/390 and z/OS: Connectivity Guide
- SAP Database Administration Guide - SAP NetWeaver 6.40
- SAP 6.40 DBA Guide
- SAP on IBM DB2 UDB for z/OS: Database Administration Guide: SAP Web Application Server, SAP NetWeaver Version 6.40
- SAP Software on UNIX: OS Dependencies
- DB2 V8 Utility Guide and Reference

Online resources

These Web sites are also relevant as further information sources:

- SAP performance analysis - Tuning SAP DB2 zSeries, Mark Gordon
  http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WPI00287
Whitepaper: SAP BW Query Performance with DB2 on ZSeries
http://www-1.ibm.com/support/docview.wss?uid=ts1wp100322

DB2/SAP Quarterly Technical Newsletter
http://www.software.ibm.com/data/partners/aelpartners/sap

Information about the latest certified SAP/IBM levels from OSS note 81737
http://www.software.ibm.com/data/db2/os390/v6apar.html

SAP Official Information about BW, News, document enhancements, and so on - especially examine the performance folder
http://service.sap.com/bw

SAP Checklist, Installation Guides of BW and the BW-BCT AddON
http://service.sap.com/instguides

SAP search for OSS notes, BW 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, BW-PI, Frontend, and so on
http://service.sap.com/ocs-download

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/
Help from SAP

SAP Support and downloads

http://service.sap.com

SAP Consultancy and other services

Index

Symbols
163

Numerics
00C900AE  208

A
ABAP Dictionary  55, 63
access path  3, 85, 118, 128, 153, 192, 208
accounting data rollup  191
Activation table  183
active ODS
data  78
table 54, 60
Active table  183
adding new partitions  206
administrative activity (AREA)  231
Administrator Workbench  17, 36, 55, 181, 186, 224
Data Access Monitor  186
Data Load Monitor  186
Meta Data Maintenance  186
Modeling function area  36
monitoring function  17
monitoring tool  243
reporting agent scheduler  224
Scheduler  186
Advanced Analytics  2
advanced business application programming (ABAP)  112
Aggregate  178–179
also see InfoCube
aggregate roll-ups  117
aggregate table  117, 160
ALTER TABLE
statement  200, 202
TRANSCOUNT  202
ALTER TABLE ADD PARTITION  206
ALTER TABLE ALTER PARTITION ROTATE FIRST TO LAST  206
API  227
Application Link
Enabling  186
Application Link Enabling  186
application server  5, 29, 100, 111, 117, 141, 143, 146, 159, 190, 215
consistent state  149
especially very large flat file  111
IBM unique hipersockets  5
internal connections  220
automatic query rewrite (AQR)  201–203
Automatic Space Management
feature  211
function  212–213

