DB2 Information Management Software

IBM

SAP Solutions on IBM DB2 UDB V8.2.2 Handbook

- Leverage highly integrated SAP and DB2 UDB for your enterprise system
- Data protection strategies and monitoring using DBA Cockpit
- Problem determination and diagnostics

Whei-Jen Chen
Jochen Donner
Edgardo G. König
Masako Konno
Beck Tang
Xiaomei Wang

Redbooks

ibm.com/redbooks

International Technical Support Organization

SAP Solutions on IBM DB2 UDB V8.2.2 Handbook

October 2005

SG24-6765-00

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

First Edition (October 2005)

This edition applies to DB2 UDB Version 8.2.2, SAP NetWeaver 2004s.

Contents

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®
DRDA®
@server®
FlashCopy®
HACMP™
Informix®
IBM®
Passport Advantage®
pSeries®
Redbooks™
Redbooks (logo) ™
Tivoli®
TotalStorage®
WebSphere®

The following terms are trademarks of other companies:

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

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

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

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

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

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

Preface

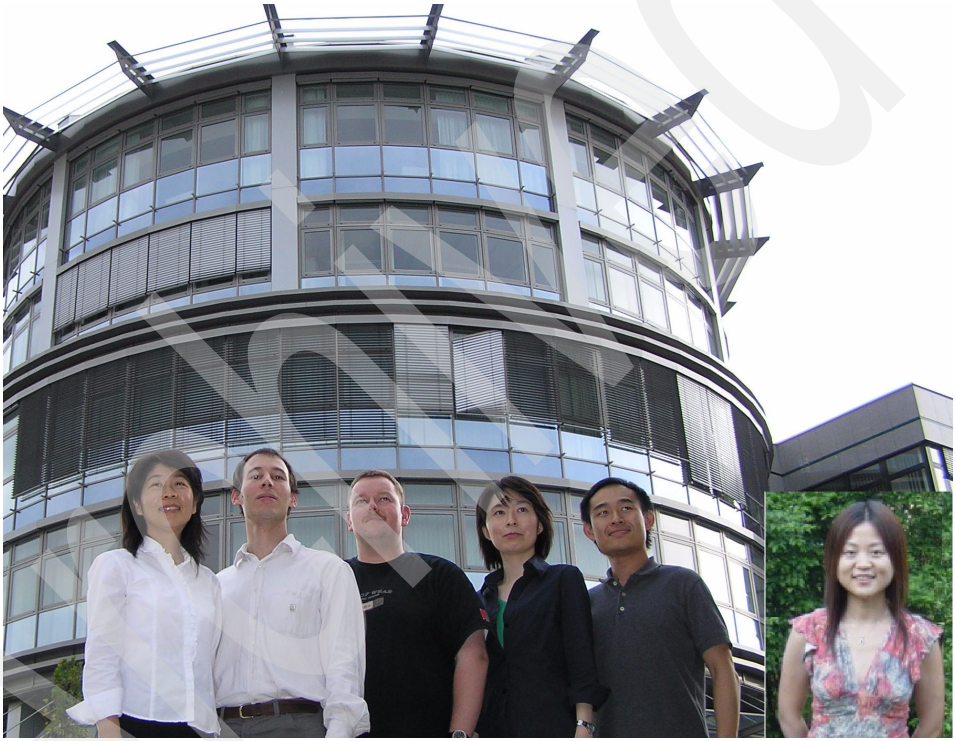
This IBM Redbook gives a broad understanding of the integration of SAP and DB2® UDB; administration, managing, and monitoring DB2 UDB under the SAP environment. This book includes the following chapters:

- **Chapter 1** introduces the DB2 UDB V8.2.2 enhancements and SAP release matrix with DB2 UDB.
- **Chapter 2** provides information about DB2 UDB architecture and its use in SAP environments. We also provide a list of common terms in DB2 UDB and the mapping of these terms to other databases.
- **Chapter 3** introduces a new SAP tool Prerequisite Checker and discusses the considerations that should be taken when installing SAP NetWeaver 2004s with DB2 UDB V8.2.2.
- **Chapter 4** describes DB2 UDB's logical and physical database objects, storage management, SAPs data classes, and space reclamation strategies. It also provides best practices for intelligent storage subsystems and examples of DB2 UDB table conversion using `DB6CONV`.
- **Chapter 5** covers the configuration of DB2 UDB instance and database from the SAP perspective. Database registry and environment variables are also discussed.
- **Chapter 6** describes how to use DBA Cockpit for your system. DBA Cockpit is one of the SAP transactions that you can use from SAPGUI to administrate your database system.
- **Chapter 7** describes DB2 UDB log file management, database backup and recovery, advanced backup techniques for large databases, high availability, and clustered solutions.
- **Chapter 8** introduces the Problem Description / Problem Source Identification (PD/PSI) methodology, and the troubleshooting approaches for common problems based on problem types. Moreover, it discusses usage of DB2 UDB and SAP diagnostic tools. It also introduces the technical support engagement model jointly developed by SAP and IBM®. A list of available knowledge base resources for assisting you in using DB2 and SAP products and troubleshooting problems in DB2 and SAP environments is included.
- **Chapter 9** describes how to upgrade DB2 UDB from V8.x to V8.2.2 on the SAP systems prior to SAP NetWeaver 2004s; explains which DB2 UDB V8.2.2 features can be used with older SAP systems; and shows how to enable the supported DB2 UDB V8.2.2 features in those SAP systems.

- **Chapter 10** discusses new features of SAP NetWeaver 2004s Business Intelligence (SAP BI) with DB2 UDB V8.2.2, including introduction of the architecture, framework, and components of SAP BI; clustering support in DB2 UDB V8.2.2, and SAP BI.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the IBM SAP International Center of Competence, Walldorf, Germany.



Right to left: Xiaomei Wang, Beck Tang, Masako Konno, Jochen Donner, Edgardo Konig, and Whei-Jen Chen

Whei-Jen Chen is a Project Leader at the International Technical Support Organization, San Jose Center. She has extensive experience in application development, database design and modeling, and DB2 system administration. Whei-Jen is an IBM Certified Solutions Expert in Database Administration and Application Development as well as an IBM Certified IT Specialist.

Jochen Donner is an IBM @server® Certified Specialist for pSeries® AIX® System Administration and Support at IBM Global Services in IBM Germany. He has worked at IBM for seven years and has seven years experience in the AIX, Windows®, SAP R/3, DB2 UDB, Informix®, and Oracle fields. His areas of expertise include AIX, HACMP™, SAP R/3, SAP R/3 migrations, DB2 UDB, and Oracle. Jochen is an SAP Certified Technical Consultant for UNIX®, Oracle, DB2 UDB, and OS/DB migrations. Furthermore, he is an IBM Certified Advanced Database Administrator for DB2 UDB V8.1.

Edgardo G. König is a DM Consultant/IT Specialist. He works for the Software Lab Services Team at IBM Argentina. He holds professional certifications in Informix, DB2 UDB, and WebSphere® products. He worked for Informix Software as a database consultant and as a Java™ and C/C++ developer. In 2001, after the acquisition of Informix Software by IBM, he started working with DB2 UDB databases. He has over eight years of experience working with relational databases, J2EE™, and XML related technologies. He has database administration and application development experience on Linux®, AIX, Solaris™, and Windows operating systems. He gives technical assistance in database administration tasks. He also provides consulting services in SAP environments using Informix and DB2 UDB databases. He can be reached at ekonig@ar.ibm.com.

Masako Konno is an IT specialist working at IBM Japan. As a certified Technical Consultant for mySAP Technology and a Technology Consultant for OS/DB migration for SAP Systems, she has worked in a multitude of SAP areas since 2002. Currently, Masako covers technical pre-sales support for DB2 on SAP solutions for Japan.

Beck Tang has four years of experience as a member of the IBM SAP Integration and Support Center at the IBM Toronto Lab. His current role includes certification testing of SAP NetWeaver with IBM DB2 UDB and assisting customers with problem analysis and troubleshooting. Mr. Tang is also a customer advocate providing custom-tailored support for large customer accounts running SAP and DB2 UDB. In addition, he is a DB2 Certified Solution Expert. He is a co-author of the white paper, “SAP/DB2 and High Availability on Sun™ Cluster 3.X” and recently developed the “Text User Interface For DB2 UDB Administration”, which is a lightweight, text-based DBA solution for database maintenance on IBM alpha Works. Mr. Tang can be reached at becktang@ca.ibm.com.

Xiaomei Wang is an IBM DB2 Certified Advanced Technical Expert and a RedHat Linux Certified Technician in the DB2 UDB Advanced Support team at the IBM Toronto Laboratory, Canada. She has over six years of experience with IBM. She handles critical DB2 customer situations worldwide, exploiting her expertise in both non-partitioned and partitioned database environments.

She spear-headed the DB2 Problem Determination and Trouble Shooting education activities across IBM DB2 support centers worldwide and has helped to establish the DB2 L2 Support Centers in AP countries. Her other experiences include working as the IBM Greater China Group Business Development Manager. She drove the business development efforts through technical, sales, marketing enablement, and channel development to increase DB2 presence on non-IBM hardware platforms. Following that, she also worked as a worldwide pre-sales IT specialist for IBM Information Management to drive incremental DB2 revenue on partner platforms (that is, Sun and HP). She currently holds a Master's degree in Computer Science from York University, Canada, and a Master's and Bachelor degrees in Computer Science from Zhejiang University, China. She has written extensively on many DB2 technical topics, such as the Web published tutorial *Problem Determination in a Multi-Node Environment*.

Acknowledgements

The authors express their deep gratitude for the help received from Martin Mezger, Thomas Fiebig, and Brigitte Bläser, who contributed advice, support, and written content:

*Martin K. Mezge*r leads the IBM side at the worldwide SAP DB2 Center of Expertise at SAP AG, Walldorf, Germany. He received his Diploma degree in theoretical computer sciences at University of Stuttgart, Germany. He has been working on DB2 in SAP since 1994. During his career, he has assumed positions in SAP DB2 benchmarking, SAP technology consulting, Advanced SAP Support, and as a Senior Software Developer.

Thomas Fiebig is a Software Engineer at the IBM Böblingen Lab. Thomas is a member of the joint IBM/SAP development team that enables SAP applications on DB2 Universal Database™ for Linux, UNIX, and Windows. His current responsibilities include the integration of new database functionality into the SAP NetWeaver Business Intelligence application. His areas of expertise are database management systems, SAP NetWeaver, and data warehousing.

Brigitte Bläser is a Software Engineer at the IBM Böblingen Lab. She is a member of the joint IBM/SAP development team that enables SAP applications for DB2 UDB for Linux, UNIX, and Windows. She has more than 10 years experience in DB2 UDB and more than 5 years in SAP. Her current responsibilities include the integration of new DB2 UDB features into SAP NetWeaver Business Intelligence.

We also thank the following people for their support and contributions to this project:

Yvonne Lyon, Editor
International Technical Support Organization, San Jose Center

Tammy Toval
IBM Information Management Marketing, Software Group, USA

Liwen Yeow
Michael Cornish
Steve Rees
Kelly Schlamb
Yasir Warraich
Mike Winer
Patrick Zheng
Liam Finnie
Adam Wilson
Lan Pham
John Kennedy
Guiyun Cao
IBM Software Group, Toronto Laboratory, Canada

Thomas Drescher
Karl Heinz Engler
Jens Seifert
Sven-Uwe Kusche
Malte Schünemann
Jörg Overbeck
Guenther Moessner
Thomas Rech
Dirk Pohl
Karl Fleckenstein
Hans-Juergen Moldowan
IBM Software Group, Germany

Torsten Ziegler
Britta Bachert
Andreas Zimmermann
Ralf Stauffer
Frank-Martin Haas
Claudia Langner
Joachim Pfefferle
Frank-Herwig Walter
Andreas Thumfart

Jan Kritter
SAP AG, Germany

Joseph P. Huchel
IBM Global Services, USA

Volker Susok
IBM Sales and Distribution, Software Sales, Germany

Peter Gronimus
IBM Global Services, Germany

Takae Momose
IBM Sales and Distribution, Software Sales, Japan

Florenz Kley
Ulf Hofemeier
Intel® Corporation

Emma Jacobs
International Technical Support Organization, San Jose Center

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. QXXE Building 80-E2
  650 Harry Road
  San Jose, California 95120-6099

1

DB2 UDB and SAP

DB2 UDB version 8.2.2 is the first DB2 product developed jointly between IBM and SAP. With this new release, both companies enter a new level of partnership, which promises enormous benefits for database customers. DB2 UDB V8.2.2 is an optimized version of IBM's DB2 Universal Database to help customers ease configuration, enhance performance, and increase availability of their SAP solutions running on DB2.

The optimized DB2 offering is tailored for both new and existing SAP customers worldwide and is perfectly positioned to support customers in implementing SAP Enterprise Services Architecture. DB2's low operation and maintenance efforts continue to allow SAP customers to benefit from significantly lower total cost of ownership.

This chapter introduces DB2 UDB Version 8.2.2 new features and functions.

1.1 Executive summary

Any successful products are a result of smart designs, strategic vision, and long term stamina and persistence. IBM is the world's largest information technology company. SAP is the world's leading provider of business software solutions. The long term partnership of SAP and IBM has developed the integrated, proven, top-to-bottom, end-to-end joint solutions that help customers navigate the path to realize immediate business value and sustainable business growth.

The joint effort involves close-tied development teams at SAP headquarters in Walldorf, Germany; and IBM laboratories in Böblingen, Germany; Toronto, Canada; and Silicon Valley, USA. In 1999, SAP announced adopting DB2 UDB as the strategic platform for SAP internal development and production systems. Following the announcement, SAP's internal development and data processing organizations opened up and formulated their requirements for DB2 UDB releases, proactively and forward-looking. IBM's development laboratories in Böblingen, Toronto, and Silicon Valley further increased their focus on SAP-related functionality and features.

The culmination of the strategy and joint development is shown in the joint announcement at Sapphire 2005, Copenhagen, to offer a database platform, DB2 UDB version 8.2.2, optimized for SAP's suite of applications and the SAP NetWeaver platform.

"*Since 1999, SAP has worked closely with IBM around DB2 to give our customers a competitive edge*," said Karl-Heinz Hess, senior vice president, Active Global Support, SAP. "*IBM's latest release of DB2 is a substantial step in our joint integration efforts. This tighter integration will allow our customers to enjoy more enhanced performance of SAP solutions at an even lower cost*."

"*This new release of DB2 is just the beginning of a multiyear technology collaboration that will enable the pre-configured DB2-SAP platform to be a growth engine into the future*," said Janet Perna, general manager, IBM Information Management Software. "*Our collaboration with SAP will remain focused on producing the advanced, integrated software solutions customers require to build on demand information infrastructures that generate value from business data*."

With DB2 UDB version 8.2.2, SAP's customers benefit the features and performance optimizations no other database vendors have been offered. DB2 V8.2.2 delivers, with facts as seen on SAP's Web site[1], unparalleled SAP performance and a wealth of features that facilitate the daily life of SAP customers.

[1] http://www.sap.com/benchmark

1.2 DB2 UDB V8.2.2 features for SAP systems at a glance

The joint understanding of customer needs produced enhancements on several areas in DB2 UDB technology. In DB2 UDB V8.2 and DB2 UDB V8.2.2, there are enhancements to ease the job of managing the database. Features like *auto-resize DMS* and *automatic storage* simplify the job in managing the database layout. Some other features, such as *improved index heuristics* and *MDC inserts/deletes*, provide improvements in performance. And there is the Autonomic Computing feature introduced in DB2 UDB V8.2.1; automatic statistic collection, which automatically performs `RUNSTATS` when necessary.

In this section, we provide an overview of new DB2 UDB features and functions introduced in version 8.2 and 8.2.2 pertinent to the SAP/DB2 environment. We provide a more detailed description in the appropriate chapters, showing how you can take advantage of the new features.

1.2.1 New features in DB2 UDB V8.2

Here are the new features introduced in DB2 V8.2 pertinent to SAP/DB2 environments:

- Automatic statistics collection:

  Since DB2 UDB V8.2, as part of the automatic maintenance features, the automatic statistics collection functionality has been introduced. The automatic statistics collection, also known as automatic `RUNSTATS`, lets DB2 UDB determine which statistics to collect and which to update. With this feature enabled, DB2 UDB will automatically run the `RUNSTATS` utility in background as required to ensure that the up-to-date statistics are available. Beginning with SAP NetWeaver 2004s, this DB2 UDB feature is supported and is recommended to be enabled. By using this feature, DB2 UDB will ensure that the DB2 optimizer has the statistical information required to choose the best access plan when resolving a query.

- New log file management:

  The SAP system has been using its own log file management tools to manage DB2 UDB transaction logs (for example, archiving logs with a customized user exit mechanism). Since DB2 UDB V8.2, there is a new log file management introduced. It provides features like log file chaining, which is not available within the SAP DB2 log file management. In a multipartition system, the new log file management also allows each database partition to be configured differently; whereas for the SAP DB2 log file management, there is only one possible configuration for all partitions.

1.2.2 New features in DB2 UDB V8.2.2

Here are the new features introduced in DB2 UDB V8.2.2:

- DB2 workload optimization for SAP environment:

  Prior to DB2 UDB V8.2.2, an administrator for an SAP system running on a DB2 UDB environment needs to refer to an SAP Note for recommendations on a set of DB2 registry variables optimized for SAP environment. With this new feature, DBA can set the registry variable `DB2_WORKLOAD` to SAP, which automatically switches on a set of SAP recommended DB2 registry variables. This feature simplifies the administrative job of keeping the DB2 registry variable in sync with the SAP recommended values. When a new DB2 version or fixpack is installed, any changes for the SAP recommended set of DB2 registry variables automatically applies. If required by SAP support, DBA can override the values of the DB2 registry variables set by `DB2_WORKLOAD`.

- Simplified DB2 UDB installation on Windows:

  In DB2 UDB V8.2.2 for Windows, the installation procedure is simplified in terms of the dialogs you need to complete during the installation. Using new options of the installation wizard, you do not need to complete all the dialogs, because the installation runs with common values and you only see its progress. This feature is planned to be used in upcoming releases of SAP.

- Administration command support and table functions:

  These are a collection of enhancements which provide the ability to issue database administration commands from an SQL statement. This includes performing `RUNSTATS` and `REORG` on tables and indexes, updating database configurations, collecting snapshot information, retrieving backup and log file information from history files, etc. Starting from SAP NetWeaver 2004s, the DBA cockpit has incorporated these administration commands and table functions to perform some of the tasks.

Simplified storage layout

Here are the new features in DB2 UDB V8.2.2 for simplifying the job of managing the database layout:

- Auto-resize DMS:

  In the past, whenever a Database Managed Space (DMS) table space is full, you must either extend a table space or add an additional container to the table space. Now, if auto-resize DMS is enabled for a DMS table space, every time this DMS table space is filled up, DB2 UDB automatically grows the DMS table space. You can control the initial size (`INITIALSIZE`), increment size (`INCREASESIZE`) and maximum size (`MAXSIZE`).

Auto-resize DMS can be enabled at the time of creating the table space using a `CREATE TABLESPACE` statement. For an existing DMS table space, an `ALTER TABLESPACE` statement with the new `AUTORESIZE` options can be issued to take advantage of the auto-resize DMS feature.

- Automatic storage:

  There is ongoing effort to simplify the administrative job for managing table spaces. Automatic storage allows you to define a set of storage paths during database creation time. Then, when you create an automatic storage table space, DB2 UDB creates the containers automatically on those storage paths. Later, when these automatic storage table spaces are filled up, DB2 UDB automatically extends existing containers or creates new containers. In SAP NetWeaver 2004s, you can choose to use automatic storage to store you data. Currently, automatic storage does not support a partitioned database environment and it can only be enabled for new databases during database creation time. After creating your database with automatic storage, you can still create table space(s) that do not use automatic storage (that is, normal DMS table space(s)).

- Uniform page size:

  In the past, DB2 UDB required at least one system temporary table space and the system catalog table space to be of a page size of 4K. This required the existence of a buffer pool of 4K page size. If you would like to use a table space of a different page size (for example, 8K, 16K, 32K), an additional buffer pool is required that matches the page size of the new table space. With this new feature, a newly created database can have an initial page size other than 4K. This allows you to create future table spaces of the same page size; and a single buffer pool is sufficient, thereby reducing the need of having multiple buffer pools.

Performance enhancements

Here are the new features introduced in DB2 UDB V8.2.2 that are related to performance:

- MDC insert/delete:

  Multi-dimensional clustering (MDC) provides an elegant method for clustering data in tables along multiple dimensions in a flexible, continuous, and automatic way. It also supports the typical data warehousing operations of loading data from external source systems (roll-in) and deleting old data no longer needed for reporting (roll-out). If a large amount of data is inserted into an MDC table, locking can be reduced by locking MDC blocks instead of single data rows. MDC also supports fast deletion of the data in an MDC cell. Complete blocks are marked as rolled-out without actually clearing the data.

- Evaluate uncommitted:

  When this feature is turned on, predicates in an SQL statement are evaluated in an uncommitted way. Locks are only acquired if those predicates qualify. This allows a higher level of concurrency and thus reduces the chance of hitting deadlocks. This feature is turned on automatically by enabling the DB2 workload optimization for SAP environment feature.

- Skip inserted:

  This feature allows index or table scan to skip the evaluation of inserted rows that have not committed yet. Like the Evaluate Uncommitted feature, *Skip Inserted* allows a higher level of concurrency and thus reduces the chance of hitting deadlocks. This feature is turned on automatically by enabling the DB2 workload optimization for SAP environment feature.

- Improved Index heuristics:

  This new DB2 UDB internal method is used to overcome sub-optimal cost estimates of the DB2 UDB optimizer because of imprecise statistical data. The heuristic methods consider several factors beside the cost estimates to restrict the number of alternative data access paths. This feature is turned on automatically by enabling the DB2 workload optimization for SAP environment feature. Efforts are being spent by SAP and IBM DB2 to refine index heuristics on a continuous basis.

- REOPT:

  Before V8.2.2, when an SQL statement uses host variables, special registers, or parameter markers for input variables, DB2 UDB uses default filter factors to optimize it. In some cases, this might not reach the optimal performance because default filter factors are estimates of how many rows will be returned at run time, and these estimates are sometimes not appropriate for actual values. In order to improve performance of such SQL statements, the REOPT bind option provides function to optimize the access path for SQL statements at the time the actual values for input variables (host variable, parameter markers, and special registers) become available. In the SAP environment, REOPT can be specified as an Open SQL hint in a native DB2 SQL.

Enhanced serviceability

Here are the new features introduced in DB2 UDB V8.2.2 that simply the job for collecting necessary information for diagnosing problems:

- `db2support` tool for optimizer problems:

  Optimizer or bad access plan problems are some of the most difficult problems to diagnose and identify the root cause. We often see multiple iterations of back and forth between the customer and the support personnel to gather the required diagnostic data for proper problem determination.

An enhancement of db2support is available in DB2 UDB V8.2.2 to automate and centralize diagnostic data collection for optimizer problems. db2support will collect the optimizer data for a problematic query when the -st, -sf or -se option is specified. In case of error or trap during optimization, the -cl option should be used to collect database catalog tables and db2look table definitions without trying to explain a problematic query. When db2support is invoked to collect optimizer data, an additional output file (db2support_opt.zip) will be archived to db2support.zip.

- Enhanced deadlock event monitor:

  In the old deadlock event monitor, the SQL statement and the application triggering the deadlock is captured in the deadlock event monitor. However, because most package solutions use dynamic SQL statements with parameter markers, the values of these parameter markers are not known. A knowledge of these values in the SQL statement from the applications participating in the deadlock will help in resolving a deadlock situation.

  In the new deadlock event monitor offered by DB2 UDB V8.2.2, a history of most of the SQL statements (**SELECT statements** with isolation level, *Uncommitted Read* are not recorded) issued by all deadlock participants since the last COMMIT was recorded. Also, you can now see the runtime values of the parameter markers for these SQL statements.

1.3 SAP release matrix with DB2 UDB platforms

The SAP Product Availability Matrix (PAM) in the SAP Service Marketplace is the best way to find out which SAP releases can be used with DB2 UDB. PAM is available at the Web site:

http://service.sap.com/pam

To access SAP Service Marketplace, an ID is required.

You can get the information for each component or product. Figure 1-1 shows the PAM. The database version that is supported can be displayed in the other window, which is brought up by clicking the Operating System name in the Database Platforms tab.

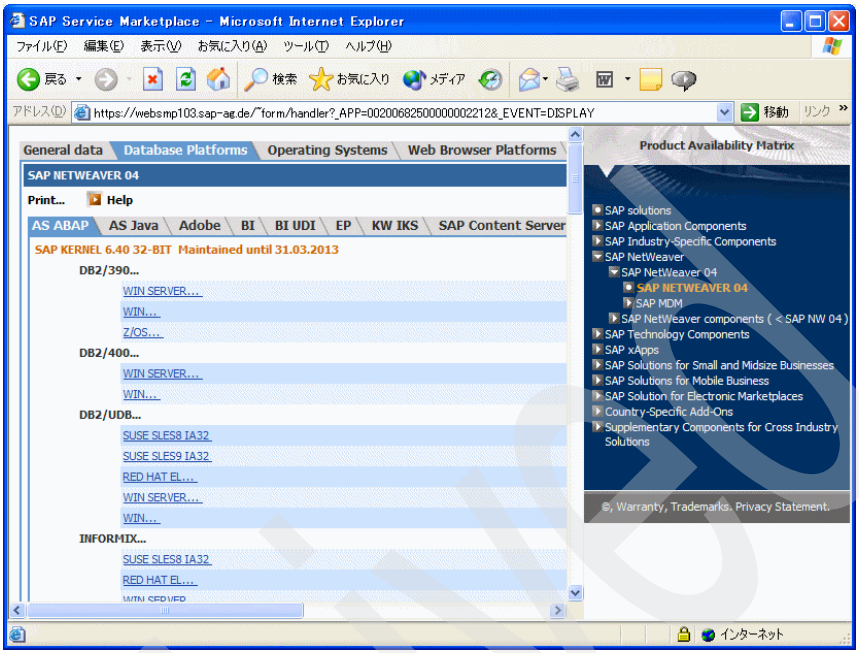


Figure 1-1 SAP Product Availability Matrix

Figure 1-2 shows a supported DB2 UDB version for SAP NetWeaver 2004 on Windows server™ 2003 for SAP kernel 6.40 32bit.

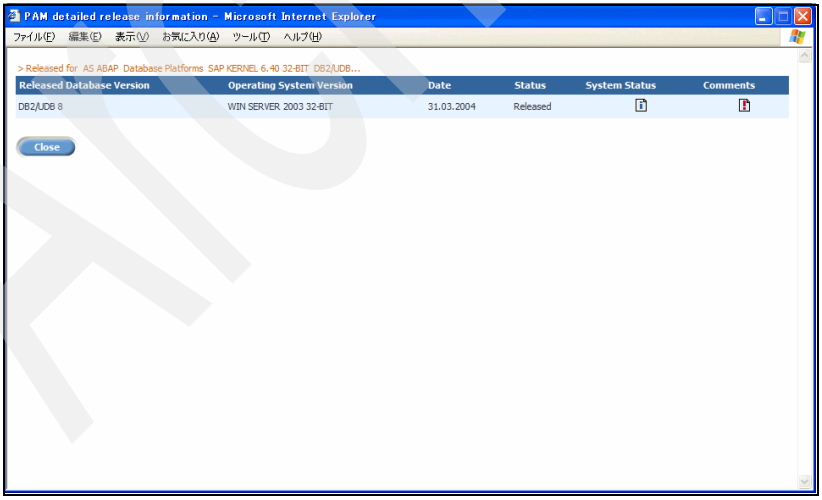


Figure 1-2 Database version information on PAM

In addition to the SAP Service Marketplace, it is necessary to check the SAP Note *101809*, *DB6: Supported FixPaks for DB2 UDB for UNIX and Windows*, in order to find out which DB2 UDB FixPak release is supported by SAP. As this SAP Note is continuously updated, we recommend that you check it regularly.

2

DB2 UDB architecture from SAP's point of view

In this chapter we describe the DB2 UDB architecture and discuss how this architecture is used in SAP systems. First, we introduce the DB2 UDB process model, memory, and storage management architecture. Secondly, we explain in detail the new features available in DB2 UDB Version 8.2.2 that simplify storage management.

We also describe the interaction between DB2 UDB and SAP systems and provide information about the use of the data partitioning feature (DPF) of DB2 UDB in SAP environments.

At the end of this chapter, you can find a group of tables describing how to map a non-DB2 UDB database term into DB2 UDB. These tables will help you apply your knowledge to understand DB2 UDB architecture.

2.1 DB2 UDB server process model

DB2 Universal Database (UDB) server activity is controlled by engine dispatchable units (EDUs). EDUs are implemented as threads in a single process on Windows-based platforms and as processes on UNIX and Linux. System and database administrators commonly examine the DB2 processes on their database servers to understand the DB2 activity and resource consumption within the server at various occasions, such as system planning and configuration, performance tuning, as well as health check and problem analysis. In this book we introduce the process model from a practical view in line with what you would normally see when you operate in DB2 environments.

2.1.1 Fault monitor facility

If you maintain your DB2 databases on UNIX platforms, it is beneficial for you to understand the fault monitor facility as the first step before you look into the core of DB2 process model (see Figure 2-1).

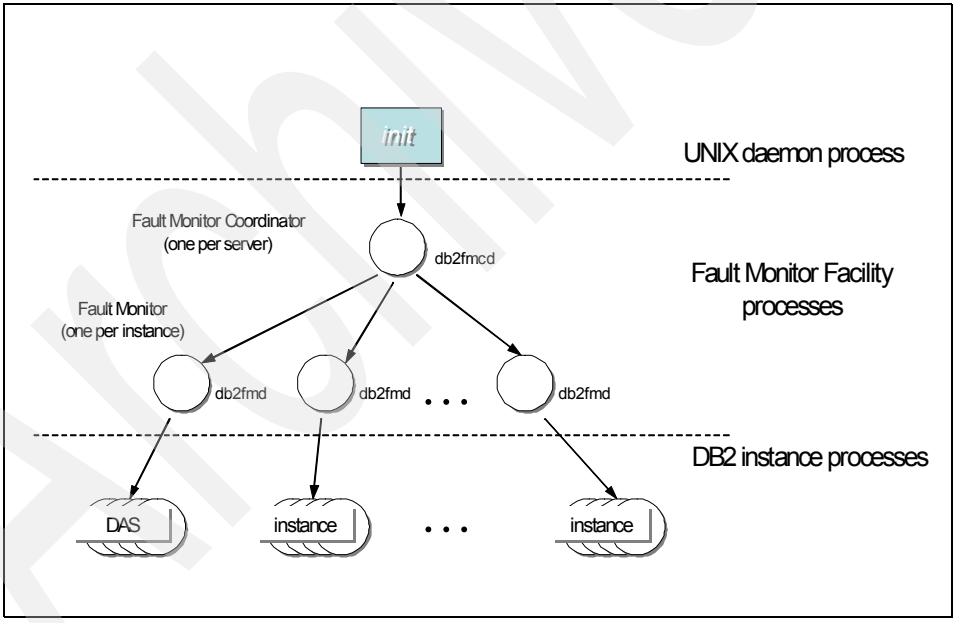


Figure 2-1 DB2 fault monitor facility process overview

On UNIX systems, the fault monitor facility manages the availability of non-clustered DB2 instance environments through a sequence of operations that work together to ensure that DB2 instances are running. That is, the UNIX *init* daemon monitors the fault monitor coordinator (FMC), the FMC monitors the fault monitors, and the fault monitors monitor the DB2 instances.

Figure 2-1 on page 12 shows an overview of DB2 fault monitor facility processes. The fault monitor coordinator or FMC daemon (*db2fmcd*) is the first DB2 process to be started at boot time on a UNIX system. The *init* daemon starts one, and only one, FMC daemon (*db2fmcd*) within the server and will restart it if it terminates abnormally. The FMC daemon then starts one fault monitor (*db2fmd*) for each DB2 instance registered with FMC. If such DB2 instance exits abnormally, the fault monitor will restart it.

Note: The fault monitor is turned off by default and is not used in an SAP system.

2.1.2 A practical view of the process model

In this section, we discuss the DB2 UDB process model in the Linux and UNIX environments. As a database administrator, your first step is to start the database manager instance environment before any DB2 operations can be performed. We can configure the DB2 instance to be autostarted at the boot sequence, or use the `db2start` command to start the instance at a later time. In SAP environments, the DB2 instance is always started using the `startsap` command. When the instance starts, it initializes the current database manager instance background processes on a single database partition or on all the database partitions defined in a multi-partitioned database environment.

In this section we use the **`ps -ef`** command to examine how DB2 grows its process model from the initial set of DB2 instance background processes into a more complicated structure when application connection requests come into the database manager engine.

Before we go any further, it is beneficial to understand some basic concepts about DB2 agents. An agent can be perceived as a “worker” process or thread that carries out the requests made by a client application. There are two main types of DB2 agents:

- *Coordinator agent*:
  Each client connection request is served by one coordinator agent (*db2agent*). The coordinator agent performs all database requests on behalf of the application.

- *Subagent*:
  In some database environments, the *intra_parallel* database manager configuration parameter is enabled, and the coordinator agents distributes the database requests to subagents (*db2agntp*). These agents perform the requests for the application. Once the coordinator agent is created, it handles all database requests on behalf of its application by coordinating subagents (*db2agntp*) that perform requests on the database. When the subagent completes its work and changes its status from active to idle, its process name also changes from *db2agntp* to *db2agnta*.

We start with the DB2 process model for a non-partitioned database, and cover the DB2 process model for a partitioned database at the end. Right after the DB2 database manager is started, an example of the initial set of DB2 backend processes in a single partition database is:

```
    root  12669     1   0 11:42:44      -  0:01 db2wdog 0
  db2axb  12671 12670   0 11:42:46      -  0:00 db2ckpwd 0
  db2axb  12673 12670   0 11:42:46      -  0:00 db2ckpwd 0
  db2axb  12672 12670   0 11:42:46      -  0:00 db2ckpwd 0
  db2axb  12677 12670   0 11:42:47      -  0:00 db2resync 0
  db2axb  12670 12669   0 11:42:46      -  0:00 db2sysc 0
  db2axb  12685 12670   0 11:42:48      -  0:02 db2hmon 0
  db2axb  12674 12670   0 11:42:46      -  0:00 db2gds 0
  db2axb  12676 12670   0 11:42:46      -  0:00 db2tcpcm 0
  db2axb  12678 12674   0 11:42:48      -  0:00 db2srvlst 0
  db2axb  12675 12670   0 11:42:46      -  0:00 db2ipccm 0
```

The DB2 watchdog process (*db2wdog*) is the parent of the entire DB2 process tree with an instance. This process handles abnormal terminations within an instance. Each time when a new DB2 process is forked, the watchdog is notified. In the event that any DB2 process terminates abnormally, it sends the signal to the watchdog, and the watchdog triggers an abort signal to shut down the entire instance.

The *db2sysc* process is the system controller of an instance, which performs the instance initialization and more importantly the main DB2 engine functions. During the instance initialization, the *db2sysc* process starts the communication listeners, *db2ipccm* as the IPC communication listener for local client connections, and *db2tcpcm* as the TCP communication listener for remote client TCP/IP connections. Depending on the communication protocol used in the client-server environment, other types of communication listener processes, such as *db2snacm* for remote SNA/APPC connection requests can be found. In the SAP environment, SAN/APPC communication protocol is not used. The system controller process (*db2sysc*) also starts the “Generic Daemon Spawner” (*db2gds*) to handle all subsequent DB2 EDU process creation requests.

Once the DB2 database manager engine is started, the DB2 process model grows further along with the underline operations of the database activation and the connection request processing when the DB2 instance starts to serve remote or local connection requests.

A means of communication between an application and the database manager must be established before the database can carry out any work on behalf of this application. The DB2 communication listeners are designed to serve this purpose. Upon the database manager receiving an application request, the appropriate communication listener is contacted to serve the request. The *db2ipccm* listener is for local client requests, and the remote communication listeners are for remote requests, among which the most common one is the *db2tcpcm* listener for remote requests in the TCP/IP client-server environment. The appropriate communication listener then assigns one coordinator agent (*db2agent*) to serve that particular application request. Meanwhile, the assigned coordinator agent would make a new connection to the database.

The first connection would activate the database. Upon the database activation, The “Generic Daemon Spawner” (*db2gds*) spawns a series of DB2 EDU processes. An example of DB2 processes after the SAP system is started and connects to the database is as follows:

```
   root  12669     1   0 11:42:44      -  0:01 db2wdog 0
 db2axb  13277 12674   0 11:45:17      -  0:00 db2pfchr 0
 db2axb  12671 12670   0 11:42:46      -  0:00 db2ckpwd 0
 db2axb  13281 12674   0 11:45:17      -  0:00 db2pclnr 0
 db2axb  13293 12670   0 11:45:19      -  0:01 db2fmp (12683) 0
 db2axb  13432 12676   0 11:46:13      -  0:00 db2agent (AXB) 0
 db2axb  12673 12670   0 11:42:46      -  0:00 db2ckpwd 0
 db2axb  13249 12674   0 11:45:13      -  0:00 db2loggr (AXB) 0
 db2axb  13284 12674   0 11:45:17      -  0:00 db2pclnr 0
 db2axb  12677 12670   0 11:42:47      -  0:00 db2resync 0
 db2axb  12672 12670   0 11:42:46      -  0:00 db2ckpwd 0
 db2axb  12682 12670   0 11:42:48      -  0:05 db2agent (AXB) 0
 db2axb  13291 12670   0 11:45:18      -  0:01 db2fmp (13255) 0
 db2axb  13272 12674   0 11:45:16      -  0:00 db2dlock (AXB) 0
 db2axb  13288 12674   0 11:45:18      -  0:00 db2event (DB6_EDLMON1) 0
 db2axb  12680 12670   0 11:42:48      -  0:00 db2agent (AXB) 0
 db2axb  12670 12669   0 11:42:46      -  0:01 db2sysc 0
 db2axb  12681 12670   0 11:42:48      -  0:00 db2agent (AXB) 0
 db2axb  13289 12670   0 11:45:18      -  0:01 db2fmp (12681) 0
 db2axb  13287 12670   0 11:45:18      -  0:01 db2fmp (13261) 0
 db2axb  13506 12676   0 11:46:34      -  0:00 db2agent (AXB) 0
 db2axb  13280 12674   0 11:45:17      -  0:00 db2pclnr 0
 db2axb  13285 12674   0 11:45:17      -  0:00 db2event (DB2DETAILDEADLOCK) 0
 db2axb  13274 12674   0 11:45:16      -  0:00 db2pfchr 0
 db2axb  13279 12674   0 11:45:17      -  0:00 db2pclnr 0
 db2axb  13296 12676   0 11:45:26      -  0:00 db2agent (instance) 0
```

```
db2axb  13275 12674   0 11:45:16      -  0:00 db2pfchr 0
db2axb  12685 12670   0 11:42:48      -  0:02 db2hmon 0
db2axb  13448 12676   0 11:46:17      -  0:00 db2agent (AXB) 0
db2axb  13253 12674   0 11:45:13      -  0:00 db2lfr (AXB) 0
db2axb  12674 12670   0 11:42:46      -  0:00 db2gds 0
db2axb  13250 12674   0 11:45:13      -  0:00 db2loggw (AXB) 0
db2axb  12676 12670   0 11:42:46      -  0:00 db2tcpcm 0
db2axb  12678 12674   0 11:42:48      -  0:00 db2srvlst 0
db2axb  12675 12670   0 11:42:46      -  0:00 db2ipccm 0
```

The key DB2 EDU processes spawned in the above example are:

- *db2pfchr*, the buffer pool prefetcher.
  It asynchronously reads ahead the data and index pages from disk into the buffer pool. The number of prefectchers per database is configured by the NUM_IOSERVERS database configuration parameter.
- *db2pclnr*, the buffer pool page cleaner.
  It asynchronously writes changed pages from the buffer pool to disk before the space in the buffer pool is required by a database agent. The number of page cleaners per database is configured by the NUM_IOCLEANERS database parameter.
- *db2loggr*, the database log reader.
  It reads the database transaction log files to handle transaction processing, restart, and rollforward recovery.
- *db2loggw*, the database log writer.
  It writes log records in the log buffer to the log files on disk.
- *db2logts*, the table space log handler.
  It collects historical information about which logs are active when a table space is modified. This information is ultimately recorded in the DB2TSCHG.HIS file in the database directory. It is used to speed up table space rollforward recovery.
- *db2dlock*, the local deadlock detector.
  One per database partition. In a multi-partitioned database environment, the *db2dlock* process runs on non-catalog partition, while an additional process called *db2glock* runs on the catalog partition to coordinate the information gathered from the *db2dlock* process on each partition.
- *db2fmp*, the fenced mode process.
  It executes fenced stored procedures and user-defined functions outside the firewall. *db2fmp* is always a separate process but may be multi-threaded depending on the types of routines it executes.
- As a summary, the core of the DB2 process model in a non-partitioned database environment is illustrated in Figure 2-2.

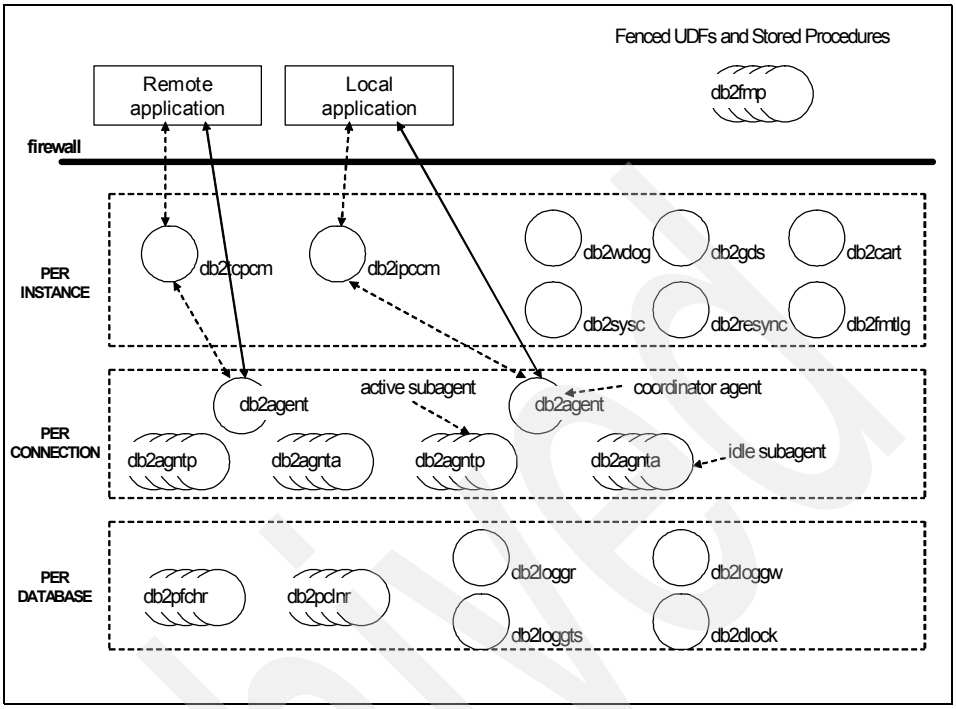


Figure 2-2 The DB2 process model for a non-partitioned database

The process model for a multi-partitioned database can be considered as a bunch of the process models for a single partition database glued together with some extra processes required. Those additional processes are only applicable in a multi-partitioned database environment to coordinate requests across database partitions and to enable the Fast Communication Manager (FCM). The two important processes applicable only in multi-partitioned database environment are:

- *db2pdbc*, the parallel system controller. It handles parallel requests from remote partitions.
- *db2fcmd*, the FCM daemon. It handles inter-partition communication.

Of course DB2 UDB offers more processes than what are listed above. However, once you understand the core of the DB2 process model, you can use that as a start point to better understand the DB2 process model even in a more complicated DB2 environment. Also, by understanding the role of each DB2 process, you can begin to see what operations DB2 is performing when you monitor the DB2 database server, which will surely help you on the system planning, configuring and tuning, as well as problem analysis.

2.2 DB2 UDB server memory architecture

Understanding DB2 UDB's memory architecture is essential when it comes to system planning and performance tuning as well as problem analysis. DB2 UDB has its own memory management layer on top of the Operating System's native memory management, both for performance reasons, and to provide a layer of abstraction so that both threaded (Windows) and non-threaded (UNIX, Linux) DB2 platforms can share the same DB2 memory architecture. The DB2 memory architecture is illustrated in Figure 2-3, where in general DB2 has four different types of memory sets:

- Database manager shared memory
- Database shared memory
- Application group shared (global) memory
- Agent private memory

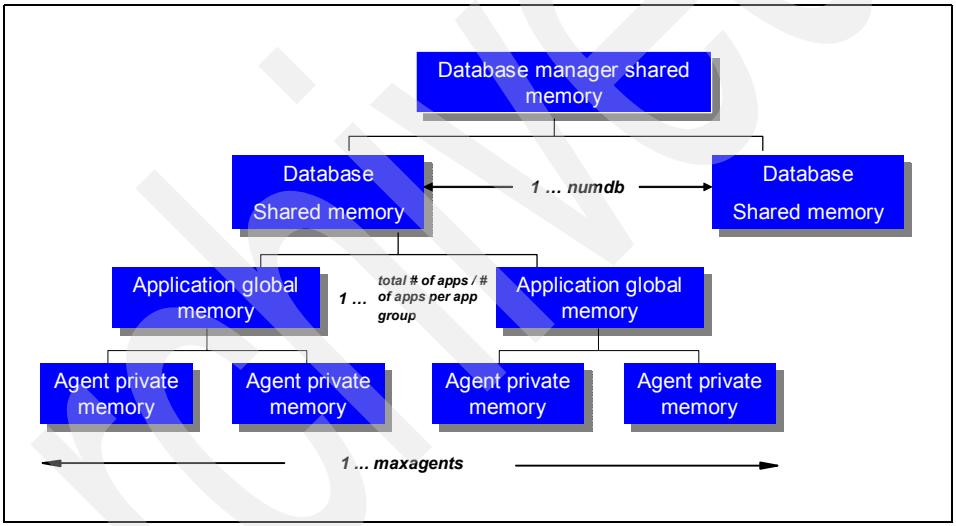


Figure 2-3 DB2 memory architecture

2.2.1 Database manager shared memory and FCM shared memory

There is one database manager shared memory set per DB2 instance. In a multi-partitioned database, it is allocated on a per-partition basis. On all non-AIX platforms, DB2 reserves a separate Fast Communication Manager (FCM) shared memory set. On AIX though, by default the FCM resources are stored in the database manager shared memory set instead unless the DB2 registry variable `DB2_FORCE_FCM_BP` is turned on, or if there is not enough room in the database manager shared memory set, then a separate FCM shared memory set will be allocated. In SAP systems, by default, the `DB2_FORCE_FCM_BP` is set to `YES`.

Both database manager shared memory and FCM shared memory get allocated when the instance is started via the `db2start` command, and freed when the instance is stopped via the `db2stop` command.

The database manager shared memory is used for instance level tasks such as monitoring and auditing, etc. The usage of the database manager shared memory is defined by a set of database manager configuration parameters:

- `INSTANCE_MEMORY` parameter: Defines the lower bound of the amount of memory that should be reserved for instance management. By default, it is set to `AUTOMATIC`, that is, DB2 will calculate the amount of instance memory needed for the current configuration. DB2 will also allocate some additional memory for an overflow buffer. The overflow buffer is used to satisfy peak memory requirements for any heap in the database manager shared memory region when a heap exceeds its configured size.
- Monitor heap (`MON_HEAP_SZ`): For storing database system monitor data when you perform database monitoring activities such as taking a snapshot, turning on a monitor switch, resetting a monitor, or activating an event monitor.
- Audit buffer (`AUDIT_BUF_SZ`): For auditing the database with the *db2audit* facility.

The FCM shared memory contains the information for the communication between partitions in a multi-partitioned database, or between agents and subagents if the database manager configuration parameter `INTRA_PARALLEL` is turned on in a single partition database.

The FCM resources consist of:

- FCM message anchors (`FCM_NUM_ANCHORS`): For sending messages among agents.
- FCM connection entries (`FCM_NUM_CONNECT`): For passing data among agents.
- FCM request blocks (`FCM_NUM_RQB`): The media through which information is passed between the FCM daemon and an agent, or between agents.
- FCM buffers (`FCM_NUM_BUFFERS`): Buffers with 4K bytes page size for internal communications (messages) both among and within database servers.

The FCM buffers by default are allocated as one per physical server, while other FCM memory structures (message anchors, connection entries, and request blocks) are allocated on a per-partition basis.

2.2.2 Database shared memory

There is one database shared memory set per database. In a multi-partitioned database, it is allocated on a per-partition basis. The database shared memory is allocated when the database is activated, and freed when the database is deactivated. The `ACTIVATE DATABASE` command or the first connection to the database would activate a database, while the `deactivate database` command or the last connection disconnect would deactivate a database.

The database shared memory is used for many database level tasks such as shared sorts, backup/restore, load, logging, locking, and SQL executions. The minimum amount that will be allocated is now controlled by the `DATABASE_MEMORY` database configuration parameter. It is usually recommended to set its value to `AUTOMATIC` so that DB2 will calculate the amount of memory needed. On 64-bit AIX, the `AUTOMATIC` value also permits DB2 to grow its database shared memory usage as needed, such as when the buffer pools grow, or when additional memory is needed for control blocks.

Please note that on Windows, `DATABASE_MEMORY` and `INSTANCE_MEMORY` simply defines an upper bound at which the corresponding sets are allowed to grow up to. Because of the multi-threaded architecture, these two memory sets do not have to be reserved up front as shared memory. If these values are set to `AUTOMATIC`, then the corresponding sets are allowed to grow as long as there is sufficient physical memory available on the machine.

Figure 2-4 shows the memory pools making up the database shared memory allocation. Each box is a memory pool, and box size does not indicate relative size of memory.

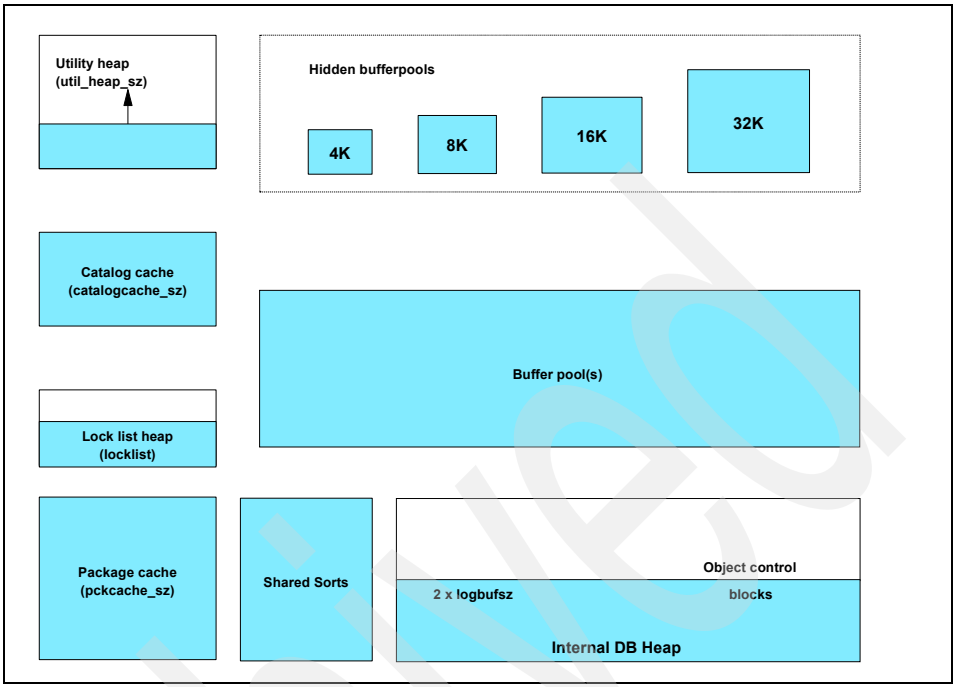


Figure 2-4 DB2 database shared memory

Among all the database shared memory pools, buffer pools and shared sort heap are the largest two components in the database shared memory set. Compared to others, these two also have the most significant impact on the database performance.

Buffer pools

The database buffer pool(s) is the memory area used to cache table and index data pages. Each database requires at least one buffer pool. For a database that has table spaces of more than one page size, additional buffer pools with matching page size need to be created.

Since most data manipulation (except large objects and long field data) takes place in buffer pools, the buffer pool area is the single most important component of the database shared memory with the largest memory consumption. We often would like to configure as much memory as possible for buffer pools in order to achieve optimal database performance.

However, DB2 UDB on 32-bit systems has a limitation on the maximum size of the database shared memory set due to the underlying virtual addressable memory constraints placed by OS. In such a case, DB2 UDB provides the Extended Storage Cache mechanism to allow large buffer pool creation, which uses additional real addressable memory beyond virtual addressable memory. The extended storage is implemented in an OS-specific manner; for instance, on Windows, DB2 UDB uses Address Windowing Extensions (AWE), and on UNIX, ESTORE memory is used.

To ensure that an appropriate buffer pool is available in all circumstances, DB2 creates four small buffer pools, one with each page size: 4K, 8K, 16K, and 32K. The size of each buffer pool is 16 pages. These buffer pools are hidden from the user, as they are not present in the system catalogs or in the buffer pool system files. You can't use or alter them directly, but DB2 uses these buffer pools in the circumstances when a buffer pool of the required page size cannot be allocated.

For example, if the buffer pools are configured too large with not enough database shared memory available and cannot be allocated at database startup, an `SQL1478W` warning is returned. DB2 then attempts to start one of each hidden buffer pool defined with a different page size. So it will allow you to connect to the database to reconfigure the buffer pool sizes or perform other critical tasks.

The quickest way to identify the information of all buffer pools created in a database is by issuing the command:

```
SELECT * FROM SYSCAT.BUFFERPOOLS
```

This returns the complete list of buffer pools within a database with their names, page size, number of pages, etc.

Shared sort heap (SHEAPTHRES_SHR)

DB2 has two basic types of sorts: shared sorts and private sorts.

Shared sorts are only available when the `INTRA_PARALLEL` database manager configuration parameter is `ON` or the connection concentrator is enabled (that is, when `MAX_CONNECTIONS` is greater than `MAX_COORDAGENTS`). They are often used when it is desirable to have multiple subagents feeding or fetching from a sort. The memory used for shared sorts is allocated from the database shared memory set.

When the `INTRA_PARALLEL` parameter is `OFF` and the concentrator is disabled, all sorts are private. The `load` and `create index` operations always use private sorts for index key sorting, regardless of the value of the `INTRA_PARALLEL` parameter. The memory used for private sorts is allocated from an agent's private memory. So a private sort can only be accessed by a single agent.

For shared sorts, the SHEAPTHRES_SHR database configuration parameter is a database-wide hard limit on the total amount of database shared memory that can be used for sorting at any one time. When the total amount of shared memory for active shared sorts reaches this limit, subsequent sorts will fail with SQL0955. If the value of SHEAPTHRES_SHR is zero, the threshold for shared sort memory will be equal to the value of the SHEAPTHRES database manager configuration parameter, which is also used to represent the sort memory threshold for private sorts. if the value of SHEAPTHRES_SHR is nonzero, then this non-zero value will be used for the shared sort memory threshold.

Database heap (DBHEAP)

It contains control block information for tables, indexes, table spaces, and buffer pools. It also contains space for the log buffer (LOGBUFSZ) and temporary memory used by utilities. The value of the DBHEAP parameter is really the maximum amount and is not all allocated at the outset, but it is "reserved". Only the minimum amount needed for the database activation is allocated from the heap. It can grow to the configured size, but there must be enough memory available at the outset for the maximum required.

Utility heap (UTIL_HEAP_SZ)

It can be used simultaneously by the backup, restore, and load (including load recovery) utilities. The UTIL_HEAP_SZ parameter specifies the maximum amount of memory across all executing utilities. When a database is first started, only about 16K bytes of memory is allocated to the utility heap.

Catalog cache (CATALOGCACHE_SZ)

It is used to cache system catalog information. By taking the default value (-1) in a server or partitioned database environment, the value used to calculate the page allocation is four times the value specified for the MAXAPPLS configuration parameter. The exception to this occurs if four times MAXAPPLS is less than eight. In this situation, the default value of -1 will set CATALOGCACHE_SZ to eight.

Lock list heap (LOCKLIST)

It contains the lock list, that is, the locks held by all applications concurrently connected to the database. There is one lock list per database.

Package cache (PCKCACHE_SZ)

It caches sections for static and dynamic SQL statements on a database. The limit specified by the PCKCACHE_SZ parameter is a soft limit. The limit may be exceeded, if required, if memory is still available in the database shared memory set.

We can take the default values for the above database shared memory pools as a starting point. However, we are always required to carefully tune the buffer pools and shared sorts.

Database shared memory consumption calculation

As we mentioned earlier, the DATABASE_MEMORY database parameter defines the lower bound of the database shared memory. Since the real database shared memory usage grows over the time, its maximum amount can be roughly calculated with the following formula:

```
buffer pools + database heap (DBHEAP) + utility heap (UTIL_HEAP_SZ) + lock
list heap (LOCKLIST) + 2 * package cache (PCKCACHE_SZ) + catalog cache
(CATALOGCACHE_SZ) + shared sort heap threshold (SHEAPTHRES_SHR) if
INTRA_PARALLEL is ON + approximate 20% overhead
```

2.2.3 Application group shared memory

The application group shared memory can also be referred to as *the application global memory*. It is a large shared memory area containing one *application control heap* per connection and a large memory pool shared by all connections called the *application group shared heap*. Application group shared memory is allocated when the first connection comes to the database, and freed when the last connection served within the application group terminates.

The application group shared memory exists only in the following environments where multiple agents may be involved in serving an application:

- The connection concentrator enabled (that is, MAX_CONNECTIONS is greater than MAX_COORDAGENTS)
- A multi-partitioned database
- A single-partition database with the INTRA_PARALLEL database manager parameter enabled.

The application group shared memory is illustrated in Figure 2-5.

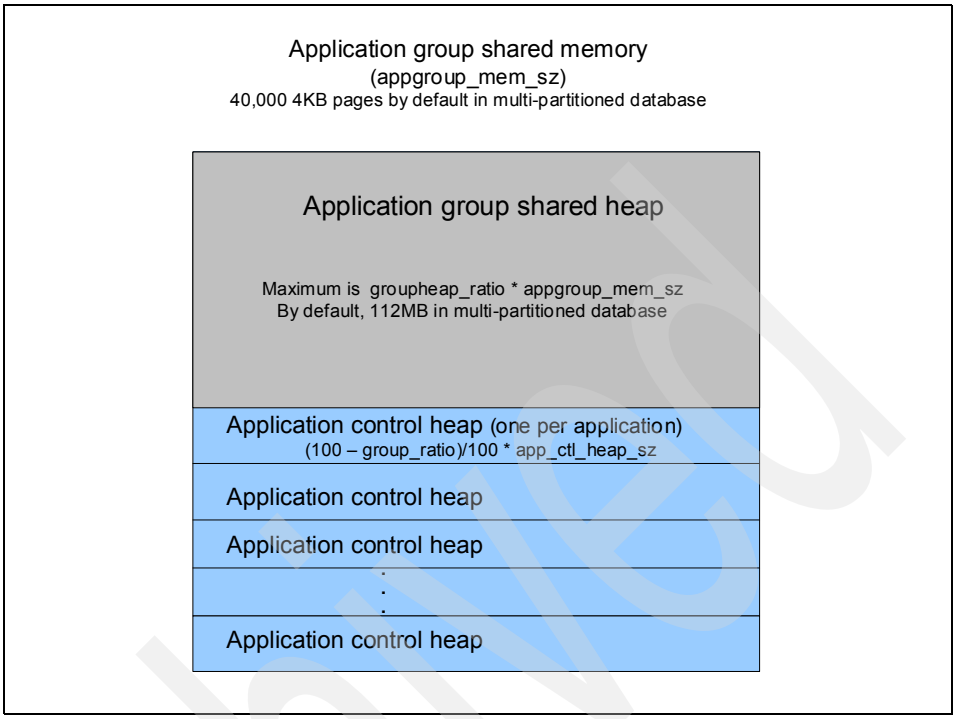


Figure 2-5 DB2 application group shared memory

The Application group shared heap is a large memory area allocated from the application group shared memory. Much of this memory is managed as a cache of SQL sections; it can be reused by any applications in the group and is known as the *shared workspace*. The shared workspace used by an application group can be monitored via DB2 application snapshots. The application group shared heap is freed when the last connection in an application group terminates.

The Application control heap is allocated from the application group shared memory and contains the information about a specific connection. Memory for table queues is also allocated from the application control heap. The application control heap is not freed when a connection terminates, but is reused by a subsequent connection.

In summary, the following three database configuration parameters determine the size of the application group shared memory:

- `APPGROUP_MEM_SZ`, the size of the shared memory for the application group.
- `GROUPHEAP_RATIO`, the percentage of the application group shared memory allowed for the application group shared heap. The maximum size of the application group shared heap is `GROUPHEAP_RATIO * APPGROUP_MEM_SZ`.

- APP_CTL_HEAP_SZ, the size of the application control heap for each application in the group. The approximate maximum amount for the Application Control heap is ((100 - GROUP_RATIO) / 100 * APP_CTL_HEAP_SZ).

2.2.4 Agent private memory

DB2 agents, such as coordinator agents or subagents, can allocate their own private memory area for memory that is only required by that specific agent, such as memory for private sorts or private application heap memory. Agent private memory is allocated for an agent when the agent is assigned as the result of a connect request or a new SQL request in a parallel environment. Agent private memory can grow as required, but freed memory is not necessarily returned to the operating system.

The following memory pools could be potentially allocated from the agent private memory:

- Application heap (APPLHEAPSZ)
- Agent stack (AGENT_STACK_SZ)
- Client I/O block (RQRIOBLK)
- DRDA® heap
- Java heap (JAVA_HEAP_SZ)
- Query heap (QUERY_HEAP_SZ)
- UDF memory
- Sort heap (SORTHEAP)
- Statement heap (STMTHEAP)
- Statistics heap (STAT_HEAP_SZ)

In many cases the amount of the agent private memory allocated per each DB2 agent could be minimal. So people tend to ignore it when doing system memory capacity planning or memory problem analysis. However, on the contrary, the aggregated agent private memory of all DB2 agents may turn out to be one of the top memory consumers in the system, especially true in environments that support hundreds or thousands of database connections, such as large-scale SAP systems.

Another fact often neglected is that by default, DB2 retains the “freed” private memory for performance reasons even when an agent becomes idle. Also, due to the DB2 agent pooling design, once the DB2 system doesn’t reach the MAXAGENT limitation, a DB2 agent doesn’t terminate when it finishes its work, but goes back to the agent pool for reuse upon the application request completion. In a DB2 environment with a large MAXAGENT configured, we may potentially see a large number of DB2 idle agents. So we may observe the “strange” behavior that the system retains high memory usage even after the workload decreases significantly.

In an SAP environment, DB2 provides some enhancement on selected platforms to avoid the above situation. AIX offers the `disclaim()` system call to release the memory. DB2 takes advantage of this, and provides two DB2 registry variables on AIX systems: `DB2MEMDISCLAIM` and `DB2MEMMAXFREE`. When `DB2MEMDISCLAIM` is turned on, DB2 will free the agent private memory if retaining the memory would result in exceeding the amount specified by `DB2MEMMAXFREE` (defaults to 8MB) and call the `disclaim()` system call so that AIX releases the memory. In the SAP environment, `DB2MEMAXFREE` is set to 2000000 by default.

Starting from DB2 UDB Version 8, a big enhancement on the Solaris platform is that via the new implementation of “disposable sets”, the memory for private sorts allocated in agent private memory set gets released when the sort operation ends. So one big sort operation won't leave behind a huge memory footprint on the system. Note that private sorts normally consume a big chunk of the agent private memory.

On AIX, DB2 UDB disclaims the memory for large sorts if the allocation is above the `DB2MEMMAXFREE` limit, and `DB2MEMDISCLAIM` has not been disabled.

On Linux, if the memory allocation is above the `DB2MEMMAXFREE` limit, DB2 UDB will free this memory back to the OS, and the OS will free the physical memory to be re-used by other applications/processes.

On HP, if the memory allocation is above the `DB2MEMMAXFREE` limit, DB2 UDB will free this memory back to the OS, but the OS will not free this memory up for other apps/processes.

On Windows, if the memory allocation is above the DBM configuration `PRIV_MEM_THRESH` (typically set much higher than `DB2MEMMAXFREE`, as DB2 UDB is threaded on Windows, and all agents share the same private memory set), then DB2 UDB will release the memory back to the OS, and the OS will free this memory up for other applications/processes to use.

2.2.5 DB2 memory architecture summary

In summary, the basic DB2 memory model is the same across all platforms. However, since the operating system memory management mechanism varies from platform to platform, the implementation of the DB2 process’s virtual memory address space layout varies as well. For example, because of the multi-threaded engine implementation, DBMS, DB, and application control group memory are private memory on Windows. As such, memory is only consumed when needed, in contrast to UNIX/Linux, where shared memory has to be allocated and reserved up front.

DB2 UDB currently supports both 32-bit and 64-bit architectures. The DB2 64-bit architecture is now fully implemented and extended. The idea in 64-bit is that there are no restrictions to the location nor the size of each memory set allocated. So the obvious benefit from a 64-bit database engine could be larger buffer pools and shared sort heap. However, in 32-bit architecture, the location and the size of each DB2 memory set need to comply with the restrictions placed by both the operating system and DB2 itself. SAP supports 64-bit DB2 UDB on AIX, HP-UX, Solaris, Linux, and Windows to allow customers to explore the advantage of 64-bit DB2 UDB.

2.3 Storage concept basics

In this section, we provide a general basic DB2 UDB storage concept. In an SAP environment, a specific database storage layout is used which comes with the SAP installation tools. DB2 UDB uses the page internally as its basic unit of storage. DB2 UDB can work with different page sizes, for example, 4K, 8K, 16K or 32K bytes are the values allowed for page sizes in DB2 UDB installations.

The page size we use can be defined at different levels: either at database creation time or at table space creation time. It is important to note that, depending on the page size we choose, we must create a buffer pool which can read these pages.

In this section we analyze DB2 UDB storage management capabilities from two points of view: logical and physical.

From a logical point of view, every piece of data stored in a DB2 UDB database is represented by a table in the system. We can create indexes in these tables and we can also create tables with special columns called BLOBs. The tables and indexes in a DB2 UDB database are created in a logical entity known as table spaces. A table space is a place DB2 used to store tables where the data is held.

Before creating table spaces, we must identify which containers we will use. A container can be assigned from a directory, a file from the file system, or a device of the server. As you can see, a container represents a physical concept. Depending on the type of table space we want to create, we must choose one of the three different container types.

2.3.1 Table space types

There are two ways to classify table spaces, either by the type of container used or by the kind of data stored in them.

Classifying them by the container type, we have:

- System managed space (SMS)
- Database managed space (DMS)

Classifying them by the information stored, we have:

- Regular table spaces
- Temporary table spaces
- Large table spaces

In the following paragraphs we describe briefly each table space type. Chapter 4, “Storage management in depth” on page 79 provides more information about when to use system managed table spaces or database managed table spaces.

System managed table space

A system managed table space uses a directory as its container, and the operating system’s file manager controls the storage space. In this directory, DB2 UDB creates the files for tables or indexes. If we decide to use an SMS table space, we cannot distribute data information in a table space and index information in another table space.

The data in the table spaces is striped by *extent* across all the containers in the system. An extent is a group of consecutive pages allocated in a database. DB2 UDB distributes data evenly across all containers in a round-robin fashion.

In SMS table spaces, DB2 UDB allocates space on demand. Beginning with DB2 UDB Version 8.2, the basic space unit allocated is an extent. In previous versions of DB2 UDB, the basic space unit allocated was a page. If we wanted to allocate an extent, run the `db2empfa` tool. This tool changes the `MULTIPAGE_ALLOC` database configuration parameter from `NO` to `YES`.

We can have many directories used as containers in an SMS table space; however, once the file system that holds the table space container is full, the complete SMS table space is considered to be full and we cannot allocate new containers. Therefore, it is important to know the size requirements of the table space and create all required containers when the table space is created. In an SAP/DB2 environment, only temporary table spaces use the SMS table space.

Database managed table space

A database managed table space uses either a file or a device as its container. In this kind of table space, DB2 UDB is responsible for managing the storage space. DB2 UDB strips the information stored to ensure an even distribution of the data in the containers. Typically containers reside in different disks to avoid I/O bottleneck problems. The space for DMS table spaces is allocated when the table space is created.

DMS table spaces can also have many containers. If a container in a DMS table space becomes full, DB2 UDB can still use the free space of the other containers. We can also add more containers or drop some of them; DB2 UDB rebalances the data automatically and asynchronously. Containers can also be extended, reduced, or resized dynamically.

Regular table spaces

Regular table spaces are used to store tables and indexes in a DB2 UDB database. Whenever we create a database, two regular table spaces are created. The first regular table space created is named `SYSCATSPACE`, and holds the system catalog tables of the database. The second regular table space created is `USERSPACE1`, which is used to store user tables and indexes. Both regular table spaces are created as SMS table spaces, but we can create them as DMS table spaces using options of the `CREATE DATABASE command`. The `USERSPACE1` table space can be dropped, if there is another user table space.

Large table spaces

This kind of table space is used to store the information of long `VARCHAR` or large object (LOB) columns of DB2 UDB tables. We must decide whether we want to accommodate the information of these columns in another table space at table creation time. Currently, SAP does not utilize large table spaces.