B
Background, which (BW)  12
BACKUP SYSTEM  167
BAPI  186
Basic InfoCube  23
BEx Analyzer  185, 224
also see Business Explorer
BEx Browser
also see Business Explorer
BEx Portfolio  226
BEx Query
Designer  226
BEx Query Designer  174
BEx Web
Analyzer  227
application  224
application wizard Assistant  226
BEx Web Application
Designer  227
Wizard  227
BI Cockpit  227
BI Java SDK  227
Buffer Pool
management  150
buffer pool  20, 150, 158, 191
new definitions  151
normal allocation  151
space usage  158
Business Application Programming Interface  24, 249
Business Application Programming Interface (BAPI)  186
Business Content  186
Business content  2
Business Document Service (BDS)  229
Business Explorer  173
BEx Browser  185
Query Designer  185
Business Information Warehouse  224
metadata objects  242
business intelligence (BI)  1, 172, 226
BW 3  55, 113, 227
BW 3.5
OSS Note  136, 221
BW administrator  3, 186
BW data
load performance  11
model design  139
object  138
target  12
BW environment  8, 136
BW group  137
BW increases (BI)  223
BW information model  171
BW installation  8, 135
BW LOAD  13, 138
<table>
<thead>
<tr>
<th>BW load</th>
<th>data statistic 103</th>
</tr>
</thead>
<tbody>
<tr>
<td>BW monitor</td>
<td>138</td>
</tr>
<tr>
<td>BW statistic</td>
<td>137, 221</td>
</tr>
<tr>
<td>BW system</td>
<td>2, 11–12, 30, 103, 115, 135, 140, 143, 145–146, 157, 189, 226</td>
</tr>
<tr>
<td>new data</td>
<td>11</td>
</tr>
<tr>
<td>regular data loads</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>cartesian join</td>
</tr>
<tr>
<td>Change log table</td>
</tr>
<tr>
<td>Change-run</td>
</tr>
<tr>
<td>characteristic value</td>
</tr>
<tr>
<td>key figures</td>
</tr>
<tr>
<td>partial quantity</td>
</tr>
<tr>
<td>particular combination</td>
</tr>
<tr>
<td>place holder</td>
</tr>
<tr>
<td>characteristics</td>
</tr>
<tr>
<td>clustering</td>
</tr>
<tr>
<td>clustering index</td>
</tr>
<tr>
<td>CLUSTER option</td>
</tr>
<tr>
<td>common table expression</td>
</tr>
<tr>
<td>Compression</td>
</tr>
<tr>
<td>Compression result</td>
</tr>
<tr>
<td>Consolidated Software Inventory (CSI)</td>
</tr>
<tr>
<td>COPY</td>
</tr>
<tr>
<td>copy pool</td>
</tr>
<tr>
<td>corresponding InfoCube</td>
</tr>
<tr>
<td>dimension tables</td>
</tr>
<tr>
<td>CSR</td>
</tr>
<tr>
<td>CSR Command</td>
</tr>
<tr>
<td>CTE</td>
</tr>
<tr>
<td>CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION</td>
</tr>
<tr>
<td>CURRENT REFRESH AGE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Access Monitor</td>
</tr>
<tr>
<td>DATA INITIALLY DEFERRED</td>
</tr>
<tr>
<td>data load</td>
</tr>
<tr>
<td>many different types</td>
</tr>
<tr>
<td>Data Load Monitor</td>
</tr>
<tr>
<td>Data Partitioned Secondary Index (DPSI)</td>
</tr>
<tr>
<td>Data Partitioned Secondary Indexes</td>
</tr>
<tr>
<td>data set</td>
</tr>
<tr>
<td>high number</td>
</tr>
<tr>
<td>data source</td>
</tr>
<tr>
<td>data target</td>
</tr>
<tr>
<td>data target</td>
</tr>
<tr>
<td>latter type</td>
</tr>
<tr>
<td>Data warehouse</td>
</tr>
<tr>
<td>data warehousing</td>
</tr>
<tr>
<td>e-busines operations</td>
</tr>
<tr>
<td>robust solution</td>
</tr>
<tr>
<td>database administration</td>
</tr>
<tr>
<td>DATABASE DSN8D81A</td>
</tr>
<tr>
<td>database management</td>
</tr>
<tr>
<td>system</td>
</tr>
<tr>
<td>dataset</td>
</tr>
<tr>
<td>DataSource</td>
</tr>
<tr>
<td>DB2 compression</td>
</tr>
<tr>
<td>detailed discussion</td>
</tr>
<tr>
<td>DB2 Java Universal Driver</td>
</tr>
<tr>
<td>DB2 optimizer</td>
</tr>
<tr>
<td>DB2 subsystem</td>
</tr>
<tr>
<td>fast volume level backups</td>
</tr>
<tr>
<td>virtual buffer pool storage</td>
</tr>
<tr>
<td>DB2 table</td>
</tr>
<tr>
<td>space</td>
</tr>
<tr>
<td>DB2 utility</td>
</tr>
<tr>
<td>complete details</td>
</tr>
<tr>
<td>DB2 V8</td>
</tr>
<tr>
<td>IBM publication</td>
</tr>
<tr>
<td>important implemenations</td>
</tr>
<tr>
<td>maximum flexibility</td>
</tr>
<tr>
<td>new code</td>
</tr>
<tr>
<td>new feature</td>
</tr>
<tr>
<td>new support</td>
</tr>
<tr>
<td>recommended method</td>
</tr>
<tr>
<td>DB8N DSNTDDIS</td>
</tr>
<tr>
<td>DB8N8.SDSN Load</td>
</tr>
<tr>
<td>DB8NDIST 0000A23C</td>
</tr>
<tr>
<td>DBA</td>
</tr>
<tr>
<td>DBA Guide</td>
</tr>
<tr>
<td>DDL option</td>
</tr>
<tr>
<td>DDL statement</td>
</tr>
<tr>
<td>DDL time</td>
</tr>
<tr>
<td>DEFAULT 0 73</td>
</tr>
<tr>
<td>deferred indexes</td>
</tr>
<tr>
<td>dimension table</td>
</tr>
<tr>
<td>dimension/snowflake</td>
</tr>
<tr>
<td>disaster recovery (DR)</td>
</tr>
<tr>
<td>disk space</td>
</tr>
<tr>
<td>large amounts</td>
</tr>
<tr>
<td>distributed relational database architecture (DRDA)</td>
</tr>
<tr>
<td>DLIST</td>
</tr>
<tr>
<td>DSN</td>
</tr>
<tr>
<td>dynamic statement cache (DSC)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>E fact table</td>
</tr>
<tr>
<td>EDM pool</td>
</tr>
<tr>
<td>other objects</td>
</tr>
<tr>
<td>EIS</td>
</tr>
<tr>
<td>END</td>
</tr>
<tr>
<td>Enterprise Portal</td>
</tr>
<tr>
<td>Business Intelligence content</td>
</tr>
<tr>
<td>Establish</td>
</tr>
<tr>
<td>ETL - see extraction, transformation, and loading</td>
</tr>
<tr>
<td>example sale</td>
</tr>
<tr>
<td>Extract, Transform, and Load</td>
</tr>
<tr>
<td>Extraction, Transfer and Load (ETL)</td>
</tr>
<tr>
<td>extraction, transformation, and loading (ETL)</td>
</tr>
</tbody>
</table>
F
fact table 31, 44–45, 182
loading activity 31
fact table 8, 23, 25, 32, 36, 41, 54, 85, 94, 117, 156, 159, 170, 176, 180, 192, 222, 232, 238
available index 192
available indexes 8
compressed data 170
compression results 43
consolidated data 43
delta values 9
Efficient access 193
initial compression 44
many records 170
package IDs 42
redundant descriptive data 192
same record 43
Shortens RUNSTATS processing time 42
fast replication 167
Front End
Processing Time 118
fullselect 201
Function module
setting 30
settings
I_IOBJNM 30
function module
RSD_IOBJ_GET 30
RSDU_ANALYZE_TOOL_DB2 117
Functional Module IDs (FMIDS) 140