Temporary table spaces

As the name implies, temporary table spaces are used to hold transient information. There are two kinds of temporary table spaces: temporary system table spaces and temporary user table spaces. Temporary system table spaces are used by DB2 UDB during sort operations, reorganizing tables, creating indexes or joining tables. On the other hand, temporary user table spaces are needed for the creation of a declared global temporary table, which is a temporary table created from an application.

When we create a database, DB2 UDB automatically creates a temporary system table space, because at least one temporary system table space must exist in the database. This SMS table spaces is called `TEMPSPACE1`. In an SAP environment, SAP installation procedure creates a temporary space `PSAPTEMP` to replace `TEMPSPACE1`.

If we plan to use declared global temporary tables, we need to create a temporary user table space.

2.3.2 DB2 UDB V8.2.2 storage management advancements

In Chapter 1, “DB2 UDB and SAP” on page 1, we briefly described the new DB2 UDB V8.2.2 features, including those mainly developed for SAP systems. In this section we cover in more detail two new features that simplify storage space management in an SAP / DB2 UDB environment.

Uniform page size

When a database is created, DB2 UDB automatically creates three table spaces, SYSCATSPACE, TEMPSPACE1, and USERSPACE1; and a buffer pool, IBMDEFAULTBP.

Prior to V8.2.2, the page size of these automatically created table spaces and buffer pool was fixed at 4K bytes. A system with very large tables may require table spaces with a page size greater than 4K bytes. In that case, the DBA must create a buffer pool with a matching page size for those table spaces. This adds some database administrative work and system overhead for multiple buffer pools.

The new uniform page size feature of DB2 UDB allows us to define a page size greater than 4K bytes at database creation time. The page size defined at database creation time will be used by the three table spaces: SYSCATSPACE, TEMPSPACE1, and USERSPACE1, which have this page size. This means that we can have a system with a uniform page size for all of its table spaces and only have one buffer pool in the system, avoiding memory fragmentation problems.

Defining the uniform page size

The PAGESIZE clause of the CREATE DATABASE command indicates the page size of the default buffer pool and the initial three table spaces. The value used in this clause is also the default page size for all CREATE BUFFERPOOL and CREATE TABLESPACE statements in that database.

Here we show a compact syntax diagram of the CREATE DATABASE command using the PAGESIZE clause:

```
>>-CREATE--+-DATABASE-+--database-name--------------------------->
           '-DB-------'

  >--+----------------------------+----------------------------><
     +-AT DBPARTITIONNUM-----------+
     '-| Create Database options |-'

     .-PAGESIZE--4096----------.
  >--+-------------------------+-------------------------------->
     '-PAGESIZE--integer--+---+-'
                          '-K-'
```

The complete syntax of the CREATE DATABASE command can be found in the DB2 UBD documentation, *Command Reference V8*, SC09-4828-01.

The values for the PAGESIZE clause are: 4096, 8192, 16384, and 32768. We can also specified the page size with the values: 4 K, 8 K, 16 K or 32 K. If the PAGESIZE value is omitted, the default value of 4096 is used.

Let us now look at some examples:

```
CREATE DATABASE RED ON D: PAGESIZE 16 K
CREATE DATABASE RED ON D: PAGESIZE 16384
```

Both examples create a database called RED using page size 16 K. The three table spaces SYSCATSPACE, TEMPSPACE1, and USERSPACE1, and the default buffer pool, IBMDEFAULTBP, are created with 16 K page size.

Automatic storage

Automatic storage, sometimes also referred as the single point of storage feature, is part of the new autonomic capabilities of DB2 UDB Version 8.2.2.

The idea behind this concept is that DB2 UDB can manage its storage space by itself. This is a very important feature for SAP systems. It allows DB2 UDB to run with minimum or no interaction of the database administrator.

You can create automatic storage databases, create automatic storage table spaces, and also modify DMS tables spaces to grow automatically.

Automatic storage databases

An automatic storage database is one in which DB2 UDB will automatically extend containers or add new ones when the table space becomes full.

Automatic storage for a database can only be enabled when the database is created and cannot be disabled afterwards. Therefore it is important to think in advance whether or not you want DB2 UDB to automatically manage the storage space.

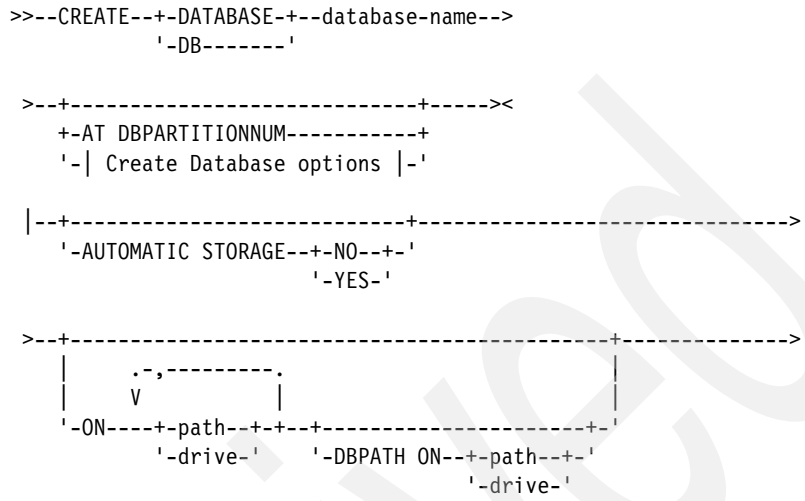
During database creation time, there are three ways to enable automatic storage. These ways are represented by three different options of the CREATE DATABASE command:

- Using the AUTOMATIC STORAGE clause
- Indicating more than one path in the ON clause
- Indicating one path in the ON clause and also specifying the DBPATH ON clause

The first is an explicit way to enable the feature, the latter two are implicit ways of enabling automatic storage in a database.

Creating automatic storage databases

A compact syntax diagram of the CREATE DATABASE command showing the AUTOMATIC STORAGE, ON and DBPATH ON clause is shown here:



The complete syntax of the CREATE DATABASE command can be found in the DB2 UDB product documentation *Command Reference V8*, SC09-4828-01.

Some examples of enabling automatic storage either explicitly or implicitly are:

Example 1:

```
CREATE DATABASE PRD AUTOMATIC STORAGE YES
```

This command creates a database named PRD with automatic storage enabled. DB2 UDB creates the DB2 control files for the database and automatic storage table spaces in the home directory of the instance owner (DB2 UDB for Linux and UNIX) or in the drive on which DB2 UDB is installed (DB2 UDB for Windows).

Example 2:

```
CREATE DATABASE PRD AUTOMATIC STORAGE YES ON D:
```

If you specified the ON clause, the DB2 control files for the database and the automatic storage table spaces are created in the path specified in this clause. For this example, the drive D: is used. If the operating system is Linux or UNIX, you can specify a directory path.

Example 3:

```
CREATE DATABASE PRD ON /db2/data1, /db2/data2
```

In this example, the AUTOMATIC STORAGE clause is not used. However, automatic storage table spaces are created in the directories /db2/data1 and /db2/data2. The DB2 control files will be stored in the first directory specified in the ON clause, /db2/data1 as shown in the example.

Example 4:

```
CREATE DATABASE PRD ON D:\db2\data DBPATH ON C:
```

For this example, the database PRD is also an automatic storage database, because the CREATE DATABASE command includes the ON and DBPATH ON clauses. This is an example where the automatic storage feature is indicated in an implicit way.

Altering automatic storage databases

As we saw in the previous section, whenever we create an automatic storage database, the paths used in the ON clause are used as the paths for the containers of the automatic storage table spaces and the DBPATH ON clause specifies the location of the database directory (which is the location of the database's metadata and control files).

It is also possible to add new paths using the ALTER DATABASE statement. This statement can be used with automatic storage databases to add new storage paths to the collection of paths used during automatic storage table space creation. DB2 UDB controls that this new path is unique with respect to the paths that are already part of the database storage.

The new path(s) are not used immediately after they are added. The database manager decides when to start using them.

The syntax of this statement is quite simple:

```
>>-ALTER DATABASE--+--------------+-------------------------->
                   '-database-name-'

                   .-,------------------.
                   V                    |
>--ADD STORAGE ON----'storage-location'-+---------------------><
```

More information about this statement can be found in the *DB2 UDB product documentation SQL Reference, Volume 2, V8*, SC09-4845-01.

For example, if we want to add a new storage path /db2/data3 to database PRD created using command in example 3, we can do it as follows:

```
ALTER DATABASE PRD ADD STORAGE ON /db2/data3
```

Automatic storage table spaces

If the database has been enabled for automatic storage, we can create table spaces using the `MANAGED BY AUTOMATIC STORAGE` clause. This new clause creates automatic storage table spaces. The `MANAGED BY SYSTEM` and `MANAGED BY DATABASE` clauses can also be used to create normal SMS and DMS table spaces without automatic storage capabilities.

The `MANAGED BY AUTOMATIC STORAGE` clause specifies that the table space is an automatic storage table space. There is no need to include a container definition at table space creation time, because the database manager decides which containers are assigned to the table space.

If the database is not an automatic storage database, the use of the `MANAGED BY AUTOMATIC STORAGE` clause generates an error.

DB2 UDB creates containers for automatic storage table spaces using the following convention:

```
<storage path>/<instance>/NODE####/<dbname>/T#######/C#######.<EXT>
```

Table 2-1 describes the meaning of each element.

Table 2-1 Automatic storage container path description

Element	Description
<storage path>	A storage path associated with the database
<instance>	The instance name under which the database was created
NODE####	The database partition number, that is: NODE0000
<dbname>	The name of the database
T#######	The table space id, that is: T000001
C#######	The container id, that is: C0000001
<EXT>	An extension based on the type of data being stored: CAT: system catalog table space TMP: system temporary table space UTM: user temporary table space USR: user or regular table space LRG: large table space:

Automatic storage table spaces can be either system managed space (SMS) table spaces or database managed space (DMS) table spaces. SMS table spaces using directory containers are created for temporary table spaces and DMS table spaces using file containers are chosen for regular or large table spaces.

Creating automatic storage table spaces

A compact syntax diagram of the `CREATE TABLESPACE` statement with the `MANAGED BY AUTOMATIC STORAGE` clause and its related attributes is shown here:

```
>>-CREATE--+----------------------+--------------------------->
           +-REGULAR--------------+
           +-LARGE----------------+
           | .-SYSTEM-.           |
           '-+--------+--TEMPORARY-'
             '-USER---'

>--TABLESPACE--tablespace-name-------------------------------->

>--+--------------------------+-------------------------------->
   '-PAGESIZE--integer--+---+-'
                        '-K-'
>--+-------------------------------------------------------+-->
   +-MANAGED BY--AUTOMATIC STORAGE--| size-attributes |------+
   +-MANAGED BY--DATABASE--| database-containers |--| size-attributes |--+


size-attributes:
|--+--------------------+--+--------------------------+----->
   '-AUTORESIZE--+-NO--+-'  '-INITIALSIZE--integer--+-K-+-'
                 '-YES-'                          +-M-+
                                                  '-G-'

>--+-----------------------------------+---------------------->
   '-INCREASESIZE--integer--+-PERCENT-+-'
                            '-+-K-+---'
                              +-M-+
                              '-G-'

>--+---------------------------+-----------------------------|
   '-MAXSIZE--+-integer--+-K-+-+-'
              |          +-M-+ |
              |          '-G-' |
              '-NONE-----------'
```

Table 2-2 describes the size attributes:

Table 2-2 Size attributes for CREATE TABLESPACE statement

Attributes	Description
AUTORESIZE	Specifies whether or not the auto-resize capability is enabled. For automatic storage table spaces, the default value is YES. For DMS table spaces, the default value is NO.

Attributes	Description
INITIALSIZE	Specifies the initial size of an automatic storage table space. If no value is provided, the database manager determines an appropriate value.
INCREASESIZE	Specifies the amount by which a table space that is enabled for auto-resize will automatically be increased when the table space is full, and a request for space has been made. If no value is provided, the database manager determines an appropriate value.
MAXSIZE	Specifies the maximum size to which a table space that is enabled for auto-resize can automatically be increased. The default value is NONE, this means that the table space can grow to file system capacity or maximum table space size.

Next we give some example statements of creating automatic storage tables spaces.

Example 1:

```
CREATE TABLESPACE RED#SAPD
```

This statement creates a table space called `RED#SAPD`. The containers are created following the convention defined in Table 2-1 on page 35 and created on the storage paths identified by the `CREATE DATABASE` command. If you don't specify any `MANAGED BY` clause, then it defaults to `MANAGED BY AUTOMATIC STORAGE`.

Example 2:

```
CREATE TABLESPACE RED#SAPD MANAGED BY AUTOMATIC STORAGE
```

In this example, the `MANAGED BY AUTOMATIC STORAGE` clause is explicitly specified. The results of this statement are the same as in the previous example.

Example 3:

```
CREATE TABLESPACE RED#SAPD MANAGED BY AUTOMATIC STORAGE INITIALSIZE 100 M
INCREASESIZE 50 M MAXSIZE 500M
```

In this example, the initial table space size, increasing size, and maximum size are specified. DB2 UDB will use the initial value to create table space and increase automatically the table space size once the table space is full. The table space size will be increased over 500M bytes.

All three examples assume that the database has the automatic storage feature enabled.

Important considerations

There are some important things to consider when working with the automatic storage feature:

- The automatic storage feature cannot be enabled or disabled for a database after it is created.
- Storage paths must be absolute path names.
- Automatic storage can only be enabled for single partitioned systems.

Making table spaces grow automatically

Beginning with DB2 UDB Version 8.2.2, there is a new feature called auto-resize DMS, which allows a DMS table space to grow automatically. This feature can be turned on after a DMS table space is created.

New clauses have been added to the `ALTER TABLESPACE` statement that turn on auto-resize DMS feature,which allows this table space to grow in size. This clause does not apply to SMS table spaces. By the nature of SMS table spaces, the size can be increased by the system in file directory containers.

The command syntax is:

```
>>-ALTER TABLESPACE--tablespace-name--------------------------->
     +-AUTORESIZE--+-NO--+-------------------------------------+
     |             '-YES-'                                     |
     +-INCREASESIZE--integer--+-PERCENT-+---------------------+
     |                        '-+-K-+---'                      |
     |                          +-M-+                          |
     |                          '-G-'                          |
     '-MAXSIZE--+-integer--+-K-+-+-----------------------------'
                |          +-M-+ |
                |          '-G-' |
                '-NONE-----------'
```

The description of the clauses is the same as for Table 2-2 on page 36.

You can find a complete syntax diagram of these statements in the *DB2 UDB product documentation SQL Reference, Volume 2, V8*, SC09-4845-01.

Now let us look at some examples of the new clauses of the `ALTER TABLESPACE` statement.

Example 1:

```
ALTER TABLESPACE RED#CLUD AUTORESIZE YES INCREASESIZE 50 M
```

This statement enables the auto-resize option of the table space RED#CLUD and sets the increase size to 50 megabytes.

Example 2:

```
ALTER TABLESPACE RED#CLUD AUTORESIZE YES MAXSIZE 256 M
```

This statement also enables the auto-resize option of the table space and sets 256 megabytes as the maximum size of the table space.

2.3.3 Directory structure for DB2 UDB in SAP environments

SAP systems have a well-defined directory structure for a DB2 UDB database. Two directory structures are used depending on the underlying operating system.

Directory structures on Windows

SAP systems running DB2 UDB on a Windows operating system use the directories described in Table 2-3 to accommodate DB2 UDB information.

Table 2-3 Directory structure on Windows

Directory	Description
<drive 1>:\sqllib	DB2 UDB software distribution binaries
<drive 2>:\db2<dbsid>	DB2 Instance directory
<drive 3>:\db2\<DBSID>\log_dir	Online log files
<drive 4>:\db2\<DBSID>\log_retrieve	Log retrieval directory during recovery
<drive 5>:\db2\<DBSID>\log_archive	Archived logs
<drive 6>:\db2\<DBSID>\db2dump	DB2 diagnostics log and service information
<drive n>:\db2\<SAPSID>\sapdata1..n	SAP database files

The values <DBSID>, <dbsid>, <SAPSID>, and <sapsid> are defined during the installation of the SAP system and represent either the database used by the SAP system (DBSID, dbsid) or the name of the SAP instance running the system (SAPSID, sapsid).

The values <drive 1>,<drive 2> and so on are the letters of the drives chosen during installing an SAP system. Even though it is not recommended, it is possible to install an SAP system in only one drive.

Directory structures on Linux and UNIX

SAP systems running DB2 UDB on a Linux and UNIX operating systems use the directories described in Table 2-4 to accommodate DB2 UDB information.

Table 2-4 Directory structures on Linux and Unix

Directory	Description
/opt/IBM/db2/V8.1 (HP-UX, Solaris, Linux) /usr/opt/db2_08_01 (AIX)	DB2 Software distribution binaries
/db2/db2<dbsid>	Home directory of user db2<sid>
/db2/<DBSID>/db2<dbsid>	DB2 Instance directory
/db2/<DBSID>/log_dir	Online log files
/db2/<DBSID>/log_archive	Archived logs
/db2/<DBSID>/log_retrieve	Log retrieval directory during recovery
/db2/<DBSID>/db2dump	DB2 diagnostics log and service information
/db2/<SAPSID>/sapdata1..n	SAP database files

2.4 DB2 UDB partitioning concepts

Beginning with DB2 UDB Version 8.1 for Linux, UNIX, and Windows, DB2 UDB combines Enterprise Edition and Enterprise-Extended Edition into a single edition DB2 UDB Enterprise Server Edition (ESE). The Database Partitioning Feature (DPF) is an optional licensing feature.

A common misunderstanding is that the DPF feature is required in order to process SQL statements in parallel. DB2 UDB supports parallel query processing with or without the DPF. If a machine has two or more CPUs, DB2 UDB can start subagents to work on a subset of the returned data. This type of parallelism provided by DB2 UDB is called intra-partition parallelism. This kind of parallelism is transparent for users.

In Figure 2-6, the user executes an SQL statement and DB2 UDB decides to process the query in parallel:

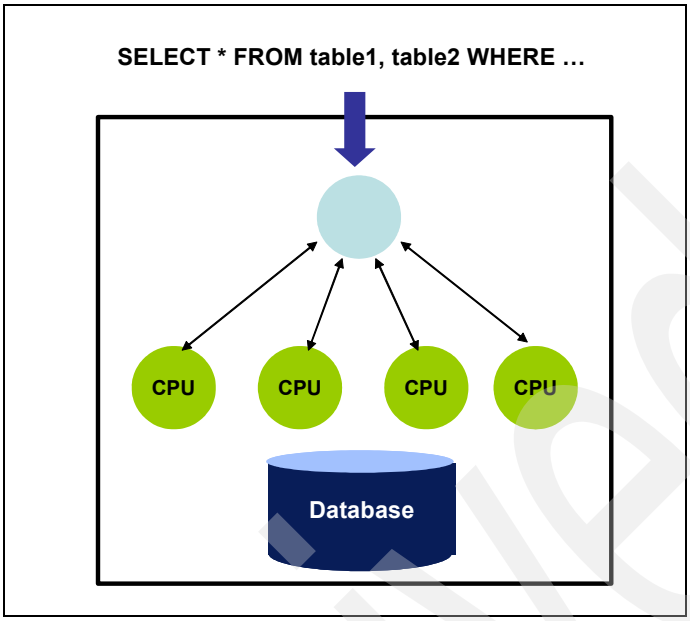


Figure 2-6 Intra-partition parallelism

With the database partitioning feature, you can partition a database in a group of machines, or create database partitions logically in the same machine. A database partition is a part of a database that consists of its own data, indexes, configuration files, and transaction logs.

Even though the database is partitioned, it appears to be a single database for users and applications. For example, using DPF, we create a partitioned database in four two-way machines and have a query partitioned and processed in parallel in each machine. This type of parallelism is called inter-partition parallelism and is shown in Figure 2-7.

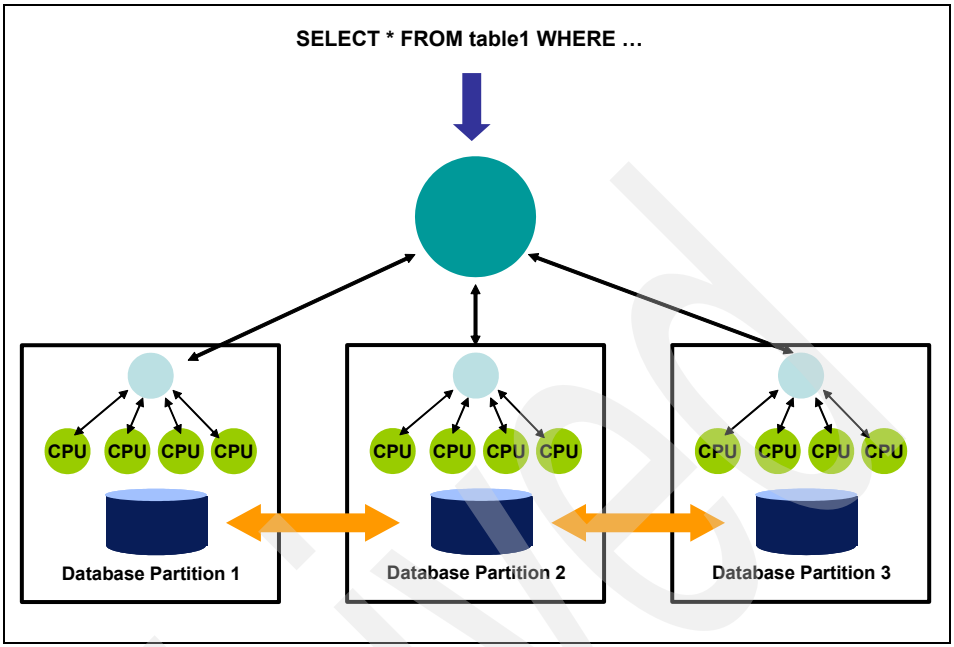


Figure 2-7 Inter-partition parallelism

2.4.1 Benefits of using the database partitioning feature

Here are some areas where you can find potential benefits using the database partitioning feature on DB2 UDB databases:

- Parallel query execution
- Architectural limits
- Data load operations
- Maintenance operations
- High availability
- Backup and restore

Parallel query execution

Using the database partitioning feature can increase the performance of query workloads and `INSERT`, `UPDATE`, and `DELETE` statements, because each database partition owns its own set of data and operations run in parallel.

The scalability provided is near linear as the data volumes or the number of processors and partitions grow.

Architectural limits

The database partitioning feature provides an option to overcome some of the DB2 UDB storage limits. Depending on the page size chosen when creating table spaces and the size of the tables created in it, it is possible to reach the maximum table space size, and you cannot add more space.

The maximum table space size is 64 GB with 4 K page size; the maximum table space size is 512 GB with 32 K page size. If we want to have tables whose size is larger than the maximum table space size, the database partitioning feature allows us to increase the maximum table space size by a factor that is equal to the number of partitions created. Even though it is possible to use this feature to overcome the maximum table space size problem, when there are table spaces with a small page size, the use of the DPF is not always the right solution.

Additionally, the database partitioning feature helps to overcome the 32-bit shared memory limit problem. Depending on the operating system, the shared memory limit of a 32-bit platform varies. Bounded by 32-bit machine architecture, you may face a situation where there is more memory than you can use because the memory is not addressable. In a DPF environment, each database partition manages its resources, so it is possible to run many logical database partitions on the same server to use all the available memory.

Database load operations

Load operations can also be run in parallel loading data into the corresponding database partitions and taking advantage of the DPF. Some situations require a very fast load process; the database partitioning feature may be an option.

Maintenance operations

The database partitioning feature can improve the elapsed time of the maintenance operations. Index creation is parallelized in DB2 UDB. Also, DPF can improve the total elapsed time by allowing DB2 UDB to work concurrently for each set of data across database partitions.

High availability

DB2 UDB with DPF uses a share-nothing architecture; this means that each database partition can use all resources available to it. In the case where a database partition fails, DB2 UDB can still access the other database partitions. This is an important advantage of the architectural design of DB2 UDB with DPF, which provides one type of high availability in the database.

Backup and restore

DB2 UDB parallelizes by default backup and restore operations. In a partitioned database, when you do a backup or restore of the database, once the catalog partition is backed up or restored, you can have DB2 UDB launch backup and restore processes or threads for each partition. Because these processes or threads run in parallel, but independently for each other, the total amount of time required for backing up a partitioned database is shorter. Additionally, restore time is also improved, because log files are not shared between partitions.

2.4.2 Using database partitioning feature in SAP systems

SAP systems can take advantage of the database partitioning feature of DB2 UDB. However, this does not mean that with each SAP system the DB2 UDB database partitioning feature can be used.

Standard SAP installations in OLTP environments provide application-side scalability through the addition of new application servers to the existing system. The OLTP SAP systems usually do not need more than one database partition. With appropriate sizing, the database server will not be the bottleneck of the system.

However, for an SAP BW system, you may want to consider using the database partitioning feature in DB2 UDB for partitioning the fact table, operational data storage table (ODS), and persistent storage table (PSA).

2.5 Database client and SAP database layer

In this section we describe the interaction between SAP application server and DB2 UDB. We describe first how SAP handles a request from an end-user and then describe the SAP database interface and its interaction with DB2 UDB.

Each request from a user is handled by a dialog work process, or a work process running in the SAP application server. Figure 2-8 shows the four components of a work process.

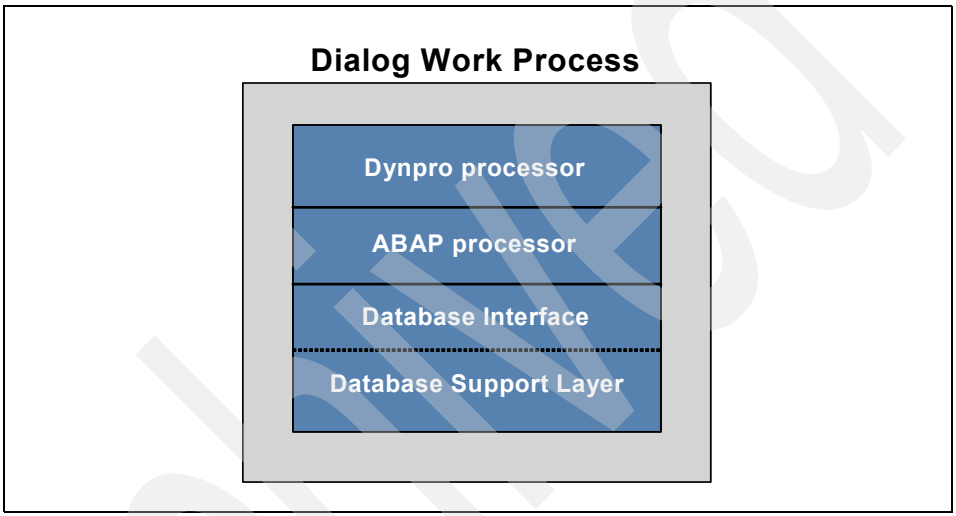


Figure 2-8 Dialog work process components

The dialog work process has a layered architecture. Each layer has a particular purpose:

- The *dynpro processor* translates information from screen into ABAP variables and vice versa. These variables are used by the ABAP processor when the input information of the request is processed. On the other hand, once the ABAP processor finishes its task, it leaves information in terms of ABAP variables for the dynpro processor to be output to screen.
- The *ABAP processor* executes the ABAP logic defined in the application. This logic can include simple tasks, but it can also include interaction with a database.
- The *database interface* is responsible for the interaction with the database in an independent way.
- The *database support layer* is the database dependent implementation of the database interface.

Each end-user connected to an SAP system is identified via a session. Every time a user wants to execute a task, the user sends a request to the SAP system. Once a request is sent to the system, the following sequence of steps occurs:

1. The request is classified by an internal process known as the *SAP dispatcher* and put into the appropriate queue.
2. The requests in the queue are sent to dialog work processes in the order that they arrive.
3. The available dialog work process, which must work with the request, invokes a subprocess called the *task handler* to bring into shared memory the user-context of the session making the request. This step of restoring the context of a session is known as “roll-in” and it allows the dialog work process to have information from the user’s current running transactions and its authorizations.
4. Once the user-context is available in memory, the task handler passes execution to a dynpro processor. The dynpro processor converts screen data into ABAP variables. The ABAP variables provide information to the ABAP processor.
5. The ABAP processor executes the coded logic in the ABAP application. For example, it invokes the coding of the process after the input module of the previous screen and the coding of the process before the output module of the next screen. If it is necessary to interact with the database, the ABAP processor also sends a request to the database interface layer.
6. Once the ABAP processor completes the request, the dynpro processor converts ABAP variables generated as output into screen fields and makes these screen fields available to the dispatcher.
7. The task handler becomes active again and saves the current user-context in shared memory. This process is known as “roll-out”.
8. Finally, the dispatcher returns the resulting data to the end-user.

Figure 2-9 shows the order followed to process a request that interacts with a database.

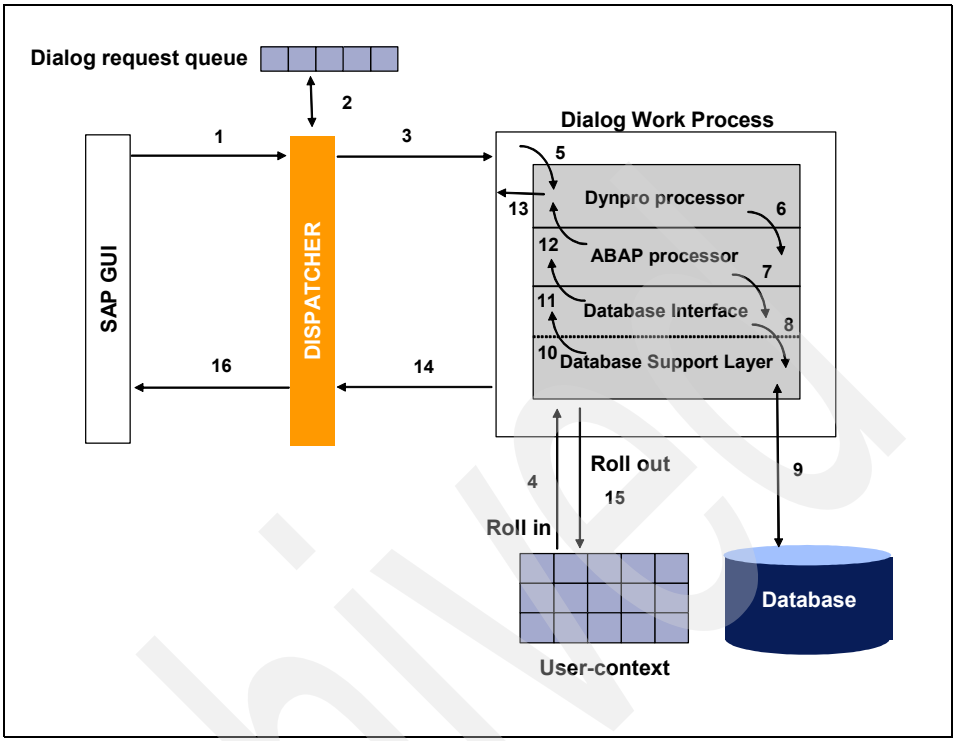


Figure 2-9 Interaction of a user with an SAP system

2.5.1 Interaction between the database interface layer and DB2 UDB

The previous section explained which tasks are executed by the SAP application server whenever a user sends a request to it. Now we explain what happens under the covers when these requests need information stored in DB2 UDB.

The database interface layer provides the ABAP processor a uniform way to communicate with a database engine. The SAP application server supports different vendor's databases; therefore the database interface layer depends on another layer, the database support layer, to access these different databases.

The database interface layer uses Open SQL statements to access databases. Open SQL is a set of SQL statements with enhancements that perform database operations; it offers a uniform syntax to access different databases, ensuring that SAP systems do not have problems when working with them.

Open SQL includes statements such as: `SELECT`, `INSERT`, `UPDATE`, `DELETE`, `MODIFY`, `OPEN CURSOR`, `FETCH`, `CLOSE CURSOR`, `COMMIT WORK`, and `ROLLBACK WORK`.

In the case of DB2 UDB, the database support layer is written using DB2 UDB Call Level Interface (CLI). DB2 UDB CLI is a set of functions written in the C programming language that allows developers to write code in C or C++ to access DB2 UDB. These programs access the database by invoking functions defined by the CLI standard. The CLI standard is in some extent similar to the ODBC standard.

Figure 2-10 shows the four components of a work process and how SAP application server interacts with DB2 UDB using its database interface.

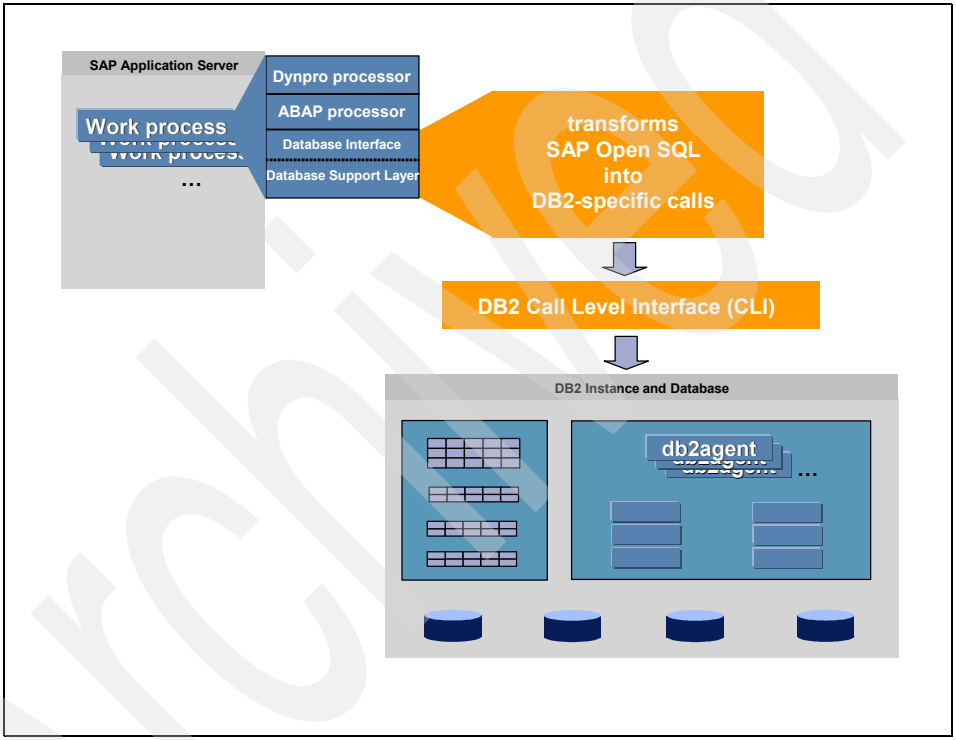


Figure 2-10 From SAP Open SQL to DB2 UDB CLI

Each work process running on an SAP system is associated with a db2agent. The db2agent is started the first time the work process connects to DB2 UDB and it is used later on for the next interactions with the database.

Before returning information back to screen, the work process issues a `commit` to the database. This strategy allows the same work process to handle requests from different end-users using the same database connection. The database transaction is finished after sending the commit. The work process always uses the same connection to avoid the overhead of connecting and disconnecting to the database each time it must handle a request from a different user.

SAP Open SQL is quite similar to regular SQL. No matter whether the SAP Open SQL statement issued is a read-only statement, such as a `SELECT`, or a statement that modifies data, such as an `INSERT`, `UPDATE` and `DELETE` statement, the interaction with DB2 UDB from the point of view of SAP systems is the same. In both cases, SAP Open SQL statements are translated into DB2 UDB CLI calls, DB2 UDB CLI calls are mapped into SQL statements, the db2agents associated with the SAP work processes receive the SQL statements, they optimize them, and finally the result set data is returned (`SELECT statement`) or modified (`INSERT`, `UPDATE`, and `DELETE` statements).

2.6 Terminology mapping with DB2 UDB

In this section we introduce frequently used terminology in the DB2 UDB environment and explain how those concepts are mapped to other relational database management systems. We provide terminology mapping tables for each of the following database servers:

- Oracle
- IBM Informix Dynamic Server
- MySQL MaxDB

2.6.1 DB2 UDB terms

Table 2-5 describes some important concepts for DB2 UDB environments.

Table 2-5 DB2 UDB terms

Term	Description
Instance	An instance is a set of processes running in the operating system and the portions of shared memory allocated when it starts. An instance must exist before a database can be created. In DB2 UDB, an instance is also known as the database manager (DBM). Many databases can be created in a DB2 UDB instance
Database	A database is the physical structure that contains the data. In DB2 UDB, many databases can be created and used concurrently in the same instance.

Term	Description
DBM and database configuration files	Files having the values for the different configuration parameters. In DB2 UDB, there is a configuration file for each instance (DBM) and one for each database. The configuration files are in binary format and cannot be edited manually. To access those files, you must use DB2 UDB commands.
Table spaces	A table space is a logical concept of storage that allows us to decide where we want to create tables and indexes.
Containers	A container is a physical concept of storage which is represented as the directories, files, or devices to actually hold the data.
Objects	Entities of the database inside the containers.
Extents	Contiguous set of pages allocated for an object.
Pages	Smallest storage unit. DB2 UDB can handle the following page sizes: 4, 8, 16 or 32 K.
System datalog	Metadata of the database. In DB2 UDB, each database creates has it owns set of system catalog tables.
SMS table space / containers	Uses a directory as its container. In this kind of table space, the operating system is responsible for managing the storage space.
DMS table space / containers	Uses a file or device as its container. DB2 UDB manages the storage space.
Buffer pools	Portions of shared memory used to buffer data from table spaces and to reduce disk I/O
Package cache	Caches prepared dynamic SQL statements.
Log files	They record the transactional activity in each database. In DB2 UDB, each database has its own set of log files.
Database manager and database shared memory	DB2 UDB allocates shared memory segments for the instance and for each database being used

2.6.2 Terminology mapping for Oracle databases

Table 2-6 describes how to map an Oracle term into a DB2 UDB concept.

Table 2-6 Oracle to DB2 UDB terminology mapping

Oracle	DB2 UDB	Description
Instance	Instance	An instance is a set of processes running in the operating system and the portions of shared memory allocated by the database manager when it starts. An instance must exist before a database can be created. Oracle instances can only have one database.
Database	Database	In Oracle, multiple instances can use the same database, and an instance can connect to one and only one database.
Control files and .ora files	DBM and database configuration files, etc.	Files that name the locations of files making up the database and provide configuration values.
Table spaces	Table spaces	Components of the database that holds the actual data.
Datafiles	Containers	Physical concept used by when creating table spaces.
Segments	Objects	Entities created inside the datafiles.
Extents	Extents	Contiguous space allocate for a segment.
Data blocks	Pages	Unit of storage.
Data dictionary	System catalog	Metadata of the database.
Datafiles	DMS containers	Files and raw devices used by the database server.
Data cache	Buffer pools	Buffers holding information from disk to reduce I/O.
Statement cache	Package cache	Memory that caches prepared dynamic SQL statements.
Redo logs	Log files	Recovery logs having records of all transaction activity.

Oracle	DB2 UDB	Description
Rollback segments	N/A	Store the old version of data for a mutating table. In DB2 UDB the old version of an updated row is stored in the log file along with the new version.
System global area (SGA)	Database manager and database shared memory	Segments of shared memory used by the database server.

2.6.3 Terminology mapping for IBM Informix Dynamic Server databases

Table 2-7 describes how to map an IBM Informix Dynamic Server term into a DB2 UDB concept:

Table 2-7 IBM Informix Dynamic Server to DB2 UDB terminology mapping

IBM IDS	DB2 UDB	Description
Instance	Instance	In IBM Informix Dynamic Server terminology, an instance is defined as a set of processes running in the operating system. The segments of shared memory are allocated to it and the disk space are used. Every database belongs to an instance and an instance can hold many databases.
Database	Database	Physical structure containing actual data.
ONCONFIG and sqlshosts file	DBM and database configuration files, etc.	In IBM Informix Dynamic Server, the ONCONFIG file stores the configuration parameters of the instance and the sqlhosts files stores the connectivity information used either by the database manager or by the database clients. There is no database configuration file.
Dbspaces	Table spaces	Logical collection of chunks. Tables and indexes can be created in different dbspaces.
Chunks	Containers	Portions of physical disk allocated to the instance.

IBM IDS	DB2 UDB	Description
Objects	Objects	Entities inside the dbspaces.
Extents	Extents	Set of contiguous space allocated for an object.
Pages	Pages	Basic unit of I/O for IBM Informix Dynamic Server.
System catalog	System catalog	Metadata of the database.
Cooked-file chunks	SMS containers	In IBM Informix Dynamic Server, files can be used to accommodate the database information. The file system manager is responsible for its access.
Raw device chunks	DMS containers	In comparison to DB2 UDB only in the case of raw devices, the database manager is responsible for the storage space.
Buffer pool	Buffer pools	In IBM Informix Dynamic Server, exists only one buffer pool for the whole instance. This buffer pool is shared by all the database in the instance and is part of a segment of shared memory.
Statement cache	Package cache	Caches the prepared dynamic SQL statements.
Log files	Log files	IBM Informix Dynamic Server also uses log files to record all the transactional activity. However, these files are not visible files in the operating systems, they reside in a dbspace of the instance.
Instance shared memory	Database manager and database shared memory	IBM Informix Dynamic Server allocates segments of shared memory for the whole instance and the databases. There is no allocation of shared memory for a database.

2.6.4 Terminology mapping for MySQL MaxDB databases

Table 2-7 describes how to map a MySQL MaxDB term into a DB2 UDB concept.

Table 2-8 MySQL MaxDB to DB2 UDB terminology mapping

MySQL MaxDB	DB2 UDB	Description
Database instance / Database manager	Instance	MySQL MaxDB makes a distinction between database instances and database manager. A database instance is a database together with the additional information required for running. The database manager is a database tool to manage database instances, database users, and database manager's operators.
Database	Database	Physical structure containing actual data.
Dbm configuration file and directories	DBM and database configuration files, etc.	MySQL MaxDB hosts its configuration file in the directory config\<database_name>. There is also a file called "dbm.cfg", that stores the configuration file of the database manager for the database instance.
Data volumes	Table spaces	In MySQL MaxDB, the data from the database is stored in data volumes.
N/A	Containers	
N/A	Objects	
N/A	Extents	Each page is stored depending on the B-Tree algorithm. There is no warranty that pages will be stored sequentially.
Pages	Pages	A page is the basic physical unit for disk I/O.However, MySQL MaxDB uses a B-Tree algorithm to store all data.
System tables	System catalog	Metadata of the database.
N/A	SMS containers	
N/A	DMS containers	

MySQL MaxDB	DB2 UDB	Description
Data cache	Buffer pools	MySQL MaxDB uses the data cache to store reading and writing data pages most recently used. This data cache is used by all users. In comparison to DB2 UDB, the data cache cannot be configured directly, because MySQL MaxDB takes the space from another cache called I/O buffer cache.
N/A	Package cache	
Log volumes / undo logs and redo logs	Log files	In MySQL MaxDB, the log entries of transactions are stored in log volumes. MySQL MaxDB uses undo logs to rollback transactions not yet committed and redo logs for rollforward purposes.
I/O buffer cache	Database manager and database shared memory	The most closed concept to the shared memory by DB2 UDB in a MySQL MaxDB is the I/O buffer cache. The I/O buffer cache is the main memory in MySQL MaxDB available for I/O operations.

3

Installing SAP NetWeaver 2004s with DB2 UDB V8.2.2

The official SAP installation documentation provides a complete list of requirements, procedures, and post-installation activities for SAP installation. In this chapter, we focus on important considerations as well as hints and tips for installing an SAP system with DB2 UDB V8.2.2.

We start by introducing a new SAP tool available for SAP NetWeaver 2004s called Prerequisite Checker. The tool will verify whether or not the system meets the SAP requirements for installation. It will generate a report listing the current configuration of the system, and the expected configuration values.

Finally, there is a section on how you might try to take advantage of the new DB2 UDB V8.2.2 features if you install SAP with older releases, for example, SAP R/3 Enterprise 4.7.

We also refer you to the official SAP installation documentation for the details.

3.1 Test environment

This section briefly describes the test environment used during the writing of this redbook. SAP NetWeaver 2004s and DB2 Universal Database (UDB) V8.2.2 for Linux, UNIX, and Windows were installed on the Windows and Linux operating systems. The installation procedure was similar for both operating systems. The Linux environment is our topic of discussion in this chapter.

3.1.1 Software configuration

This section describes the software and versions installed on the Linux machine used during the writing of this redbook.

The software environment consists of the components shown in Table 3-1.

Table 3-1 Software environment

Software component	Specification
Operating system	Linux SUSE LINUX Enterprise Server V9 Kernel version 2.6.5
Database	DB2 UDB V8.2.2 for Linux operating systems
SAP component	SAP NetWeaver 2004s
Java	Java Runtime Edition (JRE) V1.4.2_08 (required for the installation of SAP NetWeaver 2004s)

3.1.2 Hardware configuration

The hardware configuration of the machine running the Linux operating system is shown in Table 3-2.

Table 3-2 Hardware components

Hardware component	Specifications
Processor	Intel (R) Xeon™ (TM) 2.80 GHz Processor 4-way machine
RAM	8 Gb
Hard disk	4 SCSI disks (approximately 110 Gb): 10 Gb were used for the operating system, 6 Gb were used as swap space and the remaining disk was assigned to three volume groups. These volumes groups were used for the SAP, DB2 UDB installation and for database files.

3.2 SAP installation guides

The SAP documentation is the first place you should look for information if you are in doubt about any step of the installation procedure, or about the SAP products in general.

Whenever you plan to install a new SAP system, it is recommended to read the installation guides first. The latest release of the SAP installation guides are available in the SAP Service Marketplace and can be accessed using the following link:

`http://service.sap.com/instguides`

To access this site, a user ID and password are required. These are usually provided for you if you are an SAP customer or partner.

3.3 System preparation

The SAP installation guide provides you with a complete system preparation checklist. In this section, we introduce you to the Prerequisites Checker Tool, a new tool that comes with SAP NetWeaver 2004s.

As the name implies, this tool checks whether the host machine on which you plan to install an SAP system is in compliance or not with the installation requirements that are documented in the installation guides.

3.3.1 The Prerequisites Checker Tool

Before the actual installation is performed, you need to make sure that your hardware and software environment meets the requirements from SAP. This can be done via the checklist provided in the installation guides. The Prerequisites Checker Tool offers a quick way to check if your system meets the SAP installation prerequisites. After running on the system, it will generate a report listing the configuration of the system and the expected configuration values. We used this tool to verify our system environment.

It is important to mention that this tool only checks the prerequisites, it does not make any changes to the configuration of the system. If it finds any prerequisite that is not met, the appropriate task must be carried out manually.

Note: This tool is only an informational tool. It provides system status but will not prevent the user from continuing the system installation when unsatisfied prerequisites are found.

3.3.2 Checking conditions

In the Prerequisites Checker Tool, a *condition* is a prerequisite that the host machine must satisfy. Each condition has a severity. The possible severity values are: low, medium, and high, and are associated with the colors yellow, orange, and red, respectively, in the status report of the tool. The severity value depends on how important it is for a specific condition to pass for installation. Some conditions are critical, while others return warning messages.

The number and type of conditions depend on the operating system. Some conditions are general for all the operating systems and some are specific for a particular operating system.

In this redbook we describe the general conditions and the operating system (OS) dependent conditions for SUSE LINUX and Windows operating systems. For the particular conditions for all operating systems, refer to the SAP documentation at the Web site:

`http://service.sap.com/instguides`

Table 3-3 describes the general conditions checked by the tool in the Linux and Windows operating systems.

Table 3-3 General conditions

Condition	Description
Host name	The host name must be an alphanumerical string of characters [a-z][A-Z], digits [0-9] or the hyphen sign (-). Although new RFCs allow host name beginning with digits, it is recommended that the host name begins with an alphabetical character. The maximum host name length is 13 characters. (SAP Note *611361*).
RAM size	Size of memory required depending on the type of SAP instance to install.
SWAP size	The swap size varies according the type of SAP instance.

Table 3-4 shows the conditions we found during a Windows installation of SAP NetWeaver 2004s with DB2 UDB V8.2.2.

Table 3-4 OS dependent conditions for Windows operating system

Condition	Description
Operating system version	This condition describes the Windows operating systems supported, including required service pack. The tool also informs you of the current operating system version.
Names of well known groups	Only English international versions of Windows are supported (SAP Note *362379*). The installation expects to find the groups: `everyone`, `users`, `guests`, `power users`. If this condition is not satisfied, it could be because the Windows systems is a localized one.
Domain controllers	The installation of SAP systems on domain controllers are not supported.

Table 3-5 shows the conditions checked during a Linux installation of SAP NetWeaver 2004s with DB2 UDB V8.2.2.

Table 3-5 OS dependent conditions for SUSE LINUX operating system

Condition	Description
LINUX distribution	Depending on the Linux release used, this condition checks whether the release is supported or not.
Kernel parameter `msgmni`	The kernel parameter, `msgmni`, defines the maximum number of message queues available in the system. A value of 1024 or larger is required for DB2 UDB on Linux.
Package `pdksh`	The package `Public Domain Korn Shell (pdksh)` must be installed for DB2 UDB installations on Linux.
`glibc` version	The tool verifies if the expected `glibc` version is in place. It also informs you what is the current `glibc` version.
Minimum `glibc` version	This condition informs you which is the minimum `glibc` version to run SAP systems.
Kernel version	The tool verifies if the kernel version is supported. It also informs you of the actual kernel version.
Minimum kernel version	This condition informs you which is the minimum kernel version to run SAP systems.
`saplocales` version	Check is applied to find out whether the `saplocales` packages are installed in the system or not. The actual `saplocale` is displayed.
Minimum `saplocales` version	This condition informs you which is the minimum `saplocales` version to run SAP systems.
Package `suse-sapinit`	Some Linux distributions, like SUSE LINUX, require an additional package with kernel modifications to run SAP systems.

3.3.3 Running modes

There are two ways to run the Prerequisite Checker Tool, it can either run as a stand-alone application or it can be invoked automatically during the installation of an SAP system.

Stand-alone mode

To execute the stand-alone version of the Prerequisite Checker Tool, you need to run the installation procedure. This means that you need to prepare all of your environment to run the `SAPinst` executable.

However, instead of choosing to install an SAP instance, you go to **System Lifecycle Management** → **Check Tools** and click **Prerequisites Check** as shown in Figure 3-1.

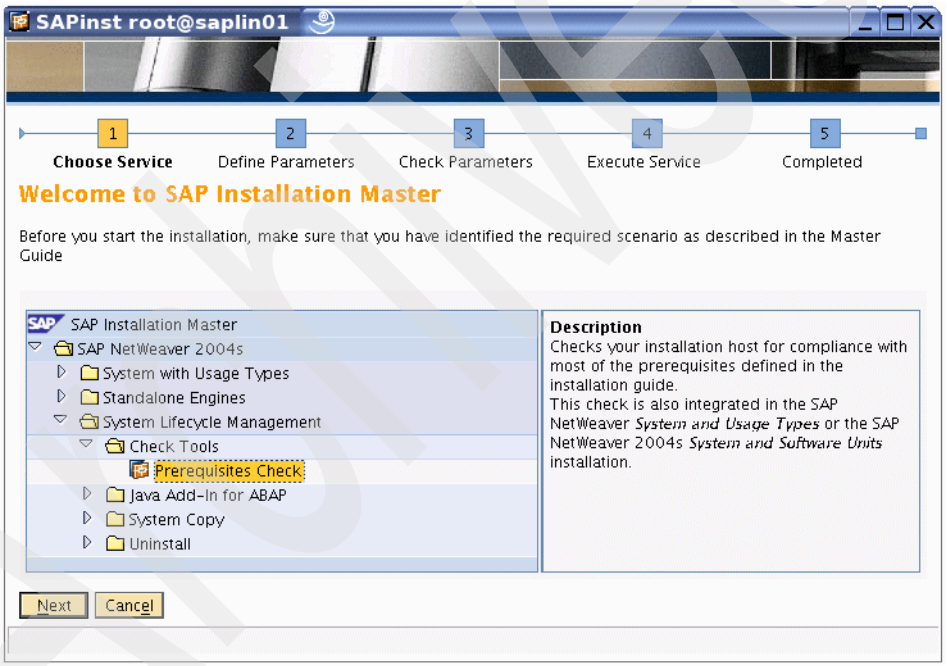


Figure 3-1 Selecting Prerequisites Check tool

Click the **Next** button to choose the Checker data file to be used to verify the prerequisites. The checker data file is an XML file that has all the information regarding the conditions the tool will check.

This design gives flexibility to apply new prerequisites if conditions are changed. You only need to point to the data file with the new checks. New versions of the XML file can also be downloaded from the SAP Web site to ensure that the latest prerequisites are checked.

To select the checker data file, click the **Browse** button Browse... and search for the file you decide to use. See Figure 3-2.

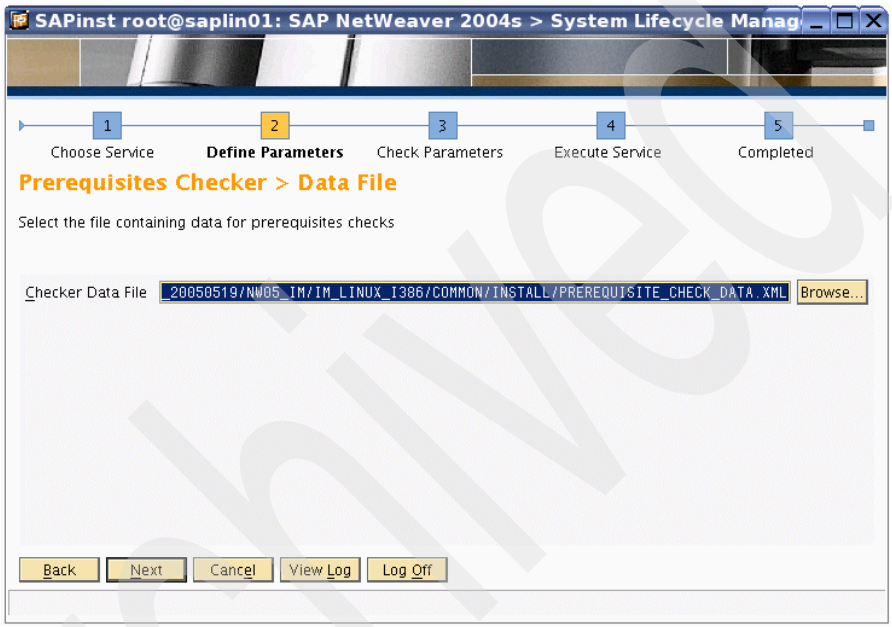


Figure 3-2 Choosing the Checker Data File

The next few screens are used to choose the type of SAP system for installation, the database system, and whether or not unicode is enabled. Figure 3-3 shows the drop-down list used to choose the database manager system to use in the SAP system. Select **DB2 Universal Database for Unix/Windows**.

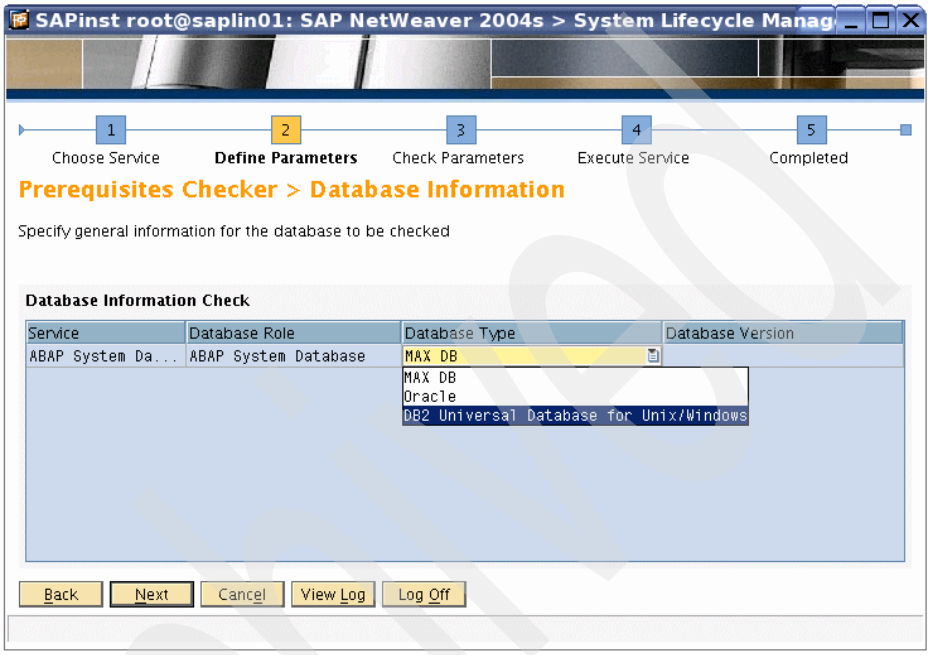


Figure 3-3 Choosing DB2 UDB as database type

Before executing the checks, the Prerequisite Checker Tool lets you modify your selections. By selecting the check-box associated with each step of the tool and clicking the **Edit** button as shown in Figure 3-4, you can make the modifications.

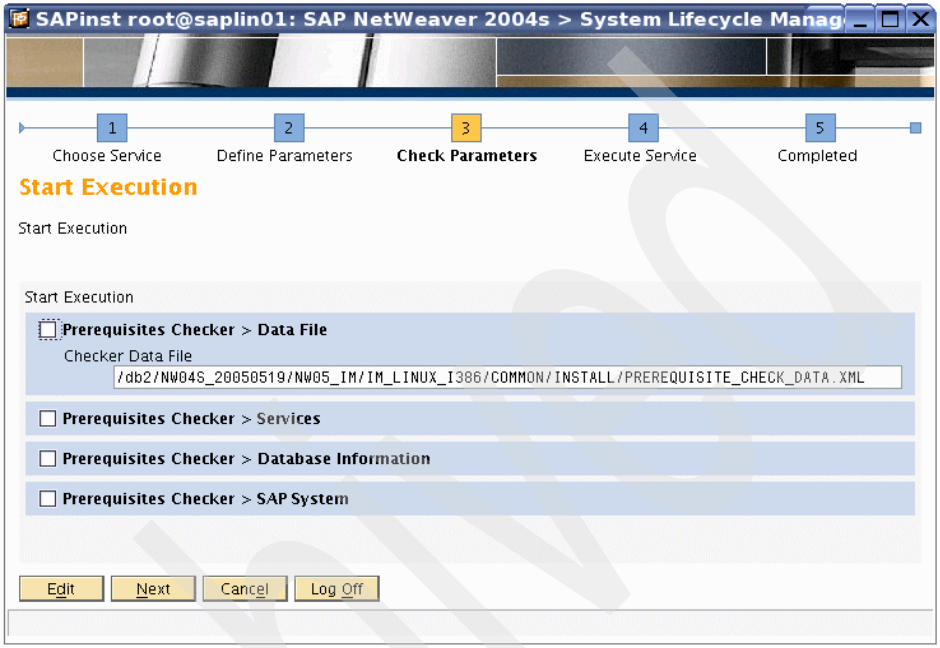


Figure 3-4 Editing parameters before launching execution

As a result of the checks made, the Prerequisite Checker Tool generates a status report. This report contains information about each condition verified and whether the condition is met or not. For each condition, the tool provides the following information:

- A result code with the values “OK” or “Condition does not hold”.
- A severity level with the values of LOW, MEDIUM, or HIGH.
- A message describing the condition, the actual condition value, and the expected value.

Figure 3-5 shows conditions that were not satisfied with different severities. If you want to be sure that your system is compliant with all the prerequisites, you should get a status report in which the result code for each condition is “OK”.

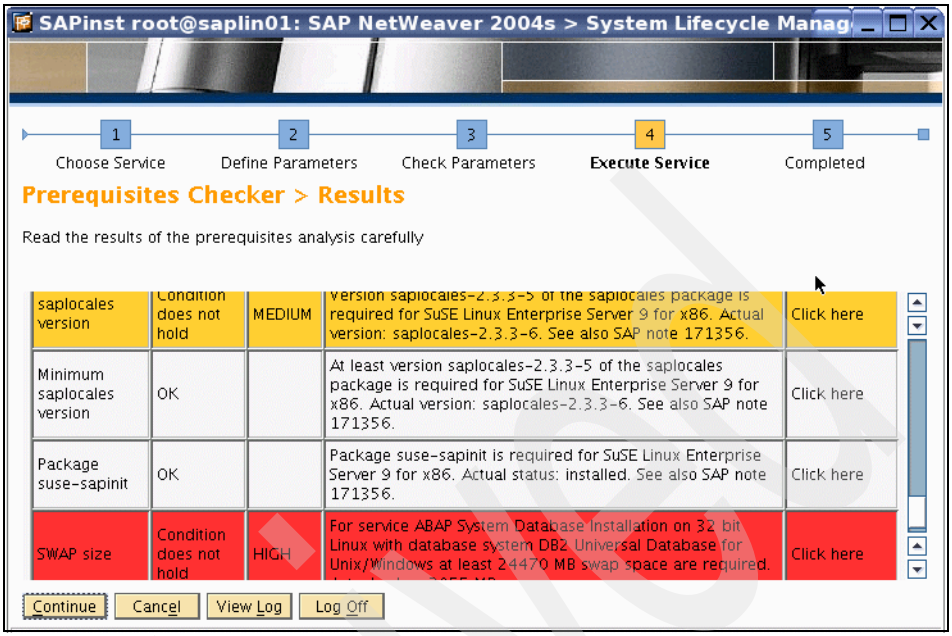


Figure 3-5 Status report

Integrated mode

Using this mode, the Prerequisite Checker Tool is automatically invoked during the installation procedure.

In the integrated mode, required information, such as the type of SAP system to install, database to be used, using a unicode database or not, is supplied in the previous installation steps. The only step you need to complete is to indicate the location of the checker data file. A status report is then generated.

3.4 Considerations: Installing SAP NetWeaver 2004s with DB2 UDB V8.2.2

This section highlights the considerations you should take when installing SAP NetWeaver 2004s with DB2 UDB V8.2.2. Most of the new version 8.2.2 features are automatically enabled during the installations. For some of the new features you are required to make a selection. An example would be whether you select automatic storage or auto-resize DMS. There are some restrictions that you should know before running the installation. Refer to the *SAP NetWeaver 2004s Installation Guide* for the complete information on SAP NetWeaver 2004s installation.

3.4.1 Features that are automatically switched on

In the SAP NetWeaver 2004s installation, some of the DB2 UDB V8.2.2 new features are automatically switched on:

- *DB2 Workload optimization for SAP Environment.* This feature in turn switches on other V8.2.2 features such as *evaluate uncommitted*, *skip inserted*, and *improved Index heuristics* by setting the corresponding DB2 profile parameters. SAP environment specific DB2 Registry variables are enabled by issuing **db2set DB2_WORKLOAD=SAP**.
- *Automatic statistic collection*
- *Uniform page size*

Uniform page size

In SAP NetWeaver 2004s, a page size of 16K bytes (`PAGESIZE 16 K`) is used in the `CREATE DATABASE` statement. Here is an example of the script `createDatabase.clp` that creates the database taken from the installation directory of our SAP NetWeaver 2004s installation on Linux:

```
create database DAP on /db2/DAP using codeset ISO8859-1 territory en_US
collate using IDENTITY pagesize 16 k dft_extent_sz 2 catalog tablespace
managed by database using (
FILE '/db2/DAP/sapdata1/NODE0000/SYSCATSPACE.container000' 12800,
FILE '/db2/DAP/sapdata2/NODE0000/SYSCATSPACE.container001' 12800,
FILE '/db2/DAP/sapdata3/NODE0000/SYSCATSPACE.container002' 12800,
FILE '/db2/DAP/sapdata4/NODE0000/SYSCATSPACE.container003' 12800)
extentsize 2 prefetchsize 2 autoresize yes maxsize none
with 'SAP database DAP';
```

Since the database is created with a page size of 16K bytes and all of the table spaces are created with 16K bytes including `SYSCATSPACE`, we only need one buffer pool that has page size of 16K bytes. In our SAP NetWeaver 2004s installation, there is only one buffer pool `IBMDEFAULTBP` created and it is assigned to all the table spaces.

Attention: With 16K bytes page size, the maximum size of the table spaces are 256 GB. If necessary, you can create new table spaces with a larger page size (for example, 32K bytes) after the installation even though the database is created with a page size of 16K bytes.

3.4.2 Storage management and table spaces: considerations

With SAP NetWeaver 2004s, there are some new DB2 UDB V8.2.2 features for configuring the initial database layout: *automatic storage* and *auto-resize DMS*. Figure 3-6 is the installation screen that shows the options.

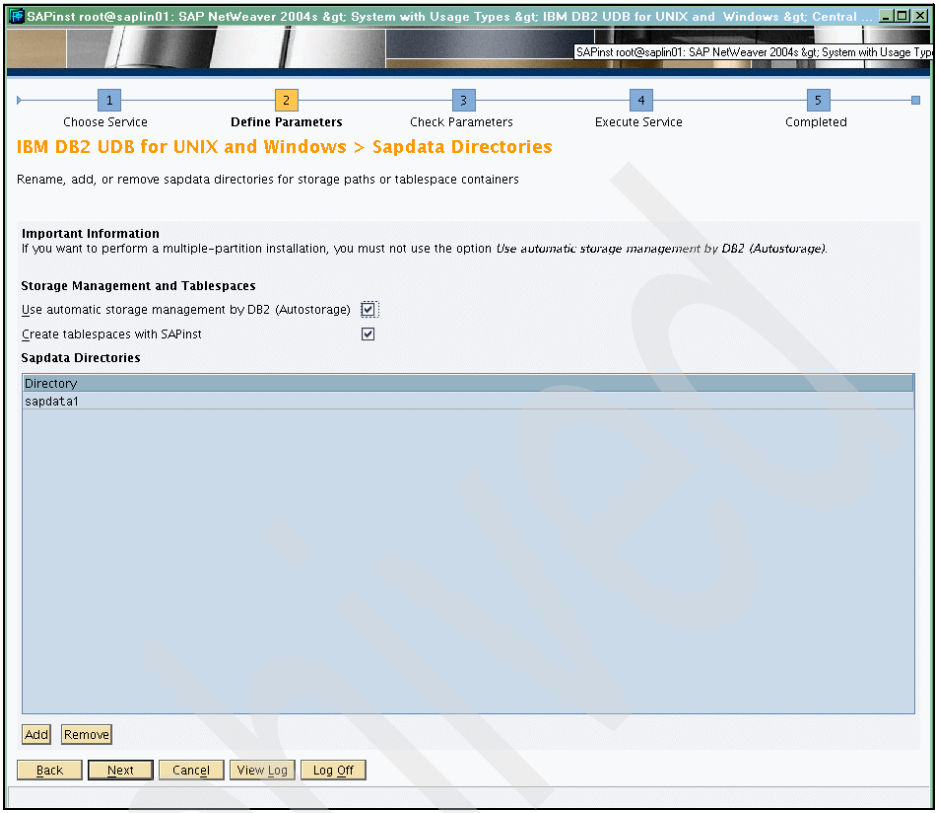


Figure 3-6 Storage Management and Table spaces: Automatic Storage option

Automatic storage

This allows the database manager to decide where to put table space containers among a set of storage paths and resize them when necessary. In SAP NetWeaver 2004s, each `sapdata<x>` represents a storage path (for example, `sapdata1` represents the storage path, `/db2/<SID>/sapdata1`). This feature is used when you select the Automatic Storage option.

Auto-resize DMS

This feature is similar to the traditional Database Managed Space in that you decide where to put the table space containers. With auto-resize DMS, you have the additional benefit that whenever the table space becomes full, the database manager automatically extends the size of the containers as long as there is enough space in the underlying file system. This feature is used when you *uncheck* the Automatic Storage option.

By default, the containers for the auto-resize DMS enabled table spaces are created across all `sapdata<x>`. For example, if you have defined `sapdata1`, `sapdata2`, and `sapdata3`; all of your auto-resize DMS enabled tables spaces will have three containers. If you would like to customize the table space layout, refer to the next section about *Customizing the table space layout*.

Customizing the table space layout

In the installation screen where you choose the Storage Management and Tablespaces (see Figure 3-6 on page 69), there is an option called *Create tablespace with SAPinst*. By default, this option is checked, meaning that the SAP installation creates the table space for you using the default layout. If you would like to customize the table spaces, you can uncheck the *Create tablespace with SAPinst* option.

`SAPinst` pauses at the point where it creates table spaces (see Figure 3-7). Notice that `SAPinst` creates a table space creation template script called `createTablespaces.clp` which contains the `CREATE TABLESPACE` statements for all the table spaces in the system in the installation directory. You can modify the script; execute it to create the table spaces manually and then continue with the installation.

Example: Customizing the table space layout

Normally, for an optimized database layout, table spaces are created to spread across as many I/O subsystems or disks as possible to take advantage of the I/O parallelism. In this example, however, the customer has four file systems for holding the database containers: the first three file systems have plenty of space but the fourth one is limited in space. Therefore, the customer does not want to spread all the table spaces across to include the fourth file system (that is, `sapdata4`). The customer chooses to use auto-resize DMS (not to use Automatic Storage).

Here are the steps of this table space customization:

1. In the installation screen where you choose Storage Management and Tablespaces, uncheck the option *Use automatic storage management by DB2 (autostorage)* and select *Create tablespaces with SAPInst* (see Figure 3-7) and continue with the rest the installation.

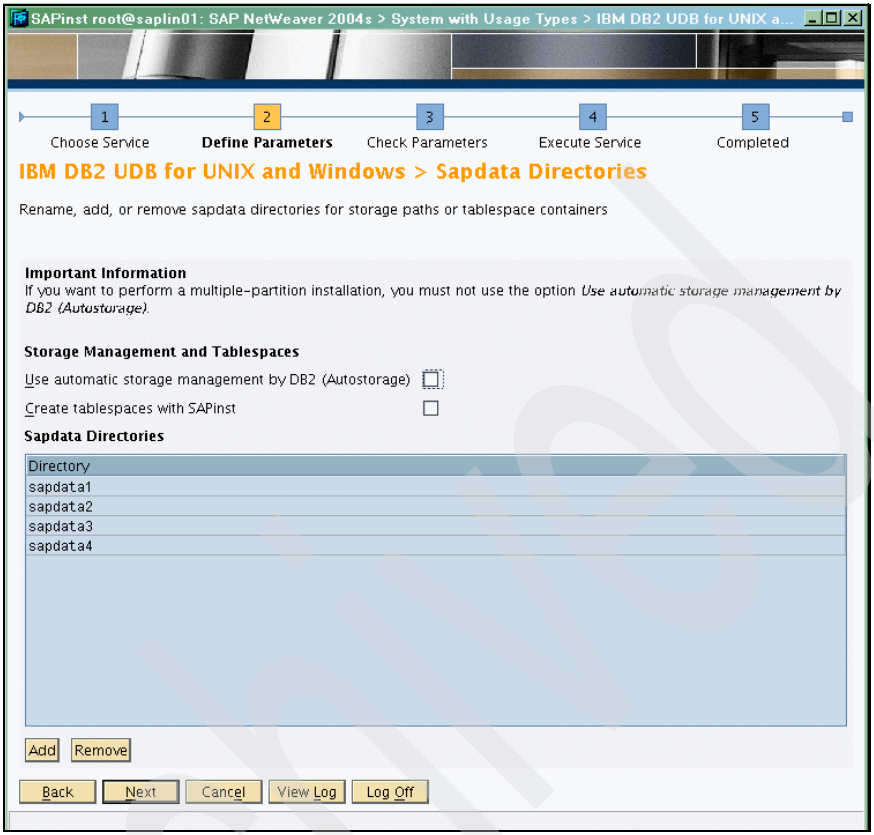


Figure 3-7 The Create tablespace with SAPinst option

2. At the phase DB2 Create Tablespace (after the DB2 instance and database is created), a message box pops up with the message, `You must create the Tablespaces now` (refer to Figure 3-8). In this figure, the SAP installation pauses, waiting for you to finish making changes to the `create tablespace` template script and execute it. This is the point when you can make changes to the table space layout.

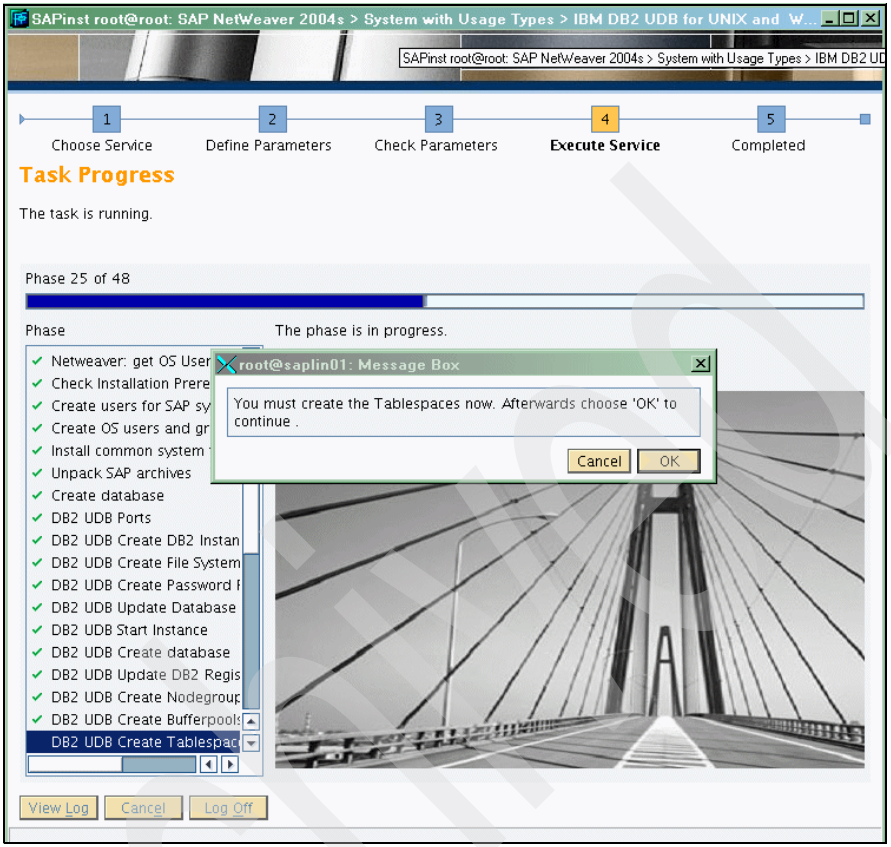


Figure 3-8 SAP installation pauses, waiting for user to finish making changes

3. Go into the installation directory (this is the directory where you start your SAP installation). You should see a file named, `createTablespaces.clp`. Use a text editor program to open the file.

4. As you can see, the default for the `SAPinst` is to create table space containers on all file systems (that is, sapdata1 to sapdata4 in this example):

```
...
create tablespace DAP#PROTI
  in nodegroup SAPNODEGRP_DAP pagesize 16k managed by database  using (
   FILE '/db2/DAP/sapdata1/NODE0000/DAP#PROTI.container000' 15 M ,
   FILE '/db2/DAP/sapdata2/NODE0000/DAP#PROTI.container001' 15 M ,
   FILE '/db2/DAP/sapdata3/NODE0000/DAP#PROTI.container002' 15 M ,
   FILE '/db2/DAP/sapdata4/NODE0000/DAP#PROTI.container003' 15 M )
  on node ( 0 )  extentsize 2 prefetchsize automatic  dropped table
recovery off
  autoresize yes maxsize none;
```

```
create tablespace DAP#ES700D
  in nodegroup SAPNODEGRP_DAP pagesize 16k managed by database  using (
   FILE '/db2/DAP/sapdata1/NODE0000/DAP#ES700D.container000' 1350 M ,
   FILE '/db2/DAP/sapdata2/NODE0000/DAP#ES700D.container001' 1350 M ,
   FILE '/db2/DAP/sapdata3/NODE0000/DAP#ES700D.container002' 1350 M ,
   FILE '/db2/DAP/sapdata4/NODE0000/DAP#ES700D.container003' 1350 M )
  on node ( 0 )  extentsize 2 prefetchsize automatic  dropped table
recovery off
  autoresize yes maxsize none;

create tablespace DAP#ES700I
  in nodegroup SAPNODEGRP_DAP pagesize 16k managed by database  using (
   FILE '/db2/DAP/sapdata1/NODE0000/DAP#ES700I.container000' 225 M ,
   FILE '/db2/DAP/sapdata2/NODE0000/DAP#ES700I.container001' 225 M ,
   FILE '/db2/DAP/sapdata3/NODE0000/DAP#ES700I.container002' 225 M ,
   FILE '/db2/DAP/sapdata4/NODE0000/DAP#ES700I.container003' 225 M )
  on node ( 0 )  extentsize 2 prefetchsize automatic  dropped table
recovery off
  autoresize yes maxsize none;
...
```

5. Edit the `CREATE TABLESPACE` statement for the table space(s) that you do not want containers to exist on sapdata4 (for example, big table spaces like `DAP#ES700D` and the corresponding index table space, `DAP#ES700I`). In the `CREATE TABLESPACE` statement for that table space, remove the container definition on `sapdata4`. In the example, the container definitions to be removed are identified with crossed line:

```
...
create tablespace DAP#PROTI
  in nodegroup SAPNODEGRP_DAP pagesize 16k managed by database  using (
   FILE '/db2/DAP/sapdata1/NODE0000/DAP#PROTI.container000' 15 M ,
   FILE '/db2/DAP/sapdata2/NODE0000/DAP#PROTI.container001' 15 M ,
   FILE '/db2/DAP/sapdata3/NODE0000/DAP#PROTI.container002' 15 M ,
   FILE '/db2/DAP/sapdata4/NODE0000/DAP#PROTI.container003' 15 M )
  on node ( 0 )  extentsize 2 prefetchsize automatic  dropped table
recovery off
  autoresize yes maxsize none;

create tablespace DAP#ES700D
  in nodegroup SAPNODEGRP_DAP pagesize 16k managed by database  using (
   FILE '/db2/DAP/sapdata1/NODE0000/DAP#ES700D.container000' 1350 M ,
   FILE '/db2/DAP/sapdata2/NODE0000/DAP#ES700D.container001' 1350 M ,
   FILE '/db2/DAP/sapdata3/NODE0000/DAP#ES700D.container002' 1350 M ,
   FILE '/db2/DAP/sapdata4/NODE0000/DAP#ES700D.container003' 1350 M )
  on node ( 0 )  extentsize 2 prefetchsize automatic  dropped table
recovery off
  autoresize yes maxsize none;
```

```
create tablespace DAP#ES70OI
  in nodegroup SAPNODEGRP_DAP pagesize 16k managed by database  using (
   FILE '/db2/DAP/sapdata1/NODE0000/DAP#ES70OI.container000' 225 M ,
   FILE '/db2/DAP/sapdata2/NODE0000/DAP#ES70OI.container001' 225 M ,
   FILE '/db2/DAP/sapdata3/NODE0000/DAP#ES70OI.container002' 225 M ,
   FILE '/db2/DAP/sapdata4/NODE0000/DAP#ES70OI.container003' 225 M )
  on node ( 0 )  extentsize 2 prefetchsize automatic  dropped table
recovery off
  autoresize yes maxsize none;
...
```

Save the `createTablespaces.clp` file and exit the text editor.

6. Run the `createTablespaces.clp` file to create the customized table spaces layout:

```
saplin01 /db2/inst_dir_resize> db2 -tvf createTablespaces.clp
create temporary tablespace PSAPTEMP16 in nodegroup IBMTEMPGROUP pagesize
16k  managed by system  using (
'/db2/DAP/saptemp1/NODE0000/temp16/PSAPTEMP16.container000' ) on node ( 0 )
extentsize 2 prefetchsize automatic  dropped table recovery off
DB20000I  The SQL command completed successfully.

create user temporary tablespace SYSTOOLSTMPSPACE in nodegroup IBMCATGROUP
pagesize 16k  managed by system  using (
'/db2/DAP/saptemp1/NODE0000/temp16/SYSTOOLSTMPSPACE.container000' ) on node
( 0 )  extentsize 2 prefetchsize automatic  dropped table recovery off
DB20000I  The SQL command completed successfully.
...
create tablespace DAP#ES70OD in nodegroup SAPNODEGRP_DAP pagesize 16k
managed by database  using ( FILE
'/db2/DAP/sapdata1/NODE0000/DAP#ES70OD.container000' 1350 M , FILE
'/db2/DAP/sapdata2/NODE0000/DAP#ES70OD.container001' 1350 M , FILE
'/db2/DAP/sapdata3/NODE0000/DAP#ES70OD.container002' 1350 M ) on node ( 0 )
extentsize 2 prefetchsize automatic  dropped table recovery off autoresize
yes maxsize none
DB20000I  The SQL command completed successfully.
create tablespace DAP#ES70OI in nodegroup SAPNODEGRP_DAP pagesize 16k
managed by database  using ( FILE
'/db2/DAP/sapdata1/NODE0000/DAP#ES70OI.container000' 225 M , FILE
'/db2/DAP/sapdata2/NODE0000/DAP#ES70OI.container001' 225 M , FILE
'/db2/DAP/sapdata3/NODE0000/DAP#ES70OI.container002' 225 M ) on node ( 0 )
extentsize 2 prefetchsize automatic  dropped table recovery off autoresize
yes maxsize none
DB20000I  The SQL command completed successfully.
...
```

7. After the table space creation is done, double-check that the customized table space layout is in place by issuing `LIST TABLESPACES` and `LIST TABLESPACE CONTAINERS` command:

```
saplin01 /db2/inst_dir_resize> db2 list tablespaces
...
Tablespace ID                        = 24
 Name                                 = DAP#ES700D
 Type                                 = Database managed space
 Contents                             = Any data
 State                                = 0x0000
   Detailed explanation:
     Normal
...
saplin01 /db2/inst_dir_resize> db2 list tablespace containers for 24

            Tablespace Containers for Tablespace 24

 Container ID                         = 0
 Name                                 =
/db2/DAP/sapdata1/NODE0000/DAP#ES700D.container000
 Type                                 = File

 Container ID                         = 1
 Name                                 =
/db2/DAP/sapdata2/NODE0000/DAP#ES700D.container001
 Type                                 = File

 Container ID                         = 2
 Name                                 =
/db2/DAP/sapdata3/NODE0000/DAP#ES700D.container002
 Type                                 = File
```

In the result, we have confirmed that this is what we want: the table space `DAP#ES700D` is only spread across `sapdata1`, `sapdata2`, `sapdata3`.

8. Switch back to the SAP installation and continue the installation by confirming the **OK** button in the message box that you see in step 2.

3.4.3 Database Partitioning Feature (DPF) considerations

Starting from SAP NetWeaver 2004s, if you want to create a DPF enabled system, you have to finish the SAP NetWeaver 2004s installation with a single partition first; and then execute the following steps:

1. Add one or more database partition(s) by re-running the `SAPinst` with the *Additional Database Partitions* option.
2. Use the new DBA Cockpit feature, **Wizard →Partition Integration** to:
   a. Assign one or more newly added database partition(s) to one or more database partition group(s).
   b. Define table space containers on the new database partitions.
   c. Redistribute the data of the table spaces if necessary.

Important: The Database Partitioning Feature is not supported by automatic storage. If you would like to install a system with DPF, you cannot use automatic storage.

3.5 Post installation activities: a DB2 UDB view

After finishing the installation of SAP NetWeaver 2004s, the activities described in the *Component Installation Guide - SAP NetWeaver 2004s* must be executed. The tasks to be done are either related to the SAP system, to DB2 UDB V8.2.2, or to both (that is, updating passwords of users created, and so on).

One of the most important activities to do after the installation of SAP NetWeaver 2004s and DB2 UDB is to enable the log retention mode of DB2 UDB databases.

Each DB2 UDB database has a set of log files recording all the transaction information done in the database. In case of a failure, DB2 UDB can use these log files to replay the operations done after the last backup and recover the transactions to a point in time.

The way you enable log retention in DB2 UDB depends on the version you are working on. The exact details and considerations of what you must do are described in 7.1, “Log file management” on page 314.

3.6 Considerations: Installing older SAP systems with DB2 UDB V8.2.2

During the Ramp-Up phase of SAP NetWeaver 2004s, SAP AG also plans to provide installation kits supporting the installation of SAP systems prior to SAP NetWeaver 2004s with DB2 UDB V8.2.2. SAP AG intends to provide these installation kits for SAP 3.1I - 4.6D releases (`R3SETUP` based installation) and for 6.10 - 6.40 releases (`SAPinst` based installation).

During the SAP installation, with these new kits, the `R3SETUP` or `SAPinst` checks the version of the installed DB2 UDB software.

With a DB2 UDB V8 version prior to V8.2.2 (FixPak 2-8), the `R3SETUP` or `SAPinst` installs the SAP system in the well known way without enabling any of the DB2 UDB V8.2.2 features.

When `R3SETUP` or `SAPinst` detects DB2 UDB V8.2.2 (FixPak 9) during the installation of the SAP systems, it activates the following new DB2 UDB V8.2.2 features:

- *Uniform page size:*
  `R3SETUP` or `SAPinst` creates the DB2 UDB database with a uniform page size of 16K, an extent size of 2K, and the prefetch size set to automatically.
- *Auto-resize DMS table spaces:*
  The SAP installation creates all Database Managed Space (DMS) table spaces of the DB2 UDB database with `AUTORESIZE` set to `YES` so that they can grow automatically when more space is needed in a table space.
- *DB2_WORKLOAD=SAP:*
  `R3SETUP` or `SAPinst` sets this DB2 registry variable so that it is no longer necessary to maintain all the registry settings individually as has been the case in DB2 UDB versions prior DB2 UDB V8.2.2.

Unlike the SAP NetWeaver 2004s installation, there is no support of *automatic storage* in the SAP system prior to SAP NetWeaver 2004s installations because the Computing Center Management System (CCMS) in these systems does not support this feature.

4

Storage management in depth

In this chapter we describe the following topics:

- DB2 UDB's logical and physical database objects: tables, indexes, views, table spaces, and table space containers
- Properties of SMS, file based DMS, and device based DMS table spaces, as well as how to work with them manually or using auto-resize and automatic storage features
- Introduction to SAPs data classes
- Space reclamation strategies and when to use them: Inplace and offline table and index reorganization and the reorganization criteria
- High water mark and how to handle it
- Best practices for intelligent storage subsystems
- SAP DB2 UDB table conversion tool `DB6CONV`, including a complete example

4.1 DB2 UDB logical and physical database objects

The DB2 Universal Database (DB2 UDB) consists of several logical and physical database objects. *Logical objects* are tables, indexes, views, and table spaces. In contrast, there are also *physical objects* such as table space containers, which can be database files, raw devices, or directories.

In Figure 4-1 the relationship between the logical and physical database objects is shown.

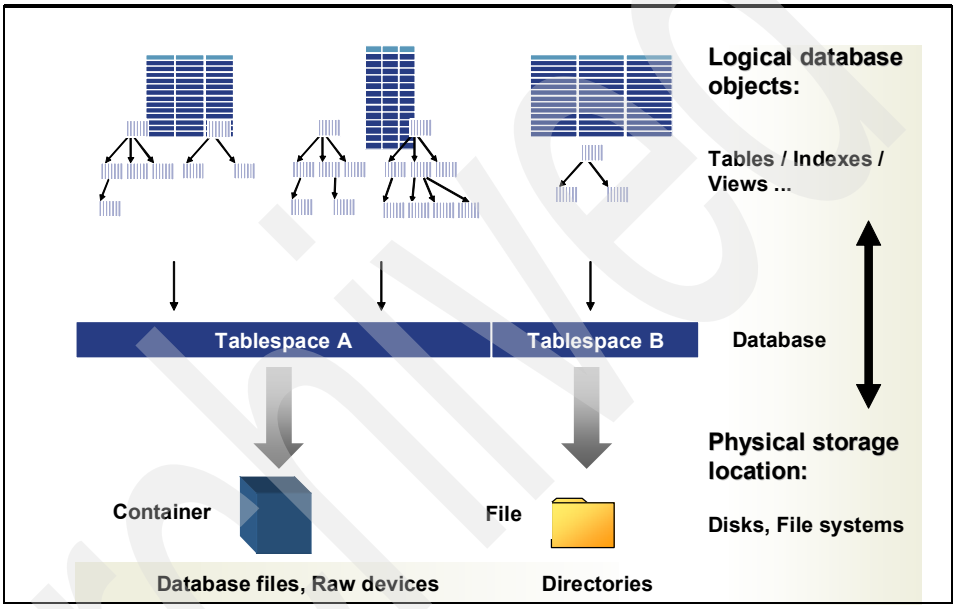


Figure 4-1 Relationship between logical and physical database objects

4.1.1 Tables, indexes, and views

The SAP system data is stored in persistent database tables. If necessary, the DB2 UDB also creates non-persistent system temporary tables holding sorted data being used for further processing.

Indexes on a table are defined on columns which are frequently accessed. This improves the access to data, because the index data structure allows efficient retrieval of small quantities of data. The SAP developers predefine the indexes for DB2 UDB tables in an SAP system, additional indexes can be added later.

A view is like a “virtual table”, which is an efficient way of representing it without needing to maintain it. A view can include all or some of the columns or rows contained in the tables on which it is based.

4.1.2 Table spaces

DB2 UDB uses table space to group the logical objects tables and indexes. All the catalog tables, which hold the database object definitions, are stored in the DB2 UDB data dictionary SYSCATSPACE table space.

DB2 UDB supports two different types of table spaces: System Managed Space (SMS) and Database Managed Space (DMS) table spaces. SAP uses the SMS table space only for the temporary table spaces; all other table spaces are created as DMS with preallocated files as containers.

We give more details about the properties of SMS and DMS table spaces, as well as how to work with them, in 4.1.3, “Table space containers” on page 83, and 4.2, “Working with SMS and DMS table spaces” on page 85.

SAP table space naming convention

As part of the installation of each SAP system, the SAP installation procedure automatically creates DB2 UDB table spaces and containers. SAP groups DB2 UDB tables by their primary usage (for example, master data, transaction data, etc.) into different table spaces. Data and indexes are in separate table spaces.

Figure 4-2 shows the table space naming convention used by SAP NetWeaver 2004s. Within SAP systems prior to R/3 4.7, the prefix of the table space name is not `<SAPSID>#` (where `<SAPSID>` is the name of the SAP system in upper case) but `PSAP`.

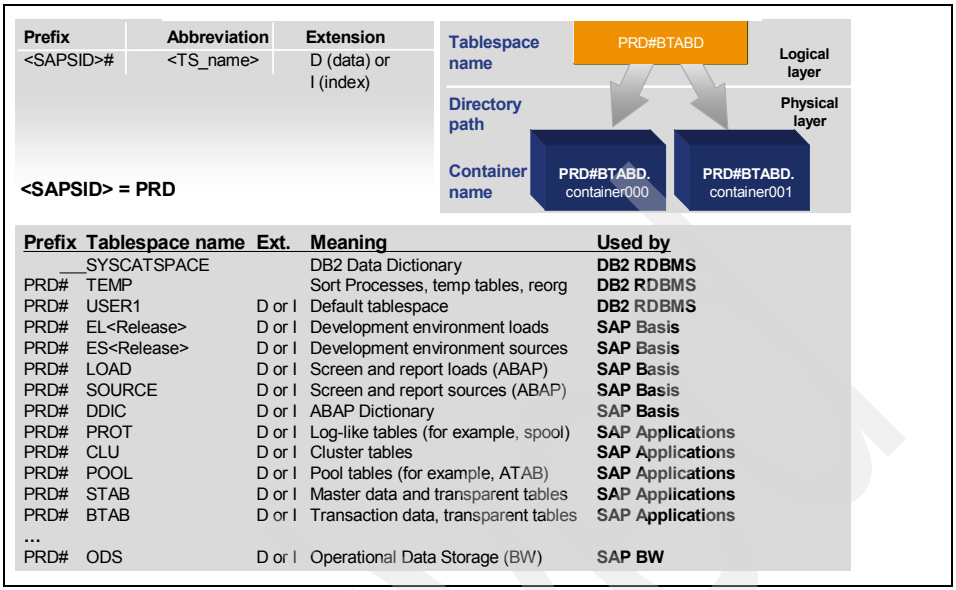


Figure 4-2 SAP table space naming convention

Table 4-1 shows the DB2 UDB table space naming convention followed by SAP systems.

Table 4-1 DB2 UDB table space naming convention in SAP environment

Prefix	Abbreviation	Extension
<SAPSID>#	<TS_name>	D (data table space) I (index table space)

Each table space is composed of three parts:

1. The first letters until the pound sign (#) is the SAP system id (`SAPSID`). `SAPSID` is the name of the SAP system given during the installation.
2. The letters after the pound sign (#) until the one before the last letter are an abbreviation of the table space name. Usually capital letters are used for this name. During SAP installation the names are defined automatically.
3. The last letter indicates whether this table space is used for data or for index.

During installation, you decide if you want to create an automatic storage database or not. If an automatic storage database is created, DB2 UDB is responsible for naming the containers using the convention described in Table 4-1. If the database is not an automatic storage database, SAP creates

containers whose names are the table space name followed by the word ".container" plus a number starting with 000.

SAP Tools recommends that you observe this naming convention, however, it is not mandatory.

Even though it is possible to use a different naming convention, observing the previously described naming convention helps the support organization to understand in a fast and efficient way the storage layout of your SAP / DB2 UDB environment.

Examples

Given an SAPSID of `RED`, typical table spaces names are:

```
RED#STABD    STAB table space used for data information.
RED#STABI    STAB table space used for index information.
```

If database RED is not an automatic storage database, the container's name used for the table space RED#STABD are as follows:

```
RED#STABD.container000
RED#STABD.container001
...
RED#STABD.container<nnn>
```

4.1.3 Table space containers

The physical objects of a DB2 UDB database are the table space containers. Depending on the type of the table space (SMS or DMS), these objects can be directories, containers, raw devices, or Windows partitions.

SMS and DMS table space containers cannot be mixed within one table space. Mixing file and device based table space containers within one table space is allowed, but is not recommended, for the reason of ease of administration.

SMS table space containers

An SMS table space container is a simple directory where DB2 UDB creates several files for each object stored in this table space. The following list provides the most common file suffixes:

- Tables are stored in `*.DAT` files.
- indexes are stored in `*.INX` files.
- LONG columns of tables are stored in `*.LF` files.
- LONG field column data and allocation information are stored in `*.LB` and `*.LBA` files.

These DB2 UDB database files grow and shrink depending on the amount of data within the object the file belongs to. The rapid file size change causes a lot of operating system (OS) overhead. Because of this, SAP uses SMS table space containers only for the temporary table spaces.

DMS table space containers

A DMS table space container can be based on file or device:

- **File based DMS table space containers:**

  A file based DMS table space container is a preallocated file providing storage for the container of a table space. Within the table space, DB2 UDB performs space management of the objects being stored in it.

- **Device based DMS table space containers:**

  The device based DMS table space containers are preallocated devices, which can be raw devices on UNIX based operation systems or partitions on Windows based operation systems. DB2 UDB performs the space management within the table space.

Figure 4-3 illustrates SMS table space containers and file based DMS table space containers.

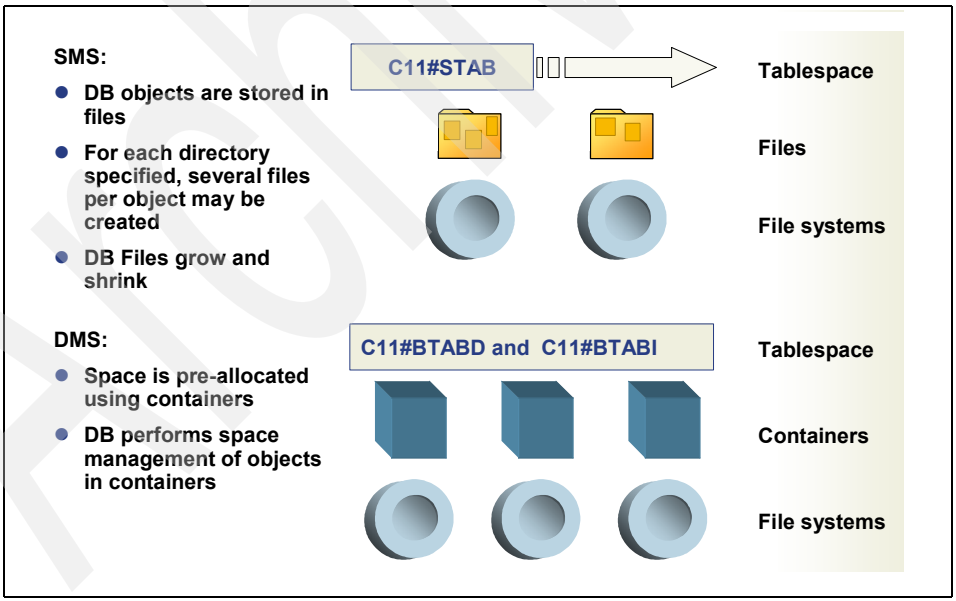


Figure 4-3 SMS and file based DMS table space containers

4.2 Working with SMS and DMS table spaces

This section provides further information about the properties of SMS and DMS table spaces and how to work with them. Execute all commands in this section as user db2<*dbsid*> (<*dbsid*> is the name of the SAP system in lower case).

One of the most important properties of a table space is the *page size*, which can be 4K, 8K, 16K or 32K. If the page size is not specified in the table space creation statement, the default value is provided by the PAGESIZE database configuration parameter specified during table space creation. It is the minimum Input/Output (I/O) unit DB2 UDB handles for each table space.

4.2.1 SMS table spaces

DB2 UDB creates several files for each database object in the containers of an SMS table space. These files grow and shrink depending on the amount of data within the objects. The file size of the database object files in an SMS table space is always the same as the amount of data within the database objects. Because of this, an SMS table space is always 100% full in the **LIST TABLESPACES [SHOW DETAIL]** command output. Monitoring the remaining free space within an SMS table space can only be done at the operating system level by checking the free space of the file systems where the SMS table space container resides.

The **LIST TABLESPACES [SHOW DETAIL]** command lists all table spaces and their properties. Example 4-1 is an excerpt of the output of this command showing the details for the SMS table space PSAPTEMP.

Example 4-1 List table space details

```
> db2 list tablespaces show detail
...
Tablespace ID                        = 3
Name                                 = PSAPTEMP
Type                                 = System managed space
Contents                             = System Temporary data
State                                = 0x0000
  Detailed explanation:
    Normal
Total pages                          = 1
Useable pages                        = 1
Used pages                           = 1
Free pages                           = Not applicable
High water mark (pages)              = Not applicable
Page size (bytes)                    = 4096
Extent size (pages)                  = 32
Prefetch size (pages)                = 32
Number of containers                 = 1
...
```

The **LIST TABLESPACE CONTAINERS FOR** tablespace-id **[SHOW DETAIL]** command shows all containers and their properties of a given table space. The table space ID can be obtained form LIST TABLESPACES command. Using the table space ID from the output of Example 4-1, we retrieve all information about the containers of the table space PSAPTEMP, as shown in Example 4-2.

Example 4-2 List table space container details

```
> db2 list tablespace containers for 3 show detail

          Tablespace Containers for Tablespace 3

Container ID                         = 0
Name                                 =
                     /db2/GR5/saptemp1/NODE0000/temp4/PSAPTEMP.container000
Type                                 = Path
Total pages                          = 1
Useable pages                        = 1

Accessible                           = Yes
```

Maximum table space size

The maximum size of an SMS table space depends on OS settings (for example, maximum file size of user db2<sid>) and the number of containers in the table space, which is specified during the creation of the table space. You cannot add additional containers to an SMS table space once the table space is created. To define additional containers for an SMS table space, a redirected restore of the database is required.

4.2.2 File based DMS table spaces

File based DMS containers are preallocated files providing storage to the table space they belong to.

Internal space allocation strategy

DB2 UDB performs the space management within a file based DMS table space on a extent wise basis. An extent is a group of multiple consecutive pages. The page size and extent size are defined during table space creation using option PAGESIZE and EXTENTSIZE. Each extent belongs to only one database object. When a new extent is allocated, it is always allocated in the next container until all containers of the table space are used. Figure 4-4 illustrates DB2 UDB extent allocation strategy within containers of a table space.

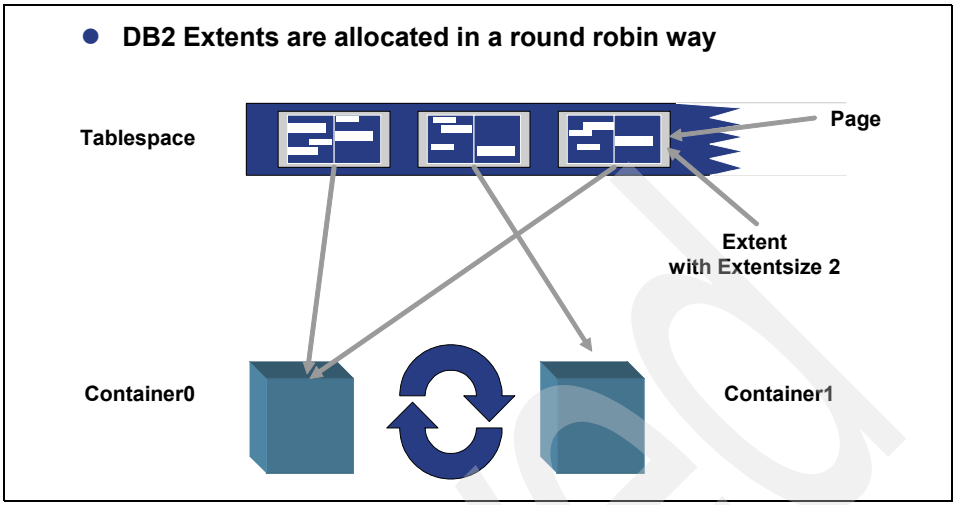


Figure 4-4 Extent allocation within containers of a DMS table space

Figure 4-5 shows the extent types and allocation of the extent types within a DMS table space. The first three extents of each table space are always *header* extent, *object table* extent, and *Space Map Page* (SMP) extent. After that the *Extent Map Page* (EMP) extent and the first data/index extent of the first object in this table space follows.

Each database object has at least one EMP extent and one data/index extent. EMP extents hold a description of the extents being used for that object. SMP is a bitmap with the status of a group of extents. Depending on the page size, the number of extents held in the bitmap varies. Table 4-2 shows the number of extents a bitmap holds based on the page size. SMP extents are allocated in these intervals.

Table 4-2 Size of extent layout bitmap within SMP based on the page size

Page size	Size of extent layout bitmap
4K	1000 extents
8K	2026 extents
16K	4074 extents
32K	8170 extents

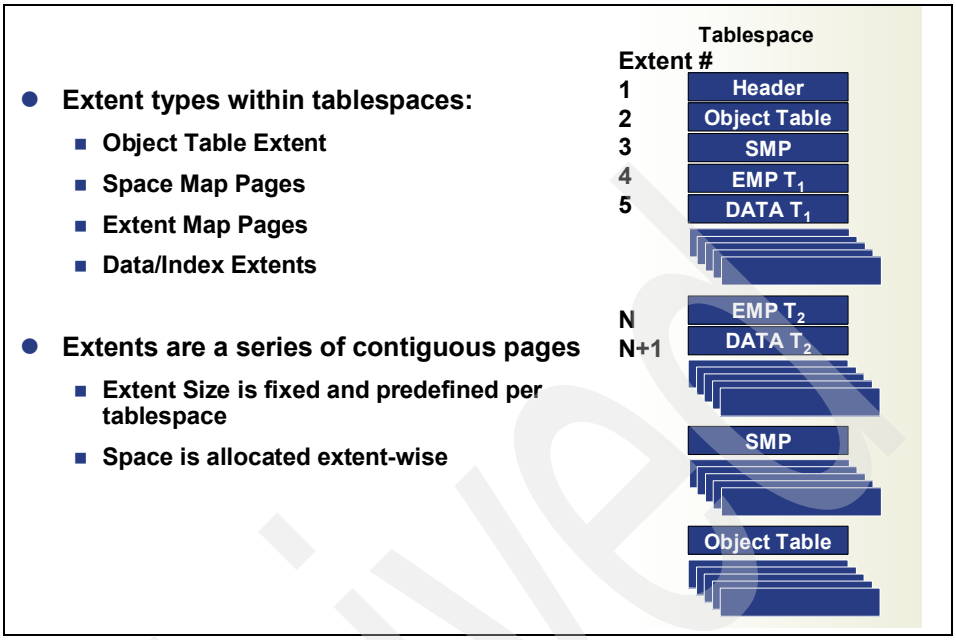


Figure 4-5 Extent allocation within DMS table spaces

Maximum table space size

The maximum table space size on each DB2 UDB database *partition* depends on the page size. SAP uses database partitioning only for Online Analytical Processing (OLAP) systems such as SAP Supply Chain Management (SCM) or Business Warehouse (BW). For Online Transaction Processing (OLTP) systems like R/3 or Enterprise SAP, SAP does not use partitions. So the maximum table space size per partition is the maximum table space size in all.

Table 4-3 shows the maximum table space size by page size per partition.

Table 4-3 Maximum table space size per partition

Page size	Maximum table space size per partition
4K	64GB
8K	128GB
16K	256GB
32K	512GB

Considering only the table space size limitation, the largest page size is the best choice for a table space to prevent it from reaching the table space maximum size when the SAP system grows. To utilize the storage space efficiently, we should also consider these factors:

- Each DB2 UDB page can have a maximum of 255 table records.
- The table's record length (sum of all table field lengths except LONG fields) must be smaller than the page size minus overhead space:
  - A 4K page can store a maximum of 4005 bytes
  - A 8K page can store a maximum of 8101 bytes
  - A 16K page can store a maximum of 16293 bytes
  - A 32K page can store a maximum of 32677 bytes

Storing a table with a small record length in a table space with a large page size wastes space because not all the space within a single page is used. For example, having a table with a record length of 10 bytes in a table space with 8K pagesize wastes 5.5K per page (`8101 bytes usable per page - 10 bytes per record entry * 255 possible entries per page`).

On the other hand, it is also not recommended to store a table with a large record length in a table space with a small page size. In this case no space will be wasted. But if the record length is bigger than the free space within a single page, part of the table will be stored in LONG fields. Because of the unbuffered access to LONG fields, there could be a performance impact.

Considering all three elements, SAP's golden mean is to use a 16K page size within the SAP NetWeaver 2004s installation. In releases prior to SAP NetWeaver 2004s, a 4K page size been used.

We give an example of how to move tables from one table space to another in 4.6, "Moving tables and indexes using DB6CONV". This can also be used to free up space in table spaces being at or near their maximum size, or to improve performance by moving a table into a table space with a larger page size to avoid the use of unbuffered `LONG` fields.

Manual storage management

One of the storage management methods is to create, drop, or resize containers manually.

Use the `ALTER TABLESPACE` command to perform manual storage management and specify the size in the command in pages, K (means KB), M (means MB) or G (means GB):

```
ALTER TABLESPACE tablespacename {ADD | RESIZE | EXTENT | REDUCE | DROP}
      ({ALL CONTAINERS |
      FILE 'full_path_and_containername' [size [{K | M | G}]]
```

```
[, FILE 'full_path_and_containername' [size [{K | M | G}]], ...])
[{ADD | RESIZE | EXTENT | REDUCE | DROP} ({...})]
```

The commands **LIST TABLESPACES [SHOW DETAIL]** and **LIST TABLESPACE CONTAINERS FOR** *tablespace-id* **[SHOW DETAIL]** provide information about the current table space and table space container configuration, as described in 4.2.1, “SMS table spaces”.

Example 4-3 shows the command to increase the size of all the containers within table space GR5#STABI by 1 GB.

Example 4-3 Increase the size of all containers in a table space

```
> db2 "alter tablespace GR5#STABI extend (all containers 1 G)"
DB20000I  The SQL command completed successfully.
```

Example 4-4 shows how to add a new container of 500 MB to table space GR5#STABI.

Example 4-4 Add a new container to a table space

```
> db2 "alter tablespace GR5#STABI
      add (file '/db2/GR5/sapdata2/NODE0000/GR5#STABI.container001' 500 M)"
DB20000I  The SQL command completed successfully.
```

Example 4-5 shows how to reduce the size of a container by 1 GB.

Example 4-5 Reduce the size of a container

```
> db2 "alter tablespace GR5#STABI
      reduce (file '/db2/GR5/sapdata2/NODE0000/GR5#STABI.container000' 1 G)"
DB20000I  The SQL command completed successfully.
```

You can combine several manual storage management operations within one command, but the operations must be either add or remove storage to one table space. Other combinations result in an error

```
SQL1763N There are multiple conflicting container operations in the ALTER
TABLESPACE statement.
```

as shown in Example 4-6.

Example 4-6 Multiple actions in one ALTER TABELSPACE statements

```
> db2 "alter tablespace GR5#STABI
      drop (file '/db2/GR5/sapdata2/NODE0000/GR5#STABI.container001')
      add (file '/db2/GR5/sapdata2/NODE0000/GR5#STABI.container002' 600 M)"
DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
```

```
SQL1763N  There are multiple conflicting container operations in the ALTER
TABLESPACE statement.  SQLSTATE=429BC
```

Extents are allocated in a *round robin* fashion. New extents for each object are allocated in the next container of a table space until all containers of the table space are used, then the first container is revisited. This is called a *stripe set* because the extents are evenly striped across the containers (Figure 4-6).

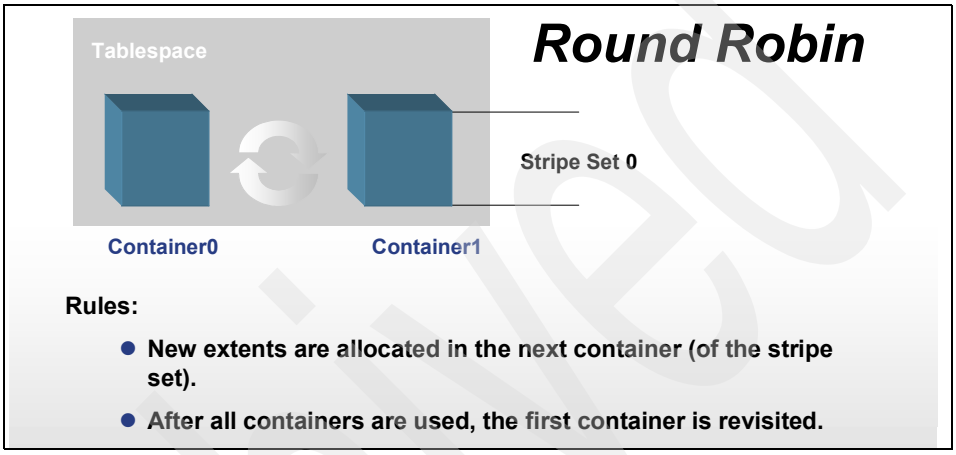


Figure 4-6 Stripe set

Figure 4-7 shows the *rebalancer*, a DB2 UDB process redistributing all used extents of a stripe set after adding one or more containers to an existing stripe set. The rebalancer moves the extents of a stripe set to redistribute them evenly, which is a long running process producing heavy I/O load on the storage subsystem where the table space resides.

Dropping containers forces an *inverse rebalancer* to run, which moves all used extents out of the containers to be dropped before they will be deleted by DB2 UDB. After a `RESIZE` or `REDUCE` operation, if the size of the container becomes too small to hold all used extents, the inverse rebalancer also moves extents out of a container.

Before starting the inverse rebalancer, DB2 UDB checks if there is enough free space in the other containers to take up the extents from the dropped, resized or reduced container. If there is not enough free space, DB2 UDB displays the error message:

```
SQL20170N There is not enough space in the table space "tablespacename" for
the specified action.
```

LIST UTILITIES [SHOW DETAIL] displays the progress of the rebalancer:

```
> db2 "list utilities show detail"
ID                                   = 2
Type                                 = REBALANCE
Database Name                        = GR5
Partition Number                     = 0
Description                          = Tablespace ID: 6
Start Time                           = 05/18/2005 11:48:04.754158
Throttling:
   Priority                          = Unthrottled
Progress Monitoring:
   Estimated Percentage Complete = 2
      Total Work                     = 49634 extents
      Completed Work                 = 986 extents
      Start Time                     = 05/18/2005 11:48:04.754350
```

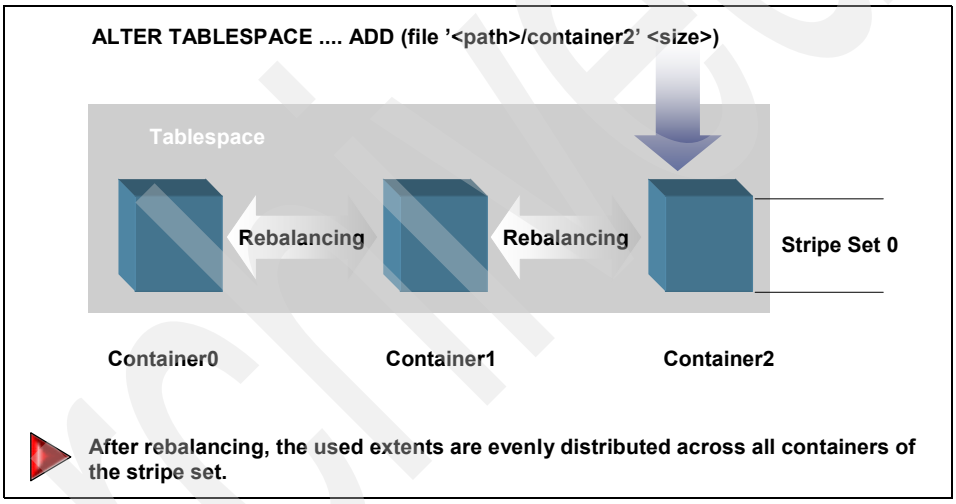


Figure 4-7 Rebalancer

Because of the heavy I/O load of the rebalancer, we recommend to avoid invoking the rebalancer by extending or resizing all containers of a stripe set equally. Figure 4-8 shows this scenario, where it is not necessary for DB2 UDB to rebalance the containers because the used extents are still evenly striped after adding the new storage.

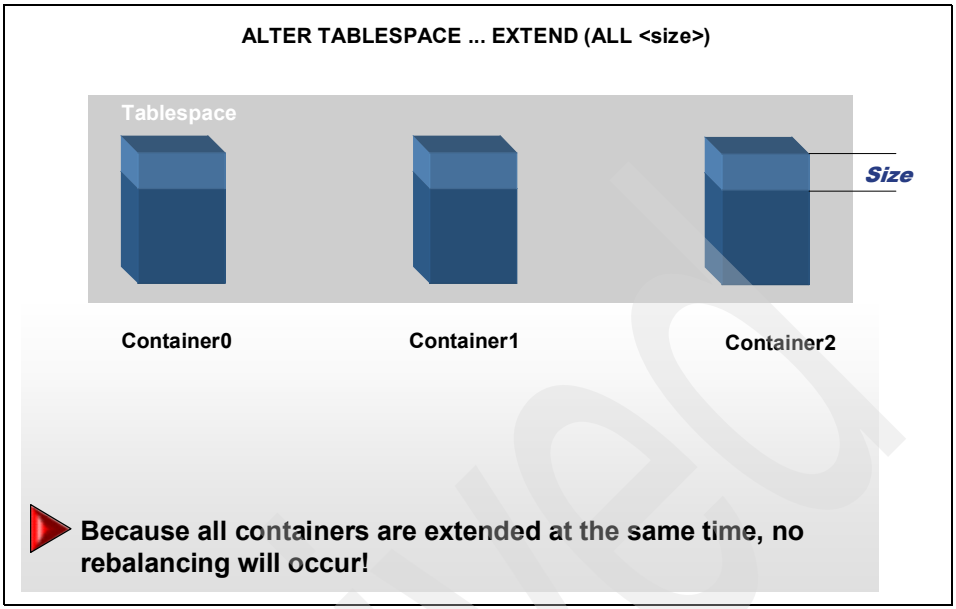


Figure 4-8 Adding storage without rebalancer

Figure 4-7 and Figure 4-8 show table space containers that are equal in size, where rebalancing will be done within the whole stripe set. With different container sizes, DB2 UDB implicitly defines ranges as shown in Figure 4-9. Initially extents are allocated and striped across both containers, but at a certain point `Container0` will be full. From this point on, new extents are allocated only in the `Container1`, which defines the beginning of a new range (`Range1`). Resizing `Container0` to the same size as `Container1` causes the rebalancer to start only within `Range1` because the extents in `Range0` are already balanced.

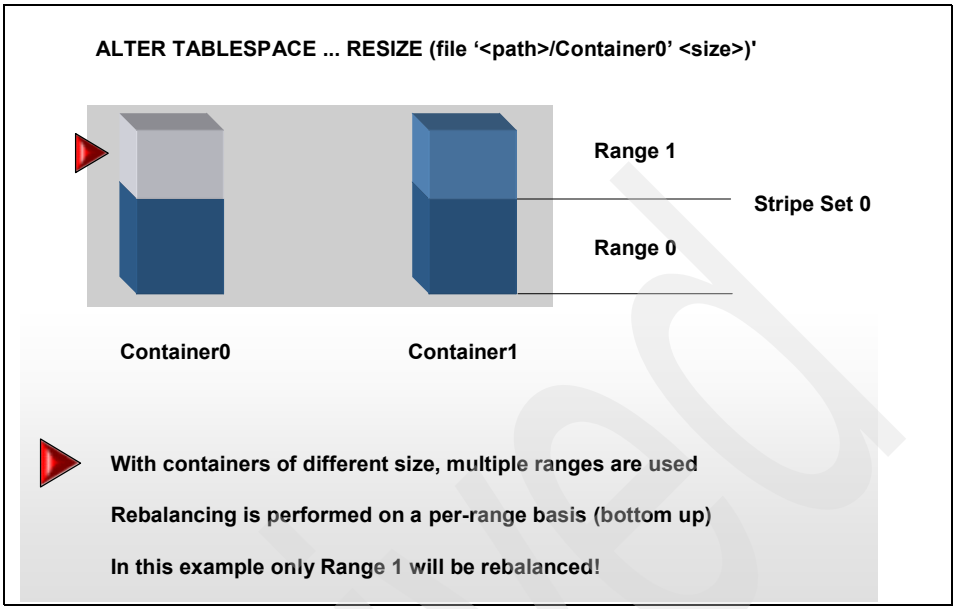


Figure 4-9 Tablespace ranges

Defining a new stripe set when adding storage to a table space also avoids rebalancing. The following command adds storage to a table space by defining a new stripe set containing one or more containers:

```
ALTER TABLESPACE tablespacename BEGIN NEW STRIPE SET
      (FILE 'full_path_and_containername' size [{K | M | G}]
      [, FILE 'full_path_and_containername' size [{K | M | G}], ...])
```

Figure 4-10 shows a table space with two containers within one stripe set and a third container (`Container2`) being added to the table space starting a new stripe set. This implicitly causes a new range to be defined. It also shows a simplified *table space map*. **`GET SNAPSHOT FOR TABLESPACES ON sapsid`** shows the table space maps as well as other snapshot information.

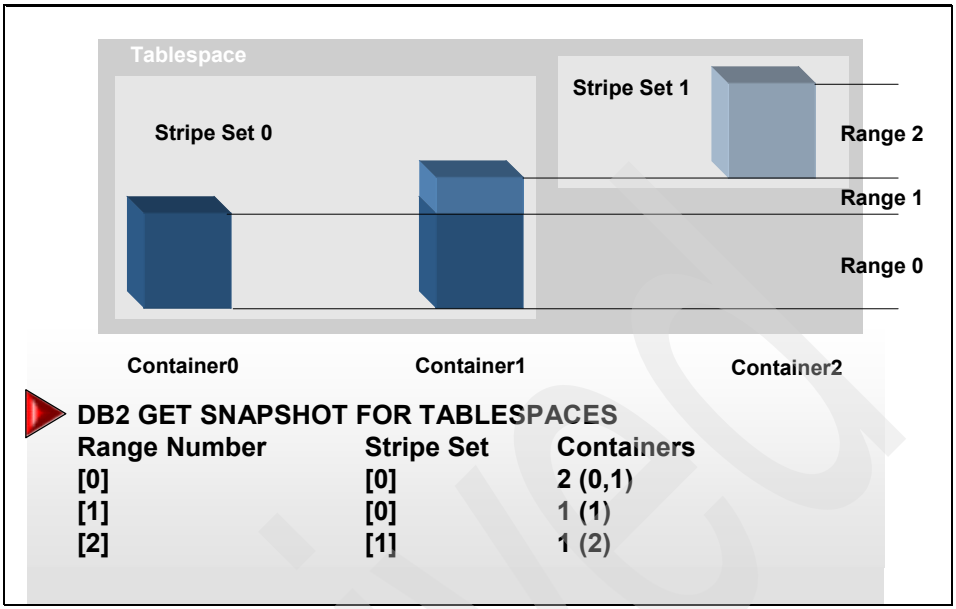


Figure 4-10 Explicit usage of stripe sets

Example 4-7 is the table space snapshot output from which the simplified table space map in Figure 4-10 is derived.

Example 4-7 Get table space map from snapshot

```
> db2 get snapshot for tablespaces on GR5
...
Tablespace name                            = GR5#STABI
...
  Number of containers                     = 3

  Container Name                           =
                        /db2/GR5/sapdata2/NODE0000/GR5#STABI.container000
      Container ID                         = 0
      Container Type                       = File (extent sized tag)
      Total Pages in Container             = 417280
      Usable Pages in Container            = 417272
      Stripe Set                           = 0
      Container is accessible              = Yes

  Container Name                           =
                        /db2/GR5/sapdata2/NODE0000/GR5#STABI.container001
      Container ID                         = 1
      Container Type                       = File (extent sized tag)
      Total Pages in Container             = 417380
```

```
    Usable Pages in Container            = 417368
    Stripe Set                           = 0
    Container is accessible              = Yes

  Container Name                         =
                        /db2/GR5/sapdata2/NODE0000/GR5#STABI.container002
    Container ID                         = 2
    Container Type                       = File (extent sized tag)
    Total Pages in Container             = 100
    Usable Pages in Container            = 88
    Stripe Set                           = 1
    Container is accessible              = Yes

  Table space map:

   Range  Stripe Stripe  Max         Max  Start  End    Adj.   Containers
   Number Set    Offset  Extent      Page Stripe Stripe
   [   0] [   0]      0  104317    834543      0  52158   0    2 (0,1)
   [   1] [   0]      0  104329    834639  52159  52170   0    1 (1)
   [   2] [   1]  52171  104340    834727  52171  52181   0    1 (2)
...
```

The table space map is easy to read. The table space consists of three containers (0, 1, 2) forming three ranges (0, 1, 2; see first column of the tablespace map). Range 0 and 1 are within stripe set 0 (see second column of tablespace map). Range 0 consists of 2 containers (0, 1; see last column), range 1 and 2 each consist of 1 container (1 and 2; see also last column).

Adding additional containers via **ALTER TABLESPACE tablespacename ADD (...)** to table spaces with more than one container always adds the new containers to the endmost stripe set. The following command supports adding additional containers to a specific stripe set:

```
ALTER TABLESPACE tablespacename ADD TO STRIPE SET stripeset
      (FILE 'full_path_and_containername' size [{K | M | G}]
      [, FILE 'full_path_and_containername' size [{K | M | G}], ...])
```

Tip: To avoid invoking the rebalancer in the manual storage management process: First, add new containers using **ALTER TABLESPACE** *tablespacename* **BEGIN NEW STRIPE SET (...)**; then enlarge these containers by **ALTER TABLESPACE** *tablespacename* **EXTEND (...)**.

Auto-resize DMS table spaces

You can switch a DMS table space from non-automatic resizing to an automatic resizing table space and let DB2 UDB automatically grow the DMS table spaces for you when needed.

The auto-resize feature for DMS table spaces is switched on and off at table space level during table space creation or via `ALTER TABLESPACE` statement after table space is created. The `ALTER TABLESPACE` statement also can be used to specify the maximum size or increase size of the table space:

```
ALTER TABLESPACE tablespacename [AUTORESIZE {YES | NO}]
      [MAXSIZE {NONE | maxsize {K | M | G}}]
      [INCREASESIZE {increase_percent PERCENT | increasesize {K | M | G}}]
```

When a table space is full, DB2 UDB always increases all containers from the last range of the table space map. DB2 UDB chooses an appropriate increasing value each time an auto-resize takes place, if `INCREASESIZE` was not specified within the command. With the maximum table space size being specified, the table space will be increased up to this size. Otherwise DB2 UDB increases it up to the maximum possible size of the table space depending on the page size of the table space.

If DB2 UDB cannot increase the containers of the last range because there is no free space in the file systems, an `SQL0289N (Unable to allocate new pages in table space "tablespacename")` error occurs. Adding more space to the file systems that are full solves that error condition. If this is not possible, modify the table space map by using container operations like **`ALTER TABLESPACE tablespacename BEGIN NEW STRIPE SET (...)`** so that the containers that could not be enlarged are no longer in the last range of the table space map.

The **`GET SNAPSHOT FOR TABLESPACES ON sapsid`** shows beside other snapshot information whether auto-resize has been enabled for each table space and which settings for `INCREASESIZE` and `MAXSIZE` are current for each table space:

```
> db2 get snapshot for tablespaces on dap
...
Tablespace name                            = SYSCATSPACE
  Tablespace ID                            = 0
  Tablespace Type                          = Database managed space
  Tablespace Content Type                  = Any data
...
  Auto-resize enabled                      = Yes
...
  Initial tablespace size (bytes)          = 33554432
  Current tablespace size (bytes)          = 234881024
  Maximum tablespace size (bytes)          = NONE
  Increase size (bytes)                    = AUTOMATIC
  Time of last successful resize           = 05/19/2005 13:04:12.362275
...
```

Automatic storage feature

Automatic storage feature can only be enabled during database creation. It enables DB2 UDB to create and resize containers automatically. Please note that automatic storage enabled databases cannot be created in multi-partition environments.

The SAP installation routine creates the automatic storage enabled DB2 UDB database using the `CREATE DATABASE` command as shown in Example 4-8. In this example the database name is `GR5`, the database path will be set to `/db2/GR5` and the storage path will be `/db2/GR5/sapdata1`:

Example 4-8 Automatic storage feature - CREATE DATABASE

```
CREATE DATABASE GR5 AUTOMATIC STORAGE YES
      ON /db2/GR5/sapdata1 DBPATH ON /db2/GR5
      USING CODESET ISO8859-1 TERRITORY en_US COLLATE USING IDENTITY
      PAGESIZE 16 K DFT_EXTENT_SZ 2
      CATALOG TABLESPACE MANAGED BY AUTOMATIC STORAGE
      WITH 'SAP database GR5'
```

The SAP installation process creates its own user and temporary table spaces after database creation and drops the default temporary and user table space created with the database afterwards.

To create automatic storage managed table spaces in an automatic storage enabled database, use the following statement:

```
CREATE {REGULAR | SYSTEM TEMPORARY} TABLESPACE tablespacename
      MANAGED BY AUTOMATIC STORAGE
      [INITIALSIZE initialsize {K | M | G}]
      [INCREASESIZE {increase_percent PERCENT | increasesize {K | M | G}}]
      [MAXSIZE maxsize {K | M | G}]
```

For table spaces managed by automatic storage, DB2 UDB creates new containers in the given storage paths. If necessary, additional properties such as `INITIALSIZE`, `INCREASESIZE`, and `MAXSIZE` can be specified.

DB2 UDB creates temporary table spaces as SMS table spaces and all other table spaces as DMS table spaces when the `MANAGED BY AUTOMATIC STORAGE` clause is specified.

If a table space managed by automatic storage becomes full, DB2 UDB resizes the existing containers or creates new containers. To avoid invoking the rebalancer, DB2 UDB creates new containers using the **BEGIN NEW STRIPE SET** clause.

Creating non-automatic storage table spaces (using options `MANAGED BY SYSTEM` or `MANAGED BY DATABASE`) and specifying their containers manually within an automatic storage enabled database can be done but is not recommended.

The **`ALTER DATABASE ADD STORAGE ON storagepath1 [, storagepath2, ...]`** statement adds one or more storage paths to the automatic storage enabled database so that DB2 UDB can use those storage paths for creating new containers if necessary.

The **`GET SNAPSHOT FOR DATABASE ON`** ***`sapsid`*** shows among other snapshot information whether automatic storage has been enabled for a database and the available automatic storage paths. If the number of automatic storage paths equals 0, automatic storage has not been enabled, otherwise the number of defined storage paths and the storage paths are displayed:

```
> db2 get snapshot for database on dap
...
Database name                                = DAP
Database path                                = /db2/DAP/db2dap/NODE0000/SQL00001/
Input database alias                         = DAP
Database status                              = Active
...
Number of automatic storage paths            = 1
Automatic storage path                       = /db2/DAP/sapdata1
...
```

Improving performance for file based DMS table spaces

Because of caching and locking mechanisms, there is an overhead during the access to data stored in file based DMS containers. Especially large file based DMS containers on UNIX file systems could lead to an i-node contention. As mentioned in SAP Note *147634*, file based DMS containers should not be bigger than 10-20 GB and of same size.

Starting with DB2 UDB V8.2, DB2 UDB now supports Direct I/O (DIO) and Concurrent I/O (CIO) on AIX and DIO on HP, Solaris, Linux and Windows. The following article gives more background information about DIO/CIO and also what to care about when using it (for example, necessary Authorized Program Analysis Reports (APARs) on AIX):

`http://www3.software.ibm.com/ibmdl/pub/software/dw/dm/db2/dm-0408lee/CIO-article.pdf`

As mentioned in the article, DIO/CIO can be enabled as a file system mount option or on table space level. We recommend using the enabling at table space level via **`ALTER TABLESPACE`** *`tablespacename`* **`NO FILE SYSTEM CACHING`** because DB2 UDB then always tries to use CIO. If CIO is not supported, DIO will be used.

To check whether DIO/CIO has been enabled at table space level, the **GET SNAPSHOT FOR TABLESPACES ON** *sapsid* should be used. The following example shows that for the SYSCATSPACE file system caching is switched on, that means DIO/CIO has not been enabled:

```
> db2 get snapshot for tablespaces on GR5
...
Tablespace name                            = SYSCATSPACE
  Tablespace ID                            = 0
  Tablespace Type                          = Database managed space
  Tablespace Content Type                  = Any data
...
  File system caching                      = Yes
  Tablespace State                         = 0x'00000000'
   Detailed explanation:
     Normal
...
```

4.2.3 Device based DMS table spaces

The device based DMS table space, also known as a raw container table space, is another type of DMS table space. It does not use preallocated files within file systems, but uses raw devices on UNIX based operation systems or partitions on Windows based operation systems as containers. In general the access to data within raw devices is faster in comparison to normal file based DMS containers because there are no overheads such as file system caching and locking mechanisms.

We recommend using raw devices only for Very Large Databases (VLDBs). You can modify the create database script of the SAP installation to create device based DMS table spaces for a new database. For existing databases, a redirected restore can be used to switch from file based DMS table spaces to device based DMS table spaces.

Please note that the DB2 UDB V8.2.2 auto-resize feature for DMS table spaces cannot be used when raw device containers are used. Also, you cannot specify raw device containers for an auto-resize enabled DMS table space during a redirected restore.

4.3 SAP data classes (TABARTs)

DB2 UDB keeps all the object information in catalog tables, including the relationship between tables and the assigned table spaces they are stored in. SAP DDIC does not have this direct relationship between tables and table spaces. The SAP DDIC assigns each table to a data class, called TABART.

Each of these TABARTs are assigned to one data table space and to one index table space. The data table space stores all tables of these TABARTs, whereas the index table space stores all indexes belonging to these tables. SAP delivers the SAP system with a list of default TABARTs that will also be used during the installation where each of the TABARTs will be loaded to the database within a single process.

In 4.6, “Moving tables and indexes using DB6CONV”, we show how to change the relationship table - TABART - table space within an SAP system.

The TABARTs can be listed via transaction SE16 (Data Browser), where you have to:

- Enter DDART for the table name and press F7 or click the **Table Contents** button.
- On the next screen, press F8 or click the **Execute** button.

The Data Browser now lists the content of the table DDART, which are the TABARTs, as shown in Figure 4-11.

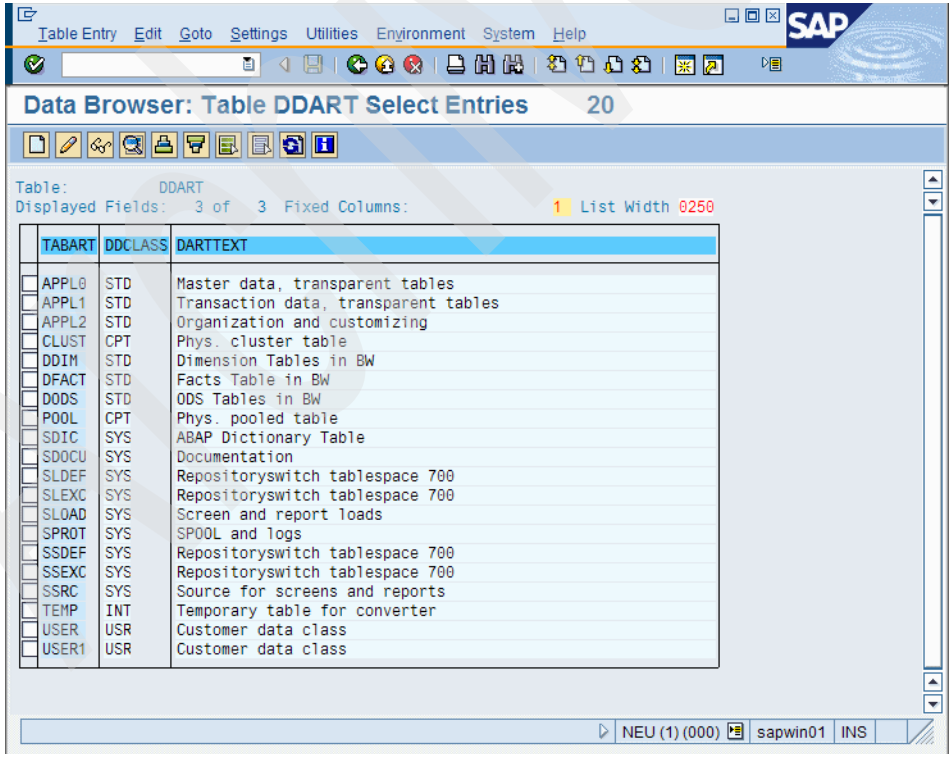


Figure 4-11 TABARTs of SAP NetWeaver 2004s

Section 4.6, “Moving tables and indexes using DB6CONV”, gives more details, in an example, about TABARTs and how to modify them when a table is moved from one table space to another.

4.4 Space reclamation strategies

After the deletion of mass data from an SAP system (for example, client delete or archival run), normally the extents of a table/index are not 100 percent empty, and the free space within the database will not increase. The free space within the extent can be used for new entries of the table/index the extent belongs to, but not for other tables/indexes. The table space sizes also cannot be reduced by the `ALTER TABLESPACE` command even though there are free spaces in the extents.

To reuse the free space within the extents for other tables/indexes or to decrease table space sizes to free up space in the sapdata file systems, reorganizing the tables/indexes is required. Reorganization not only can reclaim space but also can improve performance. A reorganization ensures that all table/index extents can be read in a sequential order and fewer extents have to be accessed when reading the data.

SAPs Structured Query Language (SQL) statements are optimized for a table access using the primary index of a table. This must be kept in mind when performing the reorganization on the command line. Specifying the primary index as order sequence on data reorganization command ensures that the table has the physical ordering based on primary key as SAP expects. By using transaction `DB13` (DBA Planning Calendar) for reorganization, SAP ensures this by calling the **`REORG`** command with the correct parameters.

DB2 UDBs reorganization is a robust operation. If something unexpected (crash of the server, not enough freespace) happens during the reorganization, DB2 UDB reconstructs the tables/indexes to the state that they can normally be used.

4.4.1 Offline table and index reorganization

Offline reorganization means that the table/index being reorganized is locked during the reorganization. This type of reorganization also rebuilds all indexes of the table being reorganized.

Offline reorganization is recommended in the SAP/DB2 UDB environment, because it is faster than the inplace reorganization. It also needs only one step to reorganize the table and rebuild all indexes belonging to the table.

Reorganization using a temporary table space

In this case DB2 UDB copies the content of the table to the specified temporary table space, before it will be copied back to the original table space. The reorganization itself will be done during the copy process. The specified temporary table space must be of the same page size as the original table space where the table is stored. The space consumption within the temporary table space could be two or three times the space of the table being reorganized, but this depends on index definitions and sizes of the tables. Figure 4-12 shows the reorganization using a temporary table space.

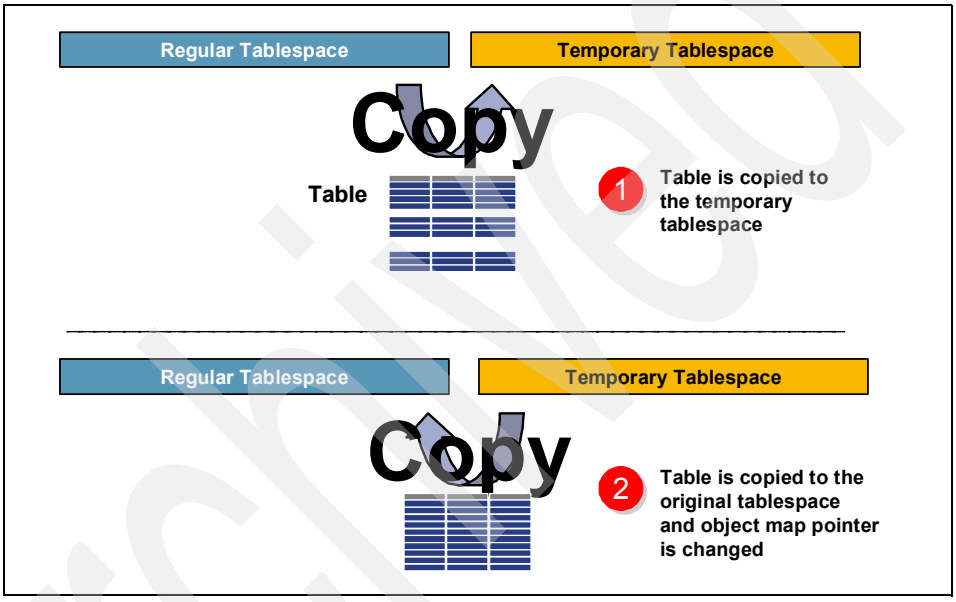


Figure 4-12 Reorganization using a temporary table space

This type of reorganization can be started within SAP by calling transaction DB13 (DBA Planning Calendar) and selecting one of the reorganizations (single table, flagged table or all tables of a table space) from the action pad. You also can select an optional temporary table space for reorganization. On the command line it can be performed using the following command:

```
REORG TABLE schemaname.tablename INDEX schemaname.primary_index
      USE temporary_tablespace
```

This command reorganizes the table *schemaname.tablename* using the *temporary_tablespace* and physically orders the entries of the table *schemaname.tablename* according to its index *schemaname.primary_index*. Besides the table reorganization, all indexes of the table *schemaname.tablename* will also be rebuilt.