G
GB Bar 150, 158, 210
agent-related storage 211
EDM pool 211
global sort pool 211
sort pool 211
GB bar
buffer pools 150
generally available (GA) 4
geographical information system (GIS) 226
geographically dispersed Parallel Sysplex (GDPS) 6, 219
given SQL statement
access path 128
granularity 183

H
hierarchy 175, 179, 181
hierarchy node 175, 226
- corresponding leafs 175
HTML page 226

I
IBIOLap interface 228
IBM DB2
UDB 146
Universal Database 131

V8 benefit 190
IBM zSeries 2, 190
identical value 34, 150
- accounting records 150
IDocs - see Intermediate Documents
IFCID 0022 205
Implementation Detail 209
index availability 59, 62
index-controlled partitioning 58, 162
Index-sequential access 61
InfoCube 8, 26, 29, 31, 33, 41, 84, 114, 137, 155, 170–172, 174, 179, 181, 200, 224
Aggregate - see Aggregate
Characteristics - see characteristics
InfoObject 170–171, 174
classes of metadata 171
See information model 171
templates 171
InfoPackage 14, 112, 139, 156, 182, 238
InfoProvider 170, 174, 226
information model 171
- business definitions 171
InfoCube - see InfoCube
InfoSource - see InfoSource
master data - see master data
Persistent Staging Area - see Persistent Staging Area
InfoSource 171–172, 174, 232
in-memory workfile 196
binary search 196
Intermediate Documents 186
Internet Engineering Task Force (IETF) 256

J
J2EE Connector Architecture (JCA) 227
JCA specification 231
JDBC driver 191, 227