Reorganization without using a temporary table space

Offline reorganization without using a temporary table space is in principle the same as those described in the preceding section where the temporary table space will be used. Here the table will not be copied to a temporary table space but copied within the original table space the table is stored. DB2 UDB copies the table only once in this case. It copies the table extents to free extents using the lowest free extents first. Sometimes this type of reorganization can be used to lower the high water mark of the table space, as discussed in 4.4.4, “High water mark”.

This type of reorganization can also be started within SAP by calling transaction `DB13` (DBA Planning Calendar) and selecting one of the reorganizations (single table, flagged table, or all tables of a table space) from the action pad. On the command line it can be performed using the following command:

```
REORG TABLE schemaname.tablename INDEX schemaname.primary_index
```

This command reorganizes the table *schemaname.tablename* and physically orders the entries of the table *schemaname.tablename* according to its index *schemaname.primary_index* within the table space where the table *schemaname.tablename* is stored. Besides the table reorganization, all indexes of the table *schemaname.tablename* will also be rebuilt.

4.4.2 Inplace table and index reorganization

Inplace reorganization means that the table and its indexes being reorganized are fully accessible during the reorganization. Using the inplace reorganization sometimes needs more than one step to achieve the same result as with the offline reorganization.

Because of this and also because of the DB2 UDB overhead to have the table accessible during the reorganization, in SAP/DB2 environment, we recommend using the inplace reorganization only on high availability systems, where tables down time is hard to schedule. In all other cases we recommend using the offline reorganization.

Reclustering the table

With this type of inplace reorganization, DB2 UDB sorts the table data during the reorganization based on the given index. Figure 4-13 shows the two phases of this inplace reorganization, also known as inplace reorganization - reclustering.

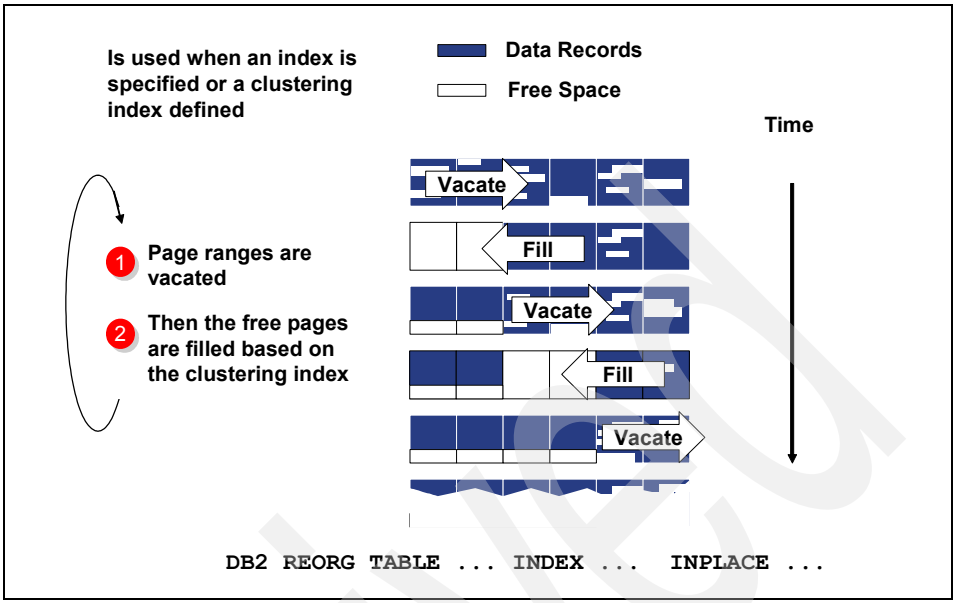


Figure 4-13 inplace reorganization - reclustering

Within the first phase, the *vacate* phase, DB2 UDB moves data out of a page range and inserts the data into other pages. During the *fill* phase DB2 UDB moves a certain amount of data (based on the index specified) back to the previously vacated range. The indexes of the table being reclustered will not be rebuilt.

This kind of inplace reorganization will also be performed when starting an inplace reorganization from the transaction DB13 (DBA Planning Calendar).

The following command reclusters the table *schemaname.tablename* based on the *schemaname.primary_index*. Any free space at the end of the table *schemaname.tablename* will not be truncated after the reclustering, because this will force a short lock on the table:

```
REORG TABLE schemaname.tablename
      INDEX schemaname.primary_index
      INPLACE NOTRUNCATE TABLE
```

It is also possible to check the state and progress of inplace reorganizations with the **GET SNAPSHOT FOR TABLES ON** *sapsid*. Furthermore, an inplace reorganization can be paused, resumed, and stopped, as shown in Example 4-9, by using the command:

```
REORG TABLE schemaname.tablename INPLACE {PAUSE | RESUME | STOP}
```

Example 4-9 inplace reorganization

```
> db2 "reorg table sapdap.reposrc
      index sapdap.reposrc~0 inplace notruncate table"
DB20000I  The REORG command completed successfully.
DB21024I  This command is asynchronous and may not be effective immediately.

> db2 get snapshot for tables on dap
...
 Table Name          = REPOSRC
...
 Table Reorg Information:
   Reorg Type        =
        Reclustering
        Inplace Table Reorg
        Allow Write Access
        No Table Truncation
   Reorg Index       = 1
   Reorg Tablespace  = 24
   Start Time        = 05/23/2005 03:01:59.565940
   Reorg Phase       =
   Max Phase         =
   Phase Start Time  =
   Status            = Started
   Current Counter   = 1443
   Max Counter       = 10309
   Completion        = 0
   End Time          =
...

> db2 "reorg table sapdap.reposrc inplace pause"
DB20000I  The REORG command completed successfully.
DB21024I  This command is asynchronous and may not be effective immediately.

> db2 get snapshot for tables on dap
...
 Table Name          = REPOSRC
...
 Table Reorg Information:
   Reorg Type        =
        Reclustering
        Inplace Table Reorg
        Allow Write Access
        No Table Truncation
   Reorg Index       = 1
   Reorg Tablespace  = 24
   Start Time        = 05/23/2005 03:01:59.565940
   Reorg Phase       =
   Max Phase         =
```

```
   Phase Start Time  =
   Status            = Paused
   Current Counter   = 4511
   Max Counter       = 10342
   Completion        = 0
   End Time          = 05/23/2005 03:03:04.249799
...

> db2 "reorg table sapdap.reposrc inplace resume"
DB20000I  The REORG command completed successfully.
DB21024I  This command is asynchronous and may not be effective immediately.

> db2 get snapshot for tables on dap
...
 Table Name          = REPOSRC
...
 Table Reorg Information:
   Reorg Type        =
        Reclustering
        Inplace Table Reorg
        Allow Write Access
   Reorg Index       = 1
   Reorg Tablespace  = 24
   Start Time        = 05/23/2005 03:48:12.199240
   Reorg Phase       =
   Max Phase         =
   Phase Start Time  =
   Status            = Completed
   Current Counter   = 10310
   Max Counter       = 10310
   Completion        = 0
   End Time          = 05/23/2005 03:49:02.979365
...

> db2 "reorg table sapdap.reposrc
      index sapdap.reposrc~0 inplace notruncate table"
DB20000I  The REORG command completed successfully.
DB21024I  This command is asynchronous and may not be effective immediately.

> db2 "reorg table sapdap.reposrc inplace stop"
DB20000I  The REORG command completed successfully.
DB21024I  This command is asynchronous and may not be effective immediately.

> db2 get snapshot for tables on dap
...
 Table Name          = REPOSRC
...
 Table Reorg Information:
   Reorg Type        =
```

```
        Reclustering
        Inplace Table Reorg
        Allow Write Access
        No Table Truncation
   Reorg Index       = 1
   Reorg Tablespace  = 24
   Start Time        = 05/23/2005 04:02:11.443299
   Reorg Phase       =
   Max Phase         =
   Phase Start Time  =
   Status            = Stopped
   Current Counter   = 334
   Max Counter       = 10342
   Completion        = 0
   End Time          = 05/23/2005 04:02:14.891573
...
```

Reclamation of freespace

Figure 4-14 shows the type of inplace reorganization that should be used to reclaim free space within the table. DB2 UDB performs a reverse scan of the table. During this it eliminates overflow records and frees the high pages of the table. This kind of inplace reorganization cannot be started out of transaction DB13 (DBA Planning Calendar), but only from the command line. DB2 UDB normally truncates the table after the reclamation of freespace, this will force a short lock on the table. If this is not acceptable, please specify **NOTRUNCATE TABLE**:

```
REORG TABLE schemaname.tablename INPLACE {NOTRUNCATE TABLE}
```

It is also possible to check the state and progress of this kind of inplace reorganization. It can also be paused, resumed, and stopped as described in the preceding section about reclustering tables.

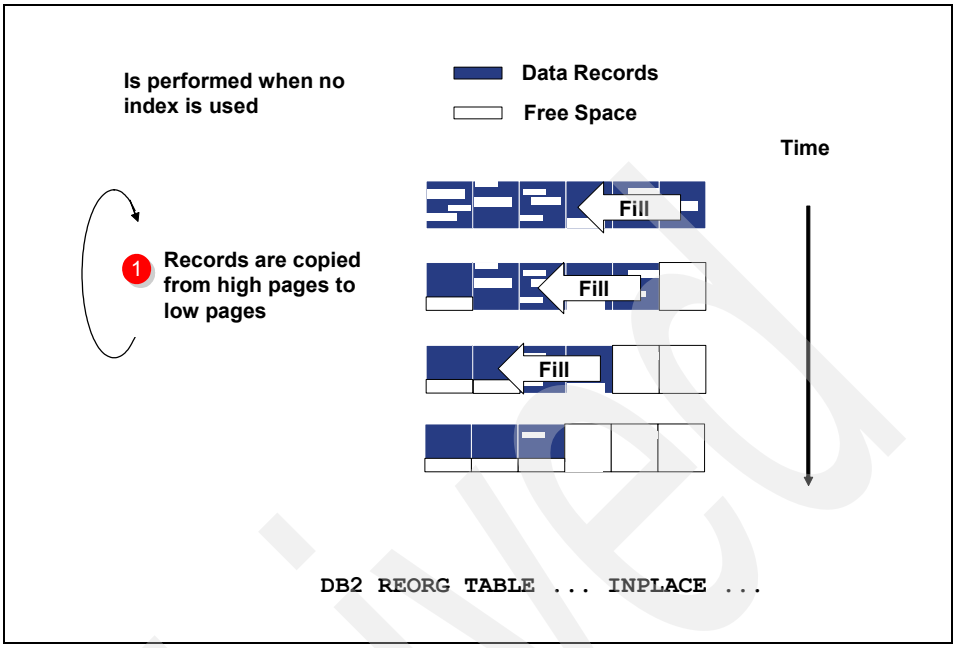


Figure 4-14 Inplace reorganization - freespace reclamation

Reorganizing the indexes of a table

In a high availability SAP system it is not only necessary to recluster the tables and to reclaim the freespace within the tables, from time to time, but also necessary to rebuild the indexes of a table. Because the inplace reorganization of a table does not handle this (as the offline reorganization does), the indexes must be reorganized by the following command if necessary:

```
REORG INDEXES ALL FOR TABLE schemaname.tablename ALLOW WRITE ACCESS
```

This command reorganizes all indexes of the table *schemaname.tablename* while allowing full access to the table and its indexes.

If there is no need to rebuild the indexes, but to clean them up, the following command can also be used:

```
REORG INDEXES ALL FOR TABLE schemaname.tablename ALLOW WRITE ACCESS
     CLEANUP ONLY [{PAGES | ALL}]
```

Certain items will be cleaned up when using each of the following commands:

- **CLEANUP ONLY**
  The indexes of the table *schemaname.tablename* will not be rebuilt, but any freed up pages will be made available for reuse by indexes defined on this table only.
- **CLEANUP ONLY PAGES**
  Committed pseudo empty pages will be searched and freed up. A committed pseudo empty page is one where all the keys on the page are marked as deleted and all these deletions are known to be committed.
- **CLEANUP ONLY ALL**
  Committed pseudo empty pages will be freed up, as well as committed pseudo deleted keys from pages that are not pseudo empty will be removed. DB2 UDB also tries to merge adjacent leaf pages, which will result in a merged leaf page.

4.4.3 Criteria for reorganization

The **REORGCHK** command, which should be normally scheduled on a regular base within transaction DB13 (DBA Planning Calendar), determines the eight reorganization criteria *F1-F8*. In releases prior to SAP NetWeaver 2004s, the DBA Planning Calendar jobs Run_DBSTATC and Runstat_All call the **REORGCHK** command.

These reorganization criteria can be viewed via transaction DB02 (Space: History - Overview). Within this transaction, you have to:

- Select **Single Table Analysis** on the left side.
- Enter the name of table in the field on the top of the right side and press **Enter**.

To view the reorganization criteria via transaction DB02:

- Select **Tables and indexes** on the left side.
- Select **Continue** in the *REORGCHK Table Check Result* window.
- Enter the Selection Criteria for the tables that should be displayed, for example, REPO* as table name and press **Enter**.
- Double-click the table REPOSRC in the list to show the table and index details of this table.

Figure 4-15 shows the details of the table REPOSRC.

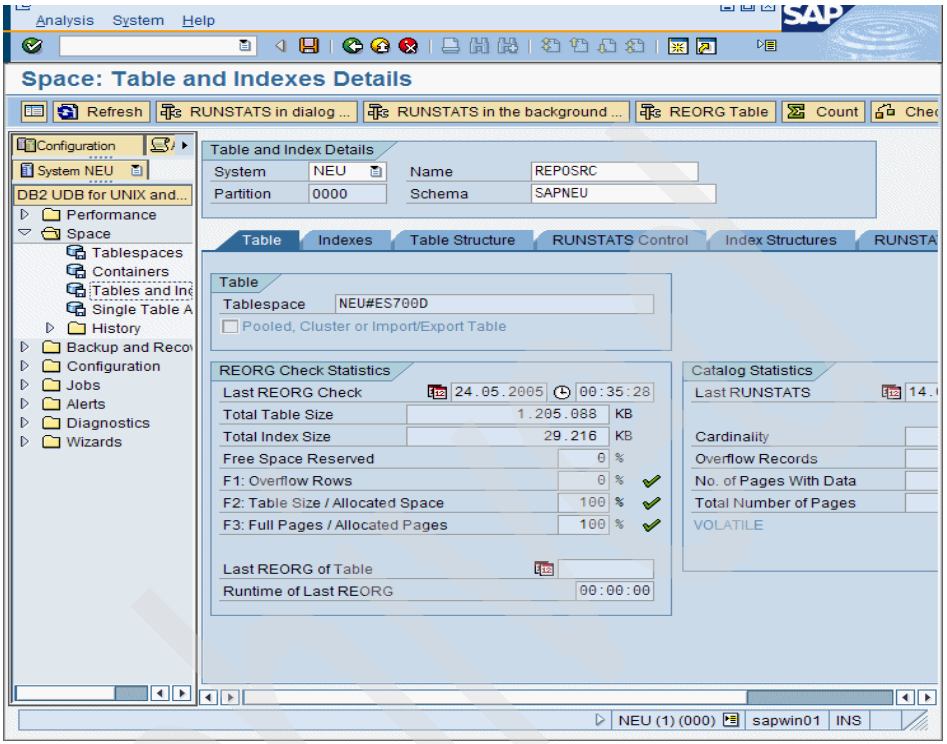


Figure 4-15 Reorganization criteria of a table

To show the details of the indexes of table REPOSRC, click the register **Indexes**. Figure 4-16 shows the details of the index REOPSRC~0.

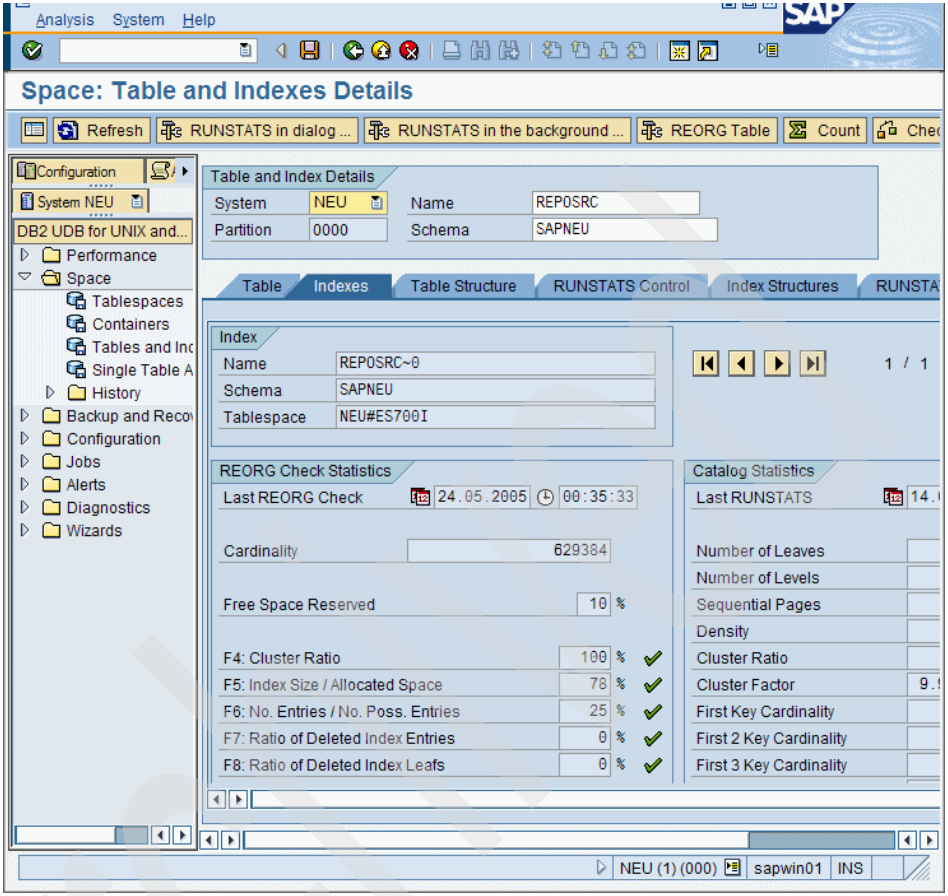


Figure 4-16 Reorganization criteria of an index

The reorganization criteria F1-F3 of a table are defined as follows:

- F1: Overflow Rows

  Reorganization criteria F1 describes how large a percent the overflow rows are taking up within a table. Table row data overflows when VARCHAR columns are updated with values that are longer than the initial values. In such cases, a pointer is kept at the original location in the row and the actual value is stored in another location indicated by the pointer stored in the original location. This impacts performance because DB2 UDB must follow the pointer to find the contents of the row. This two-step process increases the processing time and might also increase the number of I/O operations required. Figure 4-17 shows the principle of overflow rows.

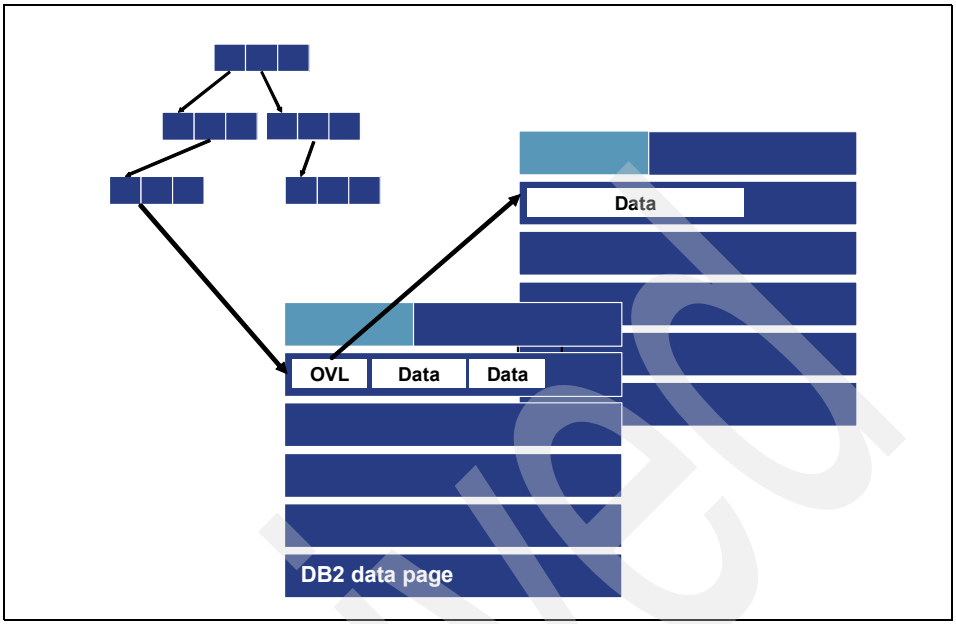


Figure 4-17 Overflow rows

Tip: DB2 UDB recommends to reorganize a table which has more than 5 percent overflow rows.

- F2: Table Size / Allocated Space

  Reorganization criteria F2 describes the ratio between the table size and the allocated space for the table within the table space.

Note: DB2 UDB recommends to reorganize a table when the table size is less than 68 percent of the space allocated in the table space.

- F3: Full Pages / Allocated Pages

  Reorganization criteria F3 describes the ratio between full pages and allocated pages for the table. During deletion of table data, pages could become empty.

Tip: DB2 UDB recommends to reorganize a table when the number of full pages is less than 80 percent of the total allocated pages for table.

The reorganization criteria F4-F8 of a index are defined as follows:

- F4: Cluster Ratio

  Reorganization criteria `F4`, `Cluster Ratio`, describes how much percent of the index sequence is identical with the table sequence. With a high clustering ratio, DB2 UDB can efficiently perform sequential prefetching techniques during the data access via the index. Figure 4-18 shows a good and a bad clustering of an index.

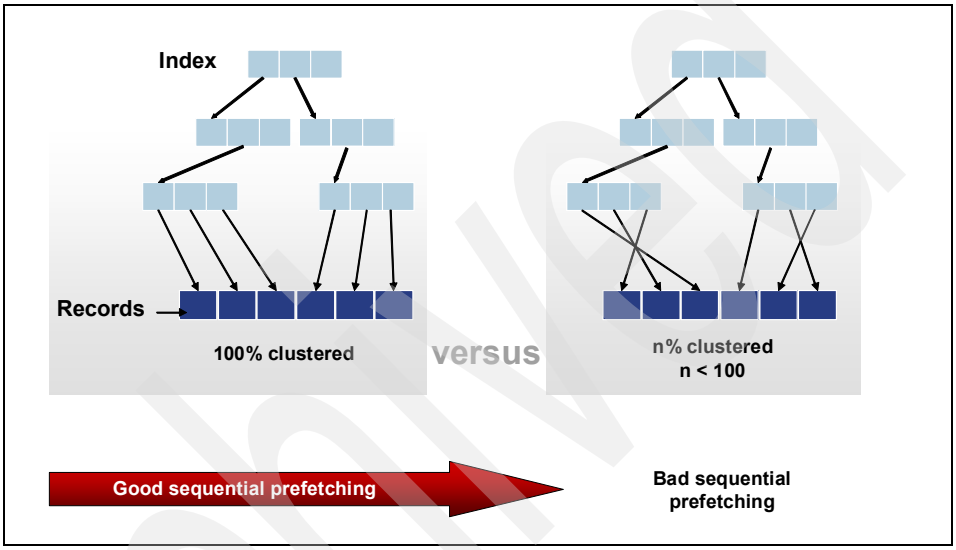


Figure 4-18 Index clustering examples

Tip: DB2 UDB recommends to reorganize an index when the clustering ratio is less than 80 percent. If a table has more than one index defined, having all indexes with good clustering ratio may not be possible to achieve. In an SAP/DB2 environment, specify the primary index when reorganizing a table.

- F5: Index Size / Allocated Space

  Reorganization criteria F5 describes the ratio between the index size and the allocated space for the index within the table space.

  Tip: DB2 UDB recommends to reorganize an index when the index size is less than 50 percent of the space allocated in the table space.

- F6: No. Entries / No. Poss. Entries

  Reorganization criteria F6 describes whether recreating the index would result in an index tree having fewer levels.

  Tip: DB2 UDB recommends to reorganize an index when its actual number of index entries is lower than 90 percent of the index entries an index tree with one fewer level can store.

- F7: Ratio of Deleted Index Entries

  Reorganization criteria F7 describes the percentage of pseudo deleted Record Identifiers (RIDs) within index pages.

  Tip: DB2 UDB recommends to reorganize an index when the percentage of pseudo deleted RIDs is higher than 20. If this is the only criteria to do a reorganization, you can use the command:

  ```
  REORG INDEXES ALL FOR schemaname.tablename ALLOW WRITE ACCESS CLEANUP ONLY
  ```

- F8: Ratio of Deleted Index Leaf pages

  Reorganization criteria F8 describes the percentage of pseudo empty leaf pages within an index tree.

  Tip: DB2 UDB recommends to reorganize an index when the percentage of pseudo empty leaf pages within an index tree is higher than 20. If this is the only criteria to do a reorganization, you use the command:

  ```
  REORG INDEXES ALL FOR schemaname.tablename ALLOW WRITE ACCESS CLEANUP ONLY PAGES
  ```

4.4.4 High water mark

The High Water Mark (HWM) of a table space is that page with the highest number within the table space containing data, no matter what kind of data this is (table or index data, space maps, object maps, ...). Sometimes, when you have the need to reduce the size of your table spaces (for example, during a system copy from production to test with less disk space on the test system), you may need to pay attention to the HWM in order to claim all the free spaces. Please note that the HWM only relates to DMS table spaces.

Reducing table space size during a redirected restore or by **ALTER TABLESPACE *tablespacename* {DROP | REDUCE | RESIZE} (...)** is only possible when not trying to bring the table space size below the HWM.

First you have to determine where the HWM of a table space currently is and whether there are free pages below the HWM. Otherwise reducing the HWM is not possible at all. This can be done by **LIST TABLESPACES SHOW DETAIL** and **db2dart *sapsid* /DHWM /TSI *tablespace-id***.

Example 4-10 shows that the table space PSAPSOURCED has a HWM of 40832 pages, but only 34650 pages are used within the table space. The output of the preceding mentioned **db2dart** command shows that there are 98 unused extents below the HWM and that a table object with the object id 782 holds the HWM. If you want to know what table this is, query the DB2 UDB system table SYSCAT.TABLES using the table space id and the given object id; for indexes query SYSCAT.INDEXES.

Example 4-10 Analyzing High Water Mark

```
> db2 list tablespaces show detail
...
 Tablespace ID                        = 9
 Name                                 = PSAPSOURCED
 Type                                 = Database managed space
 Contents                             = Any data
 State                                = 0x0000
   Detailed explanation:
     Normal
 Total pages                          = 42030
 Useable pages                        = 41664
 Used pages                           = 34560
 Free pages                           = 7104
 High water mark (pages)              = 40832
...


> db2dart W34 /DHWM /TSI 9
```

```
        The requested DB2DART processing has completed successfully!
                        Complete DB2DART report found in:
/db2/W34/db2dump/DART0000/W34.RPT


> cat /db2/W34/db2dump/DART0000/W34.RPT
...
Dump highwater mark processing - phase start.

Number of free extents below highwater mark: 98
Number of used extents below highwater mark: 540

Object holding highwater mark:

 Object ID: 782
 Type:      Table Data Extent


Dump highwater mark processing - phase end.
...

> db2 "select tabschema, tabname from syscat.tables
      where tableid=782 and tbspaceid=9"

TABSCHEMA       TABNAME
--------------- ---------------
SAPR3           TRDIRE

  1 record(s) selected.
```

The **db2dart** *sapsid* **/LHWM /TSI** *tablespace-id* **/NP 0** command creates a report with steps that must be performed to lower the HWM as much as possible. Normally these steps are reorganizing some tables or rebuilding some indexes. Example 4-11 is a continuation of Example 4-10 showing the `db2dart` output.

Example 4-11 db2dart - high water mark

```
> db2dart W34 /LHWM /TSI 9 /NP 0

        The requested DB2DART processing has completed successfully!
                        Complete DB2DART report found in:
/db2/W34/db2dump/DART0000/W34.RPT


> cat /db2/W34/db2dump/DART0000/W34.RPT
```

```
_____                    DART                    _____

  D a t a b a s e   A n a l y s i s   a n d   R e p o r t i n g   T o o l

                        IBM    DB2    6000
```

```
DART (V8.1.0)  Report:
2005-05-24-15.17.34.194413

            Database Name: W34
            Report name: W34.RPT
            Old report back-up: W34.BAK
            Database Subdirectory: /db2/W34/db2w34/NODE0000/SQL00001
            Operational Mode: Database Inspection Only (INSPECT)
```

```
Action option: LHWM
Tablespace-ID: 9; Desired Highwater Mark (Number-pages): 0

Connecting to Buffer Pool Services...

Highwater mark processing - phase start.


NOTES:  All highwater mark values and/or object sizes listed below are
        given in extents and not pages (unless explicitly stated).

        The object ID and object type are shown for each extent listed.

        Extents marked with an asterisk (*) hold the first page of an
        object and these extents can only be moved by dropping and
        recreating that object.

        Extents marked as belonging to objects with ID equal to 65534 or
        65535 are SMP extents or object table extents and they are not
        movable.
```

```
        After following a step and before continuing on to the next one,
        disconnect and reconnect to the database.


Highwater Mark:  40832 pages, 638 extents (extents #0 - 637)


Lower highwater mark processing - phase start.
     Current highwater mark:                     637
     Desired highwater mark:                     0
     Number of used extents in tablespace:       540
     Number of free extents below original HWM:  98
     Number of free extents below desired HWM:   0
     Number of free extents below current HWM:   98


Step #1:  Object ID = 782

 => Offline REORG of this table using the LONGLOBDATA option (do not
    specify a temporary tablespace).

    Table: SAPR3.TRDIRE

     DAT object size:   6
     INX object size:   0
     LF object size:    0
     LOB object size:   0
     LOBA object size:  0
     BMP object size:   0

     Total size of object parts: 6
     Minimum number of extents that will move by this operation: 6

     Current highwater mark:                     631
     Desired highwater mark:                     0
     Number of used extents in tablespace:       540
     Number of free extents below original HWM:  98
     Number of free extents below desired HWM:   0
     Number of free extents below current HWM:   92


Step #2: Object ID = 779

 => Offline REORG of this table using the LONGLOBDATA option (do not
    specify a temporary tablespace).

    Table: SAPR3.TMCRT
```

```
     DAT object size:   2
     INX object size:   0
     LF object size:    0
     LOB object size:   0
     LOBA object size:  0
     BMP object size:   0

     Total size of object parts: 2
     Minimum number of extents that will move by this operation: 2

     Current highwater mark:                     629
     Desired highwater mark:                     0
     Number of used extents in tablespace:       540
     Number of free extents below original HWM:  98
     Number of free extents below desired HWM:   0
     Number of free extents below current HWM:   90


Step #3:  Object ID = 524

 => Offline REORG of this table using the LONGLOBDATA option (do not
    specify a temporary tablespace).

    Table: SAPR3.SMODISRC

     DAT object size:   10
     INX object size:   0
     LF object size:    0
     LOB object size:   0
     LOBA object size:  0
     BMP object size:   0

     Total size of object parts: 10
     Minimum number of extents that will move by this operation: 10

     Current highwater mark:                     541
     Desired highwater mark:                     0
     Number of used extents in tablespace:       540
     Number of free extents below original HWM:  98
     Number of free extents below desired HWM:   0
     Number of free extents below current HWM:   2


Step #4:  Object ID = 16

 => Offline REORG of this table using the LONGLOBDATA option (do not
    specify a temporary tablespace).

    Table: SAPR3.DIRTREE
```

```
     DAT object size:   2
     INX object size:   0
     LF object size:    0
     LOB object size:   0
     LOBA object size:  0
     BMP object size:   0

     Total size of object parts: 2
     Minimum number of extents that will move by this operation: 2

     Current highwater mark:                     539
     Desired highwater mark:                     0
     Number of used extents in tablespace:       540
     Number of free extents below original HWM:  98
     Number of free extents below desired HWM:   0
     Number of free extents below current HWM:   0


Final highwater mark:  Extent #539 (540 extents, 34560 pages).

** This cannot be lowered further as there are not enough free extents
   to move the object holding the highwater mark.


Lower highwater mark processing - phase end.

Highwater mark processing - phase end.

                    ____________________________________

                  DB2DART Processing completed with warning(s)!
                     Warning(s) detected during processing.
                    ____________________________________

                       Complete DB2DART report found in:
/db2/W34/db2dump/DART0000/W34.RPT

    ______   D A R T   P R O C E S S I N G   C O M P L E T E   ______
```

Sometimes, when the HWM is held by a Space Map Page (SMP), reorganizing tables and rebuilding indexes will not help to lower the HWM. In this case you have to use the command **db2dart *sapsid* /RHWM /TSI *tablespace-id***.

A HWM held by an object map cannot be lowered with DB2 UDB at the moment.

For further information, please refer to SAP Notes *152531* and *486559*.

4.5 Best practices for intelligent storage subsystems

This section provides some recommendations regarding how to distribute table spaces and their containers on intelligent storage subsystems. Please keep in mind that before using large containers, we recommend to enable DIO/CIO as described in section 4.2.2, “File based DMS table spaces”.

Here we only discuss file based DMS table spaces, because all table spaces within an SAP database are this type except the temporary table spaces, which are SMS table spaces.

For the temporary table spaces, we also recommend to configure more than one container on different disks or RAID-5 arrays when using manual storage management so that DB2 UDB can access them in parallel.

4.5.1 Database stored on single SSA/SCSI disks

Figure 4-19 shows the recommended distribution of table spaces on a group of individual disks. By having one large container per table space and disk, DB2 UDB parallel access of data on disks can be best used. Creating too many small containers per table space and disk leads to a scattered read and write access of data.

The best way to achieve this is to create an `sapdata` file system on each physical disk and then create one container per table space in each of the `sapdata` file systems. Also, after adding new disks to the database server, we recommend that you create an `sapdata` file system on each of these new disks. Use the **`ALTER TABLESPACE`** *`tablespacename`* **`BEGIN NEW STRIPE SET (...)`** to add storage on the new disks to the table spaces to avoid starting the rebalancer. This can only be done if the table space is not managed by the automatic storage database.

Within an automatic storage enabled database, you can only add new storage paths to the database, but the distribution of the containers cannot be influenced. The best practice in this case is also to create `sapdata` file systems on each disk and then add these `sapdata` file systems as storage paths to the database using **`ALTER DATABASE ADD STORAGE ON storagepath1 [, storagepath2, ...]`**.

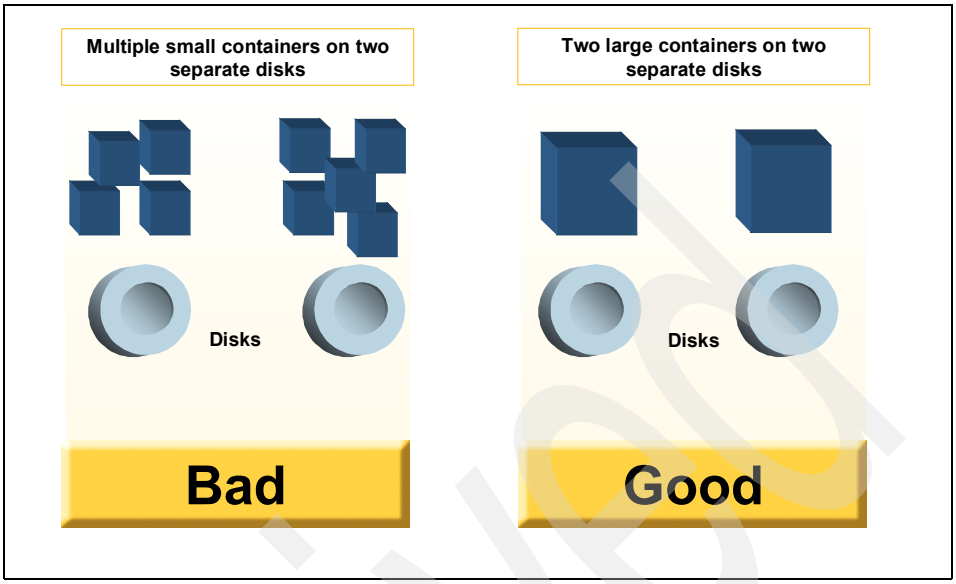


Figure 4-19 table space on single disks

4.5.2 Database stored on RAID-5 arrays

The recommended distribution of a table space on a Redundant Array of Independent Disks, Level 5 (RAID-5) can be seen in Figure 4-20. The containers of each table space should be distributed over all existing `sapdata` file systems, where each consists of a RAID-5 array. The procedure of enlarging the table space is similar to those with single disks.

Create a new `sapdata` file system on each newly added RAID-5 array. After that use the **`ALTER TABLESPACE`** ***`tablespacename`*** **`BEGIN NEW STRIPE SET (...)`** command to add storage on the new RAID-5 arrays to the table spaces to avoid the rebalancer. This process only applies to table spaces managed by a non-automatic storage enabled database. For an automatic storage enabled database, each `sapdata` file system should be used as a storage path for the database.

To improve database performance on RAID-5 arrays, the following conditions should be met:

- The `EXTENTSIZE` of each table space should be equal to (or a multiple of) the RAID stripe size.
- The `PREFETCHSIZE` of each table space should be the RAID stripe size multiplied by the number of RAID parallel disk drives (or a whole multiple of this product) and a multiple of the `EXTENTSIZE`.

- The DB2_PARALLEL_IO registry variable should be used to enable parallel I/O for all table spaces: **db2set DB2_PARALLEL_IO=*.** This is set as default during SAP installation.

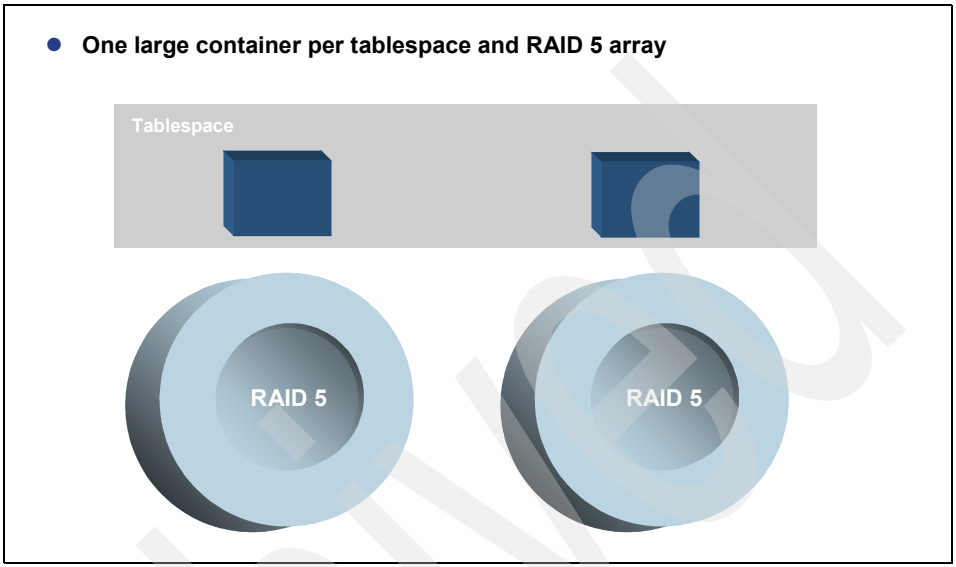


Figure 4-20 table space on RAID-5 arrays

4.6 Moving tables and indexes using DB6CONV

Moving tables and indexes from one table space to another is required, for the following reasons:

- The original table space becomes full and cannot be enlarged because the maximum table space size has been reached.
- The page size defined is smaller than the table's record length, causing some fields of the table to be stored in unbuffered **LONG** fields.
- I/O contention: The table must be moved to another table space residing on other physical disks or RAID-5 devices.

4.6.1 Principle of moving tables and indexes using DB6CONV

The following steps must be followed to move tables and indexes from one table space to another. Depending on whether you have used the tool DB6CONV once before in this system and whether some prerequisites are fulfilled, it is not necessary to do them all:

- Import the latest version of the SAP report DB6CONV to the SAP system. The latest version of report DB6CONV can be found in the SAPNet as attachment to SAP Note *362325*. Instructions how to import the report into the SAP system can be found in the mentioned SAP Note.
- To use different page size for the target table space of the table to be moved, a buffer pool and a temporary table space with the same page size are needed. Create buffer pool and temporary table space with that page size if the system does not have one yet. For the naming conventions, refer to SAP Note *147634*.
- If not already existing, the target table spaces (at least one for the table and one for the indexes) must be created and they must be granted for public usage. For the naming conventions refer to SAP Note *147634.*
- Use transaction SE38 (ABAP Editor) to start the report DB6CONV and use it to move the table and indexes to the target table spaces. Information about the usage of the report DB6CONV can be found in SAP Notes *362325* and *817709*.
- Create a new TABART for the table and indexes being moved or update the assignment to an existing one so that the TABART configuration within the SAP systems reflects the locations of the table and indexes in table spaces. With SAP releases up to 4.6D you have to create the new TABART manually as described in the example for a better understanding. Starting with SAP release 4.7 new TABARTs can also be created within the DBA Cockpit, which we will show at the end of the example.

4.6.2 Example: Moving table PCL2 to improve performance

In this example, we provide SQL statements for creating buffer pool, table space, and verification. These SQL statements can be used for both SAP NetWeaver 2004s and the systems prior to SAP NetWeaver 2004s. Note that for SAP NetWeaver 2004s, these operations can also be carried out using DBA Cockpit. In prior systems, DBA Cockpit does not support these operations.

First of all, we download and import the latest version of report DB6CONV into our SAP system as described in SAP Note 362325.

Within this example the moving of the table PCL2 (HR Cluster 2) and its indexes from their original table spaces GR5#BTABD and GR5#BTABI to new ones, ZGR5#PCL2D and ZGR5#PCL2I, will be shown. The original table space has a page size of 4K, because of this the SAP Database Support Layer (DBSL) has created the field CLUSTD of the table as **LONG VARCHAR**, otherwise the record length would not have fitted into a single page. To improve performance and have an buffered access to the table PCL2, it may as well be moved in a table space with (at least) 8K page size. Most customers may not want to do this, but high-end systems will benefit from buffered access.

Figure 4-21 shows the table structure of the PCL2 with 4K page size, which you can see in transaction DB02 and

- Select **Single Table Analysis** on the left side
- Enter PCL2 as tablename on the right side and press **Enter**
- Select the register **Table Structure**

The field CLUSTD in the last row has an SAP type X with a length of 3900 bytes, but on the database it has been created as **LONG VARCHAR** with 32700 bytes.

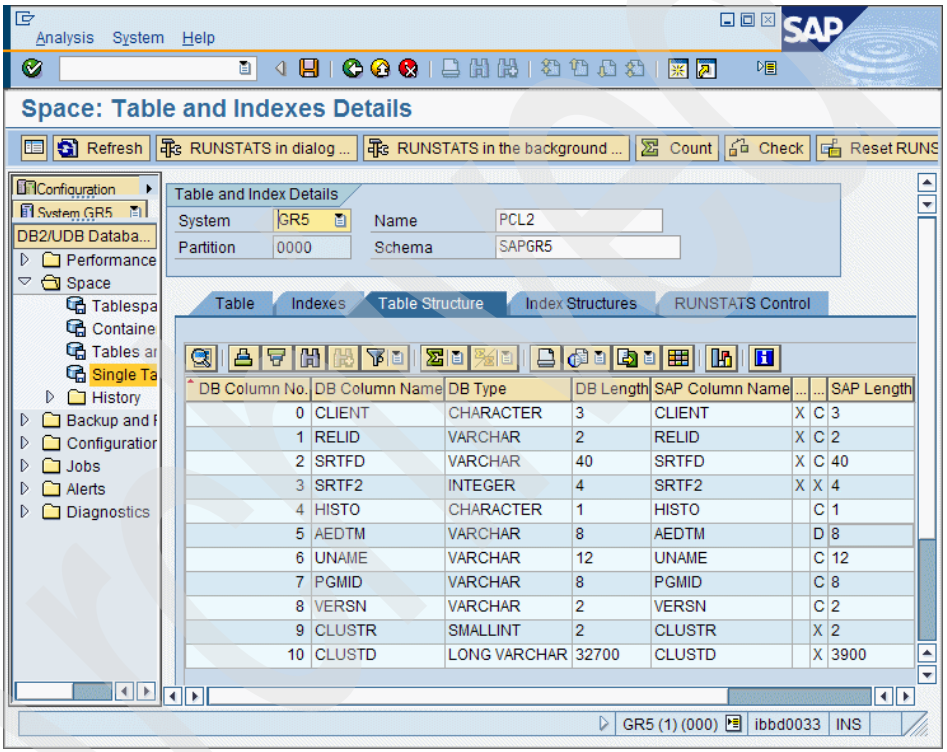


Figure 4-21 Table structure of PCL2 on 4K pagesize

The target table spaces ZGR5#PCL2D and ZGR5#PCL2I should have a page size of 8K, because of this we need a buffer pool and a temporary table space with 8K page size.

To view the existing buffer pools, we query the DB2 UDB system table SYSCAT.BUFFERPOOLS as shown in Example 4-12. We found only a 4K and a 16K buffer pool are defined.

Example 4-12 View existing buffer pools

```
> db2 "select bpname, npages, pagesize from syscat.bufferpools"

BPNAME                         NPAGES      PAGESIZE
------------------------- ----------- -----------
IBMDEFAULTBP                    10000        4096
BP_STD_16K                       5000       16384

  2 record(s) selected.
```

Tip: The column NPAGES shows the size of the buffer pool. Having a size of -1 means that this buffer pool has the size of the **DB CFG** parameter BUFFPAGE. We recommend not to use such a configuration when having more than one buffer pool, because each buffer pool configured with a size of -1 will be influenced when changing the **DB CFG** parameter BUFFPAGE, that may not be the intention.

Checking the temporary table space can be done with DB2 UDB commands **LIST TABLESPACES** and **LIST TABLESPACE CONTAINERS FOR** *tablespace-id*. In our example system there are two temporary PSAPTEMP4 and PSAPTEMP16, both using the /db2/GR5/saptemp1 file system, as shown in Example 4-13.

Example 4-13 View file system of temporary table spaces

```
> db2 list tablespaces
...
 Tablespace ID                        = 3
 Name                                 = PSAPTEMP
 Type                                 = System managed space
 Contents                             = System Temporary data
 State                                = 0x0000
   Detailed explanation:
     Normal

 Tablespace ID                        = 4
 Name                                 = PSAPTEMP16
 Type                                 = System managed space
 Contents                             = System Temporary data
 State                                = 0x0000
   Detailed explanation:
     Normal
...


> db2 list tablespace containers for 3

            Tablespace Containers for Tablespace 3
```

```
 Container ID                         = 0
 Name                                 =
                      /db2/GR5/saptemp1/NODE0000/temp4/PSAPTEMP.container000
 Type                                 = Path


> db2 list tablespace containers for 4

            Tablespace Containers for Tablespace 4

 Container ID                         = 0
 Name                                 =
                   /db2/GR5/saptemp1/NODE0000/temp16/PSAPTEMP16.container000
 Type                                 = Path
```

For creating a new buffer pool, use the **CREATE BUFFERPOOL** *bufferpoolname* **IMMEDIATE SIZE** *npages* **PAGESIZE** *pagesize* statement (Example 4-14).

Example 4-14 Creating buffer pool

```
> db2 "create bufferpool SAPBP8 immediate size 5000 pagesize 8 k"
DB20000I  The SQL command completed successfully.


> db2 "select bpname, npages, pagesize from syscat.bufferpools"

BPNAME                    NPAGES      PAGESIZE
------------------------- ----------- -----------
IBMDEFAULTBP                    10000        4096
BP_STD_16K                       5000       16384
SAPBP8                           5000        8192

  3 record(s) selected.
```

The command **CREATE SYSTEM TEMPORARY TABLESPACE** creates the necessary system managed temporary table space PSAPTEMP8, for which we share the file system /db2/GR5/saptemp1 with the existing temporary table spaces PSAPTEMP4 and PSAPTEMP16. See Example 4-15.

Example 4-15 Creating temporary table space

```
> db2 "create system temporary tablespace PSAPTEMP8
      in database partition group IBMTEMPGROUP
      pagesize 8 K managed by system using
      ('/db2/GR5/saptemp1/NODE0000/temp8/PSAPTEMP8.container000')
      bufferpool SAPBP8"
DB20000I  The SQL command completed successfully.
```

Now we can create our target table spaces ZGR5#PCL2D and ZGR5#PCL2I using CREATE REGULAR TABLESPACE command. Depending on the storage management method (manual, auto-resize, or automatic storage) of the system, proper parameters must be used. In this example we have a system with manual storage management. After creating the table spaces, they must be granted for public usage. See Example 4-16.

Example 4-16 Creating regular table spaces

```
> db2 "create regular tablespace ZGR5#PCL2D
      in database partition group SAPNODEGRP_GR5
      pagesize 8 K managed by database using (
      file '/db2/GR5/sapdata1/NODE0000/ZGR5#PCL2D.container000' 1000 K,
      file '/db2/GR5/sapdata2/NODE0000/ZGR5#PCL2D.container001' 1000 K,
      file '/db2/GR5/sapdata3/NODE0000/ZGR5#PCL2D.container002' 1000 K,
      file '/db2/GR5/sapdata4/NODE0000/ZGR5#PCL2D.container003' 1000 K)
      extentsize 8 prefetchsize 8
      bufferpool SAPBP8 dropped table recovery off"
DB20000I  The SQL command completed successfully.

> db2 "grant use of tablespace ZGR5#PCL2D to public"
DB20000I  The SQL command completed successfully.

> db2 "create regular tablespace ZGR5#PCL2I
      in database partition group SAPNODEGRP_GR5
      pagesize 8 K managed by database using (
      file '/db2/GR5/sapdata1/NODE0000/ZGR5#PCL2I.container000' 1000 K,
      file '/db2/GR5/sapdata2/NODE0000/ZGR5#PCL2I.container001' 1000 K,
      file '/db2/GR5/sapdata3/NODE0000/ZGR5#PCL2I.container002' 1000 K,
      file '/db2/GR5/sapdata4/NODE0000/ZGR5#PCL2I.container003' 1000 K)
      extentsize 4 prefetchsize 4
      bufferpool SAPBP8 dropped table recovery off"
DB20000I  The SQL command completed successfully.

> db2 "grant use of tablespace ZGR5#PCL2I to public"
DB20000I  The SQL command completed successfully.
```

Tip: Before creating the table space, check the extent size and prefetch size of the original table space using **LIST TABLESPACES SHOW DETAIL**.. To decide the new extent size and prefetch size, SAP recommends to divide the extent size and prefetch size of the current table spaces by the same factor as you multiply the page size for the new table space.

Tip (continued): Here is an example of this computation:

```
original table space:
page size=4K, extentsize=16, prefetchsize=16

new table space:
page size=8K, means page size will be multiplied with factor 2
new table space: extensize=8, prefetchsize=8 (divided by factor 2)
```

Having all prerequisites completed, we can start transaction SE38 (ABAP Editor), enter DB6CONV in the Program field and press **F8** or click the **Execute** button. Figure 4-22 shows the entry screen of the report DB6CONV (DB2 UDB Table Conversion Tool).

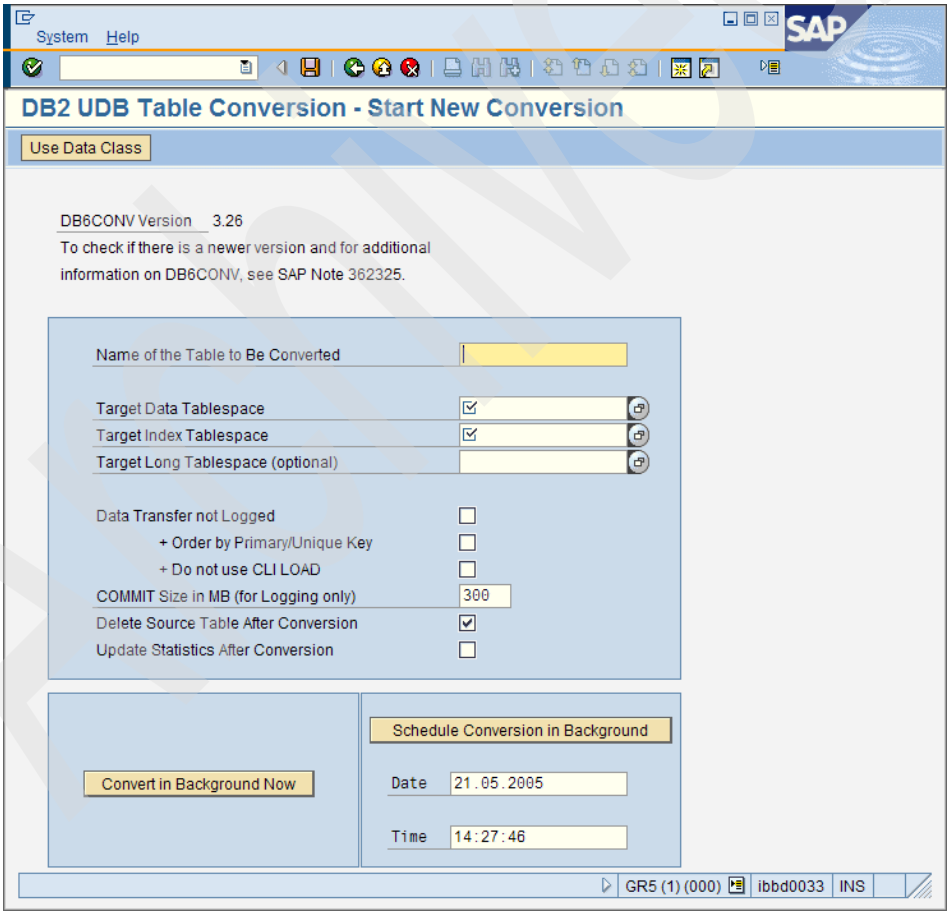


Figure 4-22 DB2 UDB table conversion tool DB6CONV

Here we can enter or select the following items:

- Name of the Table to Be Converted
- Target Data Tablespace
- Target Index Tablespace
- Target Long Tablespace (optional):
  Enter this field if you want the tool to store the LONG fields of the table to another table space.
- Data Transfer not Logged:
  With this option selected, the tool runs in a no-logging mode which will convert the table faster. After the conversion, it is required to perform a backup at least of the source and target table spaces. This option also enables the conversion of tables not having an primary or unique key.
- + Order by Primary/Unique key:
  This option can be used together with *Data Transfer not Logged* so that the conversion will be done according the primary or unique key to have an implicit reorganization of the table during the conversion.
- + Do not use CLI LOAD:
  In some rare cases there were problems with the DB2 UDB CLI LOAD function used by `DB6CONV` together with *Data Transfer not Logged.* SAP recommends to use this option only when you have experienced problems with the CLI `LOAD` function during a table conversion.
- COMMIT Size in MB (for Logging only):
  Specifies the size of the data packages to be converted before tool sends a **`COMMIT`** to the database.
- Delete Source Table after Conversion:
  This option tells the tool to delete the renamed source table after the successful conversion.
- Update Statistics After Conversion:
  This option tells the conversion tool to create new statistic for the table and its indexes after the conversion.
- Use Data Class:
  Using the Use Data Class button you can enter an existing Target Data Class (TABART) instead of the target table spaces. With this option the conversion tool also changes the data class of the table during the conversion.

To convert our table PCL2 and its indexes from the original table spaces `GR5#BTABD` and `GR5#BTABI` to the target table spaces `ZGR5#PCL2D` and `ZGR5#PCL2I`, we use the default options but choose **Update Statistics After Conversion**, as shown in Figure 4-23.

It is possible to start the conversion immediately in background or to schedule it for a specific point in time. In this case we decide to start immediately by pressing the **Convert in Background Now** button.

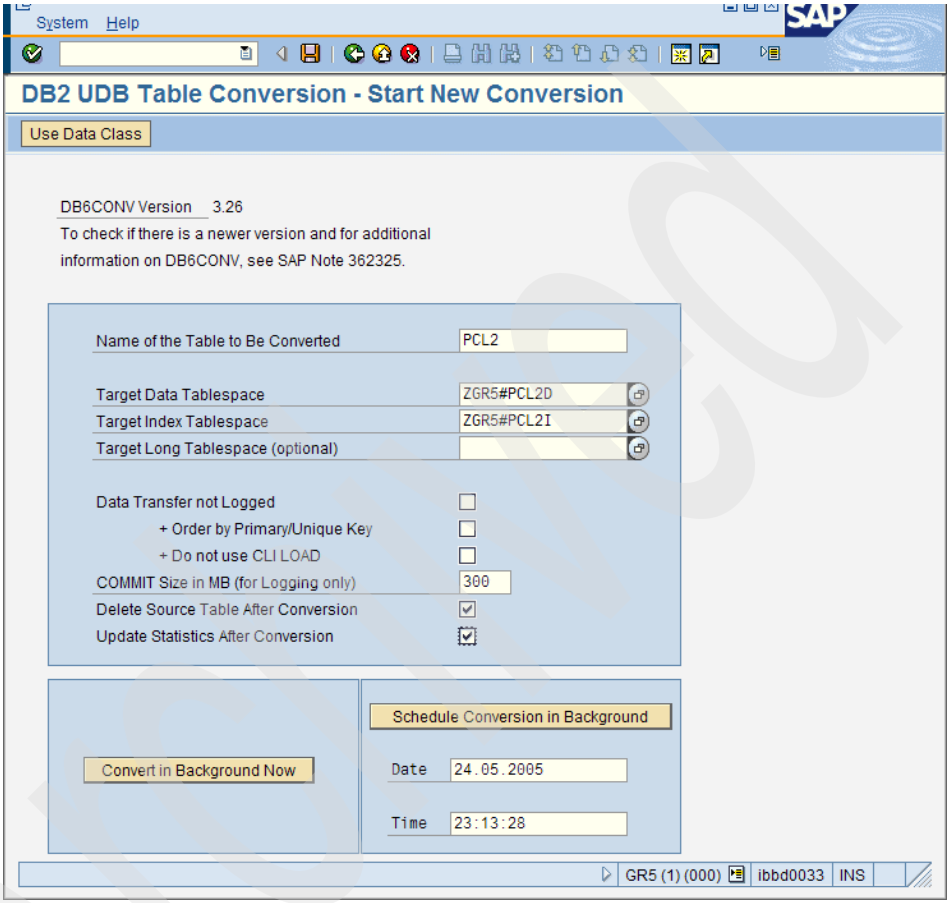


Figure 4-23 Conversion of table PCL2

The conversion is now running in background, and the tool displays the *DB2 UDB Table Conversion - Overview* (Figure 4-24). To watch the progress of the conversion, press **F5** or click the **Refresh** Refresh button.

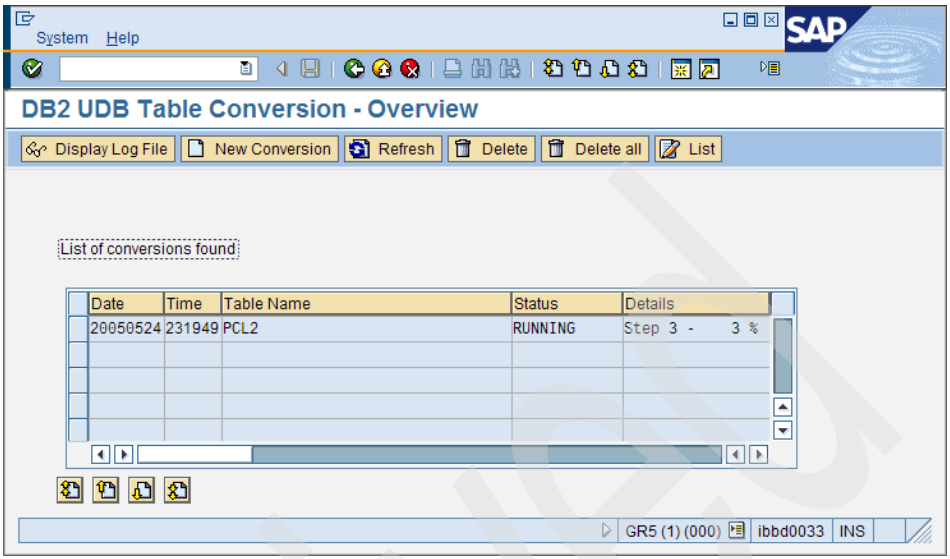


Figure 4-24 Conversion progress

Once the conversion job has completed, the finished status is displayed (see Figure 4-25).

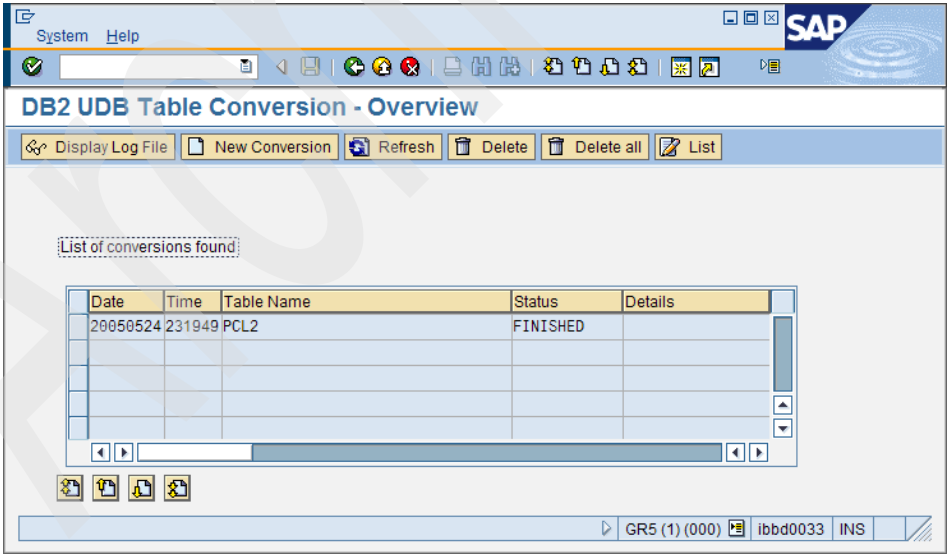


Figure 4-25 Conversion finished

Report DB6CONV log file provides the conversion details. To open the log file, highlight the row of interest, PCL2 row in our example, in the conversion list, by clicking the box : in the first column. After that, click the **Display Log File** Display Log File button. Example 4-17 shows the log content. For more detailed information about the working method of DB6CONV, refer to SAP Note *362325*.

Example 4-17 DB2CONV log

```
====================NEW CONVERSION - START==========================
Version of DB6CONV            : 3.26
Starttime                     : 20050524 231949
Conversion type               : Batch immediate
Conversion defined using      : Tablespaces
Tablename                     : PCL2
DATA Tablespace               : ZGR5#PCL2D
INDEX Tablespace              : ZGR5#PCL2I
LONG Tablespace (optional)    :
Use 'Data Transfer not logged': No
Conversionsize                : 300
Delete original               : Yes
Update Statistics After Conv. : Yes
*****************************************************************
20050524 231951 Start of Step 0
20050524 231951 check_insert_select
20050524 231951 conversion via INSERT-SELECT (without 'Data Transfer not
            Logged') is possbible
20050524 231951 index PCL2~0 will be used in step 5
20050524 231951 Data Class of original table      : APPL1
20050524 231951 Data Tablespace of original table : GR5#BTABD
20050524 231951 Index Tablespace of original table: GR5#BTABI
20050524 231951 Long Tablespace of original table : <not defined>
20050524 231951 Pagesize of original tablespaces  :   4096 Byte
20050524 231951 Pagesize of target tablespaces    :   8192 Byte
20050524 231952 >>> Describe Table PCL2
20050524 231952 >>> (colno, colname, typename, length, scale, nulls, codepage)
20050524 231952 >>> ---------------------------------- ------------------------
20050524 231952 >>> 0 CLIENT CHARACTER 3 0 N 819
20050524 231952 >>> 1 RELID VARCHAR 2 0 N 819
20050524 231952 >>> 2 SRTFD VARCHAR 40 0 N 819
20050524 231952 >>> 3 SRTF2 INTEGER 4 0 N 0
20050524 231952 >>> 4 HISTO CHARACTER 1 0 N 819
20050524 231952 >>> 5 AEDTM VARCHAR 8 0 N 819
20050524 231952 >>> 6 UNAME VARCHAR 12 0 N 819
20050524 231952 >>> 7 PGMID VARCHAR 8 0 N 819
20050524 231952 >>> 8 VERSN VARCHAR 2 0 N 819
20050524 231952 >>> 9 CLUSTR SMALLINT 2 0 N 0
20050524 231952 >>> 10 CLUSTD LONG VARCHAR 32700 0 Y 0
20050524 231952 DDIC-tableclass    : TRANSP
20050524 231952 Kernel Release     : 640
20050524 231952 DB6 Version        : 08.02.0000
20050524 231953 Step 0 completed successfully
*****************************************************************
20050524 231953 Start of Step 1
20050524 231954 No views to delete
20050524 231954 Step 1 completed successfully
*****************************************************************
20050524 231954 Start of Step 2
20050524 231955 RENAME TABLE SAPGR5."PCL2" to "QCMPCL2"
20050524 231955 Indexes of original table:
20050524 231955 PCL2~0 ; P ; REG ; +CLIENT+RELID+SRTFD+SRTF2
20050524 231955 1 indexes found on database
20050524 231955 Step 2 completed successfully
*****************************************************************
20050524 231955 Start of Step 3
```

```
20050524 232046 CREATE TABLE "QCM8PCL2" ("CLIENT" SAPDB6FIXCHAR(000003) DEFAULT
            '000' NOT NULL , "RELID" SAPDB6VARCHAR(000002) DEFAULT ' ' NOT NULL ,
            "SRTFD" SAPDB6VARCHAR(000040) DEFAULT ' ' NOT NULL , "SRTF2" INTEGER
            DEFAULT 0 NOT NULL , "HISTO" SAPDB6FIXCHAR(000001) DEFAULT ' ' NOT
            NULL , "AEDTM" SAPDB6VARCHAR(000008) DEFAULT '00000000' NOT NULL ,
            "UNAME" SAPDB6VARCHAR(000012) DEFAULT ' ' NOT NULL , "PGMID"
            SAPDB6VARCHAR(000008) DEFAULT ' ' NOT NULL , "VERSN"
            SAPDB6VARCHAR(000002) DEFAULT ' ' NOT NULL , "CLUSTR" SMALLINT
            DEFAULT 0 NOT NULL , "CLUSTD" SAPDB6RAW(003900) FOR BIT DATA) IN
            "ZGR5#PCL2D" INDEX IN "ZGR5#PCL2I"
20050524 232047 ALTER TABLE "QCM8PCL2" VOLATILE
20050524 232047 tables QCMPCL2 and QCM8PCL2 are idential (#columns, syscat.columns-colname
                + syscat.columns-colno)
20050524 232047 Step 3 completed successfully
****************************************************************
20050524 232047 Start of Step 4
20050524 232047 >>> INDEX MAPPING START
20050524 232047 >>> source table       target table       DDIC remarks
20050524 232047 >>> PCL2~0             PCL2^0             0    ---
20050524 232047 >>> INDEX MAPPING END
20050524 232048 CREATE UNIQUE INDEX "PCL2^0" ON "QCM8PCL2"
                ("CLIENT","RELID","SRTFD","SRTF2" ) ALLOW REVERSE SCANS
20050524 232048 ALTER TABLE "QCM8PCL2" ADD CONSTRAINT "PCL2^0" PRIMARY KEY
                ("CLIENT","RELID","SRTFD","SRTF2" )
20050524 232048 COMMIT
20050524 232048 1 indexes created on database
20050524 232048 Number of Indexes created on QCM8PCL2 = Number of Indexes
            defined for PCL2
20050524 232048 Step 4 completed successfully
****************************************************************
20050524 232048 Start of Step 5
20050524 232048 >>> Counting entries of QCM<tab> (may take a while)...
20050524 232050 Rowcount of table QCMPCL2 before conversion : 0
20050524 232050 Conversion of an emtpy table - no action required in this step!
20050524 232050 >>> Counting entries of QCM8<tab> (may take a while)...
20050524 232051 Rowcount of table QCM8PCL2 after INSERT : 0
20050524 232051 Rowcount of table QCM8PCL2 = Rowcount of original table
20050524 232051 Step 5 completed successfully
****************************************************************
20050524 232051 Start of Step 6
20050524 232051 RENAME TABLE SAPGR5."QCM8PCL2" to "PCL2"
20050524 232051 Step 6 completed successfully
****************************************************************
20050524 232051 Start of Step 7
20050524 232051 DB_SAVE_ALL_VIEWS
20050524 232056 No views recreated
20050524 232056 Number of Views created by DB6CONV = Number of Views using
            table PCL2 before conversion
20050524 232056 Step 7 completed successfully
****************************************************************
20050524 232056 Start of Step 8
20050524 232057 DROP TABLE "QCMPCL2"
20050524 232057 RENAME INDEX "PCL2^0" TO "PCL2~0"
20050524 232057 >>> Updating Statistics for Table PCL2 (may take a while)...
20050524 232103 Update Statistics for Table PCL2 successfully completed
20050524 232103 Step 8 completed successfully
****************************************************************
Status  : Conversion successfully completed.
Stoptime: 20050524 232103
====================END OF SECTION=================================
```

In general, the conversion consists of nine steps:

- Step 0 - Check if conversion can be done or not.
- Step 1 - Delete views of original table.
- Step 2 - Rename original table (add prefix QCM to tablename).
- Step 3 - Create target table (tablename with prefix QCM8).
- Step 4 - Create index mapping and create them on the target table.
- Step 5 - Move data from original table to target table.
- Step 6 - Rename target table (remove prefix QCM8).
- Step 7 - Recreate views.
- Step 8 - Change data class of target table (if selected) and drop original table (if selected).

The goal of this conversion was to move the table PCL2 into a table space with 8K page size so that the SAP DBSL does not need to use **LONG** fields in the table. Figure 4-26 shows the table structure of the PCL2 after conversion. To view the table structure, use transaction /NDB02 and proceed as follows:

- Select **Single Table Analysis** on the left side.
- Enter PCL2 as tablename on the right side and press **Enter**.
- Select the register **Table Structure**.

The field CLUSTD in the last row has an SAP type X with a length of 3900 bytes, and also on the database, it has been created as **VARCHAR** with 3900 bytes.

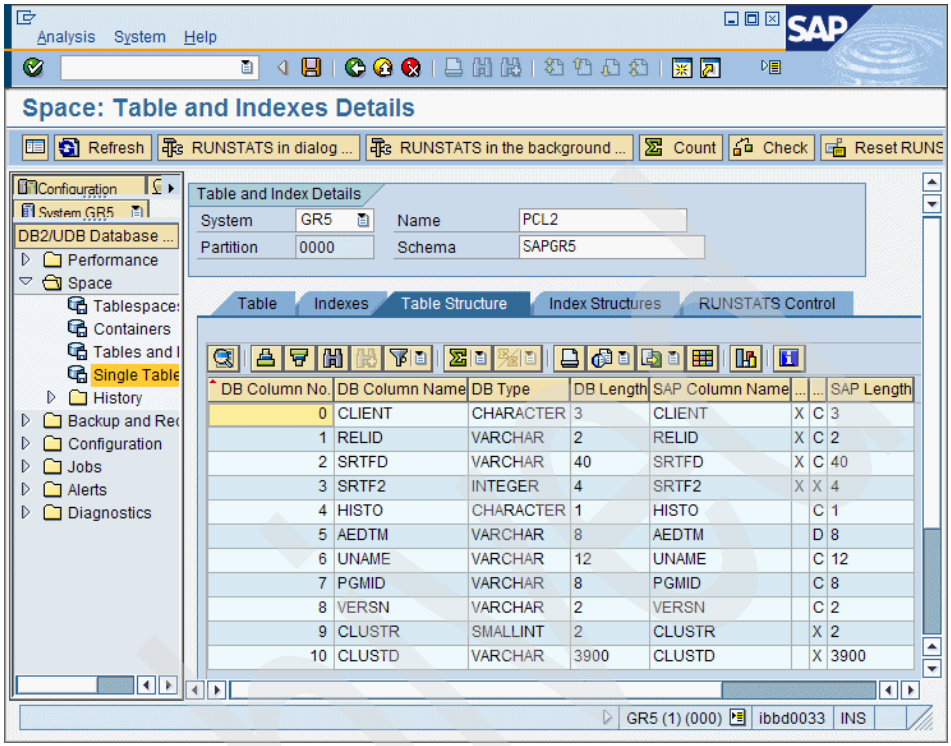


Figure 4-26 Table structure of PCL2 on 8K pagesize

Once the tool has completed moving the table, one important task to do is to update the TABART configuration within the SAP systems to reflect this change. Otherwise this could lead to problems in the future when changing table structures in the SAP system and activating them or when performing an export of the database using the **R3load** tool.

Use `ZPCL2` to create a new `TABART` for the newly created target table spaces. To do this, we need to maintain table `DDART` using transaction `SE16` (Data Browser) as follows:

- Enter `DDART` for the tablename and press **Enter**.
- Press **F8** or click the **Execute** button.
- Press **F5** or click the **Create** button.
- Enter `ZPCL2` for the TABART, `USR` for the data class category and `Data class for table PCL2` for the description as shown in Figure 4-27.
- Press the save button .

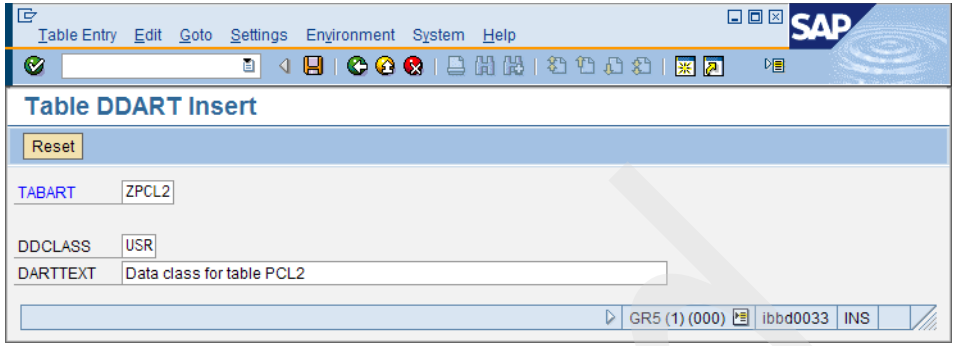


Figure 4-27 Creating TABART ZPCL2

- Press **F3** or the execute button, and the newly defined TABART is now within the TABARTs list as shown in Figure 4-28.

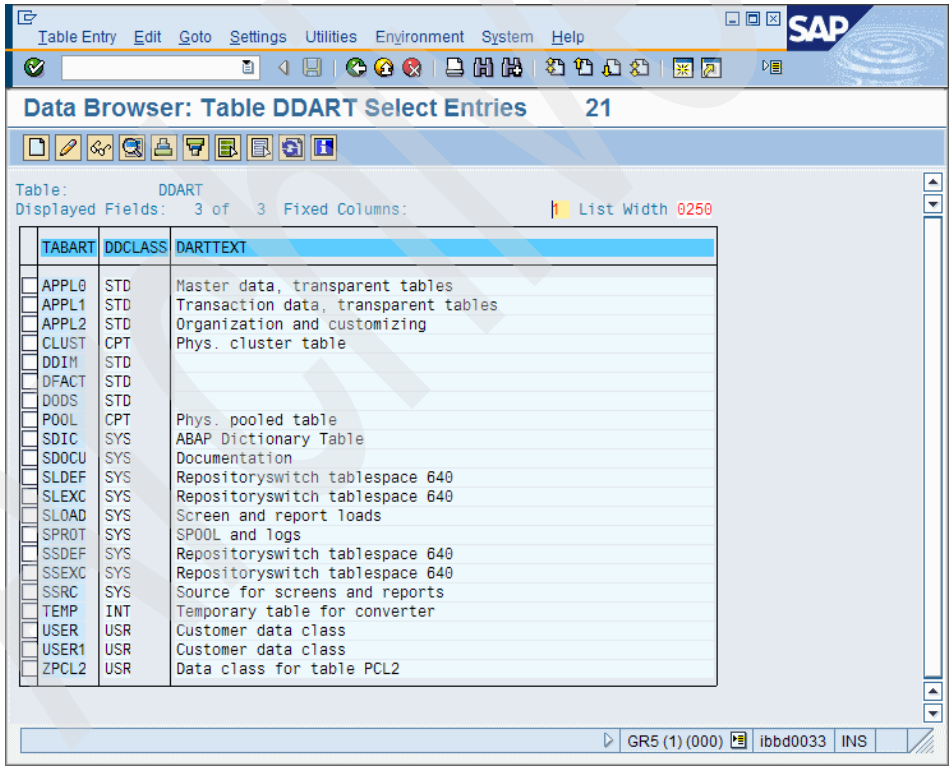


Figure 4-28 List of TABARTs including ZPCL2

In the next step we have to publish our newly created table spaces within the SAP system by maintaining table TSDB6 in transaction SE16 (Data Browser) as follows:

- Enter TSDB6 for the tablename and press **Enter**.
- Press **F8** or click the **Execute** button.
- Press **F5** or click the **Create** button.
- Enter ZGR5#PCL2D for the name of the data table space and ZGR5#PCL2I for the name of the index table space as shown in Figure 4-29.
- Press the save button.
- Press **F3** or execute button and the newly defined table spaces are now within the list of table spaces.

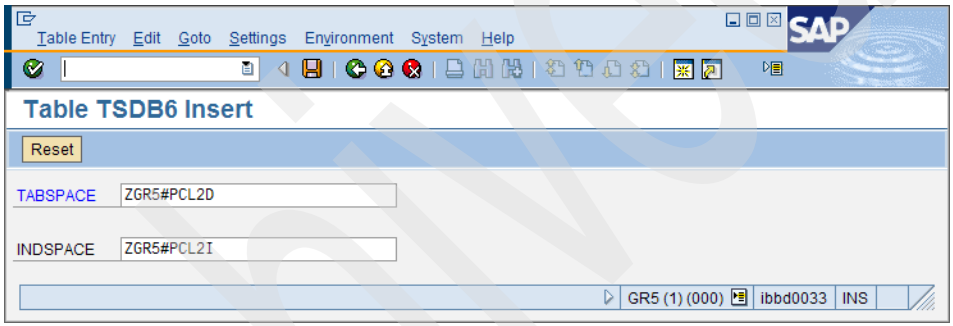


Figure 4-29 Publishing newly created table spaces

Now we have to assign the TABART ZPCL2 to the data table space ZGR5#PCL2D by maintaining table TADB6 within transaction SE16 (Data Browser) as follows:

- Enter TADB6 for the tablename and press **Enter**.
- Press **F8** or click the **Execute** button.
- Press **F5** or click the **Create** button.
- Enter ZPCL2 for the name of the TABART and ZGR5#PCL2D for the name of the data table space as shown in Figure 4-30.
- Press the save button.
- Press **F3** or the execute button and the assignment of TABART ZPCL2 to the data table space ZGR5#PCL2D is now within the list.

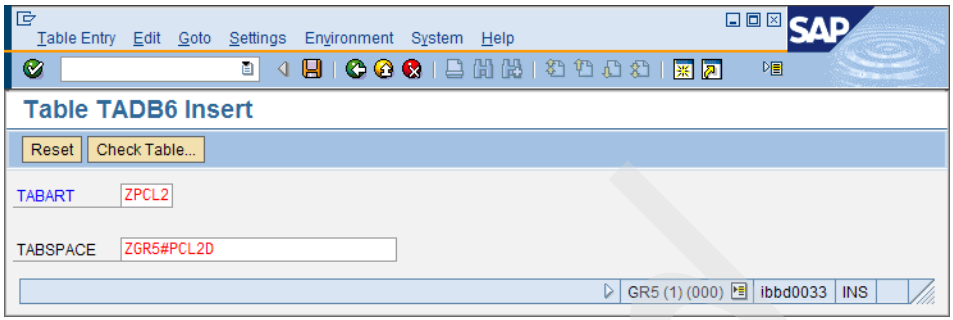


Figure 4-30 Creating assignment of TABART ZPCL2 to data table space ZGR5#PCL2D

The same process should be applied to the index table space. We have to assign the TABART `ZPCL2` to the index table space `ZGR5#PCL2I` by maintaining table `IADB6` within transaction `SE16` (Data Browser) as follows:

- Enter `IADB6` for the tablename and press **Enter**.
- Press **F8** or click the **Execute** button.
- Press **F5** or click the **Create** button.
- Enter `ZPCL2` for the name of the TABART and `ZGR5#PCL2I` for the name of the data table space as shown in Figure 4-31.
- Click the **Save** button.
- Press **F3** or click the **Execute** button and the assignment of TABART ZPCL2 to the data table space `ZGR5#PCL2I` is now within the list.

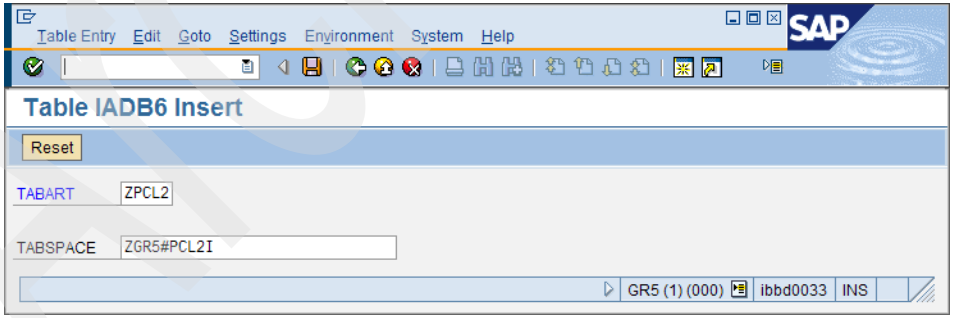


Figure 4-31 Creating assignment of TABART ZPCL2 to index table space ZGR5#PCL2I

The definitions of the TABART ZPCL2 are now all available. In the last step, the table PCL2 must be assigned to the newly created TABART. To do so, call transaction SE11 and proceed as follows:

- Enter PCL2 as tablename.
- Click the **Display** Display button.
- Click the **Technical Settings** Technical Settings button.
- Click the **Display <-> Change** button.
- Confirm the information popup with **Enter**.
- Change the data class to ZPCL2 as shown in Figure 4-32.
- Click the **Save** button to save the changes.
- If requested, create a workbench request.

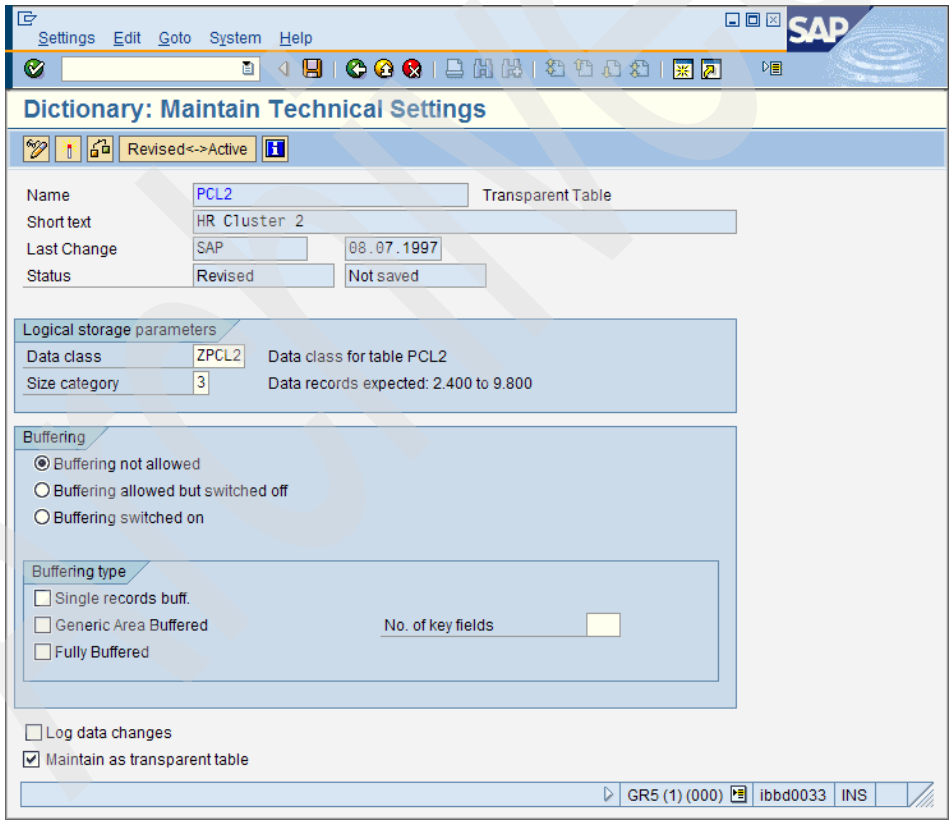


Figure 4-32 Assigning the TABART ZPCL2 to the table PCL2

Creating a new TABART using the DBA Cockpit

As mentioned in the beginning of the section, starting with SAP release 4.7, it is also possible to create a new TABART using the DBA Cockpit. To do so, call transaction `ST04` to open the DBA Cockpit, after that open the `Configuration` folder. Double-clicking `Data Classes` displays a list of all defined data classes (also known as TABARTs) and their assigned table spaces within your SAP system. Using the `Add` Add button, a pop-up window `Add Data Class` appears, as shown in Figure 4-33. Within this window you can enter the name and the description of the new data class, and also assign the new data class to a data and an index table space. Pressing the `Create` button creates the new TABART.

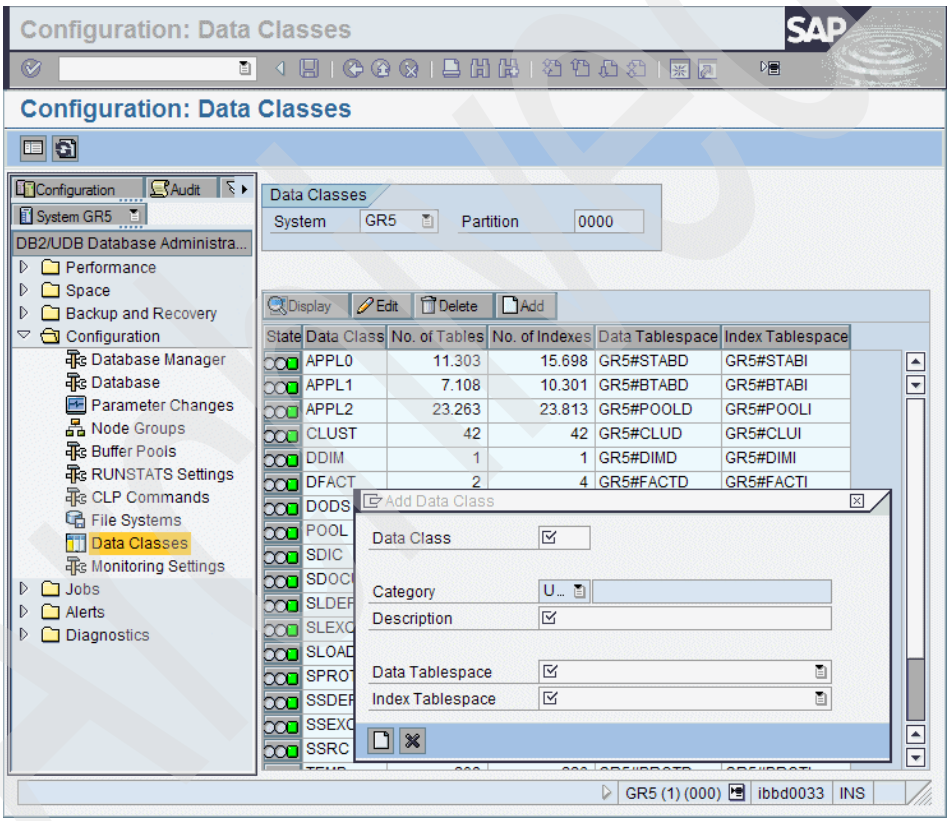


Figure 4-33 Creating a new TABART using the DBA Cockpit

5

DB2 UDB configuration from SAP's perspective

This chapter covers the configuration of the DB2 UDB instance and database from SAP's perspective. We discuss these topics:

- Database manager configuration
- Database configuration
- Database profile registry variables

5.1 DB2 UDB's approach to database parameters

Configuration on DB2 UDB database servers is performed on three levels. The first level, *database manager configuration* (DBM CFG), which focuses on the database manager, forms the controlling process for the DB2 database environment. Here, global settings are maintained, like the name of the TCP/IP port for DB2 UDB's “listener”. The second level is used to configure the database and its processes. It is called *database configuration* (DB CFG). Here, the configuration of the database, such as memory areas, number of auxiliary processes for I/O, etc., are set.

Both database manager and database configuration settings are stored in binary configuration files within the DB2 UDB instance and database directories. These files cannot be edited directly.

The third level of configuration is *database profile registry variables* (also referred as *database registry variables*). Prior to DB2 UDB V8.2.2, various registry variables had to be set individually as required by SAP. DB2 UDB V8.2.2 has made things very easy. Having the `DB2_WORKLOAD` registry variable set to *SAP*, all internals of the DB2 database server are set up in a way that facilitates the operation of your SAP system.

Once the DB2 UDB instance and database are created, DB2 UDB is set up by SAP installation tools using a default configuration. SAP requests that you update the database configuration further using the values specified in SAP Note *584952*. You can review and update the settings using SAP GUI, DB2 UDB GUI Control Center, or CLP with DB2 UDB commands. But we recommend using the SAP GUI with transaction `ST04`, also known as DBA Cockpit. This transaction allows you to view both the database manager and the database configuration parameters. You can also update the database configuration parameters using `ST04`, but not the database manager parameters.

When updating the settings, a sanity check on the parameters is performed and error messages returned if an error is detected. Most of the changes take effect immediately. In some situations, you need to recycle your database manager or database (`DB2STOP/DB2START`).

Having SAP's service offerings implemented, the next adjustments of the configuration settings are based on recommendations from SAP's GoingLive reports. It is good practice to have the configuration parameters from the SAP Service Reports implemented via the normal change management procedures in your system landscape as soon as possible. This helps you to avoid the expedited implementation of the settings in performance, or support cases required by SAP.

While small and medium size system landscapes can be maintained via ST04 efficiently, special considerations should be made, when managing large system landscapes. Here, a scripted approach to setting the DB CFG or DBM CFG is a valid investment.

Note: SAP has its own proprietary database logical and physical design. To have optimal database performance, we recommend that you follow the DB2 UDB configuration setting guidelines published in SAP Note *584952*. This Note is updated from time to time. We recommend that you check this Note on a regular basis.

We elaborate on standard OLTP configurations in SAP systems in the following sections. For special considerations that apply to OLAP systems, such as SAP BW, refer to the SAP Note *584952* OLAP configuration sections.

5.2 DB2 UDB database manager configuration

In this section we discuss the configuration of the database instance, also called the database manager. Use the following DB2 UDB command to update the database configuration parameters (you cannot use ST04 to change these parameters):

```
UPDATE DBM CFG USING <parameter> <value> [IMMEDIATE | DEFERRED]
```

There are two modes that allow the execution of DB2 commands, Interactive command prompt and the DB2 CLP window:

- Interactive command prompt:

  You can begin DB2 interactive mode by issuing **db2** from any command line. This opens the interactive CLP. You will see the following prompt, *db2 =>*. To execute a DB2 command, enter the command at the prompt. For example, to modify DBM CFG parameters, enter **update dbm cfg using maxagents 200 deferred**.

- DB2 CLP window:

  To enter commands from a DB2 CLP window, first issue **db2cmd** at any command prompt. This will open a new window entitled *DB2 CLP*. Since this interface is outside the interactive mode, you must preface all commands with **db2** — for example: **db2 'update dbm cfg using maxagents 200'**.

Most of the parameters can be changed online. With **IMMEDIATE**, which is the default option, all parameters that are changeable online will take effect right away. With **DEFERRED**, you can defer all the changes until the next restart of the database manager.

To list the database manager configuration parameters, use the command:

GET DBM CFG [SHOW DETAIL]

The SHOW DETAIL option allows you to see also the deferred values.

Example 5-1 shows the parameter list of an SAP system with DB2 UDB V8.2.2.

Example 5-1 Database manager configuring parameters

```
Database Manager Configuration

     Node type = Enterprise Server Edition with local and remote clients

 Database manager configuration release level            = 0x0a00

 CPU speed (millisec/instruction)             (CPUSPEED) = 5.943666e-07
 Communications bandwidth (MB/sec)      (COMM_BANDWIDTH) = 1.000000e+02

 Max number of concurrently active databases     (NUMDB) = 8
 Data Links support                          (DATALINKS) = NO
 Federated Database System Support           (FEDERATED) = NO
 Transaction processor monitor name        (TP_MON_NAME) =

 Default charge-back account          (DFT_ACCOUNT_STR) =

 Java Development Kit installation path       (JDK_PATH) = /usr/java14_64

 Diagnostic error capture level              (DIAGLEVEL) = 3
 Notify Level                               (NOTIFYLEVEL) = 3
 Diagnostic data directory path               (DIAGPATH) = /db2/TDR/db2dump

 Default database monitor switches
   Buffer pool                           (DFT_MON_BUFPOOL) = ON
   Lock                                     (DFT_MON_LOCK) = ON
   Sort                                     (DFT_MON_SORT) = ON
   Statement                                (DFT_MON_STMT) = ON
   Table                                   (DFT_MON_TABLE) = ON
   Timestamp                             (DFT_MON_TIMESTAMP) = ON
   Unit of work                              (DFT_MON_UOW) = ON
 Monitor health of instance and databases   (HEALTH_MON) = ON

 SYSADM group name                           (SYSADM_GROUP) = DBTDRADM
 SYSCTRL group name                         (SYSCTRL_GROUP) = DBTDRCTL
 SYSMAINT group name                       (SYSMAINT_GROUP) = DBTDRMNT
 SYSMON group name                           (SYSMON_GROUP) =

 Client Userid-Password Plugin          (CLNT_PW_PLUGIN) =
 Client Kerberos Plugin                (CLNT_KRB_PLUGIN) =
 Group Plugin                             (GROUP_PLUGIN) =
 GSS Plugin for Local Authorization    (LOCAL_GSSPLUGIN) =
 Server Plugin Mode                    (SRV_PLUGIN_MODE) = UNFENCED
 Server List of GSS Plugins      (SRVCON_GSSPLUGIN_LIST) =
 Server Userid-Password Plugin        (SRVCON_PW_PLUGIN) =
 Server Connection Authentication          (SRVCON_AUTH) = NOT_SPECIFIED
 Database manager authentication        (AUTHENTICATION) = SERVER_ENCRYPT
 Cataloging allowed without authority   (CATALOG_NOAUTH) = NO
 Trust all clients                      (TRUST_ALLCLNTS) = YES
 Trusted client authentication          (TRUST_CLNTAUTH) = CLIENT
 Bypass federated authentication            (FED_NOAUTH) = NO

 Default database path                       (DFTDBPATH) = /db2/db2tdr

 Database monitor heap size (4KB)          (MON_HEAP_SZ) = 128
```

```
Java Virtual Machine heap size (4KB)     (JAVA_HEAP_SZ) = 2048
Audit buffer size (4KB)                  (AUDIT_BUF_SZ) = 0
Size of instance shared memory (4KB)  (INSTANCE_MEMORY) = AUTOMATIC
Backup buffer default size (4KB)            (BACKBUFSZ) = 1024
Restore buffer default size (4KB)           (RESTBUFSZ) = 1024

Sort heap threshold (4KB)                  (SHEAPTHRES) = 20000

Directory cache support                     (DIR_CACHE) = NO

Application support layer heap size (4KB)   (ASLHEAPSZ) = 16
Max requester I/O block size (bytes)         (RQRIOBLK) = 65000
Query heap size (4KB)                   (QUERY_HEAP_SZ) = 2000

Workload impact by throttled utilities(UTIL_IMPACT_LIM) = 10

Priority of agents                            (AGENTPRI) = SYSTEM
Max number of existing agents                (MAXAGENTS) = 1024
Agent pool size                         (NUM_POOLAGENTS) = 10
Initial number of agents in pool        (NUM_INITAGENTS) = 5
Max number of coordinating agents     (MAX_COORDAGENTS) = (MAXAGENTS - NUM_INITAGENTS)
Max no. of concurrent coordinating agents  (MAXCAGENTS) = MAX_COORDAGENTS
Max number of client connections      (MAX_CONNECTIONS) = MAX_COORDAGENTS

Keep fenced process                         (KEEPFENCED) = YES
Number of pooled fenced processes          (FENCED_POOL) = MAX_COORDAGENTS
Initial number of fenced processes     (NUM_INITFENCED) = 0

Index re-creation time and redo index build  (INDEXREC) = RESTART

Transaction manager database name         (TM_DATABASE) = 1ST_CONN
Transaction resync interval (sec)     (RESYNC_INTERVAL) = 180

SPM name                                      (SPM_NAME) =
SPM log size                            (SPM_LOG_FILE_SZ) = 256
SPM resync agent limit                   (SPM_MAX_RESYNC) = 20
SPM log path                               (SPM_LOG_PATH) =

TCP/IP Service name                           (SVCENAME) = sapdb2TDR
Discovery mode                                (DISCOVER) = SEARCH
Discover server instance                 (DISCOVER_INST) = ENABLE

Maximum query degree of parallelism   (MAX_QUERYDEGREE) = 1
Enable intra-partition parallelism     (INTRA_PARALLEL) = NO

No. of int. communication buffers(4KB)(FCM_NUM_BUFFERS) = 4096
Number of FCM request blocks               (FCM_NUM_RQB) = AUTOMATIC
Number of FCM connection entries      (FCM_NUM_CONNECT) = AUTOMATIC
Number of FCM message anchors         (FCM_NUM_ANCHORS) = AUTOMATIC

Node connection elapse time (sec)         (CONN_ELAPSE) = 10
Max number of node connection retries (MAX_CONNRETRIES) = 5
Max time difference between nodes (min) (MAX_TIME_DIFF) = 60

db2start/db2stop timeout (min)        (START_STOP_TIME) = 10
```

5.2.1 SAP recommended database manager configuration settings

Table 5-1 shows the SAP recommendations for DB2 UDB database manager settings. These recommendations are also in SAP Note *584952*. Here we provide a short description for each parameter. Refer to the IBM DB2 UDB publication: *Administration Guide: Performance V8*, SC09-4821-01, for details.

In that publication, Table 41 lists all the configurable database manager configuration parameters and shows if the parameters can be updated online.

Table 5-1 SAP recommended database manager configuration settings

Parameter	Value	Description
MAXTOTFILOP	16000	This parameter is removed starting FP7; for maximum total file handles, DB2 UDB allowed this parameter. Use the recommended setting. Since it is configured automatically starting FP9, it is removed from the configuration.
CPUSPEED	<value>	<*value*> is a generated value, set during installation or migration. After a hardware change it might make sense to have DB2 recompute this value. This can be done by setting it to -1. The cost based optimizer (CBO) uses this value to assess the cost of CPU cycles. During the generation of an SQL access plan, the possible plans are enumerated and weighted according to DB2 UDB's cost calculation model. Assume that you have a slower system, having a lower CPUSPEED parameter. Then, the plans generated by the CBO will be more conservative in using CPU cycles. It is a good practice to have this parameter set to -1, whenever upgrading the CPUs on your database server. Then, DB2 will run a small evaluation – a few cycles only – of your system performance and will update CPUSPEED accordingly. This parameter does not play a role in database clients.
NUMDB	8	For SAP systems, the number of databases in a DB2 UDB instance is exactly ONE. On older SAP Releases, using the legacy log file management, a second *Admin* database was created and used as well in the same instance.
DATALINKS	NO	SAP does not use datalinks, therefore leave this at NO.
FEDERATED	NO	SAP does not use federated databases.

Parameter	Value	Description
TP_MON_NAME	<empty>	This parameter is for the use of a transaction monitor. Since SAP has a built-in transaction monitor, leave this empty.
DFT_ACCOUNT_STR	<empty>	This parameter is used, when using a transaction monitor.
JDK_PATH	<value>	This value is OS-dependent. When using JDK™ stored procedures, this parameter is used. SAP does not use this parameter.
DIAGLEVEL	3	The amount of diagnostic information provided in the DB2 diagnostic log file can range from mute (0) to verbose (4). SAP's Support expects this to be set to 3, and ongoing communication between SAP and IBM Labs ensures that all necessary details are contained at this trace level.All errors will be registered, but no informational data written. Having too much write activity onto db2diag.log as with level 4 and many events per second, will harm system performance.
NOTIFYLEVEL	3	The notification log is new to DB2 UDB V8 and contains administrative notifications. Level 3 is SAP's recommendation.
DIAGPATH	<value>	On UNIX and Linux platforms enter /db2/<DBSID>/db2dump; On Windows Platforms enter <DRIVE>:\db2\<DBSID>\db2dump The SAP CCMS expects the db2diag.log file to be at above mentioned path. Do not change this. But you may want to prune the db2diag.log file by calling db2diag -A command. This will create an archive of the diagnostic log file. In addition, DB2 will create a new diagnostic log file.
DFT_MON_BUFPOOL	ON	This parameter must be switched on for SAP CCMS to work properly. There is no use in disabling the buffer pool monitor switches, the performance gain will be very small. In turn, SAP users will loose the ability to tune their system.

Parameter	Value	Description
DFT_MON_LOCK	ON	These parameters must be set to on.
DFT_MON_SORT	ON	
DFT_MON_STMT	ON	
DFT_MON_TABLE	ON	
DFT_MON_TIMESTAMP	ON	
DFT_MON_UOW	ON	
HEALTH_MON	OFF	SAP currently does not use the DB2 Health Monitor. Since system resource requirements have to be considered in SAP systems, disable all unnecessary activity, as with this parameter.
SYSADM_GROUP	DB<DBSID>ADM	The group name for the DB2 UDB super users is defined here by SAPInst. Do not change this without need. In fact, never change this unless you perform a system copy. The DB2 instance user is a member of this group, with a user ID db2<*dbsid*>.
SYSCTRL_GROUP	DB<DBSID>CTL	The group name for the DB2 control users is defined here by SAPInst. Do not change this without need. The OS user sap<*sid*> is a member of this group.
SYSMAINT_GROUP	<value>	<*value*> is <empty> for basis releases < 6.40 <*value*> is DB<DBSID>MNT for basis releases >= 6.40. With the new RUNSTATS implementation of SAP systems, the SAP connect user is calling RUNSTATS via API. Therefore, sap<*sid*> or sapr3 must be member of the new system maintenance group.Older SAP releases call RUNSTATS via external command, which connects to the database with the <*sapsid*>adm user id.
AUTHENTICATION	SERVER_ENCRYPT	Authentication to the DB2 UDB server is performed in encrypted passwords.
CATALOG_NOAUTH	NO	Cataloging of databases and nodes should not be allowed for all users. Do not change this setting.

Parameter	Value	Description
TRUST_ALLCLNTS	YES	This parameter does not play a role, since AUTHENTICATION is set to SERVER_ENCRYPT. With AUTHENTICATON=CLIENT, this would play a role.
TRUST_CLNTAUTH	CLIENT	This parameter does not play a role, since AUTHENTICATION is SERVER_ENCRYPT.
FED_NOAUTH	NO	This parameter must be set to NO, since it would allow unauthenticated access to the database server.
DFTDBPATH	<value>	On UNIX and Linux platforms, enter /db2/<DBSID> On Windows platforms, enter <DRIVE>: This is the path for the SAP database on storage.
MON_HEAP_SZ	128	The SAP recommended monitor heap is 128 pages.
JAVA_HEAP_SZ	2048	This parameter should not be changed, but SAP does not use the JAVA interpreter through DB2 UDB.
AUDIT_BUF_SZ	0	SAP does not utilize the DB2 UDB's database audit facility. The use of this facility has not been tested.
INSTANCE_MEMORY	AUTOMATIC	Memory management for the database instance is performed AUTOMATIC.
BACKBUFSZ	1024	The recommendation is only an initial value, load-dependent tuning might be required for this parameter. With DB2 UDB V 8.2.2, this parameter is deprecated.
RESTBUFSZ	1024	The recommendation is only an initial value, load-dependent tuning might be required for this parameter. With DB2 UDB V 8.2.2, this parameter is deprecated.
AGENT_STACK_SZ	64	This parameter only exists on Windows Platforms. SAP does not recommend to change this parameter.
MIN_PRIV_MEM	32	This parameter only exists on Windows Platforms. SAP does not recommend to change this parameter.

Parameter	Value	Description
PRIV_MEM_THRESH	1296	This parameter only exists on Windows Platforms. SAP does not recommend to change this parameter.
SHEAPTHRES	20000	32-bit: `#wp x sortheap` (max 60000) 64-bit: `#wp x sortheap` The recommendation is only an initial value, load-dependent tuning might be required for this parameter. Since SAP uses hash-joins in SQL plans, an SAP system must be enabled to allow `SORTHEAP` to be used for all database connections. Otherwise SQL Error `SQL0955` will be raised for work processes. Constraining the sort heap threshold is OK on 32-bit systems.
DIR_CACHE	NO	SAP does not use LDAP, therefore no caching is necessary.
ASLHEAPSZ	16	This is the size of the shared memory area used for communication between the SAP work processes, CLI routines, and the db2agent EDU. SAP is traditionally running on a very small `ASLHEAPSZ`. Do not change this tested setting without need.
RQRIOBLK	65000	Communication blocks of 65000 bytes are large enough for SAP's blocked transfer between client and server. This setting has been tested by SAP for more than 10 years and should not be changed without need
QUERY_HEAP_SZ	2000	The recommendation is only an initial value, load-dependent; tuning might be required for this parameter. The query heap must be increased for more complex SQL statements. You will do this as soon as SQL error codes tell you to do so
UTIL_IMPACT_LIM	10	This is to limit the performance impact of a throttled utility to make sure that SAP operations are not impacted. Most of the DB2 UDB utilities invoked by SAP are still called via older APIs which do not use the `UTIL_IMPACT_PRIORITY`. However, for DB2 UDB's new Automatic RUNSTATS, the reduced priority is used.

Parameter	Value	Description
AGENTPRI	SYSTEM	DB2 EDUs are run at normal operating system priority. Do not change this value.
MAXAGENTS	1024	The maximum number of DB2 agents allowed for the instance. This value must be adjusted for larger systems.
NUM_POOLAGENTS	10	For agents left in the agent pool (initialized, available agents), 10 is a good setting.SAP work processes maintain a permanent connection to their DB2 Agent counterpart, so SAP does not disconnect from database agents. Therefore, the pool will not be used in normal situations.
NUM_INITAGENTS	5	At the point of database activation, Five DB2 agents will be started. This parameter does not have too much of an impact.
MAX_COORDAGENTS	MAXAGENTS	For OLTP Systems, Intra-Query Parallelism and multi-partitioning is not used. Therefore this parameter must be set to MAXAGENTS.
MAXCAGENTS	MAX_COORDAGENTS	This is the parameter restricting the number of concurrently executing agents. Since, in SAP environments, applications are expecting the database agents to provide instant results, this parameter is set to MAX_COORDAGENTS, which translates in to MAXAGENTS. This means, all agents are allowed to execute in parallel.
MAX_CONNECTIONS	MAX_COORDAGENTS	Here, you are able to restrict the number of database connections. It should translate into MAX_COORDAGENTS, which is MAXAGENTS at the end.
KEEPFENCED	YES	Fenced stored procedures are used by the monitoring environment of SAP. When a stored procedure is loaded, it will be kept allocated in memory when setting this parameter to YES. It is clear that this facilitates memory management and reduced subsequent calling overhead.
FENCED_POOL	MAX_COORDAGENTS	All database connections should be able to call stored procedures. Therefore it is set to MAX_COORDAGENTS.

Parameter	Value	Description
NUM_INITFENCED	0	Since fenced stored procedures are called by the SAP system instantly after start, it is not necessary to initialize the processors for fenced stored procedures. Keep it set to 0.
INDEXREC	RESTART	If indexes have been invalidated by DB2 UDB, they will be recreated at database restart time. This ensures that there is no delays during subsequent SQL processing.
TM_DATABASE	1ST_CONN	SAP has its own transaction manager. So, keep the default here.
RESYNC_INTERVAL	180	This is for configuration of syncpoint managers. SAP does not allow you to run a SPM, therefore leave the default at 180.
SPM_NAME	<empty>	SAP does not use syncpoint managers. Please leave these parameters as recommended.
SPM_LOG_FILE_SZ	256	
SPM_MAX_RESYNC	20	
SPM_LOG_PATH	<empty>	
SVCENAME	sapdb2 <*DBSID*>	This entry must be maintained in the *services* file. This is the name of the service which is then referenced as numerical TCP port in the /etc/services file on UNIX systems and win_installdir>/system32/drivers/etc/services on Windows.
DISCOVER	SEARCH	For database clients, it is possible to discover DB2 databases and their properties. Since some users may want to use the DB2 UDB Control Center (CC), leave this at SEARCH to allow the discovery by the CC.
DISCOVER_INST	ENABLE	For database clients, it is also possible to discover DB2 instances and their properties, as described above. You might want to allow this by setting this parameter to ENABLE.
MAX_QUERYDEGREE	1	For normal SAP OLTP systems, the degree of parallelism for queries is 1. This should not be changed, and is even disregarded, because of the parameter INTRA_PARALLEL is set to NO.

Parameter	Value	Description
INTRA_PARALLEL	NO	For SAP's OLTP systems, intra-query-parallelism is not suggested and also not tested. You may want to consider allowing this with SAP BW systems based on the number of CPUs available to your database or database partition.
FCM_NUM_BUFFERS	4096	Initial settings. The fast communication manager is not used in SAP's OLTP systems, therefore leave the default settings. For SAP OLAP systems, these initial settings should be used. Only change them if your knowledge about multi-partitioned databases allow you to assess the FCM performance or if you are asked to do so by SAP.
FCM_NUM_RQB	AUTOMATIC	
FCM_NUM_CONNECT	AUTOMATIC	
FCM_NUM_ANCHORS	AUTOMATIC	
CONN_ELAPSE	10	These two parameters are for communication management between database partitions in a partitioned DB2 database. With this setting, after 5 times 10 seconds (50 seconds), an error is returned if a TCP/IP connection cannot be established.
MAX_CONNRETRIES	5	
MAX_TIME_DIFF	60	With this setting, the maximum time difference allowed between partitions of a partitioned DB2 database is 60 minutes. This should be changed to 1 in SAP's partitioned OLAP databases, but is OK in a single-partitioned OLTP database.
START_STOP_TIME	10	When starting or stopping a partitioned database, all partitions should be started or stopped within 10 minutes. 10 is an acceptable value, since restart recovery will be covered within this value.

5.3 DB2 UDB database configuration

A recommended way to manage the configuration of your DB2 UDB server is via the SAP GUI using transaction ST04. The DB2 UDB command to set the database configuration parameters is:

```
UPDATE DB CFG FOR <database name> USING <parameter> <value> [IMMEDIATE | DEFERRED]
```

The UPDATE DB CFG command can be entered via interactive mode or CLP command options.

Most of the parameters can be changed online. With **IMMEDIATE**, which is the default option, all parameters that are changeable online will take effect once it is changed. With **DEFERRED,** you can defer all the changes until the database is reactivated.

You can see the database configuration parameters using the following commands:

GET DB CFG FOR <database name> [**SHOW DETAIL]**

The SHOW DETAIL option allows you to see all the deferred values.

Example 5-2 shows the actual and delayed settings in our SAP database TDR.

Example 5-2 Listing database configuration details

```
db2 get db cfg for tdr show detail

       Database Configuration for Database tdr

 Description                                 Parameter   Current Value       Delayed Value
 ------------------------------------------------------------------------------------------
 Database configuration release level                    = 0x0a00
 Database release level                                  = 0x0a00

 Database territory                                      = en_US
 Database code page                                      = 819
 Database code set                                       = ISO8859-1
 Database country/region code                            = 1
 Database collating sequence                             = BINARY                      BINARY
 Alternate collating sequence                 (ALT_COLLATE) =
 Database page size                                      = 16384                       16384

 Dynamic SQL Query management              (DYN_QUERY_MGMT) = DISABLE                     DISABLE

 Discovery support for this database          (DISCOVER_DB) = ENABLE                      ENABLE

 Default query optimization class            (DFT_QUERYOPT) = 5                          5
 Degree of parallelism                         (DFT_DEGREE) = 1                          1
 Continue upon arithmetic exceptions      (DFT_SQLMATHWARN) = NO                         NO
 Default refresh age                      (DFT_REFRESH_AGE) = 0                          0
 Default maintained table types for opt   (DFT_MTTB_TYPES) = SYSTEM                     SYSTEM
 Number of frequent values retained        (NUM_FREQVALUES) = 10                         10
 Number of quantiles retained               (NUM_QUANTILES) = 20                         20

 Backup pending                                          = NO

 Database is consistent                                  = YES
 Rollforward pending                                     = NO
 Restore pending                                         = NO

 Multi-page file allocation enabled                      = YES

 Log retain for recovery status                          = NO
 User exit for logging status                            = NO
```

```
 Data Links Token Expiry Interval (sec)      (DL_EXPINT) = 60                        60
 Data Links Write Token Init Expiry Intvl(DL_WT_IEXPINT) = 60                        60
 Data Links Number of Copies             (DL_NUM_COPIES) = 1                         1
 Data Links Time after Drop (days)        (DL_TIME_DROP) = 1                         1
 Data Links Token in Uppercase                (DL_UPPER) = NO                        NO
 Data Links Token Algorithm                   (DL_TOKEN) = MAC0                      MAC0

 Database heap (4KB)                           (DBHEAP) = 25000                     25000
 Size of database shared memory (4KB)  (DATABASE_MEMORY) = AUTOMATIC(116964)
AUTOMATIC(116964)
 Catalog cache size (4KB)              (CATALOGCACHE_SZ) = 2560                    2560
 Log buffer size (4KB)                        (LOGBUFSZ) = 1024                    1024
 Utilities heap size (4KB)               (UTIL_HEAP_SZ) = 10000                   10000
 Buffer pool size (pages)                     (BUFFPAGE) = 10000                   10000
 Extended storage segments size (4KB)    (ESTORE_SEG_SZ) = 16000                   16000
 Number of extended storage segments   (NUM_ESTORE_SEGS) = 0                       0
 Max storage for lock list (4KB)              (LOCKLIST) = 20000                   20000

 Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 128000                  128000
 Percent of mem for appl. group heap   (GROUPHEAP_RATIO) = 25                      25
 Max appl. control heap size (4KB)     (APP_CTL_HEAP_SZ) = 1600                    1600

 Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = (SHEAPTHRES)            (SHEAPTHRES)
 Sort list heap (4KB)                         (SORTHEAP) = 2048                    2048
 SQL statement heap (4KB)                     (STMTHEAP) = 5120                    5120
 Default application heap (4KB)            (APPLHEAPSZ) = 3072                    3072
 Package cache size (4KB)                  (PCKCACHESZ) = 5120                    5120
 Statistics heap size (4KB)              (STAT_HEAP_SZ) = 12000                   12000

 Interval for checking deadlock (ms)         (DLCHKTIME) = 300000                  300000
 Percent. of lock lists per application       (MAXLOCKS) = 90                      90
 Lock timeout (sec)                        (LOCKTIMEOUT) = 3600                    3600

 Changed pages threshold                  (CHNGPGS_THRESH) = 40                    40
 Number of asynchronous page cleaners    (NUM_IOCLEANERS) = 6                     6
 Number of I/O servers                    (NUM_IOSERVERS) = 6                     6
 Index sort flag                             (INDEXSORT) = YES                    YES
 Sequential detect flag                      (SEQDETECT) = YES                    YES
 Default prefetch size (pages)         (DFT_PREFETCH_SZ) = 32                    32

 Track modified pages                        (TRACKMOD) = YES                    YES

 Default number of containers                           = 1                     1
 Default tablespace extentsize (pages)    (DFT_EXTENT_SZ) = 2                     2

 Max number of active applications           (MAXAPPLS) = AUTOMATIC(40)          AUTOMATIC(40)
 Average number of active applications       (AVG_APPLS) = 30                    30
 Max DB files open per application            (MAXFILOP) = 1950                  1950

 Log file size (4KB)                         (LOGFILSIZ) = 16380                 16380
 Number of primary log files                (LOGPRIMARY) = 20                    20
 Number of secondary log files               (LOGSECOND) = 40                    40
 Changed path to log files                  (NEWLOGPATH) =
 Path to log files                                       = /shr/TDR/log_dir/NODE0000/
/shr/TDR/log_dir/NODE0000/
 Overflow log path                     (OVERFLOWLOGPATH) =
 Mirror log path                         (MIRRORLOGPATH) =
 First active log file                                   =
 Block log on disk full                (BLK_LOG_DSK_FUL) = YES                   YES
 Percent of max active log space by transaction(MAX_LOG) = 0                     0
 Num. of active log files for 1 active UOW(NUM_LOG_SPAN) = 0                     0
 Group commit count                           (MINCOMMIT) = 1                     1
 Percent log file reclaimed before soft chckpt (SOFTMAX) = 300                   300
 Log retain for recovery enabled             (LOGRETAIN) = OFF                   OFF
 User exit for logging enabled                (USEREXIT) = OFF                   OFF

 HADR database role                                      = STANDARD              STANDARD
```

```
 HADR local host name                  (HADR_LOCAL_HOST) =
 HADR local service name                (HADR_LOCAL_SVC) =
 HADR remote host name                (HADR_REMOTE_HOST) =
 HADR remote service name              (HADR_REMOTE_SVC) =
 HADR instance name of remote server  (HADR_REMOTE_INST) =
 HADR timeout value                       (HADR_TIMEOUT) = 120                        120
 HADR log write synchronization mode     (HADR_SYNCMODE) = NEARSYNC                   NEARSYNC

 First log archive method                (LOGARCHMETH1) = OFF                        OFF
 Options for logarchmeth1                 (LOGARCHOPT1) =
 Second log archive method               (LOGARCHMETH2) = OFF                        OFF
 Options for logarchmeth2                 (LOGARCHOPT2) =
 Failover log archive path               (FAILARCHPATH) =
 Number of log archive retries on error   (NUMARCHRETRY) = 5                          5
 Log archive retry Delay (secs)         (ARCHRETRYDELAY) = 20                         20
 Vendor options                              (VENDOROPT) =
 Auto restart enabled                     (AUTORESTART) = ON                         ON
 Index re-creation time and redo index build  (INDEXREC) = SYSTEM (RESTART)          SYSTEM
(RESTART)
 Log pages during index build           (LOGINDEXBUILD) = OFF                        OFF
 Default number of loadrec sessions    (DFT_LOADREC_SES) = 1                          1
 Number of database backups to retain   (NUM_DB_BACKUPS) = 12                         12
 Recovery history retention (days)     (REC_HIS_RETENTN) = 60                         60

 TSM management class                    (TSM_MGMTCLASS) =
 TSM node name                            (TSM_NODENAME) =
 TSM owner                                   (TSM_OWNER) =
 TSM password                             (TSM_PASSWORD) =

 Automatic maintenance                      (AUTO_MAINT) = ON                         ON
   Automatic database backup             (AUTO_DB_BACKUP) = OFF                        OFF
   Automatic table maintenance           (AUTO_TBL_MAINT) = ON                         ON
     Automatic runstats                   (AUTO_RUNSTATS) = ON                         ON
     Automatic statistics profiling      (AUTO_STATS_PROF) = OFF                        OFF
       Automatic profile updates          (AUTO_PROF_UPD) = OFF                        OFF
     Automatic reorganization               (AUTO_REORG) = OFF                        OFF
```

5.3.1 SAP recommended database configuration

Table 5-2 below shows the SAP recommended DB2 UDB database configuration settings for databases in the SAP environment. We provide a short description for each parameter. You should evaluate your system carefully before making configuration changes. Refer to the IBM DB2 UDB publication: *Administration Guide: Performance V8*, SC09-4821-01 for details. Table 43 in that publication lists all the configurable database configuration parameters and shows if the parameters can be updated online.

Table 5-2 SAP recommended DB2 UDB database configurations

Parameter	Value	Description
DYN_QUERY_MGMT	DISABLE	The DB2 Query Patroller should not be used in SAP environments. It has not been tested up to now and SAP's support will not cover problems submitted on this software component.
DISCOVER_DB	ENABLE	Since database discovery is not used in SAP environments, it is not critical to alter this parameter. But please be aware that the DB2 UDB Control Center GUI uses the database discovery. Therefore, SAP's recommendation is to leave it enabled.
DFT_QUERYOPT	5	Since query optimization is set on both the connection level and individual statement level by SAP work processes, this parameter does not play a role in SAP environments. If the standard optimization level for DB2's Cost Based Optimizer needs to changed, this must be done in the SAP default profile.
DFT_DEGREE	1	This parameter ensures that intra-partition parallelism is not used in "normal" SAP environments. The overhead of plan execution for multiple agents for SAP's SQL statements is considered to be too high. Some considerations for BW systems in SMP systems apply.
DFT_SQLMATHWARN	NO	SAP applications expect SQL Errors from databases for conversion errors and mathematical exceptions. Therefore this parameter must be set to NO to ensure that error situations are reported as errors and not as warnings.
DFT_REFRESH_AGE	0	During normal operations, DB2 material query tables (MQT) are not used. Therefore this parameter must be set to zero.
DFT_MTTB_TYPES	SYSTEM	The CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION of DB2 MQTs is not used and is therefore set to the default value SYSTEM.

Parameter	Value	Description
NUM_FREQVALUES	10	Distribution Statistics will be collected during RUNSTATS … WITH DISTRIBUTION invocation. This is not used in OLTP environments. Therefore SAP recommends to set this parameter to the default value.
NUM_QUANTILES	20	The default value is used, since SAP does not use distribution statistics in OLTP environments. Quantiles are used to understand the distribution of values within a column
DL_EXPINT	60	DB2 Data Links are not used in SAP environments and SAP therefore asks customers to leave these values on the default settings.
DL_WT_IEXPINT	60	
DL_NUM_COPIES	1	
DL_TIME_DROP	1	
DL_UPPER	NO	
DL_TOKEN	MACO	
DBHEAP	<value>	*<value>* is 25000 for all basis releases >= 4.7 AND NOT HP-UX; 35000 on HP-UX. *<value>* is 12000 for all other cases. The above recommendation is only an initial value, load-dependent. Tuning might be required for this parameter. The database heap contains control block data structures for many database objects, like buffer pools, tables, and indexes. With growing requirements from SAP Releases, this value increases frequently. If DB2 UDB error messages report the database heap to be too small, you may increase this value at your own discretion.
DATABASE_MEMORY	AUTOMATIC	DB2 Shared Memory size is calculated automatically when having this parameter set, based on the heaps and memories contained within. Since important memory areas like LOCKLIST and buffer pools are contained within, SAP recommends to have AUTOMATIC set.

Parameter	Value	Description
CATALOGCACHE_SZ	2560	The recommendation is only an initial value, load-dependent. Tuning might be required for this parameter. For SQL plan generation, the DB2 Cost Based Optimizer uses the catalog cache in DB2 UDB which provides a packed description of DB2 tables, views, and indexes. Using this compiled representation is more efficient than accessing the DB2 System Catalog. SAP recommends to monitor the catalog cache hit ratio in ST04 and increase this parameter if needed.
LOGBUFSZ	1024	This DB2 parameter defines the size of the log buffer memory which is used to store log records during transaction processing until they are committed. It used to be complicated to tune this parameter for optimal value – but the rule “bigger is better” can be observed. With SAP NetWeaver 2004s, the new ST04 logging strip in the performance section shows if your log buffer was configured too small, which forces premature flushing of log records to log file.
UTIL_HEAP_SZ	<value>	<*value*> = 1000 for DB2 UDB V8 FP7 and higher. <*value*> = 10000 for older FixPaks. When using DB2 utilities BACKUP and RESTORE, buffers will be allocated which reside in the “utility heap”. Since the utilities are also DB2 UDB applications, they compete for system resource with all other applications. It is a good practice to set this value to the maximum calculated memory consumption of the utility buffer. With this, denial-of-service attacks are not possible through inappropriate utility usage.

Parameter	Value	Description
BUFFPAGE	10000	The use of this parameter should be superseded by the ALTER BUFFERPOOL SQL Statement (and ST04), see SAP Note *99509.* This value is sizing dependent, a well tuned system should have a buffer pool hit ratio > 96%. The BUFFPAGE parameter is not used in modern SAP systems, since SAPInst is already setting the individual buffer pool sizes. It is clear that buffer pool size is one of the most important settings to look for in an SAP database server. With buffer pools being too small, your page caching will not be sufficient and excessive I/O is the consequence.
ESTORE_SEG_SZ	16000	Do not use ESTORE on SAP systems. These two parameters have never been used in SAP systems and as well not tested for use. They have been implemented as workaround in 32-bit systems, but SAP's database servers are shipped as 64-bit software. For Windows-based 32-bit systems, AWE may be a short-term consideration.
NUM_ESTORE_SEGS	0	
LOCKLIST	<value>	<*value*> >= 10000 on 32-bit systems. <*value*> >= 20000 on 64-bit systems. The above recommendations are only initial values, load-dependent, tuning might be required for this parameter. Database locks are maintained in-memory using the LOCKLIST. If this list is configured too small, you will get notification through ST04 and should increase this dynamic parameter immediately. With modern 64-bit architecture, there are no constraints on LOCKLIST sizes and therefore you should increase this parameter freely to the above SAP recommended value.
APPGROUP_MEM_SZ	<value>	<*value*> = 40000 for 32-bit systems. <*value*> = 128000 for 64-bit systems. These three parameters should be set as recommended by SAP. Aapplication groups (agents working on the same SQL task) will not play a role in your SAP systems unless you are using BW-style systems.
GROUPHEAP_RATIO	25	
APP_CTL_HEAP_SZ	1600	

Parameter	Value	Description
SHEAPTHRES_SHR	SHEAPTHRES	Shared sorts will not happen on SAP systems, until you use intra-partition parallelism. The recommended setting should be used.
SORTHEAP	2048	The recommendation is only an initial value, load-dependent; tuning might be required for this parameter. This parameter describes the maximum number of pages allocated to sort operations within a db2agent process. Since this is a per-agent allocation, conservative changes must be considered, since each increase must be multiplied times the number of agents or work processes.
STMTHEAP	<value>	<*value*> = 4096 for 32-bit systems. <*value*> = 5120 for 64-bit systems. The above recommendation is only an initial value, load-dependent; tuning might be required for this parameter. A statement heap is allocated whenever SQL statements are compiled within the db2agent process. The recommended settings are based on customer feedback, but you may need to increase this parameter for your system, as soon as you receive a message that the statement to compile is too complex.
APPLHEAPSZ	3072	When executing compiled DB2 statements (sections), db2agents are accessing these statements via a locally cached version which is copied from database global statement cache. These local copies are stored in the application heap. In DB2 UDB releases prior V8.1, SAP recommended setting this parameter to the size of the statement cache, but DB2 UDB V8-based systems have proven to be able to run on smaller application heap sizes. This application heap is allocated per db2agent and should therefore only carefully be increased.

Parameter	Value	Description
PCKCACHESZ	5120	Increase if quality of package cache is below 98%. The global memory area for caching prepared SQL statements (packages / sections) is the package cache. DB2 Packages were designed to be very compact to be used in DB2's shared nothing architecture. If your package cache is too large (30000 pages), ST04 will have problems in displaying the cache contents.
STAT_HEAP_SZ	12000	RUNSTATS uses the statistics heap during the collection process. It is de-allocated after the program has finished. Please use the SAP recommended settings. If SQL codes point to a statistics heap which is too small, you may gradually increase this parameter by increments of 1000 pages.
DLCHKTIME	300000	With the recommended deadlock check time (in milliseconds), every 5 minutes DB2 UDB's deadlock detector is triggered to ensure that deadlocked transactions are resolved. The deadlock monitor in ST04 will show you deadlock history and diagnostics data. It is a good practice to follow up on these cases for problem resolution.
MAXLOCKS	90	After 90% of the locklist is used, DB2 UDB will search though the locklist to find the table with most locks allocated. It then releases all the row-level locks after setting a single table level lock of same or equivalent type for this table. This will free up most of the space in the locklist but may reduce concurrency considerably. These lock escalations must be followed-up with and should not happen.
LOCKTIMEOUT	3600	A lock timeout should never happen on DB2 UDB systems, since locks are only acquired for very short time periods. Still, if an erroneous connection maintains locks for too long, after 60 minutes (3600 seconds), the transaction will be rolled back with an error message SQL0911. You should leave this SAP recommendation as it is.

Parameter	Value	Description
CHNGPGS_THRESH	40	The recommendation is only an initial value, load-dependent; tuning might be required for this parameter. This means, after 40% of the pages in the buffer pool are changed, DB2 page cleaners are started to collect modified pages in buffer pool. They are then copied to containers, which is persistent storage. Having a lower CHNGPGS_THRESH value means a slightly higher system load for page cleaner queue management and higher I/O. With a higher value, your page cleaners may take longer, since more changed pages will be written after the page cleaners have been triggered. Also, soft checkpoints will take longer.
NUM_IOCLEANERS	6	Set this to the number of physical devices up to a maximum of 20. With a higher number of I/O cleaners, SMP systems may be more efficient in page cleaning. Having asynchronous I/O in place, the number of page cleaners is not a major critical performance factor. SAP's experiences show that having more than 20 page cleaners will not improve performance (even on largest SMP).
NUM_IOSERVERS	6	Set this to the number of physical devices up to a maximum of 20. For prefetching, I/O servers will work prefetch requests provided by db2agents. When having a high capacity I/O subsystem and a larger SMP server, you may want to increase this parameter. Especially in OLAP systems where prefetching is seen more frequently. Otherwise leave it as recommended above.
INDEXSORT	YES	Always leave this parameter set to YES. It ensures, that index keys are sorted first, before an index tree is built. With NO, index trees are built incrementally during index creations which can be slow at times.

Parameter	Value	Description
SEQDETECT	YES	This parameter enables the sequential detection in the page read patterns of the database server. SAP has always left this parameter in the default setting YES. Although well tuned OLTP transactions should not read too much data anyway, and therefore not perform prefetching in a large scale, it is good practice to leave this enabled.
DFT_PREFETCH_SZ	<value>	<value> = 32 with Fixpaks < FP7; <value> = AUTOMATIC with Fixpaks >= FP7. When placing prefetch requests onto the queues of the I/O servers, the default size of the requests is defined here. Since DB2 UDB V8 architecture is that IOSIZE= PREFETCHSIZE/EXTENTSIZE, and multiple requests of IOSIZE should be dispatched to the prefetcher queues, the calculation should always factor in the number of containers. The enhancement of V8.2 was that this is performed automatically on per-table-space basis – which is much more detailed. Except if PREFETCHSIZE is set to a value during table space creation. Our recommendation: just leave this at AUTOMATIC, which is SAP's default. Since SAP is also creating each table space with the AUTOMATIC setting for PREFETCH, all will be calculated correctly.
TRACKMOD	ON	Setting this parameter to ON enables your database for incremental backup. On the other hand, it introduces a slight complication: you *must* first back up a table space whenever it is created. This can be done with a full DB backup or a table space backup. If you do not plan to run incremental backups, then you may also leave this OFF.
DFT_EXTENT_SZ	<empty>	Leave this parameter empty. It does not play a role, since SAP's SAPInst tool will always designate a specific extent size for a table space. Also, with ST04 it is not possible to create a table space without a specific extent size.

Parameter	Value	Description
MAXAPPLS	AUTOMATIC	This recommendation is for V8.2.2 Systems, since it only defines an upper bound. There is a security aspect to this: For SAP systems in a well guarded environment, it should be explicitly set to ensure that there is no thread of over-eager connection creation which might force a database server into paging.
AVG_APPLS	30	As one of the more occult parameters in DB2 UDB, AVG_APPLS is responsible for a certain aspect of cost based optimization: memory consumption. With a higher value, the CBO creates plans which consume less pages in buffer pool and prefer more I/O. With a buffer pool hit ratio of more than 95% in a typical SAP system, 30 is a good setting. In BW situations, queries will consume more buffer pool. Here, the value is reduced.
MAXFILOP	1950	For SMS-based databases, the maximum number of open files (containers, etc.) is defined here. Since SAP used DMS – based databases, please leave this parameter as it is.
LOGFILSIZ	16380	The above recommendation is only an initial value, load-dependent tuning might be required for this parameter. DBA wisdom says, that large log files are better. This is not important on DB2 UDB servers, since log file switches will not trigger I/O storms during checkpointing. A log file size of 64 MB, as above, is reasonable for most systems. New log files are not created, but renamed from old log files, so file allocation overhead does not happen. Nevertheless, you might want to double the size of log files in larger systems (> 1TB) and reduce the number of log files.

Parameter	Value	Description
LOGPRIMARY	20	The recommendation is only an initial value, load-dependent, tuning might be required for this parameter. This is the number of pre-allocated log files, your SAP database is running on. It should be as much as the typical transaction workload of your SAP system will keep as open log files. Having more than one open log files is the case, when a transaction becomes a long-running transaction, because it holds the oldest open log file and other transactions continue to use newer log files.
LOGSECOND	40	The recommendation is only an initial value, load-dependent, tuning might be required for this parameter. Secondary log files are created, when all primary log files are in use. In fact, in SAP environments a system should not use secondary log files because of the ongoing file allocation overhead.
NEWLOGPATH		You should not have to use this parameter, since moving the log path will deviate your system from the standard SAP layout requirements.
OVERFLOWLOGPATH		The parameter should be set to `/db2/<DBSID>/log_retrieve`. Here, log files are stored from tape or storage management system during rollforward operations. For details on Dual Logging, please refer to SAP Note *409127*.
MIRRORLOGPATH		When using Dual Logging, you will set this parameter to `/db2/<DBSID>/log_dir2`. Dual Logging is not mandatory, since `/db2/<DBSID>/log_dir` should be set up on RAID-1 or -5 devices. Nevertheless, this feature helps to prevent operator errors and can be a real life-safer during the “lost logfile” human intervention error syndrome.
BLK_LOG_DSK_FUL	YES	This will ensure “archiver stuck” behavior, if `log_archive` is full. This parameter must be set to yes. It ensures, that the DB2 UDB Log Manager – when unable to continue writing into log files – stops log processing until space is made available.

Parameter	Value	Description
MAX_LOG	0	You may adjust these two above parameters to their needs. SAP's recommendation is to use 0 here. These parameters have become available in DB2 UDB V8.2 and can be used to define the maximum number of log files, and the maximum percentage of total active log space a single transaction is allowed to span. You might want to have this set to a more restrictive setting in QA or development environments, to catch transactions which are not committing frequently enough.
NUM_LOG_SPAN	0	
MINCOMMIT	1	See SAP Note *371317.* When having high write activity to log files, you may want to increment this parameter by one or two. Then, commit statements by multiple SAP work processes will be batched until `MINCOMMIT` is reached – or be written after one second. We do not recommend to change this parameter, unless high transactional activity is continuously seen in the SAP system. During periods of low activity, commits below this value will have to wait for up to one second. This is not good for SAP transactions.
SOFTMAX	300	The recommendation is only an initial value, load-dependent, tuning might be required for this parameter. After 300% of one DB2 log file have been used by the DB2 logger, a soft checkpoint will be performed, moving modified pages from buffer pool(s) to DB2 containers. This is a good start, but you may want to change this. More aggressive page cleaning can be obtained with values of 50, while substantial reduction in write activity on DB2 containers can be obtained with values of 1000. This in turn guarantees a maximum number of log files to redo during restart recovery between ½ log file and 10 log files. It is a customer decision between performance during operations and recovery.

Parameter	Value	Description
LOGRETAIN	RECOVERY	Never run an SAP production system with this parameter on OFF. If you do not have this set, there is no way you can apply log files during database recovery. SAP will test for this parameter.
USEREXIT	<value>	<value> = ON is required when using legacy log file management <value> = OFF when using the new log file management For HADR settings, please refer to SAP Note *780546.* For LOGARCHMETH settings, please read SAP Database Administration Guide *IBM DB2 Universal Database for UNIX and Windows - New Log File Management.* This parameter ensures that either the "legacy" log file manager db2uext2 is used. Otherwise, in combination with LOGARCHMETH settings, DB2's modern log file management architecture is used.
AUTORESTART	ON	SAP wants this setting to be ON. It means, that DB2 UDB will automatically perform redo of all transactions in the log files during restart recovery, also known as crash recovery. At the end of this process, DB2 UDB will undo all open, uncommitted, transaction and the database is made available again. If you set this parameter to NO, you must issue the RESTART DATABASE command, which is not included in SAP's start scripts.
INDEXREC	RESTART	Indexes which have been invalidated will be recreated during database restart. RESTART is the recommended SAP setting, since any corrupted indexes will be rebuilt after database restart. In our experience, rarely indexes break just "like this".
DFT_LOADREC_SES	1	This parameter is used during recovery of table load sessions. This parameter does not play a role in SAP environments.
NUM_DB_BACKUPS	12	In the DB2 recovery history file, DB2 UDB will mark all backups older then the last as "expired". This setting is important when using the DB2 Tape Manager or TSM.

Parameter	Value	Description
REC_HIS_RETENTN	60	Set TSM parameters only if using TSM, otherwise leave empty. This parameter defines the number of days to keep backup history for. Having NUM_DB_BACKUPS at 12, you may even have more recovery information than 6 days. This parameter should be left empty, or adjusted to work with the NUM_DB_BACKUPS parameter.
AUTO_MAINT	<value>	Parent parameter for all new V8.2.2 automatic operations. Switch it ON on SAP SAP NetWeaver 2004, and newer releases. Otherwise OFF. For older SAP releases, SAP Note *584952* will be updated over time.
AUTO_DB_BACKUP	OFF	SAP does not want uncontrolled backup activities.
AUTO_TBL_MAINT	<value>	Automatic table maintenance has been tested for SAP NetWeaver 2004, older releases will be enabled via updates in SAP Note *584952*.
AUTO_RUNSTATS	<value>	Automatic RUNSTATS has been tested for SAP NetWeaver 2004, older releases will be enabled via updates in SAP Note *584952*. This functionality allows to remove a lot of scheduling work in SAP Transaction DB13 – database planning calendar.
AUTO_STATS_PROF	OFF	SAP does not support automatic statistics profiling.
AUTO_PROF_UPD	OFF	As above, this parameter plays a role in statistics profiling and must not be used in SAP environments.
AUTO_REORG	OFF	<*value*> = ON with FP >= FP7 and SAP Basis > 6.40 <*value*> = OFF otherwise until further notice from SAP. AUTO_REORG is performed using the traditional offline REORG – which is very fast. SAP discourages customers to use this feature for SAP systems, because concurrency of the SAP system will suffer.