K
KB 30, 151, 158, 162, 194
Key figure 2, 26, 34, 42, 176, 223
key figures 176

L
line item dimension 177
local predicate 28, 60, 80, 85, 91, 94, 193
master data tables 93
Only fields 60

M
MAINTAINED BY SYSTEM 203
MAINTAINED BY USER 203
master data 8, 29, 85, 112, 137, 171, 184, 230
- single documents 252
master data table 34, 177
materialized query table 200
isolation level 203
materialized query table (MQT) 200, 202
maximum number 12, 110, 162, 206
mean time to failure (MTF) 4
Meta Object Facility
Java mapping 239
Meta Object Facility (MOF) 232
metadata model 239, 241
Metamodel Repository 242
Mobile business intelligence 2
MOF specification 242
MQT 200
creating 200
query rewrite 203
registering 202
special registers 204
MQTs 200
Multidimensional models 171
Multiple LPARs 5
multiprovider 10, 170, 174, 243
Multi-row operation 209
MXQBCE 197

N
navigation attribute 33, 179, 224
modified version 181
needed InfoObjects
Master Data 138
NORMAL COMPLETION 163, 165
DSN END DSNTIAD 163

O
Object Log 68
Object Management Group
MOF specification 242
Object Management Group (OMG) 231
object version (OBJEVERS) 246
OD60XHNT.AODS TESX 73
ODS - see Operational Data Store
ODS object 2, 23, 55, 58, 78, 90, 171–172, 182, 200, 233
multiple layers 183
query performance 61
rerelevant fields 182
secondary indexes 55
semantic view 238
typical queries 78
ODS query
example 60
performance 53, 78
OLAP data
provider 245
source 224, 241
OLAP Engine 116, 185
OLAP reporting 227
BEx Query Designer 227
OLE DB
interface 223
OMG homepage 232
current CWM specification 232
current XMI specifications 258
Online Analytical Processing (OLAP) 173, 230
Online Transaction Processing (OLTP) 172
only, the decisions should (ODS) 80, 102
Operational Data Store 182
ODS object 170, 172, 174, 182
Operational data store (ODS) 12, 172, 246
optimization 210
OSS note 136, 141, 152
130696 115
16083 139
417307 139
633075 123
633832 117
778437 156
81737 136
out-of-space condition 212
iteratively deal 212

P
Package id 42, 170
Compression results 42, 170
page set 150, 152, 167, 213
Parallel loading 29
Parallel Sysplex 4, 218
PART 72, 166
partition elimination 25, 60, 93
Partition rotation
new concept 165
partition rotation 20, 164
partitioned E fact table
time restriction 93
partitioning 205
Partitioning Key 59, 165
leading column 62
partitions
adding new 206
rotating 206
Peer-to-peer remote copy (PPRC) 6
performance problem 3, 7, 29, 97, 99, 102, 111, 114,
127, 137, 151, 154
reliable indicator 127
Persistent Staging Area (PSA) 12, 171–172, 174, 246
PITR 167
planning level 247
complete data selection 247
planning package 247
pre-computed data 200
pre-summarized data 117
process chain 8, 11, 100, 102, 138, 155, 221, 225
ABAP program 221
DB statistics compilation 221
further optimized RUNSTATS capability 8
RSPC graphical view 113
PSA - see Persistent Staging Area
PSA record 178
attribute values 178
key figures 178
PSA Table
Name 19, 22
partition 112
PTF checker 140
Q
Query Designer 185
query optimization 210
query performance   3, 8, 25, 33, 41, 52, 58, 78, 90, 113–114, 153, 177, 196
  benefit 33
degradation 62
  indicators RSDDSTAT 3
main component 119
significant benefits 25
query rewrite 202, 205
query time 44, 159
query view 225
  filter values 235