5.4 DB2 profile registry

It is very easy to set the DB2 registry on new DB2 V8.2.2 systems, as you will see in the following sections. For this, we reference SAP's Note *582952* again:

5.4.1 DB2 registry variables and recommended settings

You can list the DB2 registry variable with the command:

```
db2set -all
```

Please note that the output format may be different when displaying the registry via ST04.

Example 5-3 shows the db2 registry variables in an SAP system.

Example 5-3 DB2 registry variables in an SAP system

```
> db2set -all
[e] DB2CHKPTR=OFF
[e] DB2CODEPAGE=819
[e] DB2COMM=TCPIP
[e] DB2COUNTRY=1
[e] DB2DBDFT=GR5
[i] DB2_WORKLOAD=SAP
[i] DB2_MDC_ROLLOUT=YES [DB2_WORKLOAD]
[i] DB2_SKIPINSERTED=YES [DB2_WORKLOAD]
[i] DB2_VIEW_REOPT_VALUES=YES [DB2_WORKLOAD]
[i] DB2_OBJECT_TABLE_ENTRIES=65532 [DB2_WORKLOAD]
[i] DB2_OPTPROFILE=YES [DB2_WORKLOAD]
[i] DB2_IMPLICIT_UNICODE=YES [DB2_WORKLOAD]
[i] DB2_INLIST_TO_NLJN=YES [DB2_WORKLOAD]
[i] DB2_MINIMIZE_LISTPREFETCH=YES [DB2_WORKLOAD]
[i] DB2_UPDATE_PART_KEY=YES [DB2_WORKLOAD]
[i] DB2_REDUCED_OPTIMIZATION=4,INDEX,JOIN [DB2_WORKLOAD]
[i] DB2NOTIFYVERBOSE=YES [DB2_WORKLOAD]
[i] DB2_INTERESTING_KEYS=YES [DB2_WORKLOAD]
[i] DB2_EVALUNCOMMITTED=YES_DEFERISCANFETCH [DB2_WORKLOAD]
[i] DB2_ANTIJOIN=EXTEND [DB2_WORKLOAD]
[i] DB2_SKIPDELETED=YES [DB2_WORKLOAD]
[i] DB2_HASH_JOIN=ON
[i] DB2MEMMAXFREE=2000000 [DB2_WORKLOAD]
[i] DB2ENVLIST=INSTHOME SAPSYSTEMNAME dbs_db6_schema DIR_LIBRARY LIBPATH
[i] DB2_RR_TO_RS=YES [DB2_WORKLOAD]
[i] DB2_FORCE_FCM_BP=YES [DB2_WORKLOAD]
[i] DB2COUNTRY=1
[i] DB2COMM=TCPIP
[i] DB2CODEPAGE=819
[g] DB2_EEE_LICENSE_POLICY=1407379178586112
```

```
[g] DB2SYSTEM=ibbd0033
[g] DB2ADMINSERVER=dasusr1
```

To alter the registry variable, use the command:

```
db2set <PARAMETER>=<VALUE>
```

For example:

```
db2set db2_workload=SAP
```

To delete existing settings, login as user db2<*dbsid*> and use the command:

```
db2set <PARAMETER>=
```

Table 5-3 shows the SAP recommended DB2 registry variable settings. We provide a short description for each variable.

Important: For DB2 V8 FP9 and higher, only set:

```
DB2_WORKLOAD=SAP
```

When using DB2_WORKLOAD=SAP, do not set other registry variables unless required by SAP support through SAP Note message.

This new aggregate registry variable must be set to SAP. If this is the case, a predefined set of registry variables is made active. Ensure that the output of db2set -all shows the aggregate name DB2_WORKLOAD for the registry variables displayed in Table 5-3.

Table 5-3 SAP recommended DB2 registry variable settings

Registry variable	Value	Description
DB2_RR_TO_RS	ON	For DB2 UDB with FixPaks lower than FixPak 9, it Is important to set this variable, if ype-1 indexes still exist (after migration to DB2 Version 8), but there is no negative impact if it is used with Type-2 indexes. As SAP systems must operate in RS isolation level, it is always a good idea to have this registry variable set to ON. This is, unless you are absolutely sure that there are no Type-1 indexes left in your system. This is the case if your system has a pure V8-legacy. So, V7 legacy customers, beware.

Registry variable	Value	Description
DB2_REDUCED_OPTIMIZATION	4, INDEX, JOIN	SAP systems must be set to 4 in this registry variable. This will ensure that the default SQL optimization level 5 will be modified in a way that complex join scenarios will be optimized using greedy enumeration of possible join alternatives. Also, DB2 UDB index heuristics are used, as well as special join enhancements.
DB2_MINIMIZE_LISTPREFETCH	ON	Tables VBHDR, VBMOD, VBDATA must be marked volatile with this setup. DB2 UDB's cost based optimizer traditionally tries to optimize for IO. For table access via qualifying index, RIDs (row identifiers) are received from the index and sorted to allow for linear increasing page access. This makes good use of disk hardware, but with SAP's very high buffer pool hit ratios, this method is way too elaborate. If this registry variable is set to ON, the DB2 database will not sort the RIDs.
DB2_INLIST_TO_NLJN	ON	With SAP's heavy usage of in-lists, the DB2 CBO will need additional guidance to favour index-based access methods.If you set this registry variable to ON, the CBO rewrite transforms single in-lists into tables, which will form the inner table of a nested loop join.
DB2_ANTIJOIN	<value>	<*value*> = ON for Fixpak levels < 4d; <*value*> = EXTEND for Fixpak levels >= 4d; With this setting, NOT IN and NOT EXISTS clauses are transformed into "anti-joins".
DB2_SKIPDELETED	ON	This allows SQL statements to skip pseudo-deleted index keys during index access and deleted rows during table access, but uncommitted pseudo-deleted keys in Type-2 indexes are not unless DB2_SKIPDELETED is also enabled.

Registry variable	Value	Description
DB2_IMPLICIT_UNICODE	ON	Only necessary if you want to install SAP Web AS - J2EE Add-In in a non-Unicode database.
DB2_EVALUNCOMMITTED	ON	Available starting at Fixpak 4d. Since DB2_EVALUNCOMMITTED must be set to ON, deleted rows are automatically skipped during predicate evaluation.
DB2_INTERESTING_KEYS	ON	Available starting at Fixpak 4d. This registry variable enables an alternate approach for the optimizer to perform key analysis.
DB2_PARALLEL_IO	*	Only set this variable, if the following conditions are met: 1) RAID devices are used for the database. 2) All table spaces in the database consist of only a single container. 3) The extent size of the table spaces is equal to the RAID stripe size. 4) DB2_STRIPED_CONTAINERS was set to ON, before the table spaces were created. Practically, you should set this variable to *. It helps DB2 UDB to run multiple prefetchers on requests towards a single table space.
DB2MEMMAXFREE	2000000	When 2 MB of unused private memory is reached in a DB2 agent, memory management will return this unused memory to the OS. Below this threshold, no deallocation will take place.
DB2_OBJECT_TABLE_ENTRIES	65532	Starting with FP7, details are in SAP Note 780546. To be able to run online backups with a large number of objects, you must bump-up this variable to the mentioned value. Otherwise, DB2 UDB table spaces are created with an object table which is not large enough for the latest SAP releases and must be extended into higher extents of the table spaces, thus being blocked during backup.

6

SAP database administration

This chapter describes how to use DBA Cockpit for your system. DBA Cockpit is one of the SAP transactions that you can use from SAPGUI to administrate your database system. Learning and using DBA Cockpit is one way to make your database administration easy. For more detailed information on DBA Cockpit, we recommend that you refer to SAP online help:

http://help.sap.com

6.1 Architectural overview

DBA Cockpit supports an extensive collection of database administration functionalities. You can monitor both local systems and remote systems. To support this flexibility, DBA Cockpit is separated, by design, into the Front End Layer and the Back End Layer, as shown in Figure 6-1.

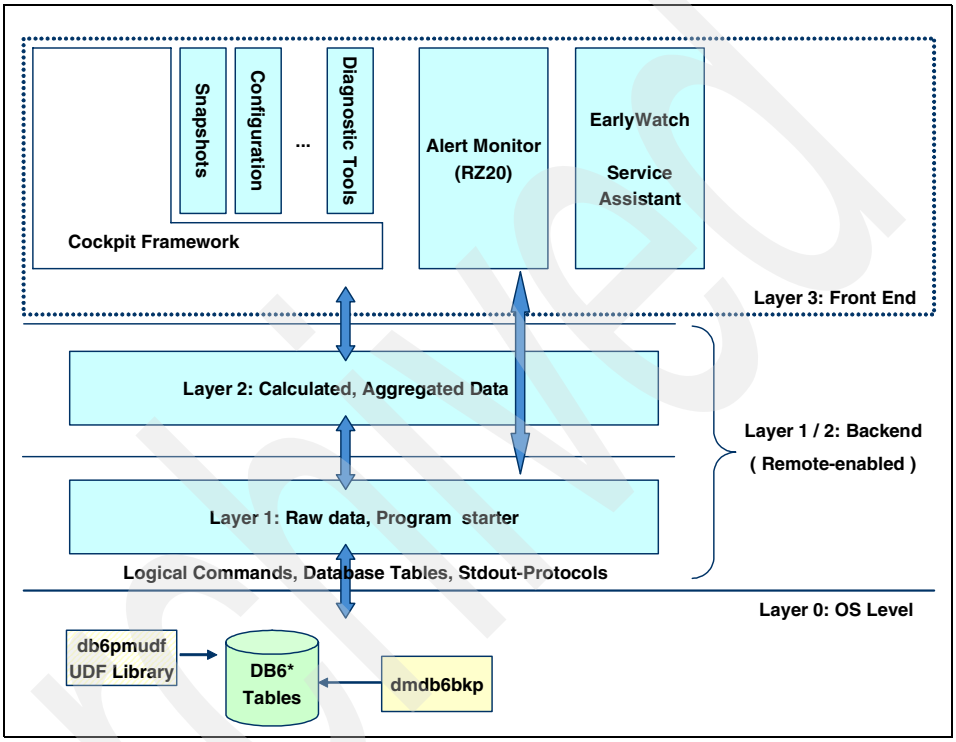


Figure 6-1 Architectural view of DBA Cockpit

- Front End Layer:

  One of the most important consumer conveniences in this layer is the graphical element of DBA Cockpit. It prints the screen and the screen elements like buttons and lists, and attaches event handlers to these screen elements to deal with actions such as double-clicking a row. When there is a need to gather data from the database, a function call to the Back End Layer is made, and the Front End Layer waits for result sets to come back from the call and populates or updates the corresponding list or text box. The Front End Layer also performs checks on the input data from users to see if the data is valid and lists any output errors if needed.

- Back End Layer:

  This layer is responsible for gathering raw data from the database on behalf of the Front End Layer, as well as manipulation of data such as aggregations and calculations. It can service function calls from local SAP system as well as the remote SAP system.

  In Figure 6-1 on page 178, we see that the Cockpit Framework, or the graphical element of DBA Cockpit, is not the sole consumer of the Back End Layer. The services provided by the Back End Layer can also be used by other modules such as *Alert Monitor* or *Early Watch TCC Service Assistant*. This decoupling of the Front End and the Back End Layers makes remote system monitoring possible.

  The `db6pmudf` UDF (DB2 UDB User Defined Function) library provides some operating system functionalities (for example, listing the content of a directory, viewing text file) to support features in DBA Cockpit such as navigating the directory structures and viewing db2diag.log file. The *dmdb6bkp* executable issues DB2 backup on behalf of DBA Cockpit.

6.2 Local and remote monitoring

You can monitor remote systems (remote SAP systems as well as remote non-SAP DB2 systems) using Remote Function Call (RFC) or remote database connection from DBA Cockpit. In this section, we describe the considerations for remote monitoring.

Note: The local system is automatically set up when DBA Cockpit is started by default.

6.2.1 Concept of remote monitoring

Because of the flexibility in DBA Cockpit architecture, you can use DBA Cockpit not only to monitor local system but also remote systems. There are two main methods to set up monitoring for remote system:

- **Remote Function Call (RFC):**

  This allows the Front End of DBA Cockpit of one SAP system to issue a remote function call to the Back End of DBA Cockpit of a remote SAP system. With the RFC configuration, you get almost the full functionality as if you are monitoring a local system.

You can configure DBA Cockpit of a SAP NetWeaver 2004s system using RFC to connect to another SAP ABAP system that is 4.6x or above. Refer to SAP Note *300828* for the details on the prerequisites. To configure a remote monitoring to an SAP release 3.1I - 4.5B system, you can only use a remote database connection. You cannot configure RFC to an SAP JAVA system.

- **Remote database connection:**

  This method uses a database connection to the DB2 UDB system. You need to choose a database user that has sufficient authority to accomplish the database task that you would like to do. (For example, if you choose to use <*sid*>adm, from the original system you will only be able to perform actions that the SYSCTRL authority is allowed to do, but not actions such as creating an event monitor, which is only available to a user belonging to SYSADM authority.)

  A remote database connection does not have access to the history tables of the remote system. Therefore, you cannot access all the functions in DBA Cockpit related to history (for example, **Space →History**). Remote database connection can be configured to monitor a target system that is an SAP JAVA System, an SAP ABAP system that is lower than SAP release 4.6x (that is, SAP release 3.1I - 4.5B), or even a DB2 UDB database that has no SAP system running on it.

Here is a list of the features that are available with DBA Cockpit for a remote system configured with RFC but NOT for remote system configured with Remote Data Connection:

- All the history related functions (for example, **Space → History**) and the corresponding *History Data configuration* (that is, **Configuration → Monitoring Settings → History Data**)
- File viewing or directory listing (for example, **Diagnostics → Database Diag Log, Diagnostics → DBSL Trace Directory**)
- File system space usage check (for example, **Configuration → File Systems**)
- **Diagnostics → Cumulative SQL Trace**
- **Diagnostics → Missing Tables and Indexes**
- **Diagnostics → Single System Check (Log Directory and Automatic Storage)**

Additionally, for SAP Java system or a DB2 UDB database that has no SAP system installed on top of it, the following function is not available:

- **Configuration → Data Classes**

Figure 6-2 illustrates all the possible target systems you can configure with remote monitoring from an SAP NetWeaver 04s ABAP system.

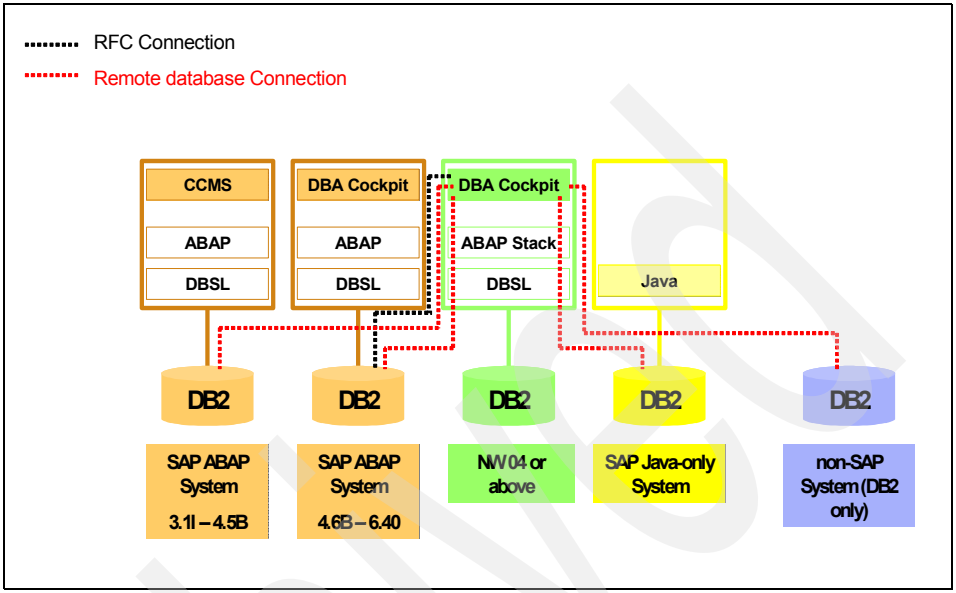


Figure 6-2 Remote monitoring from an SAP NetWeaver 2004s ABAP System

Tip: In the new Basis Support Package for your older SAP systems (for example, an SAP 4.6B system), there may only be an update on the new Back End (for example, a 6.20 system) but not a new Front End to DBA Cockpit of your system. To take advantage of the new features provided by the new Front End (that is, 6.20 in this example), you can configure remote monitoring with RFC from another system with a higher SAP release (for example, an SAP 6.20 system) to this system.

6.2.2 Configuration for remote system monitoring

Here we describe how to set up the remote monitoring.

Click **Configuration** in the navigation frame. You see the initial screen as Figure 6-3. The local system is automatically configured when DBA Cockpit is first executed.

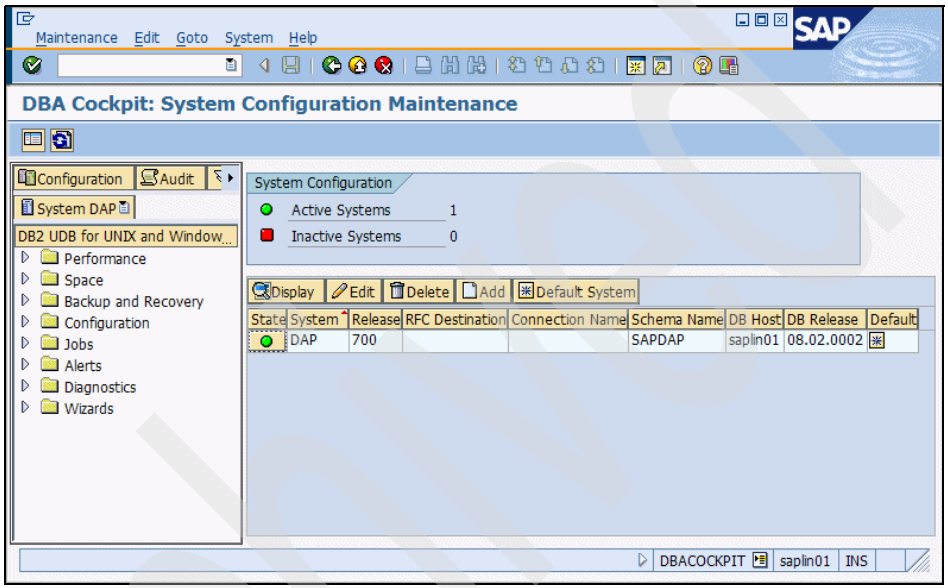


Figure 6-3 System Configuration

To add a remote system for monitoring, click **Add**. The Add System Entry screen comes up. See Figure 6-4.

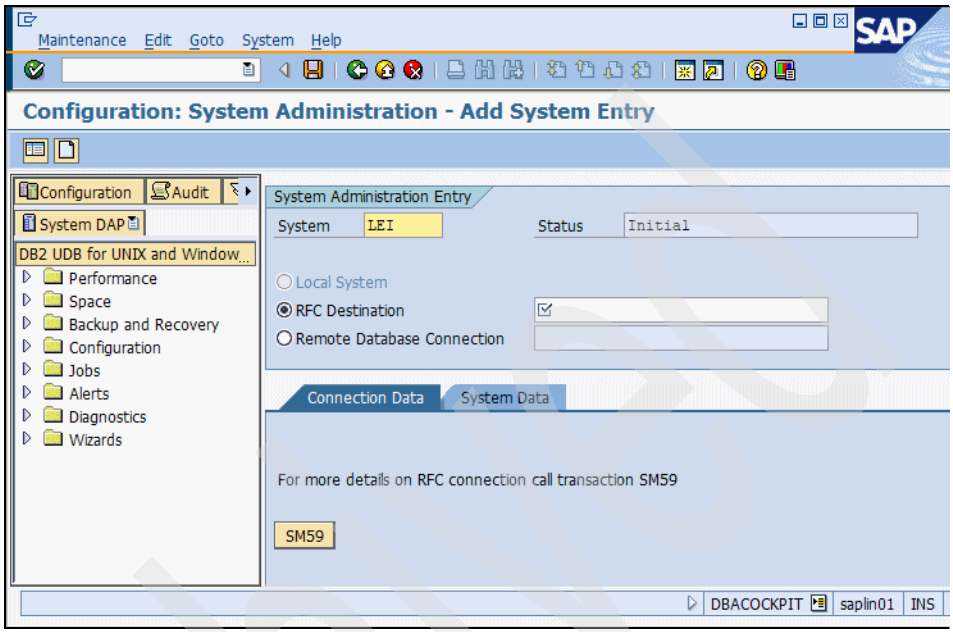


Figure 6-4 Add system entry

Using RFC

At first choose **RFC Destination**. If you already set up RFC destination for the remote system, enter RFC destination which you want to use and enter the name for this connection in System field. If you do not set up RFC destination yet, go to transaction SM59 by clicking **SM59** button in *Connection Data* tab. You see SM59 and click **Create** button. Then the screen (Figure 6-5) comes up. Enter the values for RFC destination name, connection type (ABAP connection), and target host name and so on. Then save the configuration. Go back to DBA Cockpit and set RFC destination name to the one you create. Then save the configuration.

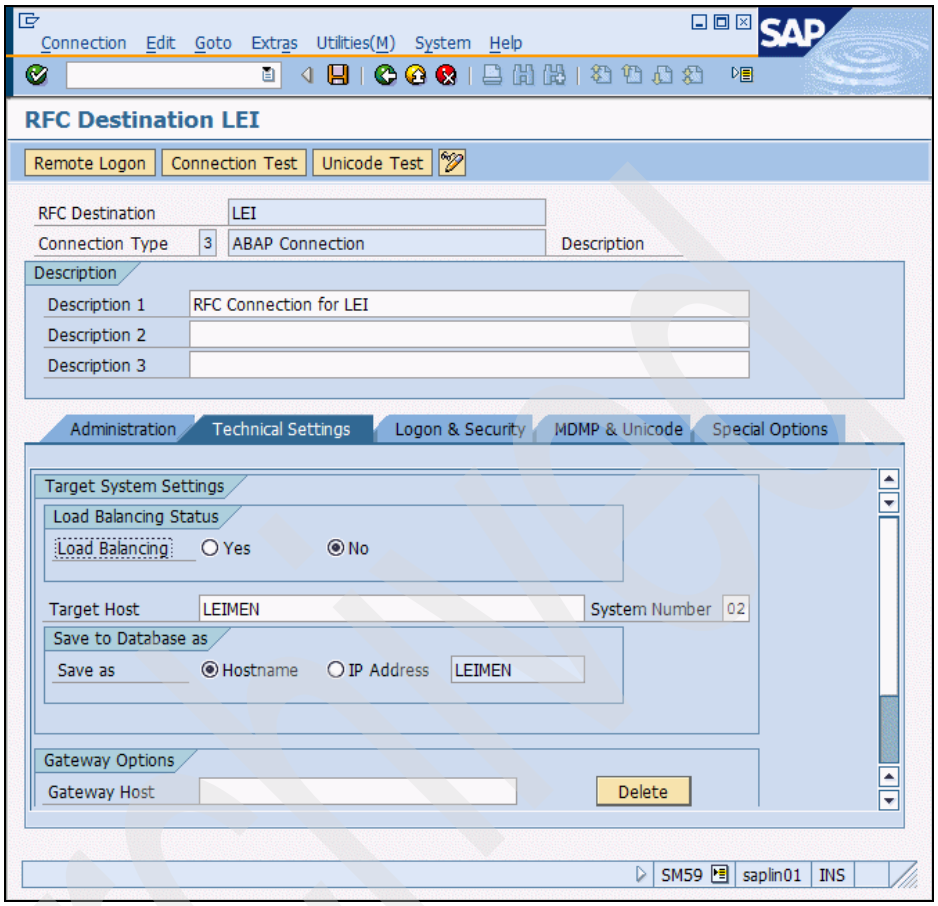


Figure 6-5 RFC Destination configuration

Using the remote database connection

Just as with RFC, you can set up your remote database connection either beforehand by transaction DBCO or set up the connection during remote database configuration in DBA Cockpit. Figure 6-6 shows the configuration for a remote database connection. You need to input user name, password, database name, service name, database host, and so on. After inputting these values, save the configuration.

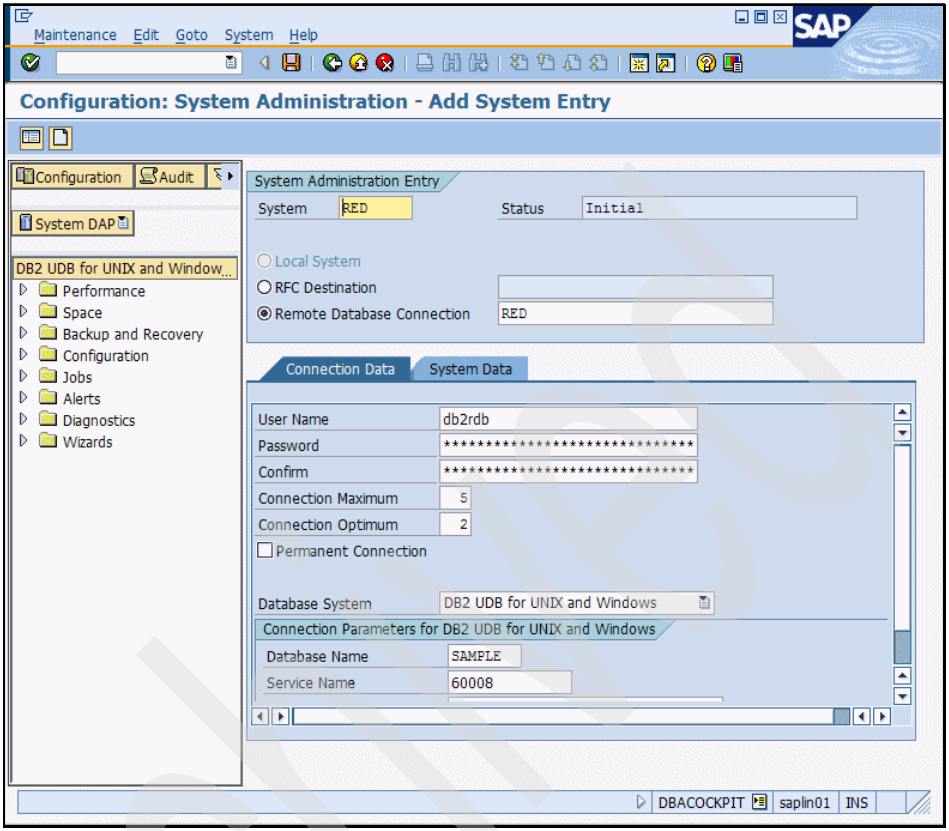


Figure 6-6 Configuring for remote database connection

6.3 Screen management

In this section, we describe common fields you see in most screens of DBA Cockpit. Then we describe buttons in the DBA Cockpit navigation frame. After that we describe how to configure basic settings for monitoring, as well as the authorization necessary to access DBA Cockpit, and to lock in DBA Cockpit.

6.3.1 Common fields

Figure 6-7 shows two common fields that are available in most of the task areas in DBA Cockpit.

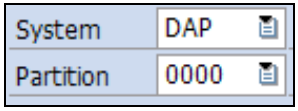


Figure 6-7 Common Fields for DBA Cockpit

- System:

  The system field displays the system name currently monitored. Click the list box in the System field and choose the system you want to monitor. You can also change the system by clicking **Selected System in the navigation frame**. If the system you want to monitor is not displayed, click **Configuration** in the navigation frame and check whether it is activated and properly configured. See Figure 6-7.

- Partition:

  This field displays the database partition (node) currently monitored. To change the partition to monitor or perform maintenance, click the list box and select a partition.

6.3.2 Buttons in the navigation frame

Here we describe the buttons in the navigation frame.

Configuration 

Click **Configuration** in the navigation frame to display and configure the systems to be monitored. The local system is automatically configured when you first execute DBA Cockpit. You can add, delete, and configure systems you want to monitor from this server. Figure 6-8 shows the System Configuration Maintenance screen. You can monitor not only local databases, but also remote databases (both SAP and non-SAP) from DBA Cockpit. This configuration function is called at first when you use transaction code `DBACOCKPIT` to DBA Cockpit.

For more information about configuring remote database monitoring, refer to 6.2, “Local and remote monitoring” on page 179.

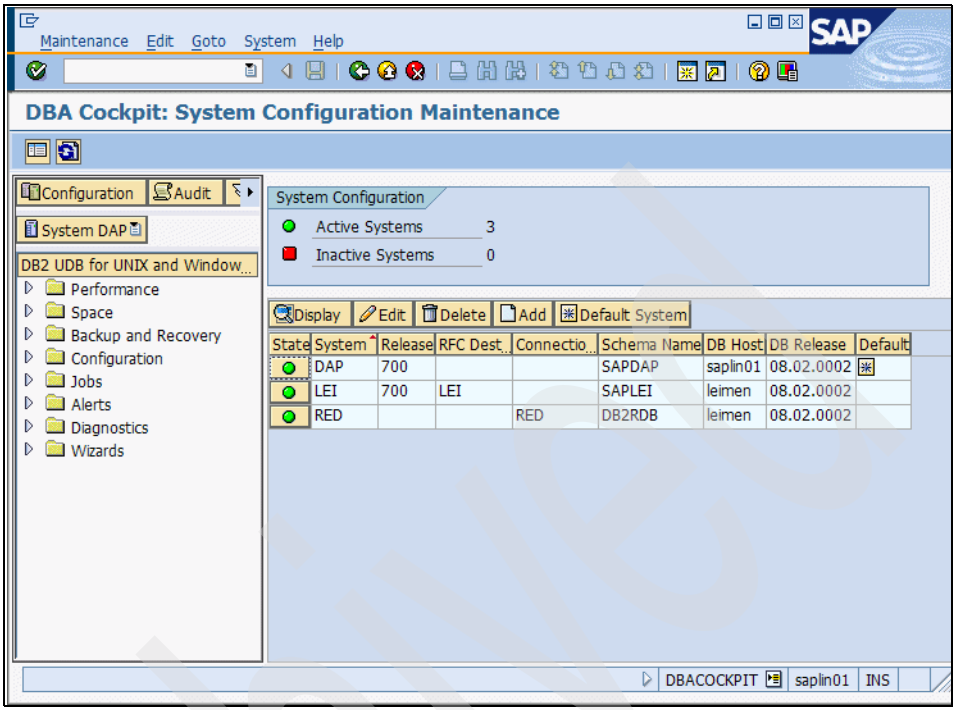
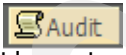


Figure 6-8 Configuration

Audit 

An audit log entry is written when you change your database systems by using DBA Cockpit, such as resizing your table space. Also, if a DB2 UDB command is executed via *CLP Command* under the *Configuration* task area or via *CLP Script Maintenance* under the *Jobs* task area in DBA Cockpit, an audit log entry is written. Click **Audit** in the navigation frame to see the audit log. To change the period displayed, double-click a day in the calendar. You can also change the Number of Days to be displayed. Audit entries created in the current week are displayed by default. To check the details of the action, select an entry and click the **Display** icon. Figure 6-9 shows the Audit Log screen.

Audit log entries are kept for 90 days.

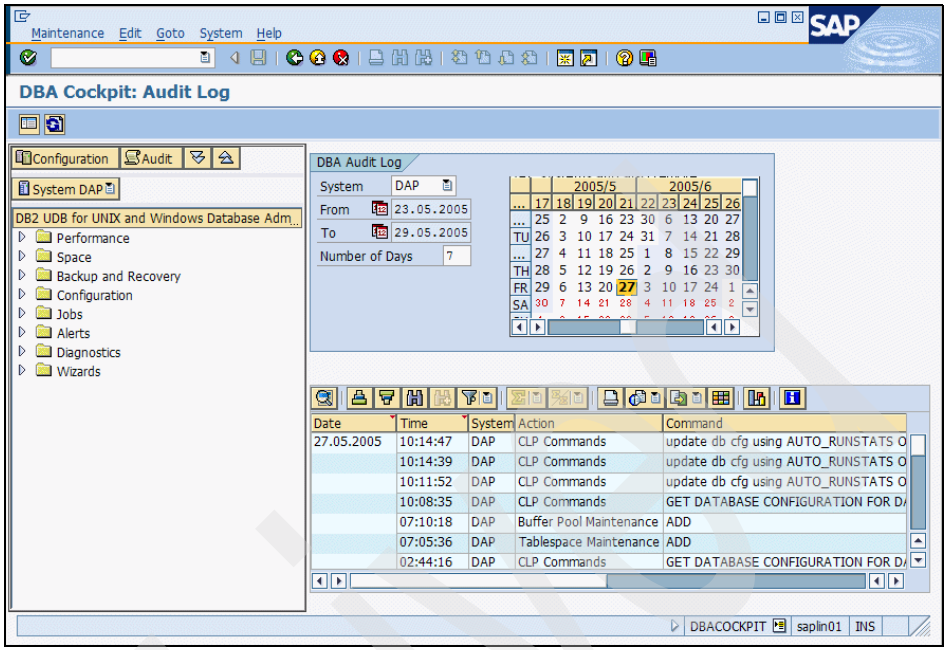


Figure 6-9 Audit Log

Expand and collapse tasks in a navigation tree

Click these buttons to expand or collapse the tasks in a navigation tree.

Full screen on/off

This button displays/hides the navigation frame. Please note that this button is not in the navigation frame, but it is almost always displayed above the navigation frame.

6.3.3 Monitoring settings

To change the basic settings of monitoring in DBA Cockpit, use *Monitoring Settings* in the *Configuration* task area. In this function, you can configure the following two settings:

- User defined functions (UDFs):

  The UDF that is necessary for DBA Cockpit is automatically set up during the first execution of DBA Cockpit for a system. You can check the location where the UDF is located and the version of UDF library `db6pmudf` from this tab. Also, you can change the path for the UDF. The path for UDF must contain SAP system ID in it or must be empty. If the path you input does not match, DBA Cockpit automatically changes the path to the path for SAP kernel. When you make the path field empty, DBA Cockpit assumes that the UDFs are in the DB2 UDF library path. Figure 6-10 shows the *UDF Configuration* tab.

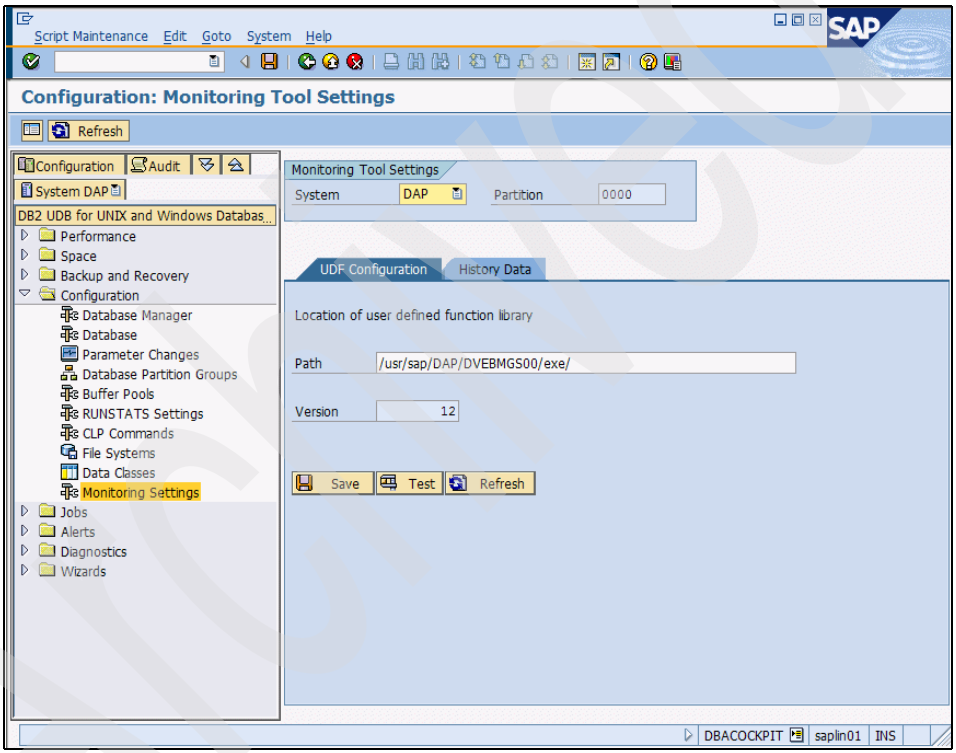


Figure 6-10 UDF configuration

- Retention period of history data for DBA Cockpit:

  You can change the retention period for history data such as database size or performance data from the *History Data* tab. Figure 6-11 shows the *History Data* tab in Monitoring Settings. Click **Reload** to get the last saved version of history data settings back.

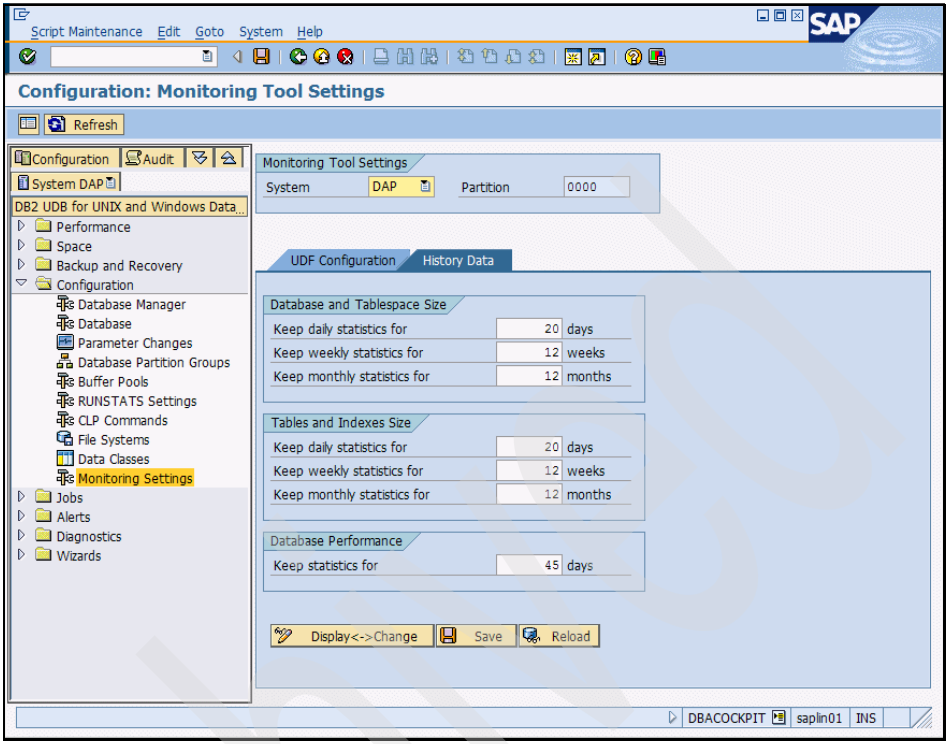


Figure 6-11 Example of history data configuration

6.3.4 Authorization and permission

You can use DBA Cockpit for not only monitoring your database, but also performing administration tasks of your database, such as changing database configuration parameters. By default, it is configured to be able to use DBA Cockpit, but it is also possible to disable DBA Cockpit. To disable DBA Cockpit, set the SAP profile parameter `dbs/db6/ccms_maintenance` to 0.

Although DBA Cockpit is enabled, you cannot use DBA Cockpit if you do not have an appropriate authorization. To use DBA Cockpit, the SAP user has to have the following authorization object and attribute in that authorization object:

- Authorization object: `S_RZL_ADM`
- Attribute: `ACTVT`

Also, some operating system users must have an appropriate authorization to perform database related tasks. Figure 6-1 shows which user performs the activities and what kind of database authorization the users need for local and remote monitoring based on RFC. For remote monitoring via remote database connections, the user specified in the remote database connection is used to perform database related activities. For SAP NetWeaver based systems, the user db2<*dbsid*> or <*sapsid*>adm has a sufficient authorization to perform administrative tasks. If only monitoring tasks are required, any user that has at least SYSMAINT authorization can be specified.

Table 6-1 Database authorization to perform database related tasks from DBA Cockpit

Tasks	User ID	Authorization
Administration	<sapsid>adm	SYSCTRL
Monitoring	sap<sapsid>	SYSMAINT

If you perform an upgrade from the previous version of SAP systems, you should check if sap<*sapsid*> user has SYSMAINT authorization, because sap<*sapsid*> does not have SYSMAINT authorization in the previous SAP systems, such as a system based on SAP Web Application Server release lower than 6.40.

To check the database authorization, use the **db2 get dbm cfg** command. The parameter "SYSCTRL_GROUP" should be set to the group to which <*sapsid*>adm user belongs. Also, the parameter SYSMAINT_GROUP should be set to the group to which the sap<*sapsid*> user belongs. Example 6-1 shows the database administrative authorization settings.

Example 6-1 Listing database administrative authorization

```
$ db2 get dbm cfg
..........
 SYSADM group name                       (SYSADM_GROUP) = DBLEIADM
 SYSCTRL group name                     (SYSCTRL_GROUP) = DBLEICTL
 SYSMAINT group name                   (SYSMAINT_GROUP) = DBLEIMNT
 SYSMON group name                       (SYSMON_GROUP) =
..........
```

6.3.5 Lock on DBA Cockpit

DBA Cockpit uses its own lock mechanism to avoid confliction of administrative tasks. For example, if user X is performing resizing a table space in DBA Cockpit, the other user Y cannot perform any table space maintenance until user X finishes maintenance. When you cannot perform maintenance due to the lock, an error message like the one shown in Figure 6-12 is displayed.

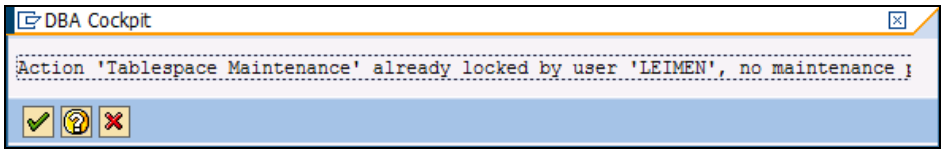


Figure 6-12 Error message for locking situation

6.4 Monitoring

This section describes the content of the *Performance* folder within DBA Cockpit, where you can monitor the whole database, schemas, buffer pools, table spaces, tables, applications, the SQL cache, lock waits, dead locks, and inplace table reorganizations.

Figure 6-13 shows the folder *Performance* within DBA Cockpit, where you can select the different tasks that you want to monitor. You can call DBA Cockpit via transaction `ST04` to access the *Performance* folder directly. In the following sections we give an overview for each of those tasks.

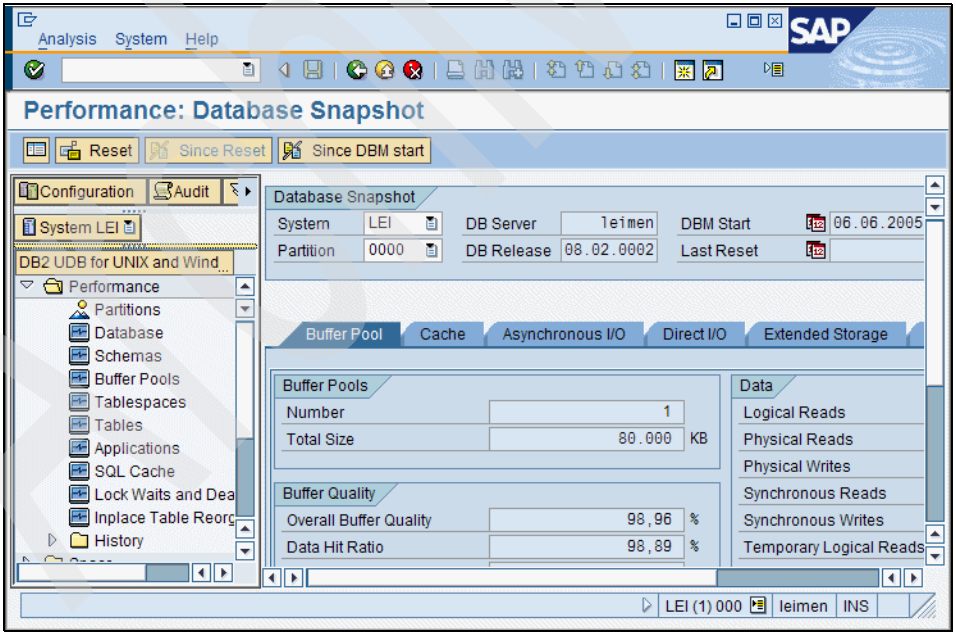
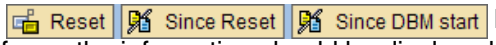


Figure 6-13 Performance monitoring in DBA Cockpit

With the 

buttons you are able to select the time frame the information should be displayed. With the Reset button, you can set a reset point, and the Since Reset button displays the values since the last reset. So it is possible to monitor a specific time frame, for example, during the execution of a job within SAP. The Since DBM start button displays all the information since the start of the database manager (DBM), which will normally be started along with the SAP system. With the Since Reset and Since DBM start buttons, you can also refresh the displayed values for the corresponding time frame.

More details can be found in the SAP documentation, *CCMS: SAP/DB2 UDB for UNIX and Windows, Release 670* from the SAPNet at:

`http://service.sap.com/instguides`

To access this Web site, you need an SAP Service Marketplace user ID and password.

Note: The pictures in the following sections do not show the folders on the left side, we used the `Full Screen`  button to maximize the right side of the screen to display as much information as possible.

6.4.1 Database monitoring

Within the database monitoring, you get a database activity overview in the area of buffer pools, the cache, the asynchronous I/O, the direct I/O, the extended storage, the locks and deadlocks, the logging, the calls, and the sorts.

More details on the monitoring of the database buffer pools, cache, locks and deadlocks, and sorts will be discussed in Chapter 8, in “ST04: key performance indicators” on page 474.

The data for the database monitoring mainly comes from the DB2 UDB command `GET SNAPSHOT FOR DATABASE ON <sapsid>`. The values returned by this snapshot are grouped by subject and displayed on various tabs. Based on the returned values, some hit ratios are also calculated and displayed.

Buffer Pool

On the Buffer Pool tab you get an overview of the activity of all the buffer pools within a database, as shown in Figure 6-14. This tab also shows the number of buffer pools and their total size. The buffer pools provide storage for data and index pages which works as cache for the database to improve the database system performance.

Assuming that your SAP/DB2 system has been up and running with a typical workload for some time, you can have a quick look at the overall buffer quality, data, and index hit ratio. The recommended values are:

`Overall buffer quality`	This value should should be above 96%.
`Data hit ratio`	This value should should be above 95%.
`Index hit ratio`	This value should should be above 98%.

The monitoring of single buffer pools is discussed in 6.4.3, “Buffer pool monitoring” on page 205.

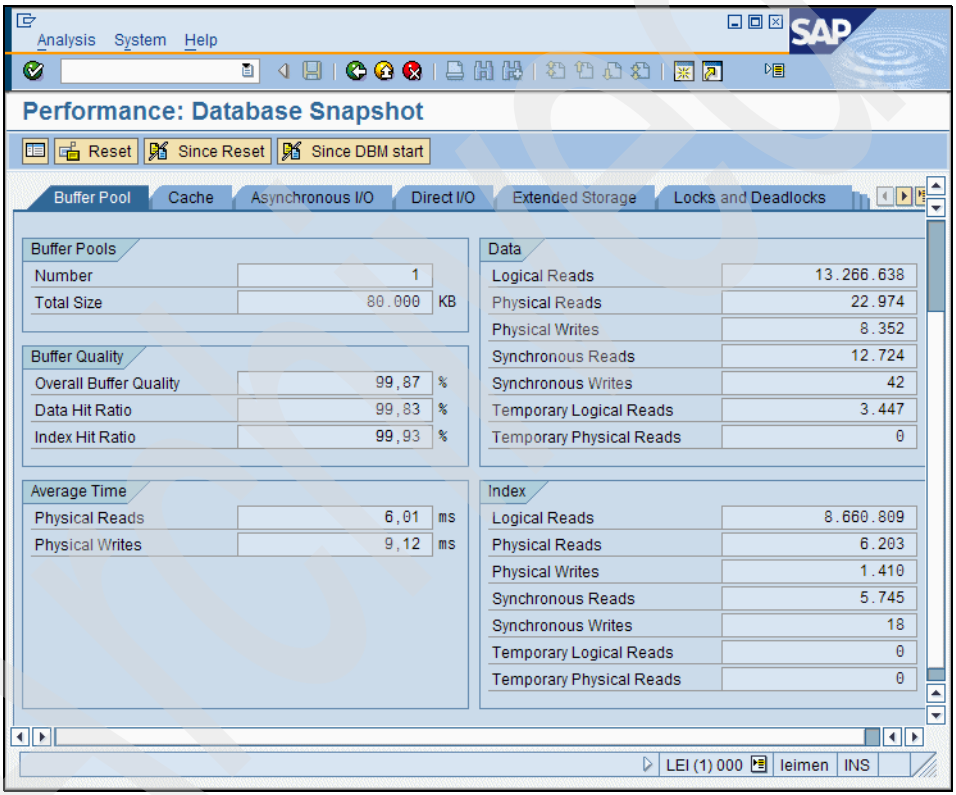


Figure 6-14 Overall buffer pool monitoring

Cache

The tab cache shows information about the package cache and the catalog cache. For the package cache and the catalog cache, the size (determined by the DB CFG parameters `CATALOGCACHE_SZ` and `PCKCACHSZ`), the quality, the lookups, the inserts, the overflows, and the high water mark are shown, as you can see in Figure 6-15.

The catalog cache contains information about tables, views, and aliases being accessed during statement preparation. The package cache is the server-side memory area where the executable embodiments of SQL statements are stored for reuse.

You can have a quick look at the following data, assuming that your SAP/DB2 system has been up and running with a typical workload for some time. The recommended value are listed:

`Catalog cache quality`:

This value should should be above 95%.

`Package cache quality`:

This value should should be above 98%.

`Overflows`:

This value should should be 0. Otherwise, you should consider increasing the cache where overflows occur.

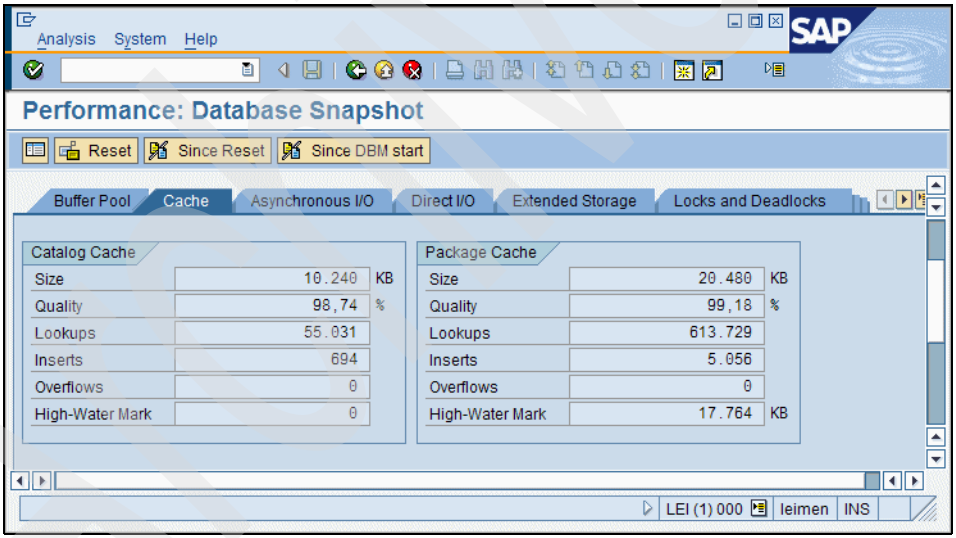


Figure 6-15 Monitoring caches

Asynchronous I/O

Asynchronous I/O tab displays activity and the configured number of the I/O servers and I/O cleaners, as shown in Figure 6-16. I/O servers, also known as prefetchers (`db2pfchr`), read asynchronously from disk to the buffer pools. I/O cleaners, also known as page cleaners (`db2pclnr`), write asynchronously from the buffer pools onto disk. In contrast to this, the `db2agent` reads and writes with so-called synchronous operations.

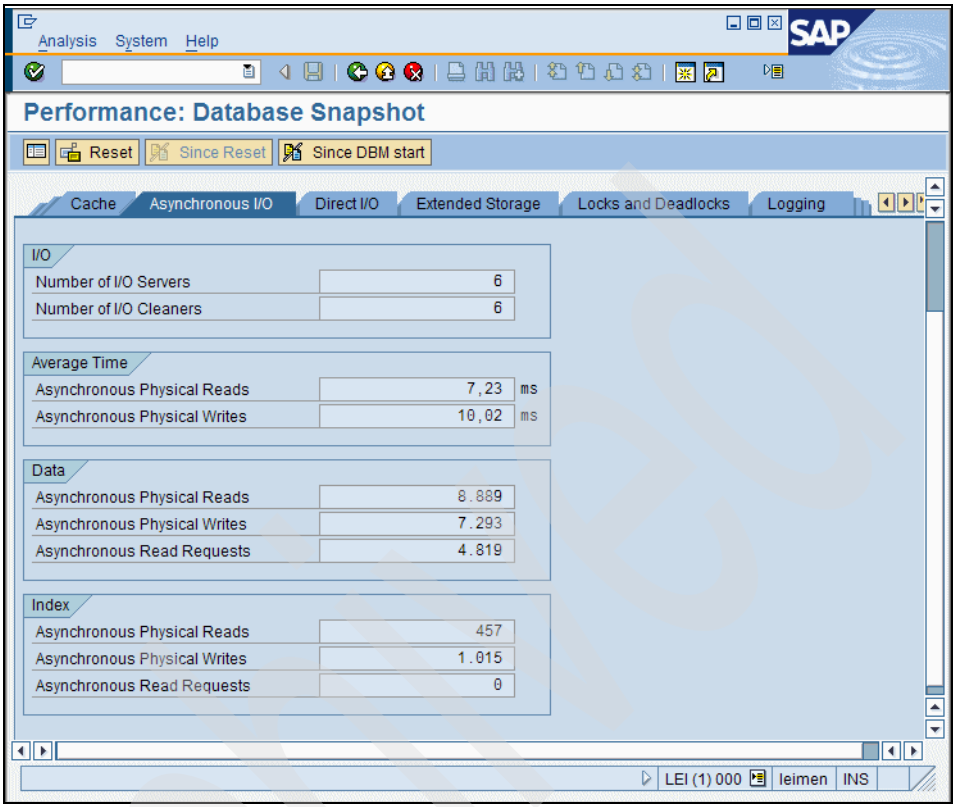


Figure 6-16 Asynchronous I/O monitoring

Direct I/O

Within *Direct I/O* (Figure 6-17) you can monitor the activity of direct read and write accesses, which do not utilize the DB2 UDB buffer pool. This includes accesses to data types `LONG VARCHAR` and `LOB`. Also, these reads and writes occur when performing a backup. Furthermore, notable cases are ABAP Sources and Loads, which are stored in `LONG VARCHAR` fields. Starting with SAP basis release 6.10, SAP uses `LOB` for ABAP object storage. Since the ABAP object storage is buffered within the SAP application servers, double-buffering through the use of DB2 UDB buffer pools is avoided with these data types.

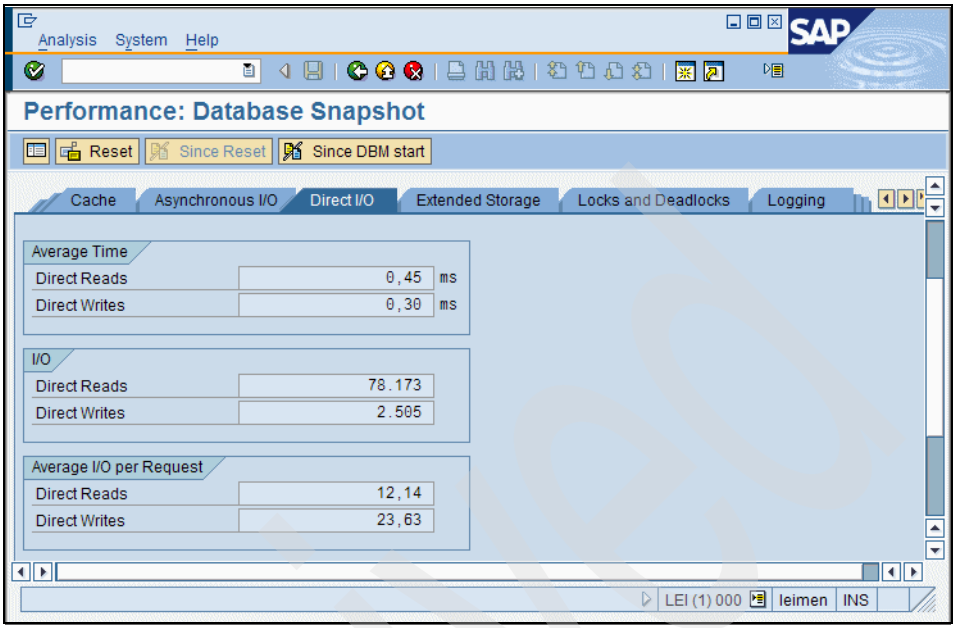


Figure 6-17 Direct I/O monitoring

Extended Storage

Within *Extended Storage* you can monitor the activity of the extended storage, especially how much data and index pages are copied from the buffer pools to the extended storage and back, as shown in Figure 6-18. The concept of extended storage was introduced with DB2 UDB Version 6 and provides a secondary level of storage for buffer pools. Storage pages are two segments that are attached or detached as needed. Pages are copied from the buffer pool to extended storage, when they are selected as victim pages. This copy process makes space for new pages in the buffer pool.

The number of extended storage segments can be defined with the DB CFG parameter `NUM_ESTORE_SEGS`. Likewise, their size can be configured with the DB CFG parameter `ESTORE_SEG_SZ`. The usage of extended storage segments for a specific buffer pool can be switched on and off with the command:

```
ALTER BUFFERPOOL bufferpoolname {EXTENDED STORAGE| NOT EXTENDED STORAGE}.
```

Tip: For DB2 UDB running with 64 bit, there is normally no need to configure extended storage because the buffer pools can be enlarged as needed.

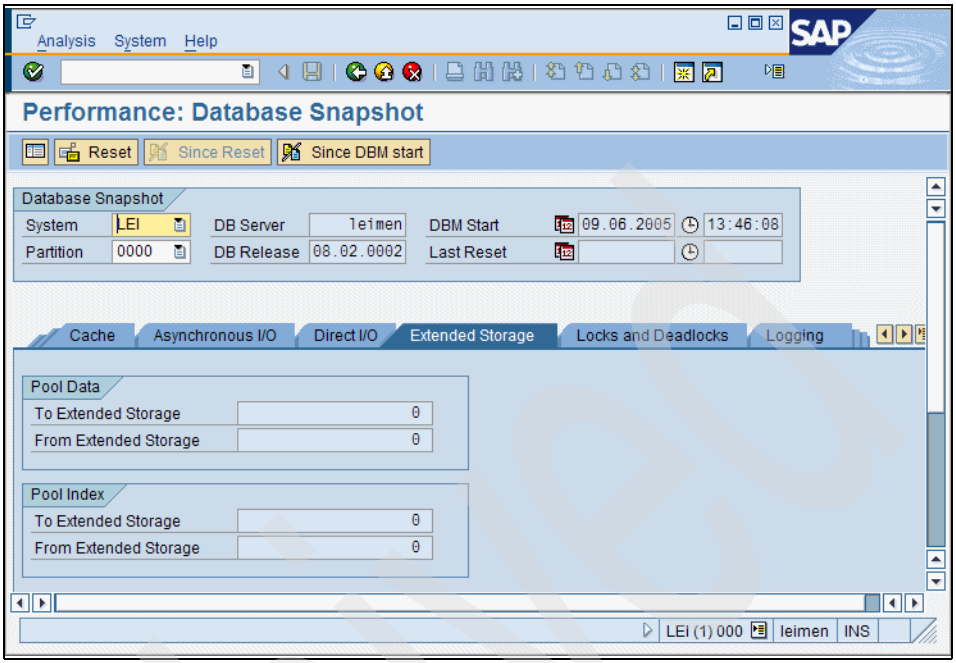


Figure 6-18 Extended storage monitoring

Locks and Deadlocks

The *Locks and Deadlocks* tab (Figure 6-19) gives an overview of the locking activity on the database system. You can see the size of the lock list (determined by the DB CFG parameter `LOCKLIST`), the current usage of the lock list, and the number of locks currently held by applications. Furthermore, you get an overview of the overall locks held by applications and the overall and average wait time on these locks, the overall lock escalations, deadlocks, and lock timeouts.

You can have a quick look at the following data, assuming that your SAP/DB2 system has been up and running with a typical workload for some time. The recommended value for each items are listed.

`Lock escalations`	This value should should be 0.
`Excl. lock escalations`	This value should should be 0.
`Lock timeouts`	This value should should be 0.

The monitoring of current lock and deadlock situations is discussed in 6.4.8, “Lock waits and deadlock analysis” on page 217.

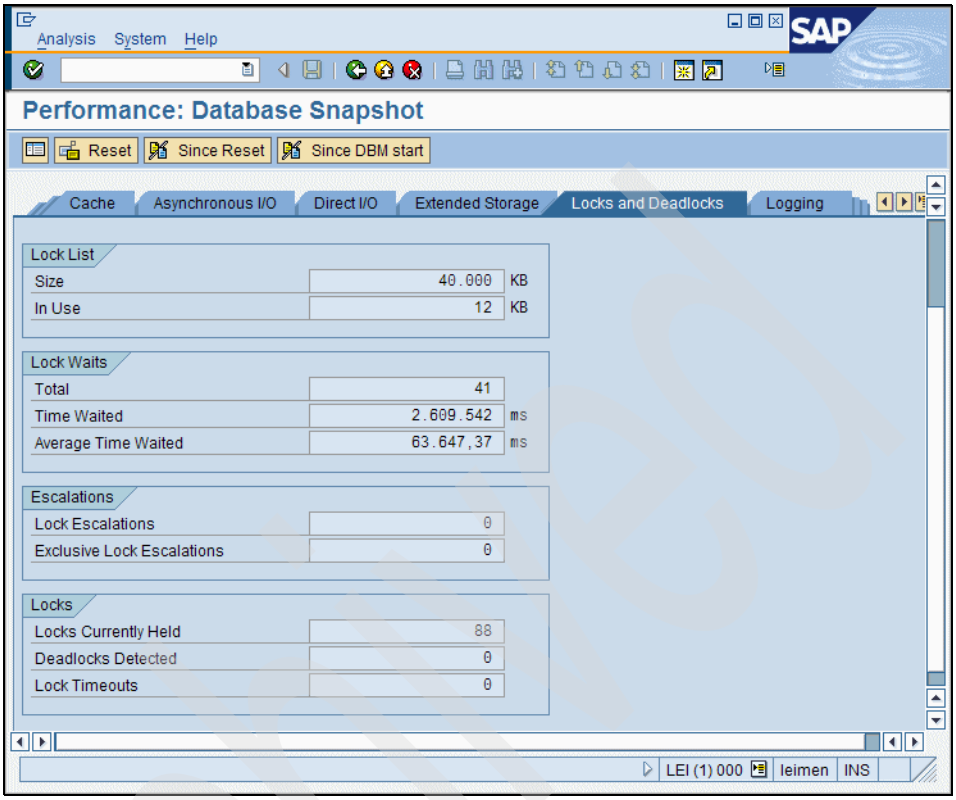


Figure 6-19 Monitoring locks and deadlocks

Logging

The *Logging* tab (Figure 6-20) shows the actual configuration of the DB2 UDB logging and its activity. You can monitor the log space consumption, the log buffer consumption, the log buffer quality, and the log buffer I/O.

SAP's new logger analysis elements in SAP NetWeaver 2004s utilize the DB2 UDB logger throughput information provided in version 8.2.2 to provide an advanced view into this previously untouched monitoring area. Now you can analyze logging information directly using SAP Cockpit instead of using operating system tools like `nmon`, `iostat`.

DB2 UDB V8.2 introduced a row of new logging monitor elements. For more details, please search the DB2 Information Center for “log_”. As an example, `log_write_time` now displays the total elapsed time spent by the logger writing log data to the disk. Knowing the number of pages written by the logger, simple mathematics brings us the average time it takes to write a page from log buffer to disk.

To make things easy, just imagine DB2 UDB's log buffer as a memory area designated to hold log records. During transaction commit, the log records, up to the record stating the commit action are transferred from log buffer to log file. This is done using 4k pages. If rollback is necessary, log records needed for the undo operations are accessed from the log buffer. In this case, rollback is a very efficient operation. If the log buffer were too small to hold all the log records of the transaction to be rolled back, the log buffer will have to write some pages to the log files. In this case, you will see log page read activity.

Since accessing storage is always a magnitude slower than accessing buffers, the rollback operation will therefore be slower than the in-memory rollback, even though having log buffer overflows cannot always be prevented. But sometimes it is sufficient to increase the log buffer size to reduce involuntary log page writes and increased rollback performance. DBAs, having the choice between optimal commit or rollback performance, will always optimize the database server for commit performance. And DB2 UDB is architectured to meet this requirement. Still, configuring a system for high log buffer hit ratio will help you to get the best possible in terms of rollback performance.

As another benefit of your new SAP DBA Cockpit, DB2 UDB's database checkpointing behavior has also been externalized. Based on new log monitor elements, you are able to assess how much more of the log space (LSN gap) will be consumed, until DB2 UDB's I/O servers will start to copy modified pages from buffer pool to disk. During this process, the restart range of the database, which is critical for restart performance will be slightly larger than the LSN gap. But at the end of the soft checkpoint, it will be identical again. Comparing the two percentages (LSN GAP % and Restart Range) will help you to understand if the time lag of the I/O servers is too high. If this is the case, you might want to increase the number of I/O servers, or ultimately improve the throughput of the storage subsystem.

The most valuable new monitor area covers the throughput for logging (Log Buffer IO). With this, SAP displays the data in microseconds instead of milliseconsd as for data page IO in ST04, since the latest storage subsystem provides page I/O latency, which is a fraction of a millisecond.

As shown in Figure 6-20, it takes DB2 UDB an average of 321 microseconds to write a single 4 K page on this database. Single page I/O is not common on log pages, so having multiple pages to write in a single request will offer a time of 363 microseconds. It shows that the initial time for single page transfer is only slightly increased for writing more pages. This is because of the overhead of using the system calls and context switches to perform this operation. Of pure academic value would be to calculate the average number of pages per I/O. But since there is no way to influence the number of pages per request from the SAP System perspective, displaying this value is not necessary.

As you may note, the time to read a log page is very low, compared to writing. This can be explained by the caching behavior of the file system and storage subsystem. It is very likely that a log page needed for rollback still resides in these buffers when it is needed.

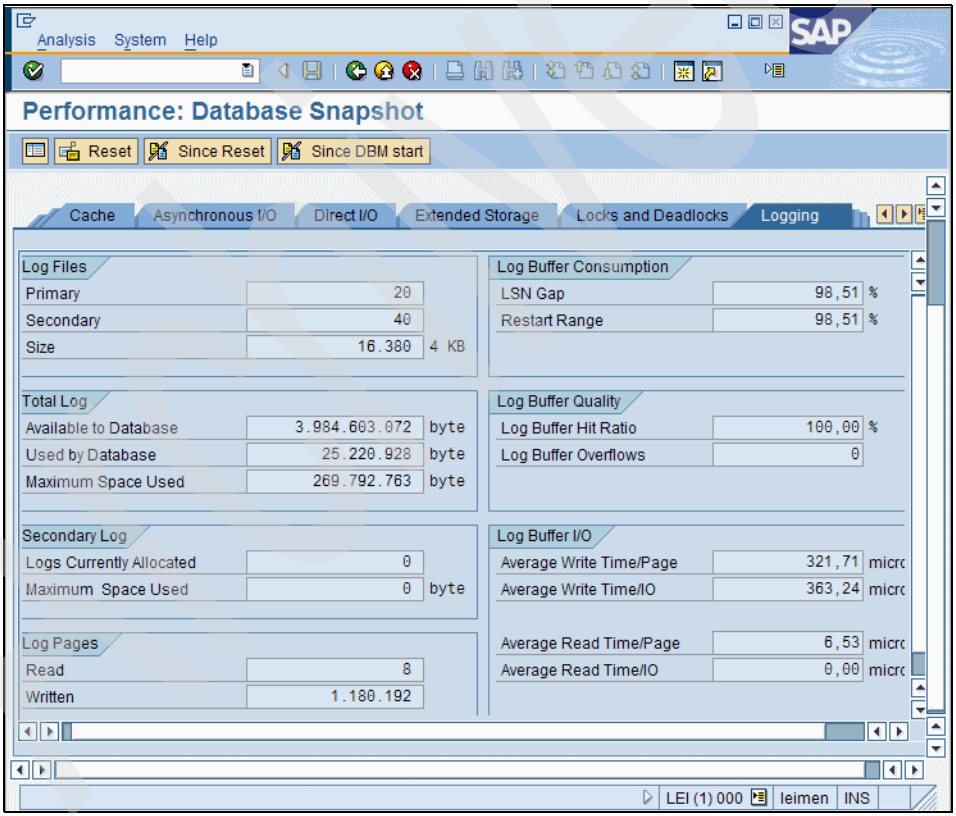


Figure 6-20 Monitoring of logging

You can have a quick look at the following data, assuming that your SAP/DB2 system has been up and running with a typical workload for some time. The recommended value for each item is listed here:

`Secondary log:`

Maximum space used should be 0. Within normal operation of the SAP/DB2 system, no secondary logs should be allocated. Because they are not preallocated, allocating secondary logs takes more time than using a preallocated primary *log file*. More information on DB2 UDB logging concept can be found in 7.1, “Log file management” on page 314.

`Log buffer hit ratio:`

This value should be 100%.

`Log buffer overflows:`

This value should be 0.

`Log buffer consumption:`

The LSN gap and restart range should be nearly equal. If the restart range is significantly bigger than the LSN gap for a longer time period, you can improve the checkpointing by increasing the number of I/O servers or by tuning the whole I/O subsystem.

Calls

The *Calls* tab gives an overview about SQL statement and row activities within your SAP/DB2 system, as shown in Figure 6-21. Here you get an overview how many SQL statements and rows have been processed within your database, and which type of processing has been done.

If your SAP/DB2 system has been up and running with a typical workload for some time, you should consider the following recommendation:

`Selected SQL:`

This value should be more than 1,000,000.00.

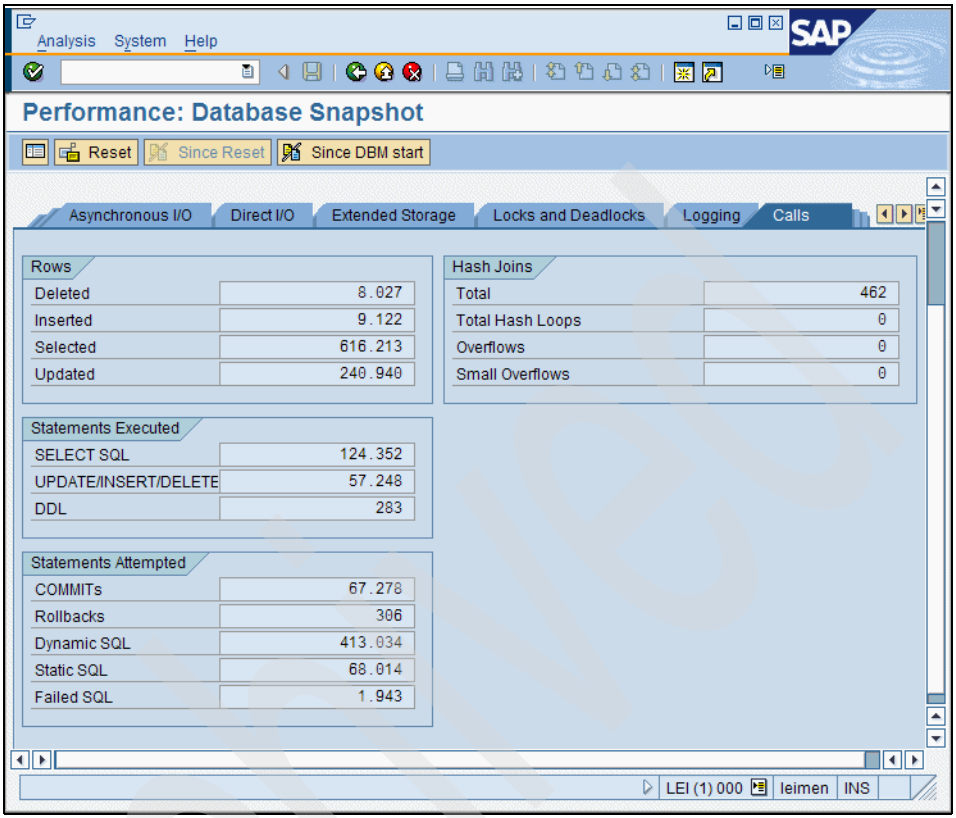


Figure 6-21 Overview of SQL statement and row activity

Sorts

The *Sort* tab (Figure 6-22) provides information about the sort activities within your database. You can determine the size of the sort heap (DB CFG parameter SORTHEAP), how much is currently used from the sort heap, sort times, and overall sorts and sort overflows.

The sort heap is used for sorting. If there is not enough space in the sort heap, a temporary table space is used to handle the sort requests. This is called a sort overflow.

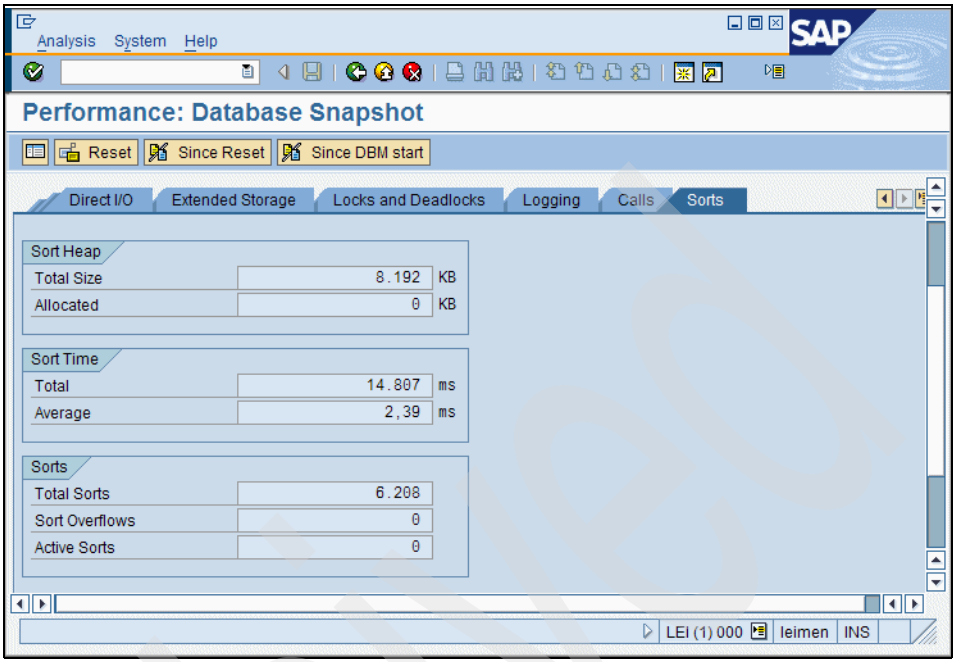


Figure 6-22 Monitoring sorts

If your SAP/DB2 system has been up and running with a typical workload for some time, you should check the following value:

`Sort overflows:`

This value should not exceed 1% of total sorts.

6.4.2 Schema monitoring

A Multiple Components in One Database (MCOD) system is an SAP system which has more than one SAP component installed, and the database contains tables for all the installed components. The differentiation between the objects of each SAP component will be done with the schema. Each SAP component has its own schema, under which all objects (tables, indexes, etc.) of this component will be created.

The schema monitoring enables you to determine the amount and percentage of physical and logical reads for data and indexes for each installed SAP component within a database. You can identify the performance-critical SAP component and the workload distribution among the components in your database using Schema monitoring.

For an MCOD system, all SAP components' objects reside within the same database, you can only monitor the whole database and tune the performance of the whole database. If one of the SAP component within the database does not perform well and interferes with the other applications, you can only try to improve the performance of the whole database. The alternative will be to separate one or more applications to another standalone SAP/DB2 system.

Figure 6-23 shows the Schema monitor of a non-MCOD system. Since it is a single component system, 100 percent of database activities are performed by only the SAP component with the schema name `SAPLEI`.

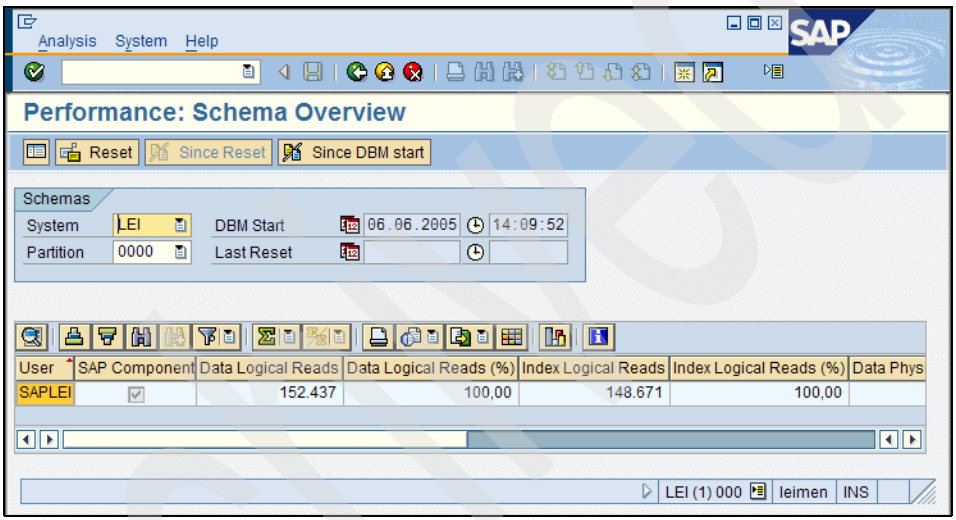


Figure 6-23 Schema monitoring

6.4.3 Buffer pool monitoring

The data for the buffer pool monitoring mainly comes from the DB2 UDB command **`GET SNAPSHOT FOR BUFFERPOOLS ON`** *<sapsid>*. As shown in Figure 6-24, a list of all buffer pools with the key values for them are displayed.

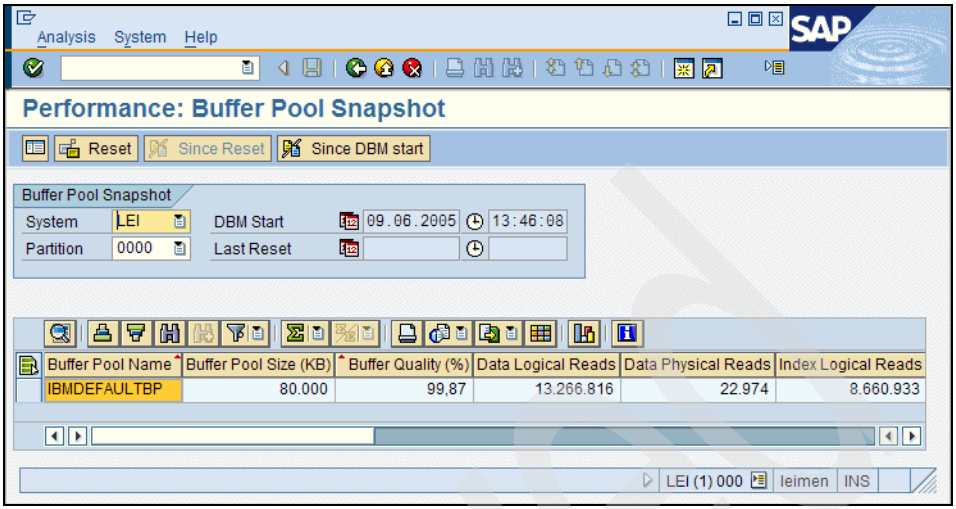


Figure 6-24 Monitoring single buffer pools

By double-clicking one buffer pool, the values returned by the snapshot for this buffer pool are grouped by subject and displayed on the tab index. Based on the returned values, some hit ratios are calculated and displayed, as shown in Figure 6-25. The tab indexes, Asynchronous I/O, Direct I/O, and Extended Storage, are the same as described in 6.4.1, “Database monitoring” on page 193. The values shown here are for the chosen buffer pool.

If your SAP/DB2 system has been up and running with a typical workload for some time, you can have a quick look at the following data. The recommended values are listed here:

`Overall buffer quality` This value should should be above 96%.

`Data hit ratio` This value should should be above 95%.

`Index hit ratio` This value should should be above 98%.

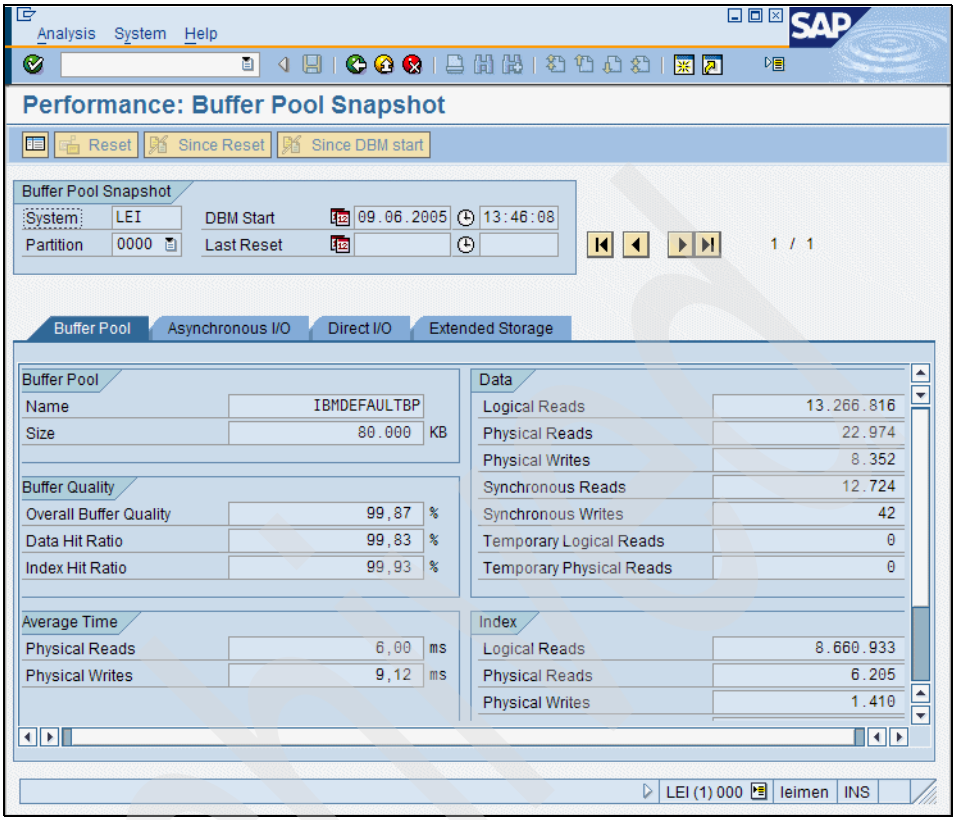


Figure 6-25 Monitoring details for one buffer pool

The monitoring of the database buffer pools is discussed in detail in Chapter 8, “ST04: key performance indicators” on page 474.

6.4.4 Table space monitoring

Within the table space monitoring, database activity and performance information is broken down on the table space level. As you can see in Figure 6-26, the list contains the buffer pool quality, and data and index physical and logical reads on the table space level.

The data for the table space monitoring mainly comes from the DB2 UDB command **`GET SNAPSHOT FOR TABLESPACES ON`** *<sapsid>*.

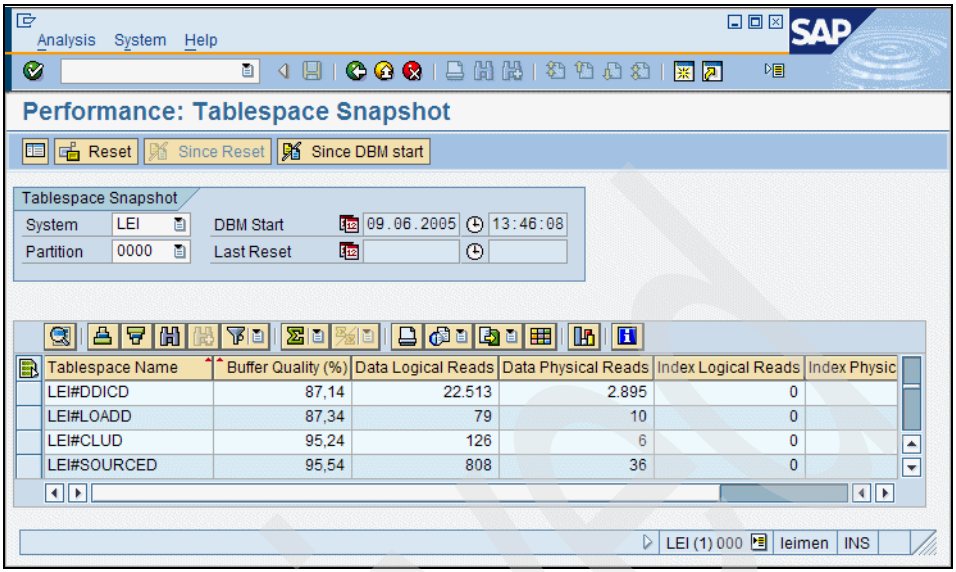


Figure 6-26 Table space monitoring

Double-clicking a specific table space shows the values returned by the snapshot grouped by subject on tab indexes, as shown in Figure 6-27. With the arrow buttons, you can cycle through the other table spaces. Based on the returned values, some hit ratios are calculated and displayed.

The tab indexes, Buffer Pool, Asynchronous I/O, Direct I/O, and Extended Storage, are the same as described in 6.4.1, “Database monitoring” on page 193. The values shown are based on the chosen table space.

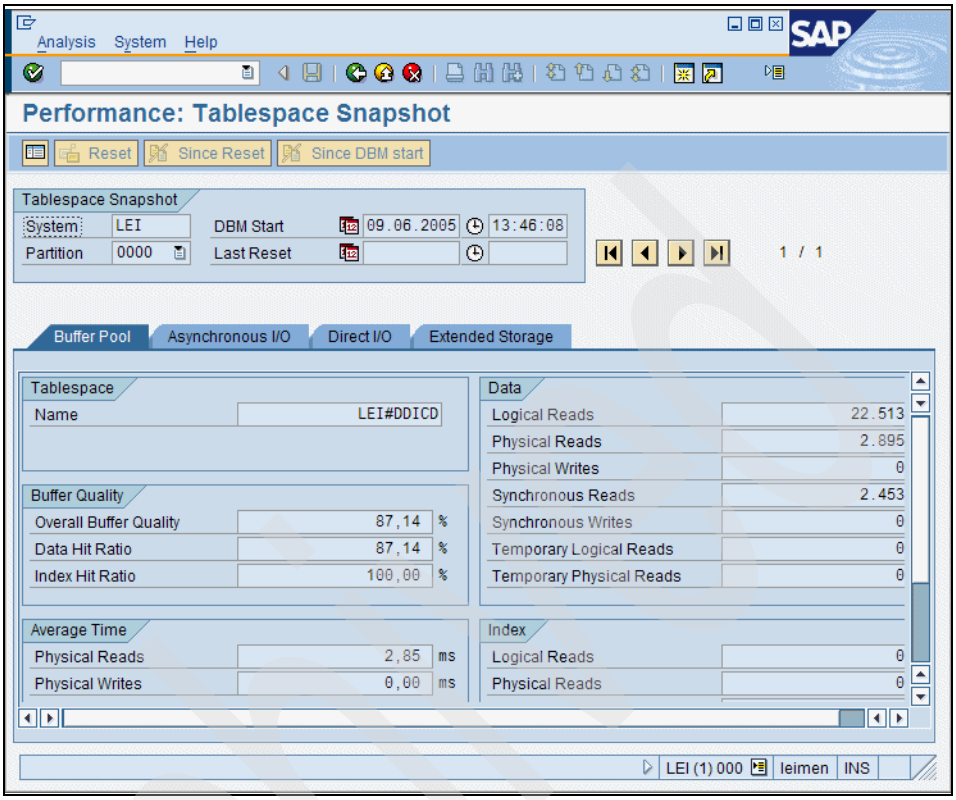


Figure 6-27 Table space monitoring details

If your SAP/DB2 system has been up and running with a typical workload for some time, you can have a quick look at the following data. The recommended values are listed here:

`Average read time` This value should be smaller than 10 ms.

`Average write time` This value should be smaller than 50 ms.

6.4.5 Table monitoring

The table monitoring function lists only data on tables that are used in the system because the data for table monitoring mainly comes from the DB2 UDB command **`GET SNAPSHOT FOR TABLES ON`** `<sapsid>`. If a table is not selected, updated, or inserted into, it will not be displayed within this list. It also displays for each table how many rows have been written or read and whether there are overflow accesses and internal page reorganizations, as shown in Figure 6-28.

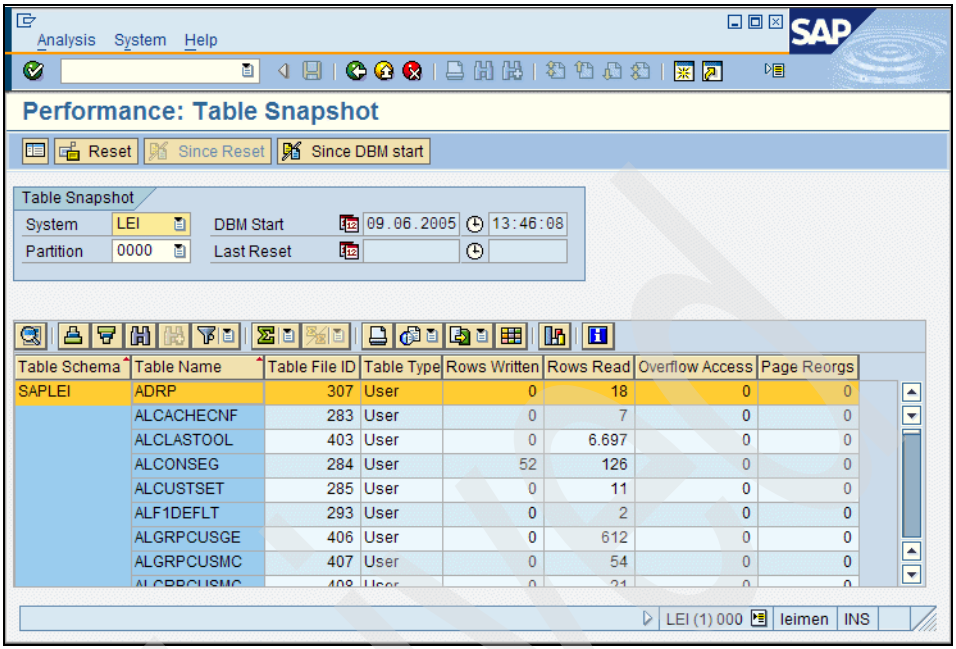


Figure 6-28 Table monitoring

By double-clicking a table, you get a tab indexed view of the properties of this table. Table monitoring is discussed more in detail in 6.5.4, “Single table analysis” on page 229.

6.4.6 Application monitoring

Application monitoring shows all applications that are connected to the database using the DB2 UDB command **`GET SNAPSHOT FOR ALL APPLICATIONS`**. Basing on the returned values, some hit ratios are calculated and displayed. To have a better overview, you should select *Active Only Applications* in the upper right screen, as shown in Figure 6-29.

Tip: With the application monitoring, you can examine running applications. For example, when an application does not perform well, you can check, via Explain, whether it uses an index or not to access the data. More information about Explains is provided in 6.11.5, “Explain” on page 291.

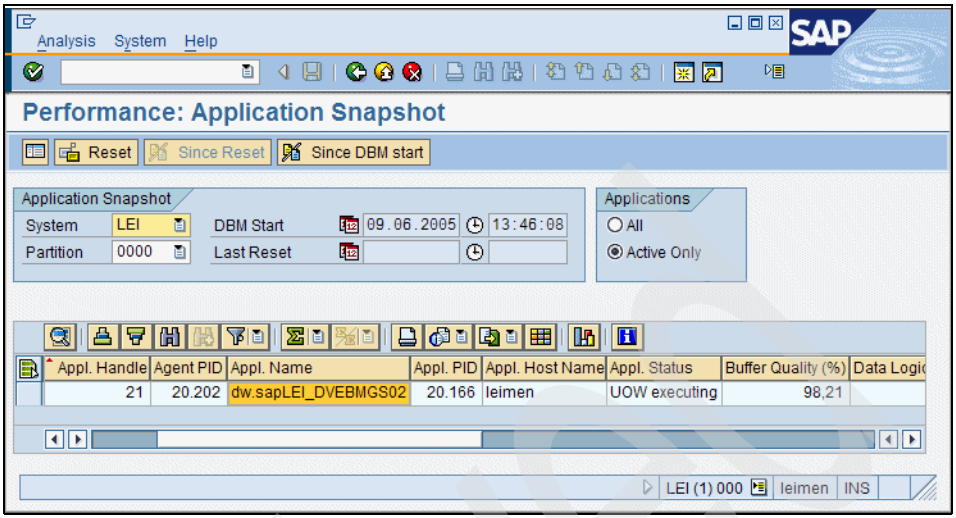


Figure 6-29 Application monitoring

By double-clicking an application you get more information in card index format. The card indexes (except Application, Agents, Unit Of Work, Statement, and Statement Text) are the same as described in the preceding section, but only the data shown is related to a single application, which is normally a workprocess in SAP/DB2 systems.

Figure 6-30 shows the *Application* index card, where you can get information about the application itself. Here you see an SAP workprocess with the process ID (PID) 20.166 running on a Linux server called `leimen`. Furthermore you see when the workprocess has connected to the database and what is the PID of the coordinating `db2agent` handling the SQL statements of this application.

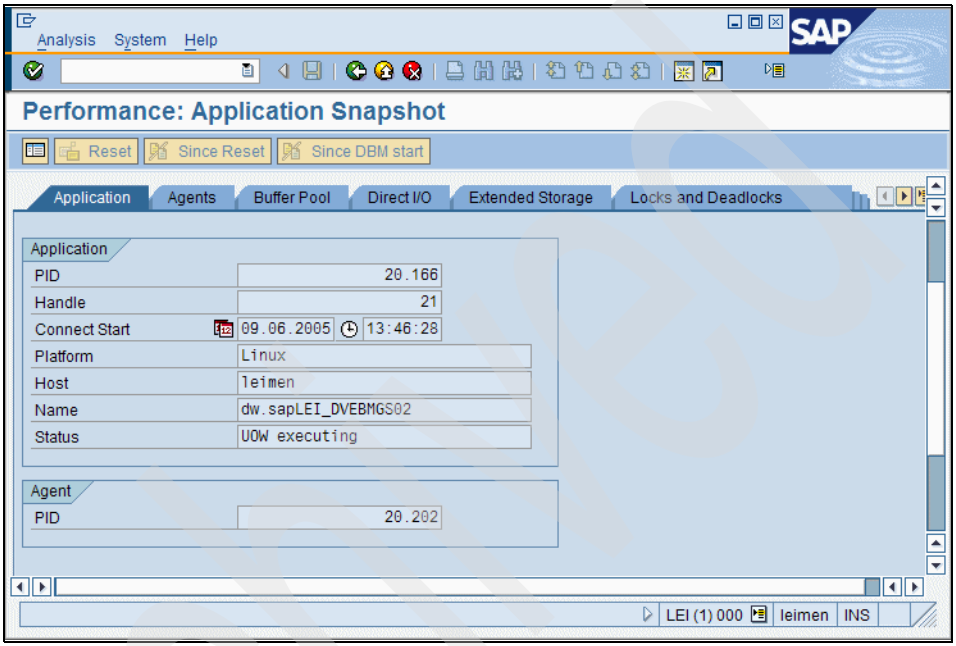


Figure 6-30 Application monitoring - application card index

On the next index card, *Agents*, you see more details about the coordinating `db2agent` handling the SQL statements of this application. As shown in Figure 6-31, the number of additional `db2agent` used, and the times used by all agents associated with this application are displayed.

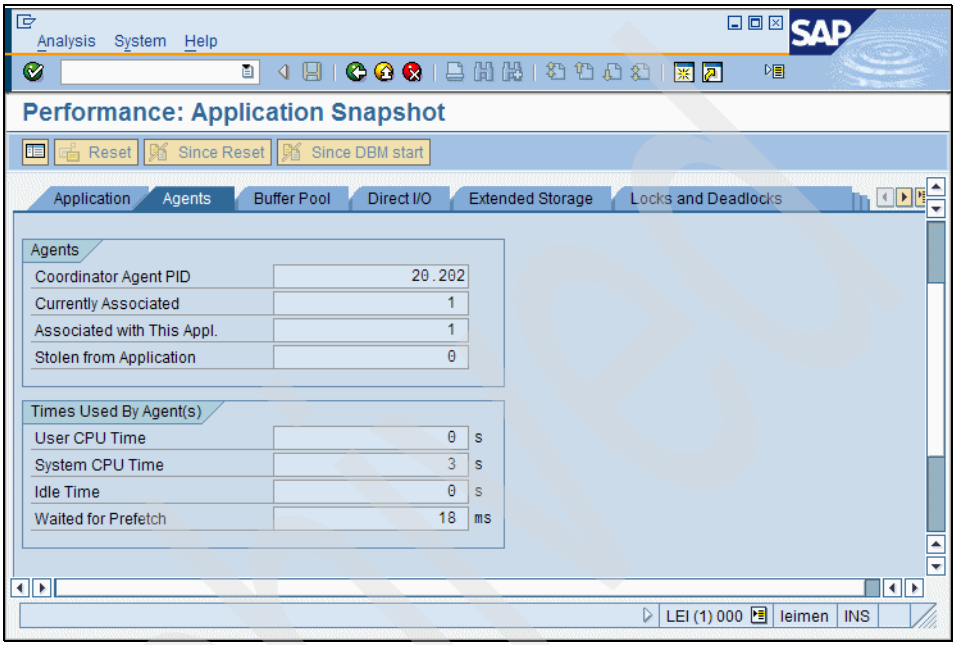


Figure 6-31 Applications monitoring - agents card index

Figure 6-32 shows the *Unit Of Work* (UOW) index card, providing information about the currently running UOW. You can determine when the UOW has been started, how much log space has been used up to now, and so on.

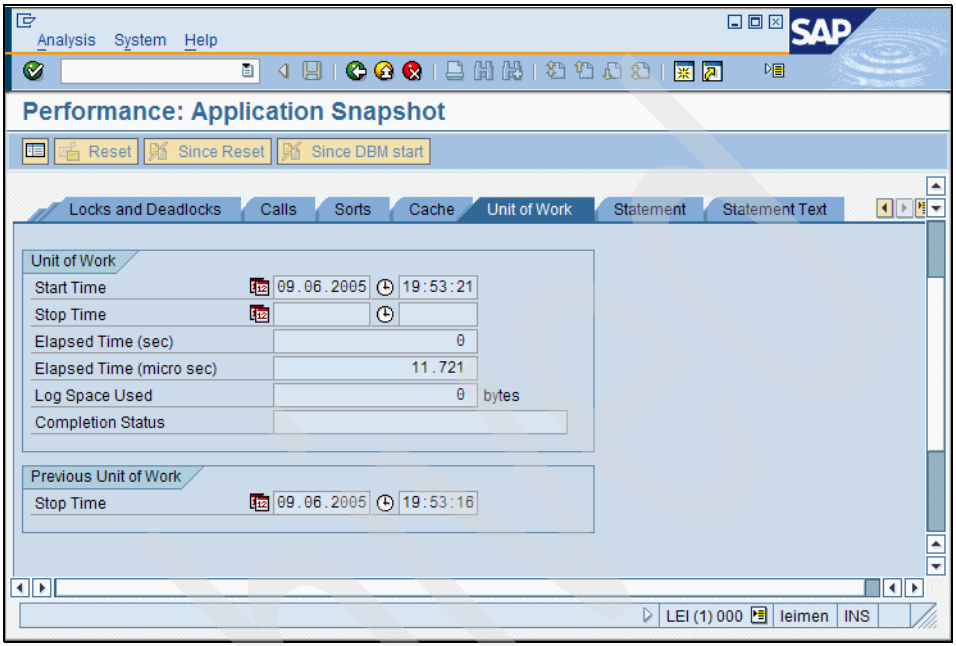


Figure 6-32 Application monitoring - unit of work index card

On the index card, *Statement*, you see an overview of all activities of the currently running statement, as shown in Figure 6-33. You can get information about sorts, rows read and written by the statement and data and index logical and physical reads.

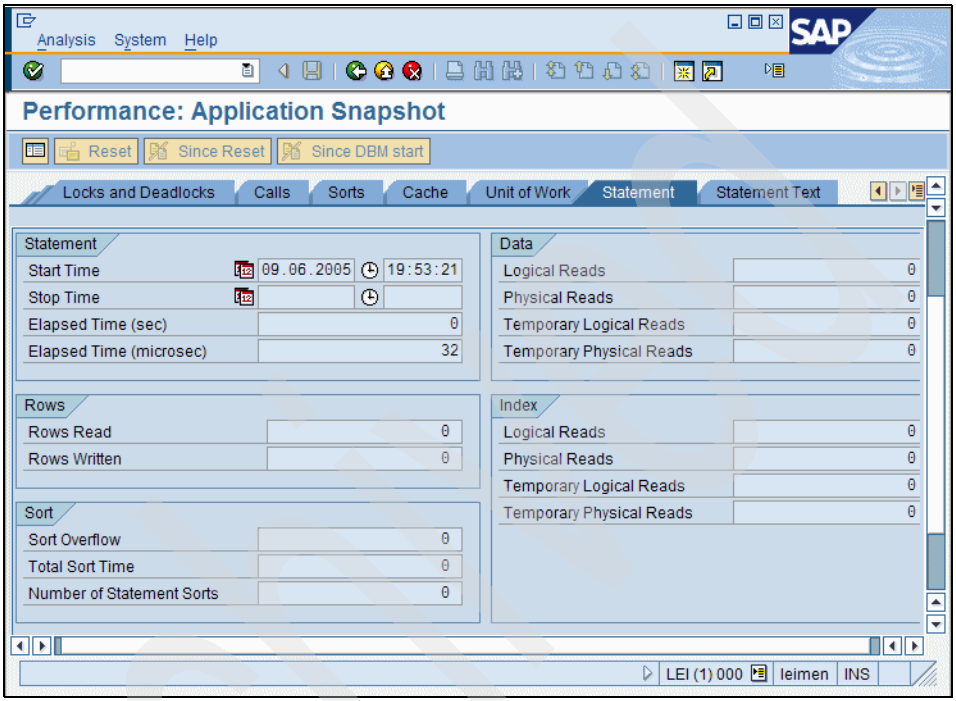


Figure 6-33 Application monitoring - statement index card

On the last index card, *Statement Text,* you see the currently executed SQL statement, as shown in Figure 6-34. Here you can also get an Explain of this running statement. More information about Explains is provided in 6.11.5, “Explain” on page 291.

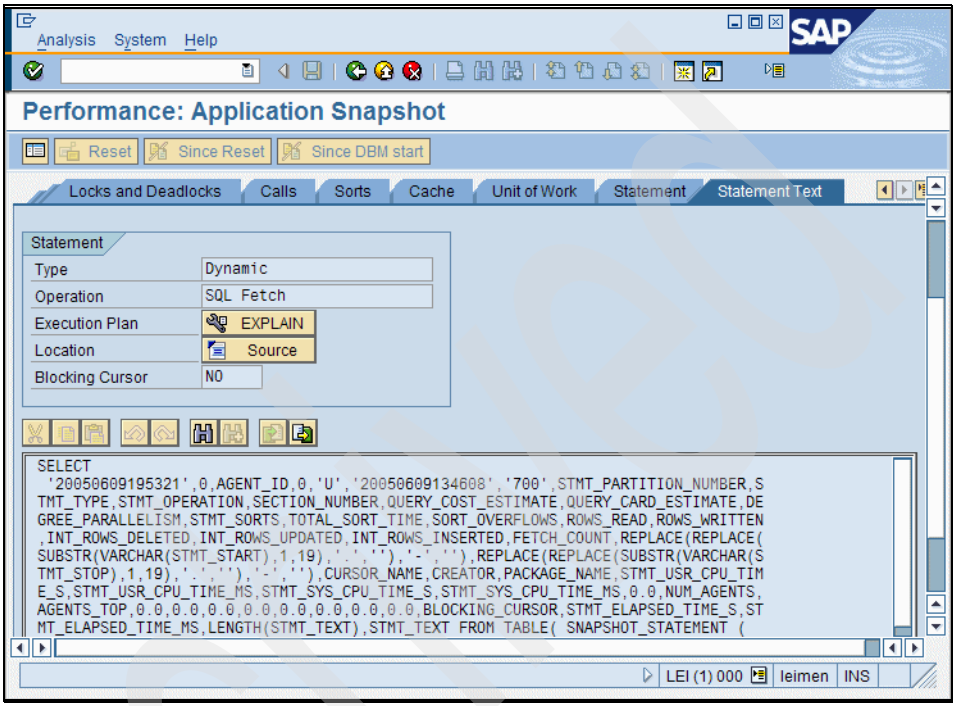


Figure 6-34 Application monitoring - statement text index card

6.4.7 SQL cache analysis

SQL cache analysis allows you to analyze the SQL statements that run by all applications within your SAP/DB2 system. As shown in Figure 6-35, you can use selection criteria to analyze the SQL cache, for example, to select only those statements that have been executed for a certain number of times or those that have been running longer than a specified amount of time. With the help of the SQL cache, you can determine expensive SQL statements, which we discuss in detail in Chapter 8, “ST04: SQL Cache - finding expensive statements” on page 482.

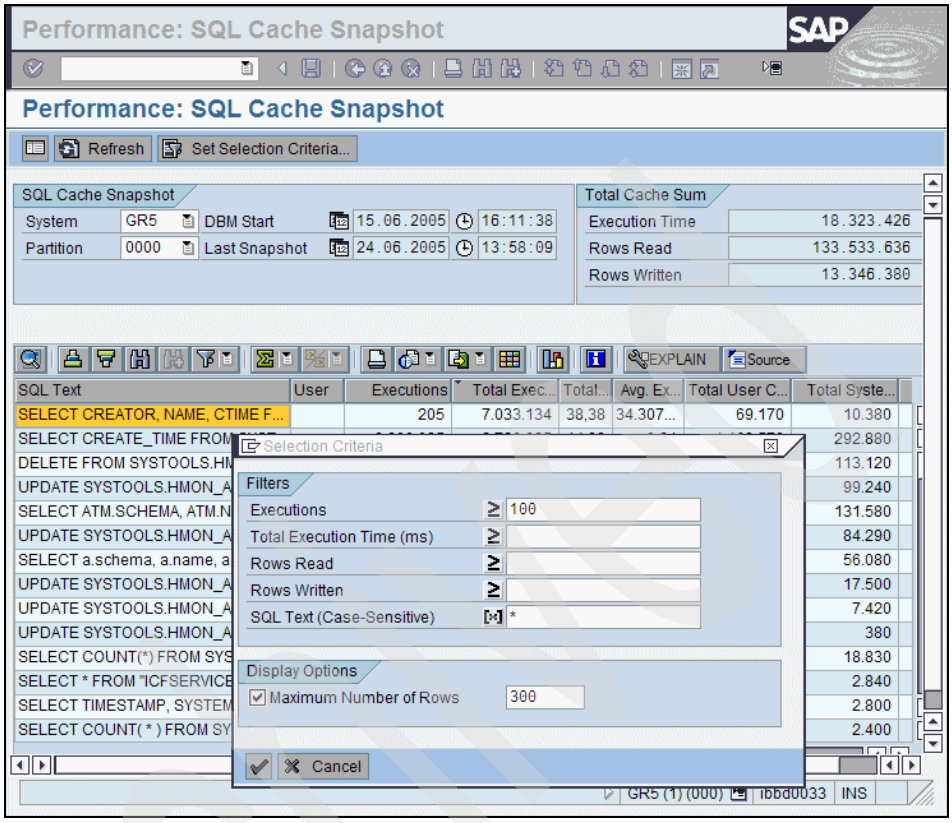


Figure 6-35 SQL cache analysis

6.4.8 Lock waits and deadlock analysis

With this analysis, it is possible to analyze current lock wait and deadlock situations. In the example in Figure 6-36, you can see that at the time the screen shot was taken, there are no deadlocks in the SAP/DB2 system. But there is a lock wait situation between two `db2bp` processes (these processes are called background processors and they are used, for example, for the command line processor).

The arrows in the graphical sketch always point to the process which holds a lock. In the example, the process with the ID 201 waits for a lock being held by the process with the ID 196. Below the graphical sketch you can see a list with the SQL statements by which each of the involved processes has been invoked. More details on lock waits and deadlocks can be found in Chapter 8, “Deadlock” on page 434.

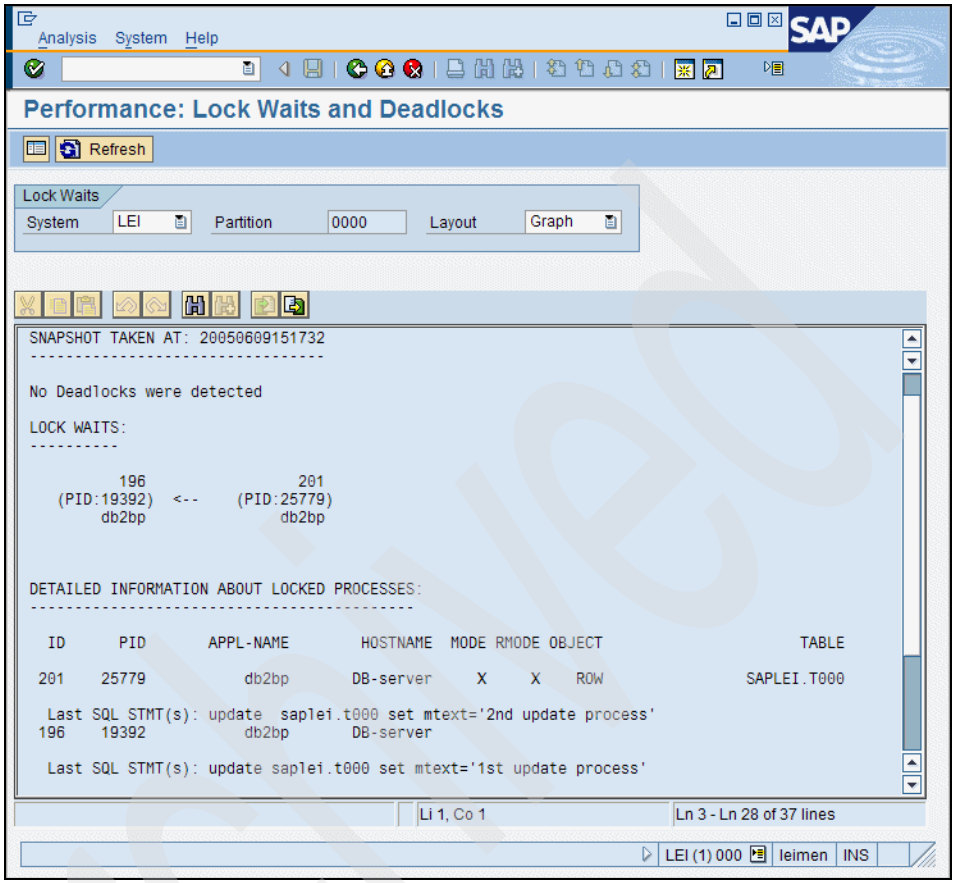


Figure 6-36 Lock waits and deadlock analysis

6.4.9 Inplace table reorganization monitoring

To get an overview of all running inplace table reorganizations and their state, you can use the inplace table reorganization monitor, which displays all started inplace table reorganizations. As shown in Figure 6-37, for each started inplace table reorganization, the schema, the table name, the status, the progress, the start date and time, the access mode, and the table space where the reorganization takes place are displayed. A reorganization can have the status *paused* for two reasons: The reorganization has been paused manually, or the SAP/DB2 system has been restarted during a running inplace table reorganization.

After restarting the DB2 UDB, the inplace table reorganizations normally are in the state *paused* and must be resumed with the command **`REORG TABLE`** *`schemaname.tablename`* **`INPLACE RESUME`**, as described in 4.4.2, “Inplace table and index reorganization” on page 104. For example, to resume the paused reorganization in Figure 6-37, you have to use **`REORG TABLE`** `saplei.dynpsource` **`INPLACE RESUME`** within the DB2 UDB command line processor.

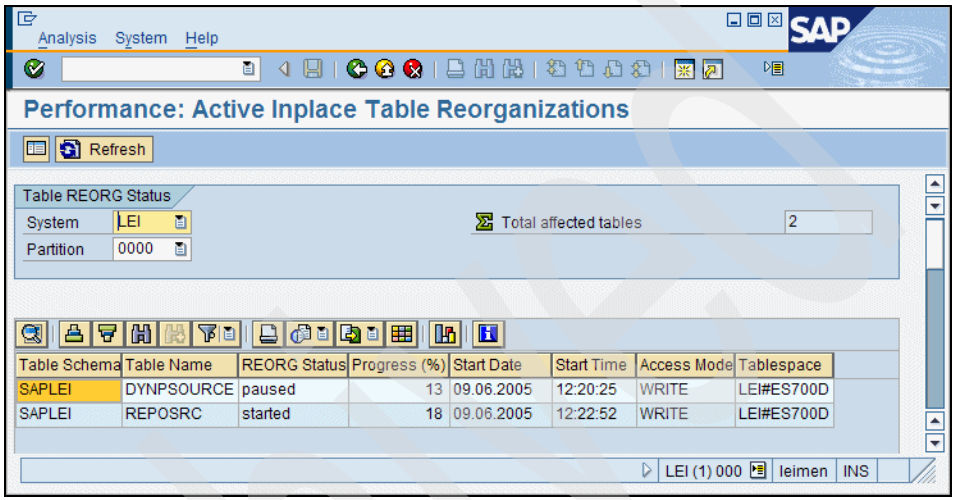


Figure 6-37 Monitoring inplace table reorganizations

6.5 Space management

Managing space is one of the most important administration tasks for the DBA, because if there is no space left, most operations can no longer be done. In the *Space* task area, administration of table spaces and table space containers can be done. Also, you can monitor each table and index carefully from DBA Cockpit. You simply go to *Space* in the navigation frame, where DBA Cockpit provides you with easy administration for space management.

6.5.1 Table spaces

Go to **Tablespaces** under *Space* in the navigation frame. With this function, you can check the status of table spaces. Also, changing attributes, adding a table space, and deleting a table space can be done from here. Operations such as changing, adding, or deleting a table space can be performed while DB2 UDB is running. Figure 6-38 shows the Tablespace configuration screen, which displays a list of table spaces and their status. This screen has two tabs, *Automatic Storage* and *DMS/SMS Tablespaces*.

- Automatic Storage:

  This tab has two sub-tabs:

  - *Tablespaces:* Here you can check each table space status and attributes. You can perform not only monitoring of a table space, but also changing a table space, adding a table space, and deleting a table space. To see the detail of table space, such as container name, double-click an entry.
  - *Storage Paths:* You can check the storage path for automatic storage. Also, it is possible to add and delete a storage path.

- DMS/SMS Tablespace:

  Here you can check each table space status and attributes, and also maintain table spaces which are not created with the automatic storage option.

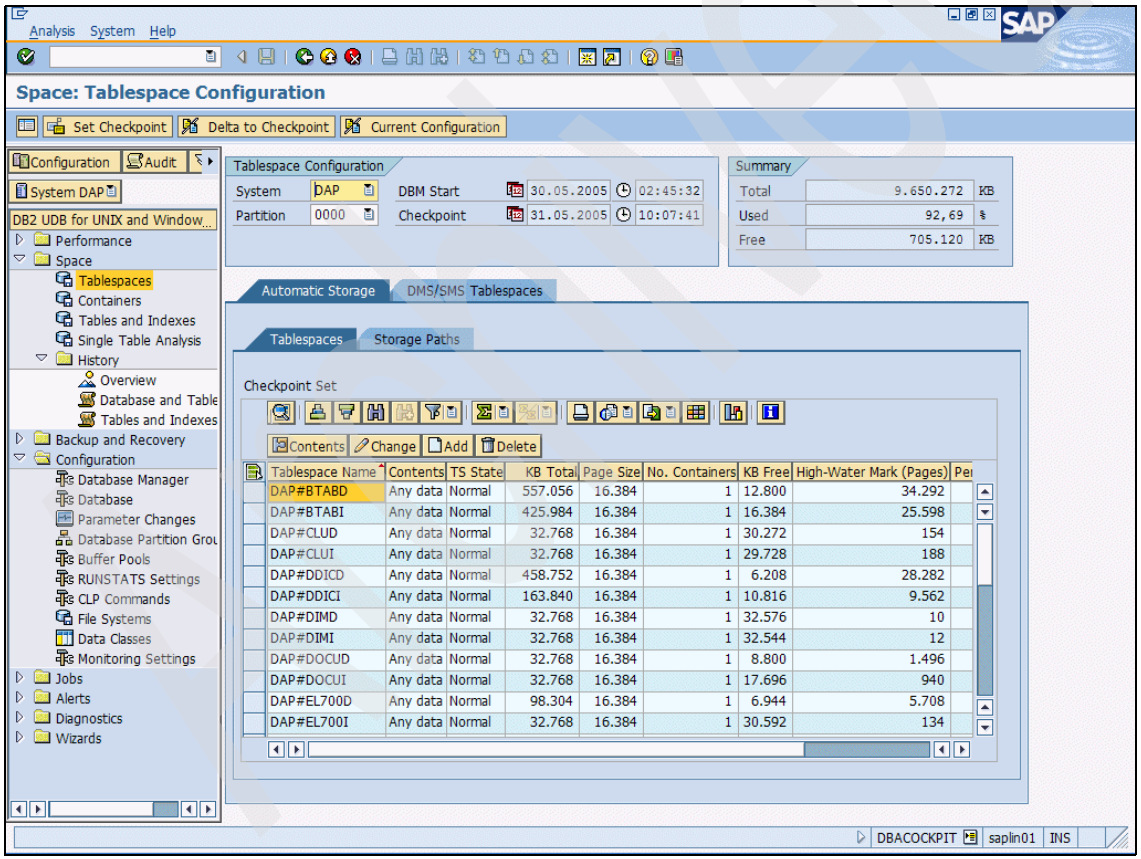


Figure 6-38 Tablespace Configuration

Monitoring table spaces

You may need to monitor the percentage used for table spaces carefully if you use auto-resize DMS. When you use the auto-resize DMS or SMS table spaces or automatic storage management with auto-resize, you need to watch the percentage used for the file systems where the containers reside. Also, you need to monitor the size of table spaces to see if the table space is approaching its maximum size or not.

From the *Tablespace Configuration* screen (Figure 6-38 on page 220), you can monitor the information on table space used percentage and size of a table space. To check file system usage, go to **Configuration** → **File Systems**. (See Figure 6-39.)

As you saw in Figure 6-38 on page 220, it is possible to set a checkpoint for table space information. If you set a checkpoint at a certain time, you can see the difference between the checkpoint time and now, such as total table space size. To set a checkpoint, click **Set Checkpoint**. The checkpoint time is displayed in the upper frame. To see the difference, click **Delta to checkpoint.** To go back to the current configuration after checking the difference, click **Current Configuration**. This checkpoint function may help you to determine how fast a table space is growing. The checkpoint set here is only valid for the table space level. To see the delta for containers, you can also set a checkpoint in the **Containers** task area. (Refer to “Containers” on page 225.)

We recommend that you monitor table space information and file system space information daily.

Tip: Percent Used for SMS table spaces always indicates 100%. This is because DB2 UDB manages the SMS table space to allocate sufficient space to the table space. In this case, you should carefully watch the file system space usage.

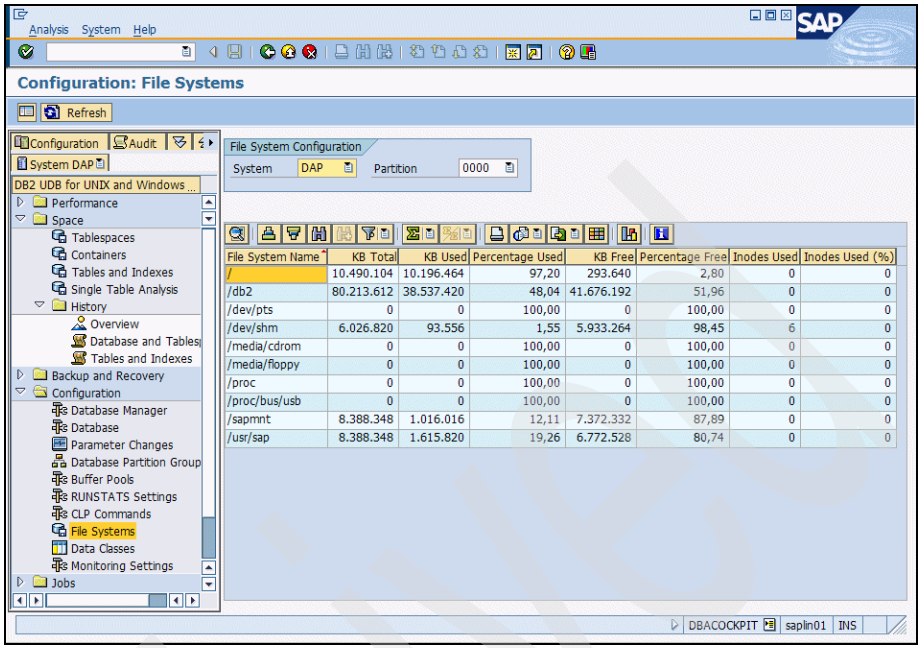


Figure 6-39 Checking File Systems

Changing/adding/deleting a table space

If a DMS table space without auto-resize is almost full or growing fast, you may need to add containers to the table space or expand existing containers. Click **Change** to add or resize a container. When you add/resize a container, we recommend that you use the same size containers for a table space. If different sized containers exist in a table space, data is not distributed equally; and this can result in poor performance.

When adding or resizing containers, rebalancing may occur. Rebalancing may affect system performance. One way to avoid rebalancing in case of adding a container is to create a new stripe set. For more information about stripe sets, refer to Chapter 4, “Storage management in depth” on page 79. You can also change some table space attributes by clicking **Change**.

Configuring the alert monitor may help you if you want to get alert messages about the percentage used of table spaces or the percentage used for the log file directory automatically. For more information about the alert monitor, refer to 6.10, “Using the alert monitor” on page 278.

Here we show you steps to add a new DMS table space.

First, click **Add** in the table space list (Figure 6-38 on page 220). Then a new screen (Figure 6-40) comes up. Enter the name of a new table space and decide which table space type, page size of the table space, extent size, prefetch size and so on.

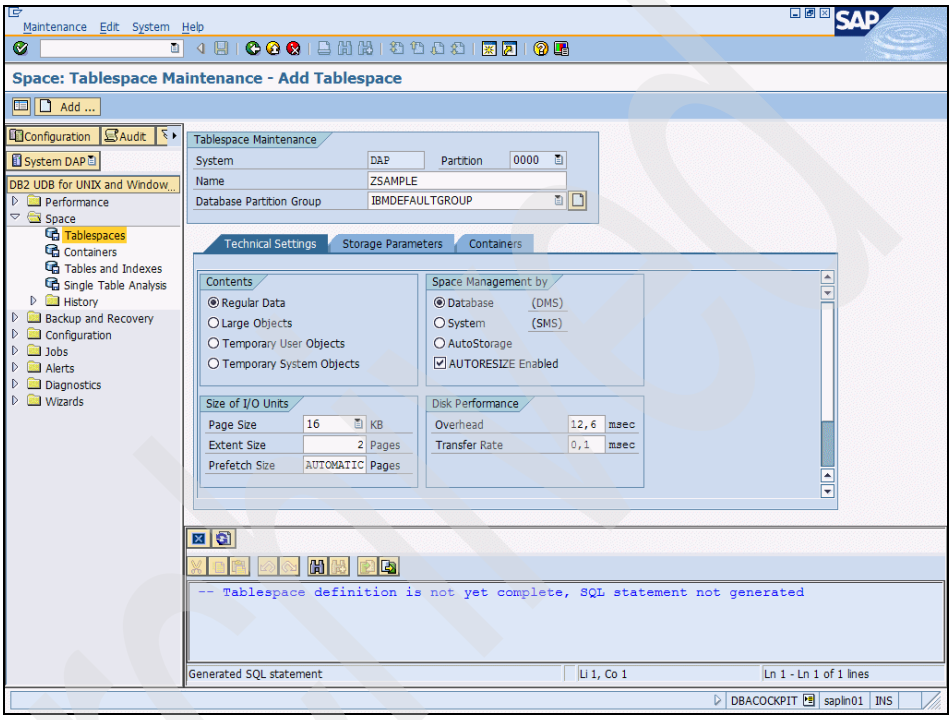


Figure 6-40 Technical setting for a new table space

If you go to **Storage Parameters** tab, you can specify maximum size and increase size.

When you go to the **Containers** tab (Figure 6-41), you can add a container by clicking **Add container**. The path for a container is automatically generated and displayed. You can change the path and size as you want. Also, here you can specify the size of a table space.

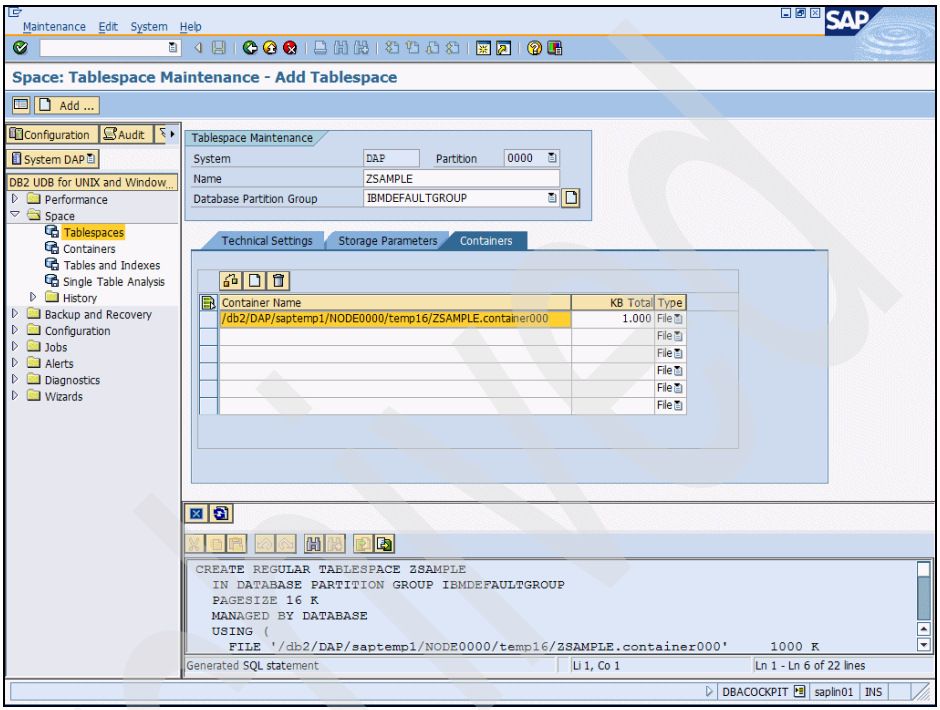


Figure 6-41 Container configuration for a new table space

As shown in the lower part of Figure 6-41, SQL statements are automatically generated and displayed in the lower half of screen after you change the configuration. These SQL statements are the SQL statements which are going to be executed by DBA Cockpit. You can save these statements as a local file.

After inputting all necessary values, click **Add**. Then DBA Cockpit starts to create a table space.

Note: If you need to delete a table space, you have to delete a data class which is related to the table space before deleting the table space.

For more information about DB2 UDB table spaces, refer to 4.1, “DB2 UDB logical and physical database objects” on page 80.

6.5.2 Containers

Sometimes you need detailed information on each of the containers. To get the information, under **Space**, double-click **Containers**. You get a list of all containers as shown in Figure 6-42. From this list you can easily get container status and attributes, such as container name, container type, etc.

In the *Container* task area, you can set a checkpoint to see the delta. To set a checkpoint, click **Set Checkpoint**. To see the difference between the checkpoint and now, click **Delta to Checkpoint**. The difference in container size or number of pages can be accessed. To go back to the current configuration after checking the delta, click **Current Configuration**. This checkpoint function helps you to determine how fast a container is growing.

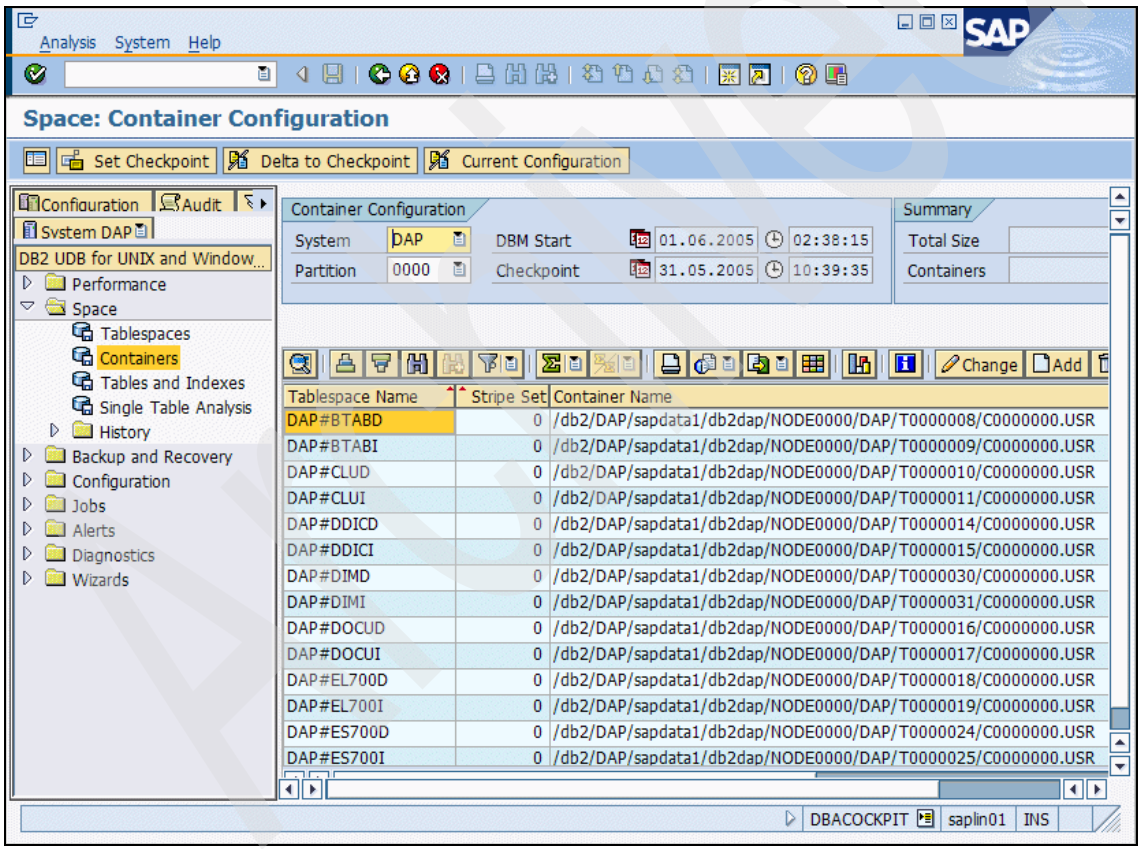


Figure 6-42 Configuring containers

Here we show you an example of adding a container to the existing DMS table space.

Select a container which belongs to a table space where you want to add a new container, and click **Add**. The *Change Tablespace* screen comes up with the Containers tab. (As the title of the screen indicates, this is the function used when a table space is changed.) As you see in Figure 6-43, an entry for a new container is automatically generated. Change it as you want. For example, if you want to add a new container to a new stripe set, select **-NEW-** in the Stripe Set column. After inputting all values, click **Execute**.

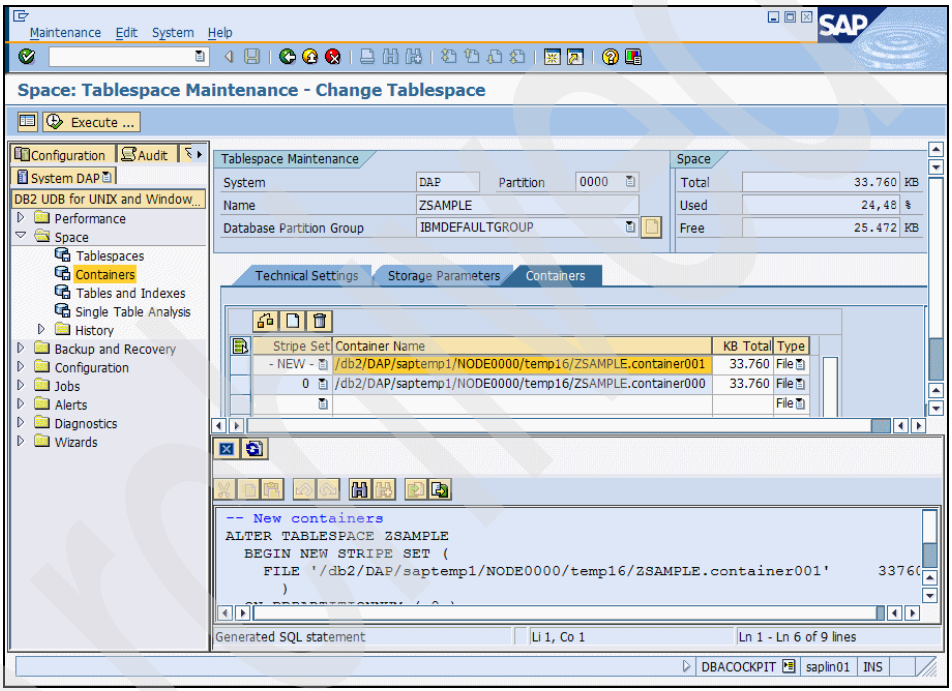


Figure 6-43 Add a container to an existing table space

Tip: In case of a multi-partitioned database, you have to add a container to all partitions of the corresponding database partition group.

For more information about DB2 UDB table space containers, refer to 4.1, “DB2 UDB logical and physical database objects” on page 80.

6.5.3 Tables and indexes

To see which table or index is recommended to be reorganized, this view can be used. Go to **Space** and click **Tables and Indexes**. See Figure 6-44. This information is stored in special tables called `DB6TREORG` and `DB6IREORG`. Therefore, this function is not available for system monitored via remote database connections. REORGCHK job in the DBA Planning Calendar writes this information to those special tables.

For more information about the DBA Planning Calendar, refer to 6.9.1, “DBA Planning Calendar” on page 266.

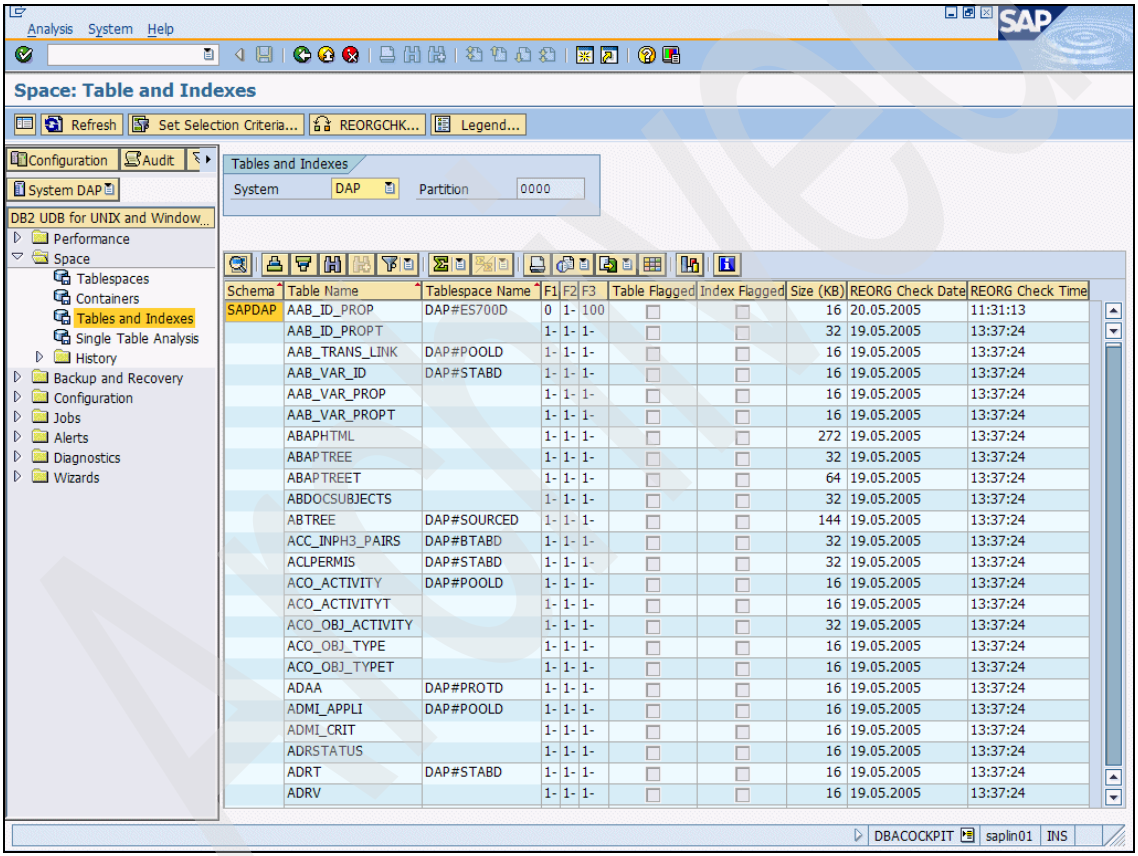


Figure 6-44 Tables and indexes

When you first start this function, you may see a dialog box as in Figure 6-45. The tables displayed in the dialog box have one of the following conditions:

- There is no entry for that table in table `DB6TREORG`.

- There is an entry in table DB6TREORG, but information in DB6TREORG is not up-to-date. (Information in DB6TREORG differs from system catalogs.)

This means that you need to adjust the information in the table DB6TREORG. To adjust information immediately, click **REORGCHK**. Then a dialog box comes up to specify the start time. You can start the REORGCHK immediately or schedule it to later. This REORGCHK job runs in background.