R
R/3 system 102, 173
  general performance 112
  memory resources 112
  slow network 112
read-only state 150
real storage 148, 151
Recursive SQL 199
recursive SQL 199
Redbooks Web site 260
  Contact us xii
REFRESH DEFERRED 202
Relational Metadata
  Model 241
Remote Function Call (RFC) 186
RemoteCube 24, 174, 249
REORP 206
response time 101, 109, 114
detailed breakdown 116
  major component 115
major components 115
QDBTIME component 116
RESTORE SYSTEM 167
road map 99, 114
roll-up 181
rotating partitions 206
RSDDSTAT entry 39
RSRT transaction 46, 120
  statistical data 46
rule of thumb (ROT) 117
Runstats enhancement 153, 210
RUNSTATS improvement 153, 210
RUNSTATS utility 155, 210

S
sample JCL 63, 158
SAP Business Information Warehouse
  Administrator Workbench 186
  business content 186
  Business Explorer 173, 185
data flow 173
OLAP engine 185
Staging Engine 185
SAP Data Dictionary 208
SAP Enterprise 208
SAP Transaction (ST) 143
SAP-specific implementation 242
  main focus 242
SCOPE PENDING 206
screen shot 33, 39
Secondary Index 54–55, 61, 90, 113, 205
  many index partitions 207
SECQTY value 161, 214
SELECT ACCTID 201
self-contained dataset 23, 237
server consolidation
  capability 217
  scenario 219
service alternative access paths (SAP) 190
Service Provider Interface (SPI) 252
SET LOG SUSPEND 167
SID 240
SID combination 178
Simple Object Access Protocol (SOAP) 228
  single point of failure (SPOF) 4
SJMXPOOL 195
SM37 job
  log 113
snowflake 192
source system 12, 29, 111, 137, 172, 232
  business client 13
  data transfer 252
  extract structure 254
  resource problem 111
  updating process 234
  work processes 112
sparse index 193–195
SQL access path (SAP) 7, 127
SQL efficiency 23, 170
SQL statement 80, 85, 111, 128, 148, 155, 185, 198
  access path 208
Staging Engine 185
Standard Generalized Markup Language
  developed subset 257
Standard Generalized Markup Language (SGML) 257
Standard ODS 54
star join 197
star schema 176–177
  also see InfoCube
  extended star schema 176–177
STARJOIN 195
STATUID 39, 123
STOGROUP REDODD
  PRIQTY 71 72
  PRIQTY 72 75
STOGROUP REDODI
  PRIQTY 12 76
  PRIQTY 16 73
Strategic Enterprise Management (SEM) 24, 240
surrogate identifier 177
system implementation 212
system-maintained MQT 203
T
Table screen  65
  table size  206
  table space  61, 154–155, 205, 207
  table-controlled partitioning  58, 162
    DB2 V8  59
TABLESPACE statement  161
TB  151, 191
TCP/IP  186
temporary tablespace  38
time basis  156
time period  42, 111, 114, 157, 164, 245
TNAME column  205
  employed MQT  205
Transaction Data
  BW LOAD  138
  correct processing  137
Transaction data
  context-sensitive documents  240
  transaction data  16, 24, 29, 137, 174, 238–239
  transaction ST03N  103
  transfer rule  102, 174, 254
Transfer Structure  183

U
Unified Modeling Language (UML)  232
UNIQUE  207
user interface
  pattern  238
  Visual presentation  255
user-maintained MQT  203

V
Virtual Infocube  25
volume dump  167

W
Web Application  222, 224
  Designer  185, 224
  Server  146, 216, 227
  Server ABAP/JAVA 6.40  136
Web application
  alert monitor  224
  geographical map  240
  new documents  240
  table Web item  250
Web Interface
  Builder  256
  Builder Tool  224
Web item  223
  Ad-hoc Query Designer  224
  paging Mechanism  256
Web item paging  256
Web template  226, 241
  consistent sections  256
  Web items  244
WML content  255
  compressed transfer  257

workfile entry  194
  total number  194
workfile record  194
workload manager  135, 143
World Wide Web
  Consortium homepage  252

X
XML format  233

Z
z/OS image  151
  real storage capacity  151
Best Practices for SAP Business Information Warehouse on DB2 UDB for z/OS V8

Benefits of SAP Business Warehouse on z/OS

This IBM Redbook describes the benefits of DB2 V8 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, ODS 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 redbook 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 redbook includes SAP and DB2 administrators. Knowledge of these products and of the z/OS environment is assumed.

For more information: ibm.com/redbooks