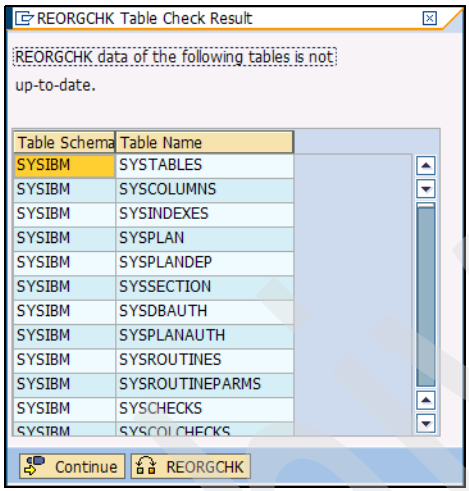


Figure 6-45 REORGCHK result

After *REORGCHK Table Check Result* dialog box, the *Selection Criteria* dialog box comes up to limit the data to be displayed. (See Figure 6-46.) You can also sort the data by using *Sort by* option. To change the selection criteria afterward, click **Set Selection Criteria**.

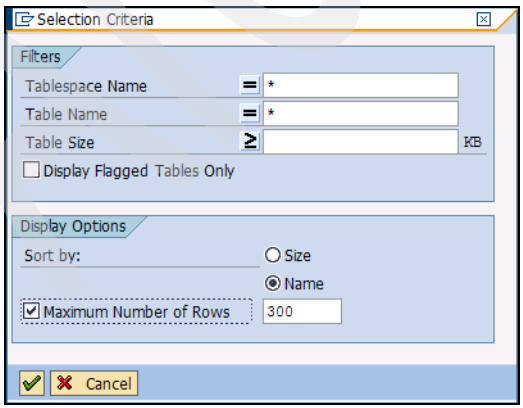


Figure 6-46 Selection criteria

To see detailed information, double-click an entry in Tables and Indexes (Figure 6-44 on page 227), then you can get information such as table structure, index structure, and so on. (See Figure 6-47.) Actually, this function is the same as the Single Table Analysis function, which is located under Tables and Indexes in the Space task area. For more information about Single Table Analysis, refer to 6.5.4, “Single table analysis” on page 229.

6.5.4 Single table analysis

You can check statistics information for each table and index, such as cardinality, number of overflow records, table size, and so on.

Go to **Space** → **Single Table Analysis**, and input name of a table in the *Name* field. Then you see the detailed statistic information in the screen (Figure 6-47). This statistical information is in the *Table* tab and the *Index* tab. Also, you can check the structure of a table or an index from the *Table Structure* tab or the *Index Structures* tab. If you have several indexes in the table, click **Next** or **Previous** to see the information for another index.

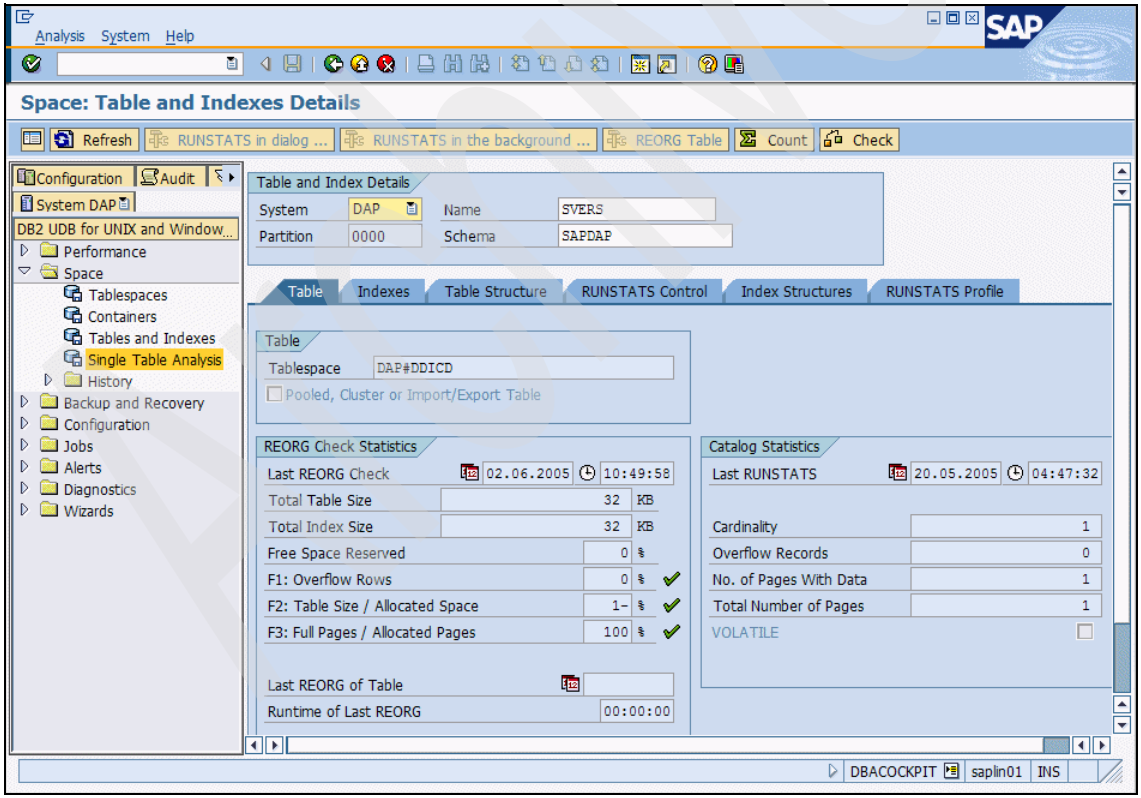


Figure 6-47 Detailed information on a table

To check the quality of the statistics for a table, you can click **Count**. This action counts the current rows in the table. After counting, two fields, “Count Rows” and “Deviations” are added to the *Table* tab. (These fields are marked in blue. See Figure 6-48.) If the deviation is more than 15%, we recommend that you perform RUNSTATS on this table.

Note: Counting rows may take a long time.

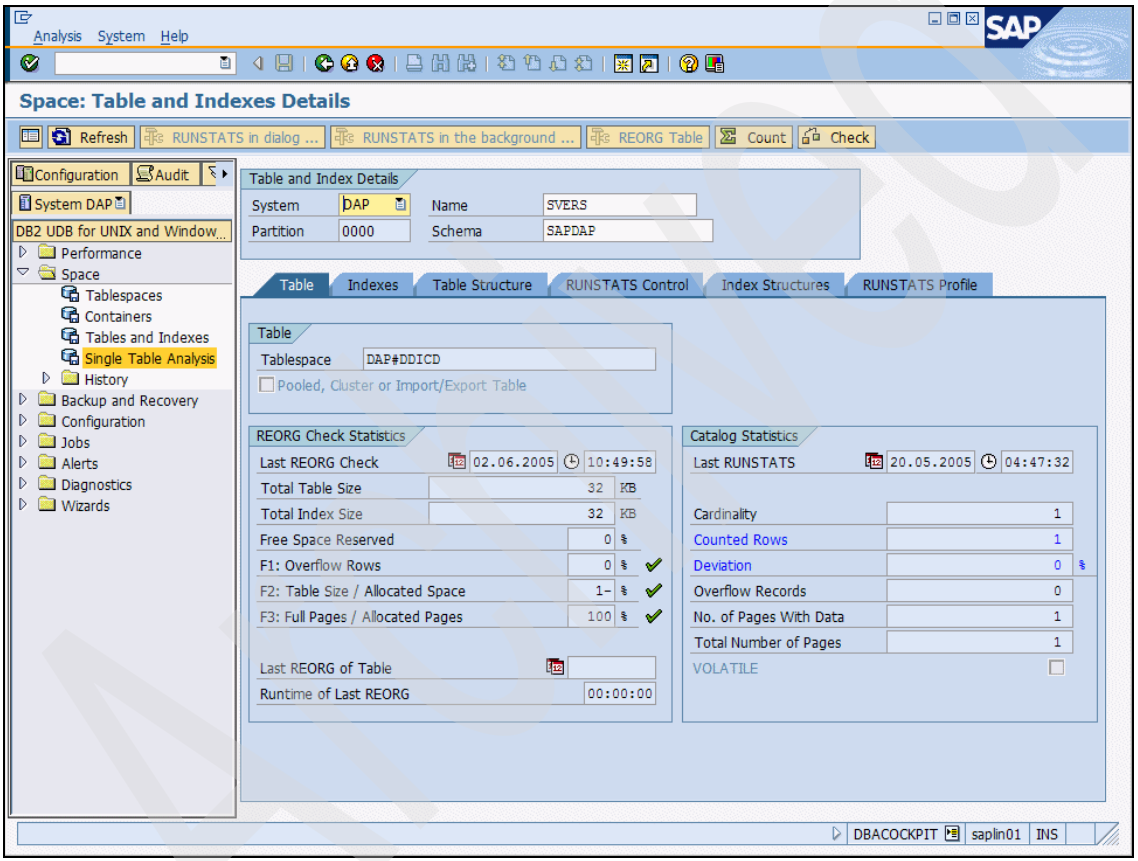


Figure 6-48 Table statistics after count

Here it is possible to perform RUNSTATS in dialog/background and REORG for each table. If you perform RUNSTATS for a large table, we recommend that you use *Runstats in the background*. Also, if you set up a RUNSTATS profile, you can use the *Runstats Profile* tab to perform RUNSTATS with your profile.

Runstats control

If automatic statistics collection is disabled, you see the *Runstats control* tab as shown in Figure 6-49. From here, some options for RUNSTATS can be changed. If you save these changes, a subsequent RUNSTATS job via DBA Planning Calendar uses the saved settings.

Tip: The option "*None*" for the table analysis method or index analysis method, means that RUNSTATS does not overwrite the statistics, but it also does not delete or deactivate old statistics.

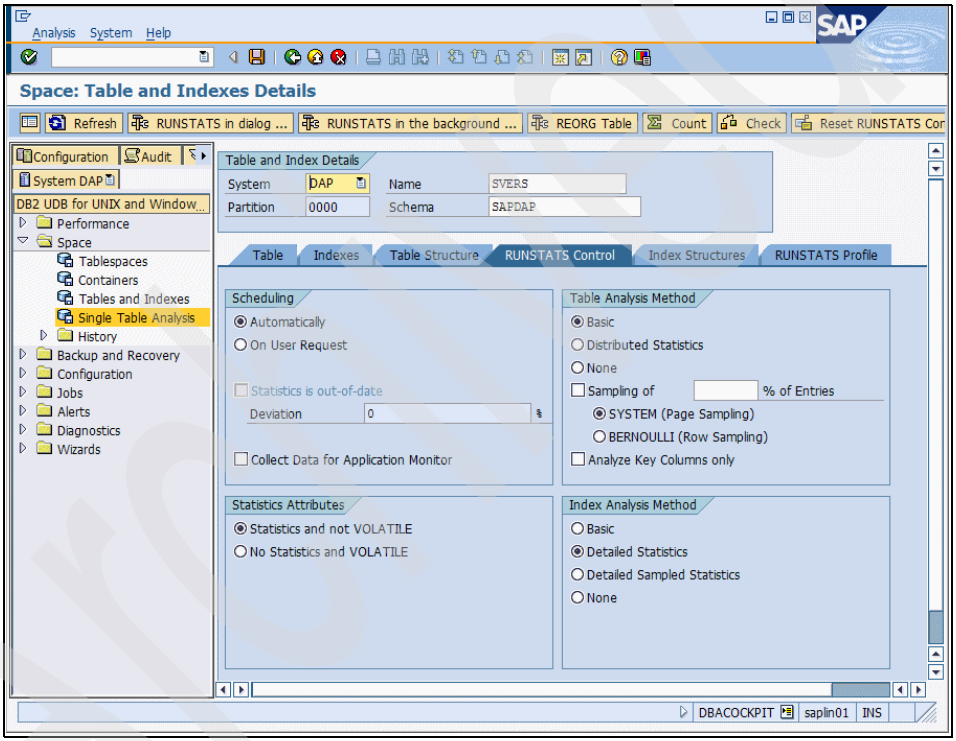


Figure 6-49 Runstats Control if automatic statistics collection is off

When you click **Runstats in background** and if the option is changed and not saved yet, you get a dialog box as shown in Figure 6-50. This dialog box asks you whether you want to use the changed option only for this RUNSTATS, or you want to use saved options for this RUNSTATS, or you want to save the changed option for subsequent RUNSTATS jobs and use the changed option for this RUNSTATS.

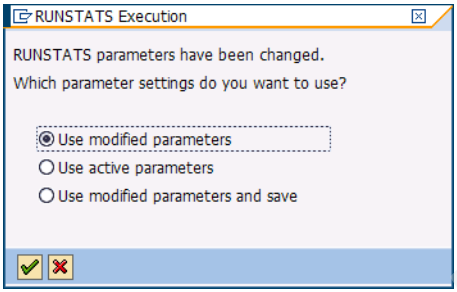


Figure 6-50 Confirmation for Runstats option change

If you want to reset the *Runstats Control* settings to the standard, click **Reset RUNSTATS Control**.

Note: SAP systems have special settings for some tables. If you use Reset RUNSTATS Control, those special settings are lost. This may result in performance degradation.

If automatic statistics collection is enabled, you see the Runstats Control tab as in Figure 6-51. Compared to Figure 6-49 on page 231, you may notice that there is no scheduling option. It is because automatic statistics collections perform RUNSTATS automatically. If you mark this table as not volatile, automatic statistics collection does not take statistics for this table.

If you change the analysis method for a table, this setting is not used by automatic statistics collection. This means that you cannot explicitly change RUNSTATS settings for automatic statistics collection.

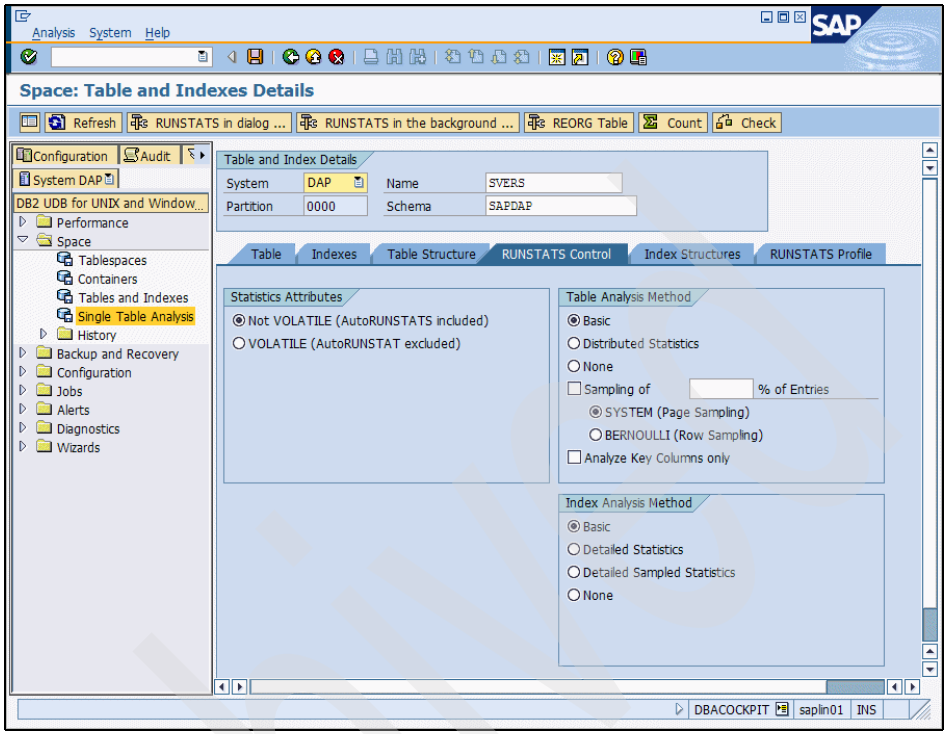


Figure 6-51 Runstats Control if automatic statistics collection is on

6.5.5 History

In the SAP environment, sometimes it is difficult to determine beforehand which tables are getting larger. To determine the change of space size, table size, or index size easily, space history data can be used. In this task area, you can see the history of a database, table space, table, and index size. This history related to space usage is kept in the special tables in the database. Therefore, this function is only available for local and remote systems monitored via RFC.

Overview

This screen just gives you current available information about database, tables, and index size. See Figure 6-52 for information on a database. This data is collected by the `DB Collector` job which is automatically scheduled by the system. Information on tables and indexes is calculated from statistics. So if the statistics are not up to date, the information in these tabs may not be accurate.

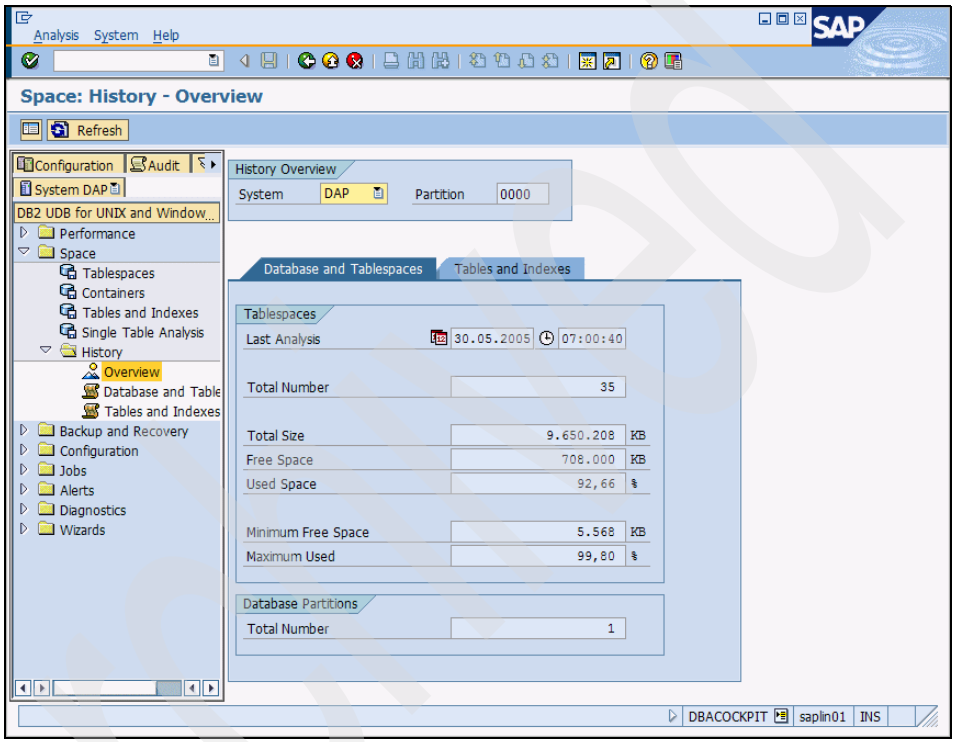


Figure 6-52 History Overview - Database and tablespaces

Database and tablespaces

To see how much a database grew last week, or how much space a table space is occupying recently, is very important for database administration. In this function you see the delta values of space usage from the database level or table space level. By default, the database level history is displayed. Figure 6-53 shows the database level space history.

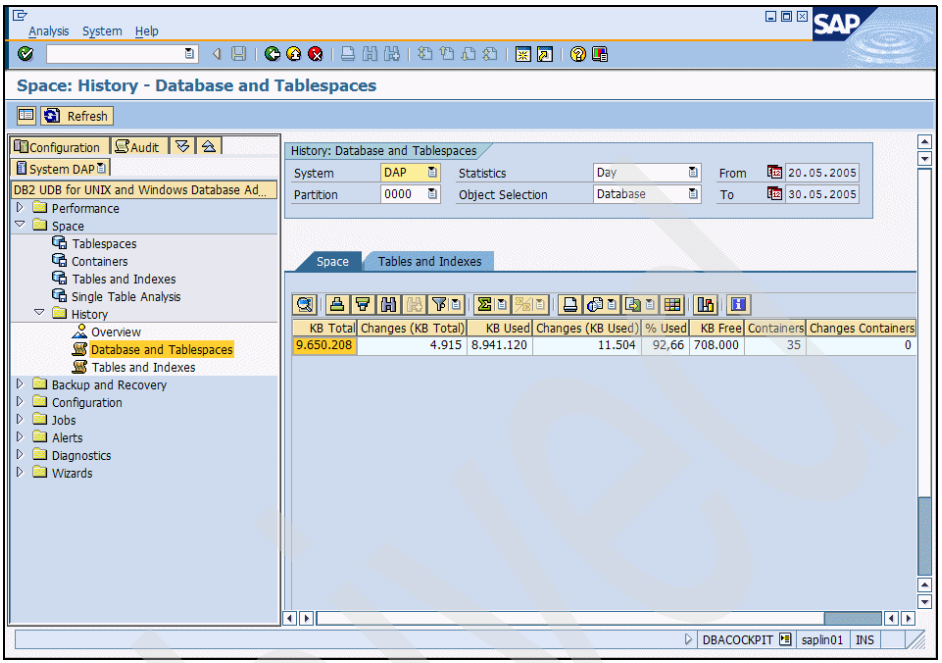


Figure 6-53 Database space history

If you want to see the delta value for each day, click **Details**. Figure 6-54 shows delta value at day level. It is also possible to display the delta value by week or month. Click **Statistics** field to make them weekly or monthly. To show table space level information, click **Object Selection** and select **Tablespace**.

The difference between two tabs, *Space* and *Tables and Indexes*, is the way the value is calculated. For the *Space* tab, it calculates the value from table space size information. For *Tables and Indexes* tab, it calculates the value from each table or index size determined by `RUNSTATS`.

It is important to know how fast the database is growing. We recommend that you monitor this space history information weekly.

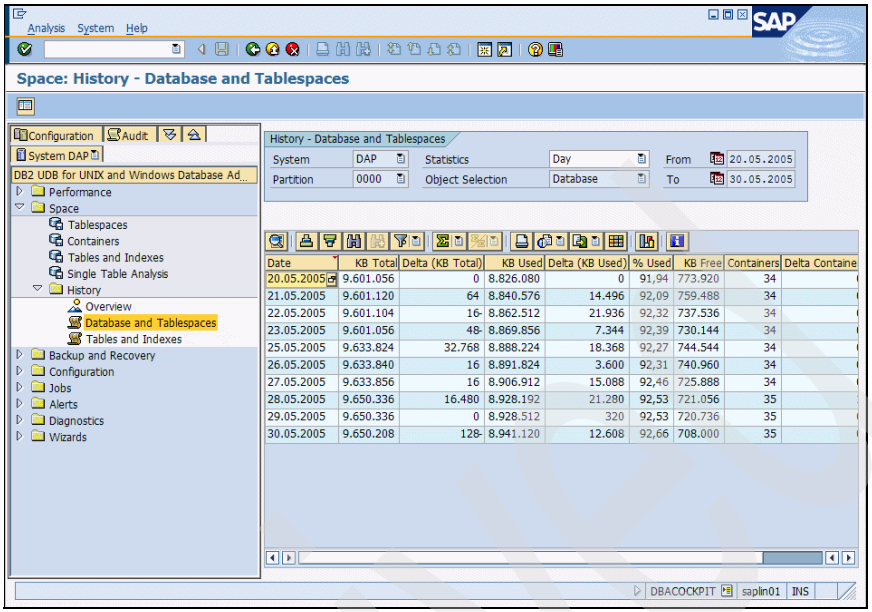


Figure 6-54 Database daily space history

Tables and indexes

How large does the table grow these days? The answer for this question can be determined by this function. You can see the delta value for each table and index size by day, week, or month.

When you first enter this function, the window Figure 6-55 comes up for selecting objects you want to get the information. Because the SAP system has so many objects, it is more efficient to limit the objects to be displayed. To change criteria afterwards, click **Set Selection Criteria**.

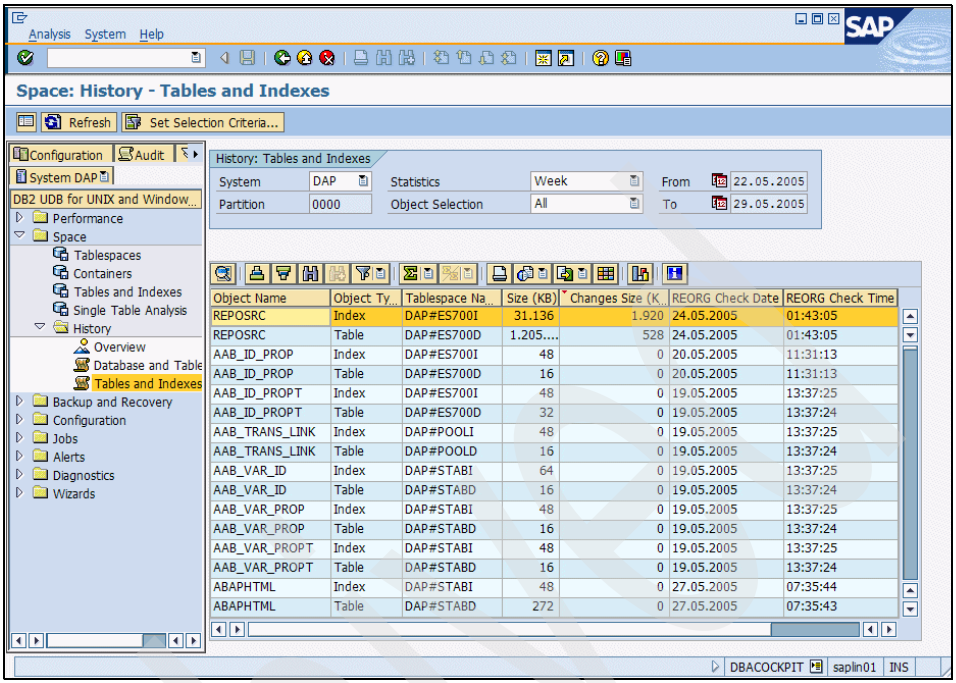


Figure 6-55 Tables and Indexes

To limit the objects displayed, select Table, Index, or All in the **Object Selection** field in the Tables and Indexes screen. (See Figure 6-56.) Even if there are several indexes in one table, there is only one entry for the indexes. The space consumed by each index is summed and displayed as one entry. To see detailed information for each object, select an object in the list and click **Details**.

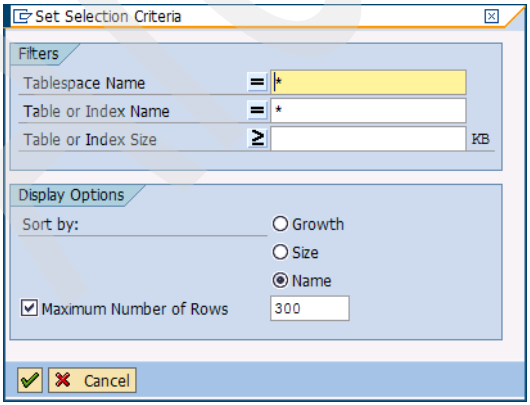


Figure 6-56 Selection criteria

6.6 Backup management

It is mandatory to check the result of a backup operation to ensure that your system can be recovered in case this becomes necessary. You can check the result of DB2 UDB backup operation and log archiving. Also, you can check logging parameters from the *Backup & Recovery* task area.

6.6.1 Backup Overview

Go to **Backup & Recovery** and double-click **Backup Overview in the navigation frame** to check the result of DB2 UDB backup and log file archiving.

Figure 6-57 shows *Database Backup* tab in Backup Overview.

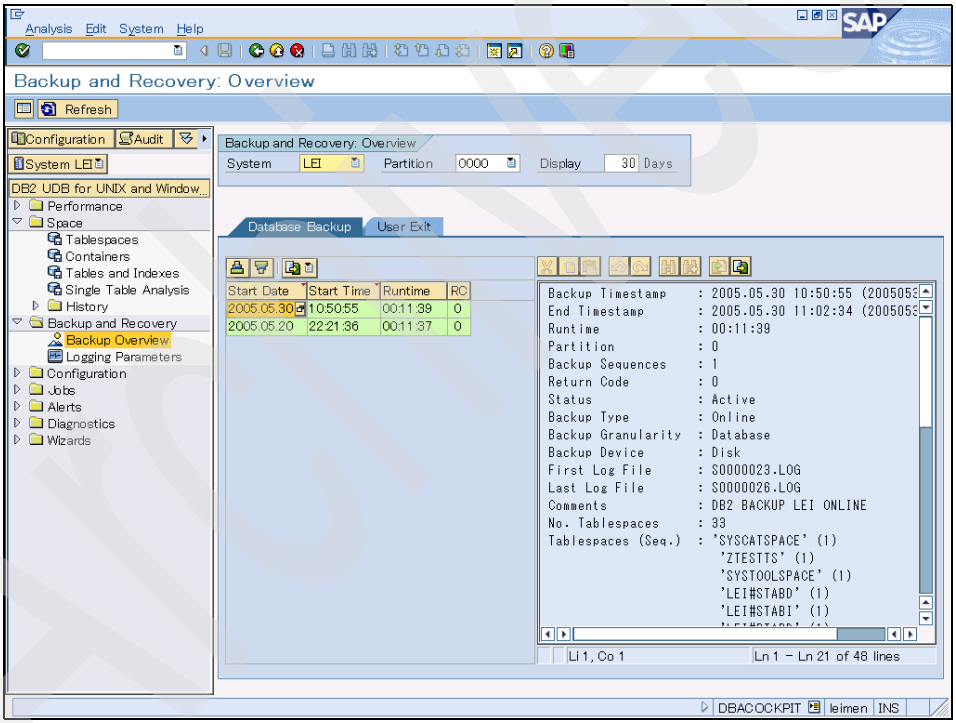


Figure 6-57 Database backup result

Database Backup

The *Database Backup* tab displays the status of DB2 UDB backup jobs. If you double-click an entry in the left table, you see the detailed information for that entry on the right frame. Also, you can check the job log for these backup jobs from transaction SM37 or you can see the Job Log from the DBA Planning Calendar. For more information for the DBA Planning Calendar, refer to 6.9, “Scheduling DBA tasks” on page 266.

Note: Information displayed in the *Database Backup* tab comes from the DB2 UDB history file. Therefore, you see the status of all DB2 UDB backups including backups executed from outside of DBA Cockpit. You do not see the job log for those backups executed from outside of DBA Cockpit in transaction SM37.

Tip: You can display all entries in DB2 backup history file using the DB2 UDB LIST HISTORY command:

```
db2 list history backup all for <DBSID>
```

The successfully completed backup jobs are shown in the left table in green. The entry shown in yellow indicates that the backup has completed with warning messages. The red colored entry shows that the backup has ended with errors.

When you perform an offline backup from DBA Cockpit (DBA Planning calendar), the entry can be marked in yellow or red, even though it has completed successfully. We recommend that you check the status of all backup jobs by reading the detailed information on the right frame or by using transaction SM37. To avoid confusion and risk, if you schedule your backup from DBA Cockpit (DBA Planning Calendar), we recommend that you use online backup jobs with the INCLUDED LOGS option instead of offline backup.

For more information about offline backups from DBA Cockpit, refer to SAP Note *427982*: *DB6: Offline backup ends with job error (DB13)*.

Log Archiving

In the *User Exit* tab, you see the information for log files that have been archived from log_dir to archive target (disk, Tivoli® Storage Management, and the other vendor library). This information comes from the DB2 UDB archive log history file. In case of errors, some entries are written to db2diag.log.

Figure 6-58 shows archiving information in the *User Exit* tab.

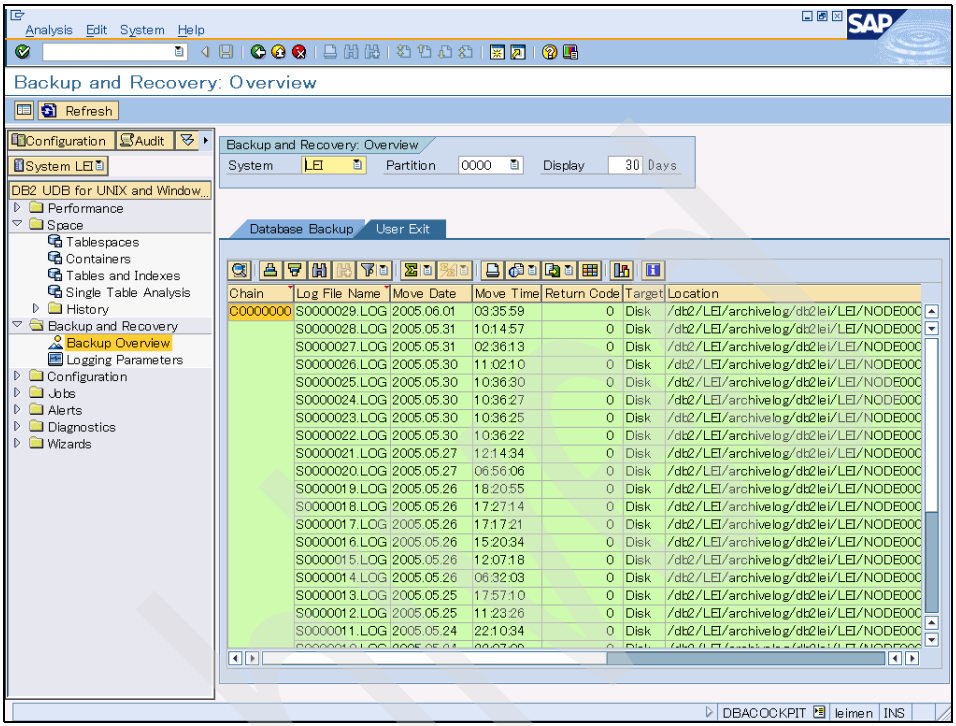


Figure 6-58 Log archiving result

Tip: You can display all entries in DB2 archive log history file using the db2 list history command:

```
db2 list history archive log all for <DBSID>
```

6.7 Log file management

To access the log file management information, open the *Backup and Recovery* option in the task area and double-click the Logging Parameters option in the navigation frame.

Using this option, you see the configuration of the logging facilities of DB2 UDB. Since SAP NetWeaver 2004s, the DB2 log manager is the default log management. We describe here how its information is shown in the SAPGUI logon utility.

For SAP systems using the DB2 log manager, you can see information on the SAP system and the partition you are monitoring at the top of the Logging Information frame.

The *Log Directory* tab displays information about the logging facilities configured in the database; see Figure 6-59. The information includes:

- Directory frame:
  - Name: This field is the current directory where all active log files are created and stored.
  - Files Total: This field indicates the total number of files created in this directory.
- File System frame:
  - Available Space: This field shows the total amount of free space, in Kb, available in the file system where the directory specified by the field Name is created. For Linux operating systems, the output of the column `Available` from the **`df -k`** command is shown.
  - Used Space: This field shows the total amount of space, in Kb, used by the active log files. For Linux operating systems, the output of the column `Used` from the **`df -k`** command is shown.
  - Filling: This field shows the percentage of used space in the file system. For Linux operating systems, the output of the column `Use%` from the **`df -k`** command is shown.
- First active log file:

  This field contains the name of the log file that is currently active. It shows the value of the `LOGHEAD` database configuration parameter.
- Size of log files:

  This field indicates what is the size of each log file. The unit of measure for this field is 4 Kb. It shows the value of the `LOGFILSIZ` database configuration parameter.
- Number of primary log files:

  This field shows how many primary logs are configured in your system. It shows the value fo the `LOGPRIMARY` database configuration parameter. To read more about this parameter, see 7.1, “Log file management” on page 314”.
- Number of secondary log files:

  This field shows how many logs will be allocated and used, if there is a need for log space during a recovery operation. It shows the value fo the `LOGSECOND` database configuration parameter. To read more about this parameter, see 7.1, “Log file management” on page 314”.

- Directory content window:

  In this text area, the contents of the log directory are shown. For Linux operating systems, the output of the **ls -la** command is shown.

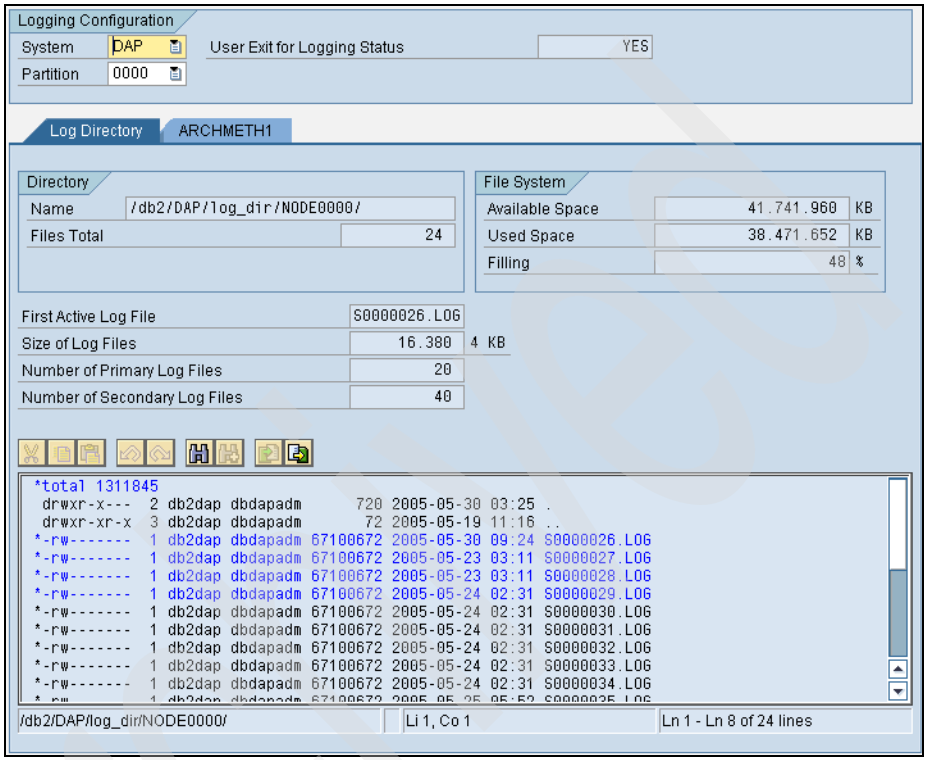


Figure 6-59 Log directory tab from Logging Parameters frame

The ARCHMETH1 tab (Figure 6-60) is a tab that corresponds to the LOGARCHMETH1 database configuration parameter. In this tab, you can find:

- Directory frame:

  In this case, as the LOGARCHMETH1 database configuration parameter is configured to DISK, the name and files total fields are shown.

- Options for LOGARCHMETH1 database configuration parameter:

  These are the options set in the LOGARCHOPT1 database configuration file.

- Directory content pan:

  This window shows the content of the target media configured in the LOGARCHMETH1 database configuration parameter. If the target media is disk directory, for a Linux operating systems, the output of the **ls -la** command is shown.

Figure 6-60 shows how this information is shown on screen, when a disk directory is used in the LOGARCHMETH1 database configuration parameter.

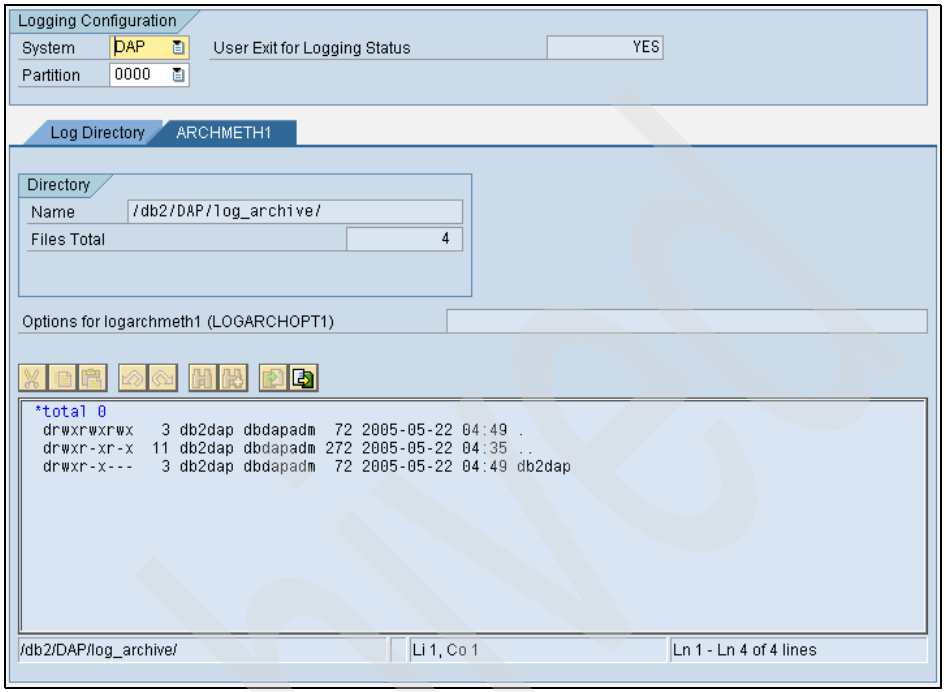


Figure 6-60 LOGARCHMETH1 tab from Logging Parameters frame

If you have set the LOGARCHMETH2 database configuration parameter, there will be another tab with the name LOGARCHMETH2. The output of this tab is very similar to the output of the LOGARCHMET1 tab; the information you see depends on the backend repository you have configured.

Note: Information displayed in the *Logging Parameters* tab comes from the database configuration file or from directories and backend repositories configured for logging purposes.

Tip: To get similar information as displayed by the *Logging Parameters* tab, you can use the following command:

```
#db2 get db cfg for <database_name>
```

Then look for the logging parameters described in 7.1, “Log file management” on page 314.

6.8 Configuration

DBA Cockpit: The Configuration task area allows you to perform these tasks:

- Viewing current and previous database configurations and database manager configurations
- Listing and maintaining the database partitioning groups and maintaining the buffer pool
- Configuring `RUNSTATS` settings
- Running various CLP commands
- Maintaining data classes
- Configuring monitor settings

6.8.1 Database Manager

This folder allows you to view the value of the DB2 UDB database manager configuration. It is organized into the following logical groups. You can access each group by choosing the corresponding tab (see Figure 6-61).

- Common:

  Common information such as release level and CPU speed

- Diagnostics:

  Information about the default monitor switches, health monitor, and diagnostic level for db2diag.log and notifying level for the database notification log

- Security:

  Information on groups and authentications of the database manager and on clients

- Memory:

  Values of various instance level memory related database manager configuration parameters

- Applications:

  Information on agents and the database application remote interface (DARI)

- Sync Point Manager:

  Information on the configuration of the synchronization manager and the transaction manager

- Communications:

  Information on network characteristics such as communication protocols and DCE directory services; in a situation involving a communication error problem, check if the `SVCENAME` database manager configuration parameter is set properly

- FCM:

  Information about the fast communication manager, only available for the Database Partitioning Feature

- Discovery Services:

  Information about the configuration of discovery mode

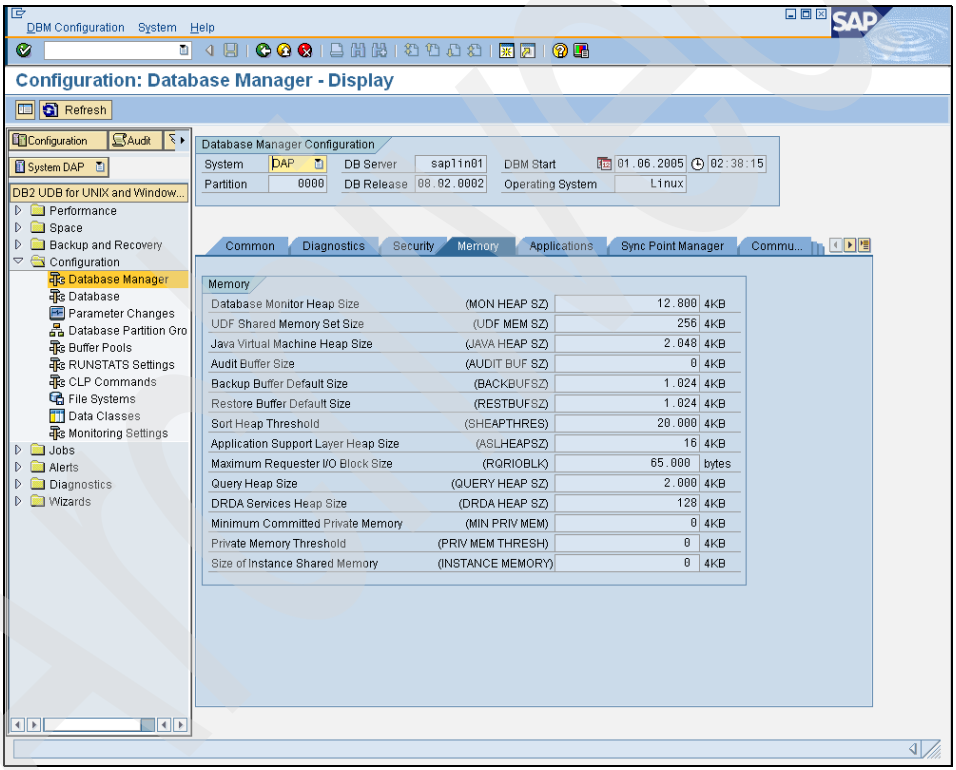


Figure 6-61 Configuration: Database Manager - Display

6.8.2 Database

This screen allows you to view and modify the value of the database configuration. It is organized into the following logical groups. You can access each group by choosing the corresponding tab (see Figure 6-62).

- Common:

  Common information for the database configuration, such as collating sequence and country code

- Performance:

  Information on optimization and I/O, such as degree of parallelism and number of I/O servers

- Memory:

  Values of various database level memory related database configuration parameters such as database heap size and sort heap size

- Logging:

  Information on logging and logging parameters, such as number of primary log files and log file size

- Log File Management:

  Information on log file management, such as first log archive method and overflow log path

- Backup and Recovery:

  Information on backup and recovery (such as index recreation time and whether there is backup pending mode); and TSM configurations

- Locks:

  Information on locking (for example, maximum storage for lock list and intervals for checking deadlocks).

- Space:

  Information on table spaces and containers, such as default number of containers and default table space extent size

- Application:

  Application related information, such as the average number of active applications maximum number of active applications

- High Availability:

  Information on the settings of High Availability Disaster Recovery (HADR) feature

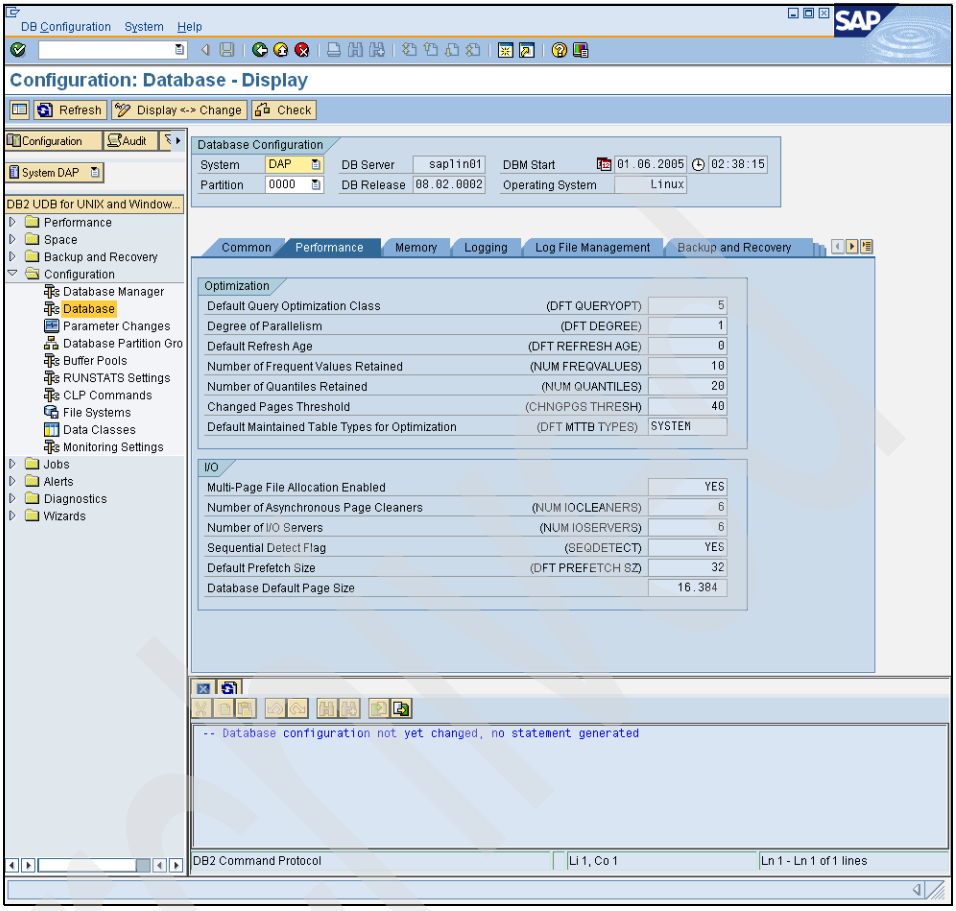


Figure 6-62 Configuration: Database - Display

You can modify the values of the database configuration parameter by clicking the **Display <-> Change** button. You now see the additional **Execute** button. The color of the fields changes to white and now you can type in a new value for the value you would like to change.

You can perform an input check for the new value by clicking the **Check** button. The **Check** button performs the following checks:

- Type checking:

  For example, characters are not allowed for a numeric database configuration parameters.

- Range checking:

  It checks if the values falls into the allowable range for a database configuration parameter. For example, the value for the database configuration parameter `dbheap` cannot be a negative number.

When you are ready, click the **Execute** button to commit the change. At this point, DBA Cockpit issues an `UPDATE DATABASE CONFIGURATION` command to update the new value for the database configuration parameter(s). You can see the generated CLP command at the lower half of the screen.

6.8.3 Parameter Changes

The main feature of the Parameter Changes feature (see Figure 6-63) is that you can view a history of the changes to the database manager configuration parameters and the database configuration parameters: approximately when the change was made and the value used at that time. Every now and then (the default is daily), a background job is run to collect a snapshot of all the *changed* values in the database configuration parameters, and the database manager configuration parameters and the delta changes are stored in a history table.

The history table keeps track of the parameter changes made in DBA Cockpit as well as the parameter changes made with the CLP (that is, `UPDATE DB CFG/UPDATE DBM CFG`). This screen is helpful in the situation when you suspect that a recent problem on the SAP system is related to a parameter change in the past. You can try to match the date/time of the change in parameter to see if the problem appears at about the date/time after the change in the parameter was made.

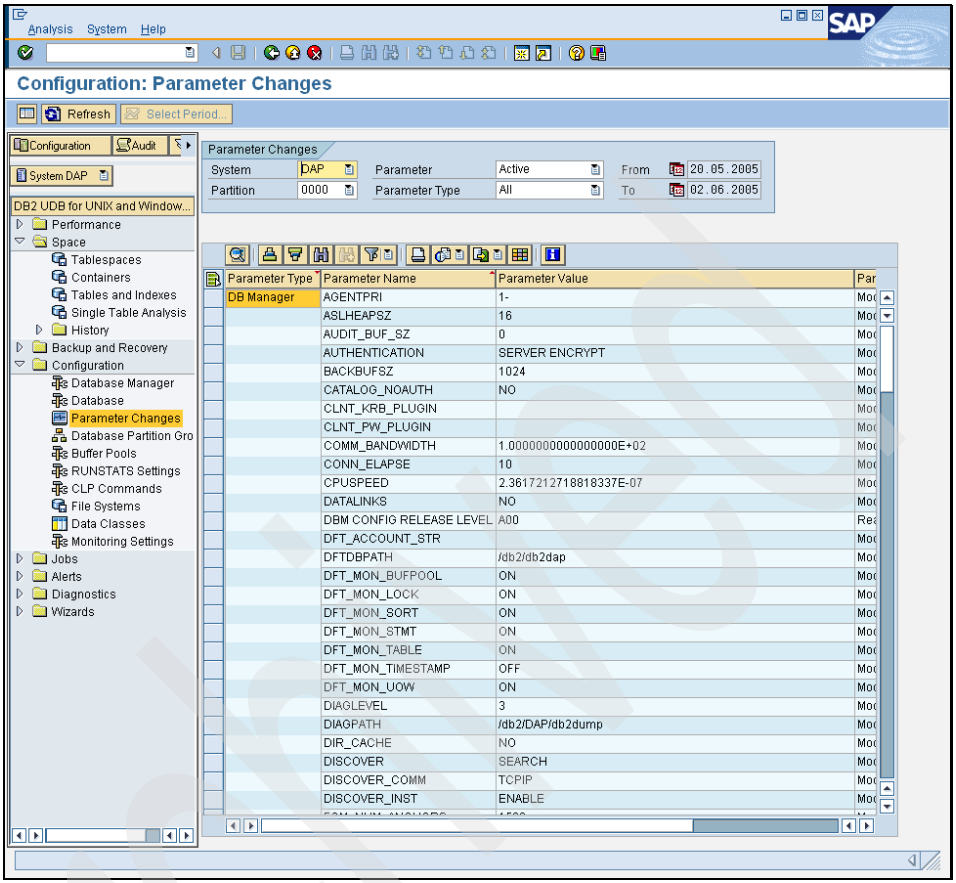


Figure 6-63 Configuration: Parameter Changes - Active Parameters

When you choose **Parameter** → **Active**, the current active values of the parameters are shown. If you can choose **Parameter** → **History**, a dialog box pops up (see Figure 6-64). There are two options:

- All:

  See all the parameter changes since the system is installed.

- Between <beginning datetime of a period> AND <ending datetime of a period>:

  See the parameters changed during a certain datetime period.

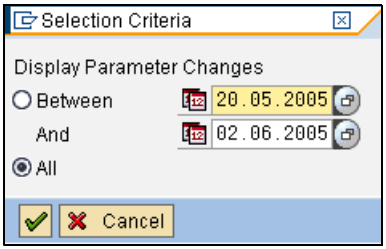


Figure 6-64 Configuration: Database -Parameter Changes - Selection Criteria

In Figure 6-65, for example, we have chosen **Parameter** → **History** → **ALL**. We see that the value of MON_HEAP_SZ was set to 128 on May 05, 2005 at 08:00:22 and the value of MON_HEAP_SZ was changed to 12800 on May 25,2005 at 13:00:25.

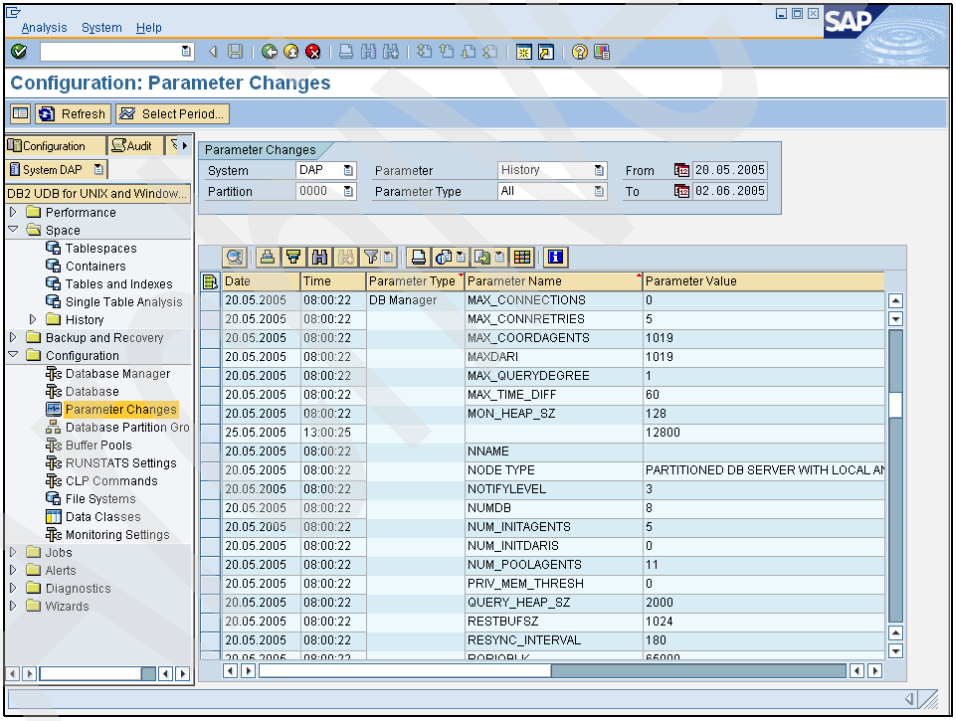


Figure 6-65 Configuration: Parameter Changes - History - All

Attention: Since the check for parameter changes is done once every day, if you make several changes to the same parameter between two checks (within a day), only the last change for this parameter is recorded in the history table for parameter changes.

6.8.4 Database partitioning group

A database partition group is a set of one or more database partitions. When you want to create tables for the database, you first create the database partition group where the table spaces will be stored, then you create the table space where the tables will be stored.

In the *Database Partitioning Group*, you can:

- View which database partitions a database partition group contains.
- Add a database partition group by clicking the **Add** button.
- Delete a database partition group by clicking the **Delete** button.
- Edit an existing database partition group by clicking the **Edit** button.

Viewing the database partition group

In the main screen of DBA Cockpit: Database Partition Group, the default behavior is to use the lowest partition number for the Partition field (usually “0000”), which means that only those database partitioning groups that have this partition are shown. If you want to see the database partition groups to database partitions assignment for all database partitioning groups and all the partitions assigned to each database partition group, you should choose *All* in the Partition field. You now see a screen similar to Figure 6-66.

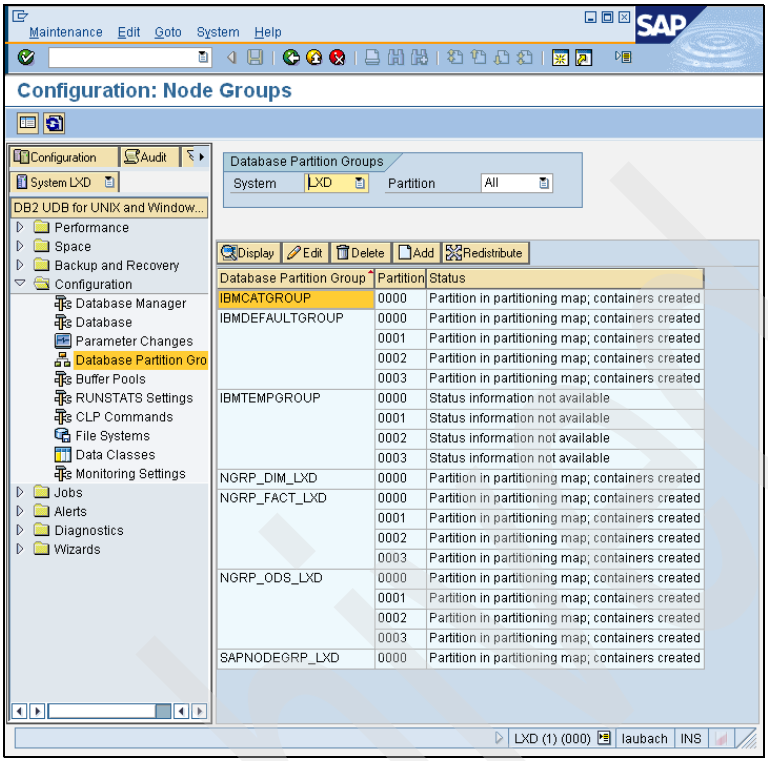


Figure 6-66 Configuration: Database Partitioning Group - view (All partitions)

For each database partition group, there are one or more rows. Each row represents one partition. For example, in Figure 6-66, the database partition group, `NGRP_FACT_LXD` has 4 partitions: `0000`, `0001`, `0002`, `0003`.

The column *Status* describes the current status of the database partition. Here is a list of the possible values for the Status column:

- Status information not available:

  Displayed for database partition group `IBMTEMPGROUP` or if the status cannot be determined.

- Partition not in partitioning map; containers not yet created:

  Partition has been created without containers and is not yet referenced in the partitioning map.

- Partition not in partitioning map; containers created:

  Partition and containers have been created, but partition is not yet referenced in the partitioning map.

- Partition in partitioning map; containers created:

  Partition will be dropped after next redistribution.

To view the detail of a database partition group, click the **Detail** button or double-click one of the lines for the database partition group. You see the following information:

- Partitions tab: Lists all the partitions assigned to this database partition group.
- Buffer Pools tab: Lists all buffer pools which are available to this database partition group.
- Tablespaces tab: Lists all the table spaces which are associated to this database partition group.

Adding database partition group

Here are the steps to create partition group:

1. In the main screen of the *Database Partitioning Group* screen, click the **Add** button.
2. You now see the screen for adding database partition group (Figure 6-67). Type in a name for the new database partition group in the field *Name*.

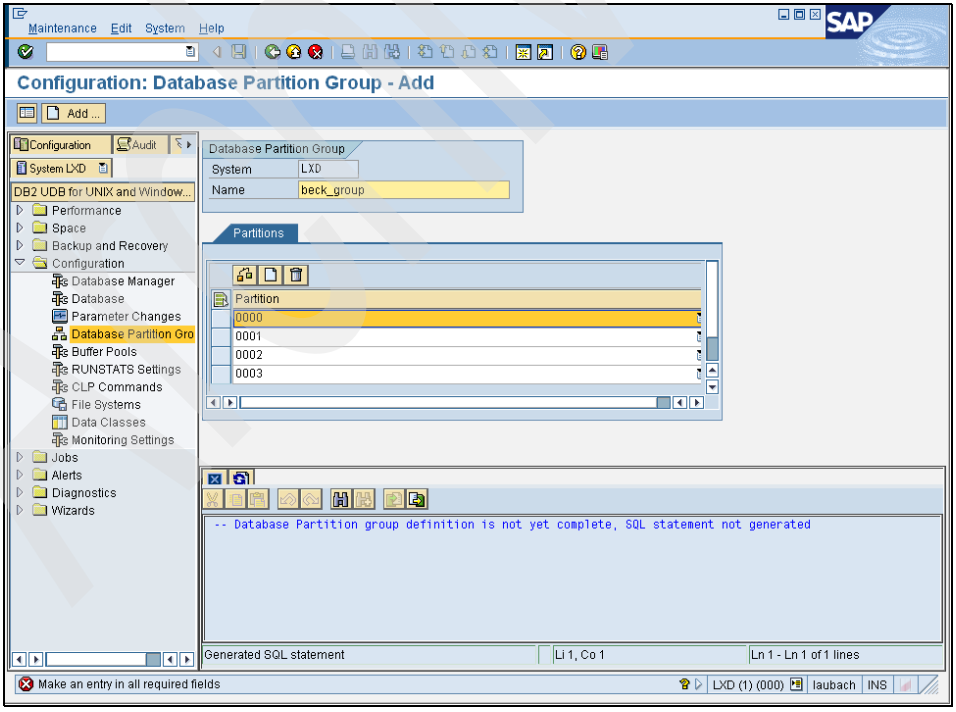


Figure 6-67 Configuration: Database Partitioning Group - Add

3. By default, all available partitions are listed to be added to the new database partition group. You can use the add partition button or delete partition button to choose the partition(s) to be used in this database partition group.
4. When you are done with the modification, click the **Add Node Group** button Add ... to create the new database partition group.

Editing database partition group

In the main screen of the Database Partitioning Group, choose one of the rows of the database partition group that you would like to change. Then, click the **Edit** button. You see a screen similar to Figure 6-68.

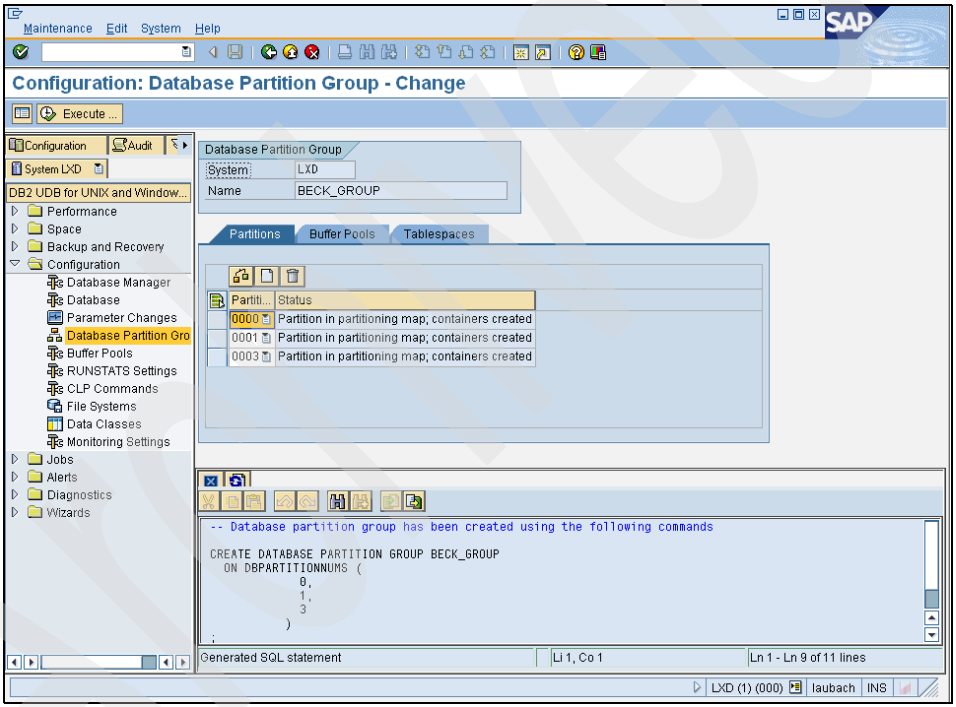


Figure 6-68 Configuration: Database Partitioning Group - Edit

In this screen, you can use the Add Partition button or Delete Partition button to change the partitions assignments. Click the **Execute...** button Execute ... to confirm the changes.

Redistributing database partition group

You can only redistribute database partition groups for which the partition and containers have been created, but the partition is not yet referenced in the partitioning map (the status *Partition not in partitioning map; containers created*). This function redistributes the data among partitions according to the partitioning map.

In the main screen of the Database Partitioning Group, choose one of the rows of the database partition group that you would like to change. Click the **Redistribute** button and a scheduling screen of the DBA Planning Calendar appears. You can choose to redistribute the database partition group immediately or at a later time.

6.8.5 Buffer Pools

DBA Cockpit: Configuration - Buffer Pools allows you to maintain the buffer pools in your database. In this folder, you can:

- Change buffer pools: Add or remove partitions, resize, and control the use of extended storage.
- Add new buffer pools by clicking the **Add** button.
- Drop existing buffer pools by clicking the **Delete** button.

Viewing buffer pools

In a Database Partitioning Feature (DPF) enabled system, a buffer pool can be used for multiple partitions; depending on the definition of the database partition group that the buffer pool belongs to. If you want to see all the buffer pool to partition assignments for all partitions, choose *All* in the Partition field (see Figure 6-69). In the list you see the value of size and the page size defined on each partition for a buffer pool.

Important: A value of -1 in the Size(Pages) column indicates that the default buffer pool size parameter from the database configuration is used (parameter `BUFFPAGE`). If this value is displayed and you want to see the real size of the buffer pool, you should use the buffer pool snapshots (DBA Cockpit: **Performance** → **Buffer Pools**.

To view the details of a buffer pool, click the **DETAIL** button or double-click one of the lines for the buffer pool. You see the following information for this buffer pool, in the following tabs:

- Technical Setting tab: The size (in pages) of the buffer pool for each partition, the page size, block size, and number of blocks pages, and whether the buffer pool uses extended storage.
- Database Partition Group tab: A list of all the partitions that this buffer pool is related to.
- Tablespaces tab: A list of all the table spaces that use this buffer pool.

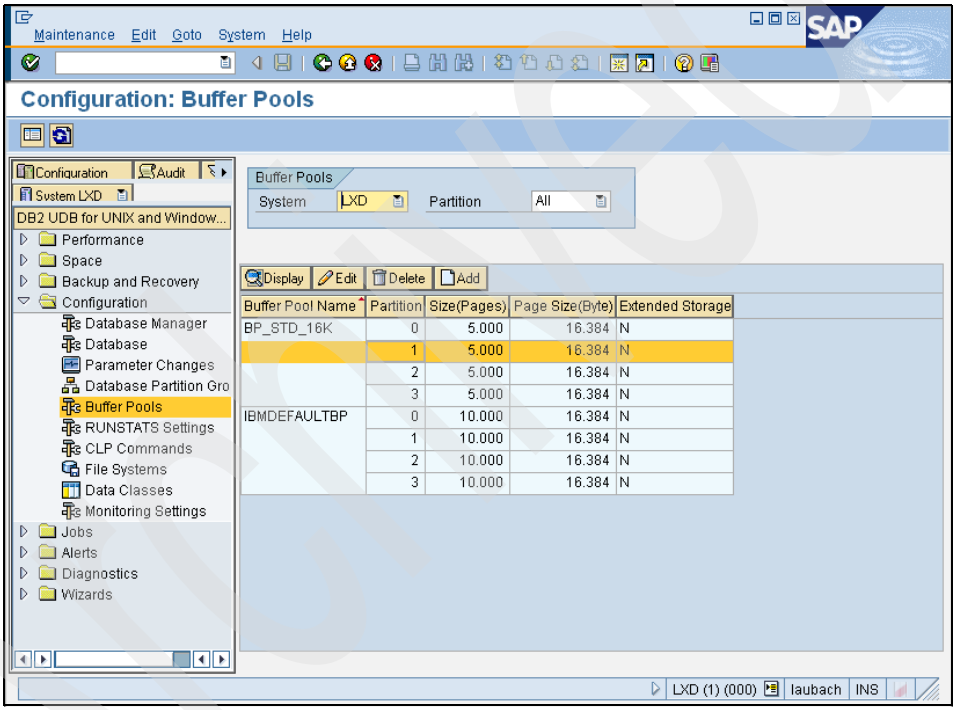


Figure 6-69 Configuration: Buffer Pools -View (All partitions)

Changing buffer pools

If you would like to modify the buffer pool (any partition that the buffer pool is related to), choose one of the rows of the buffer pool and click the **Edit** button. You see a screen similar to Figure 6-70.

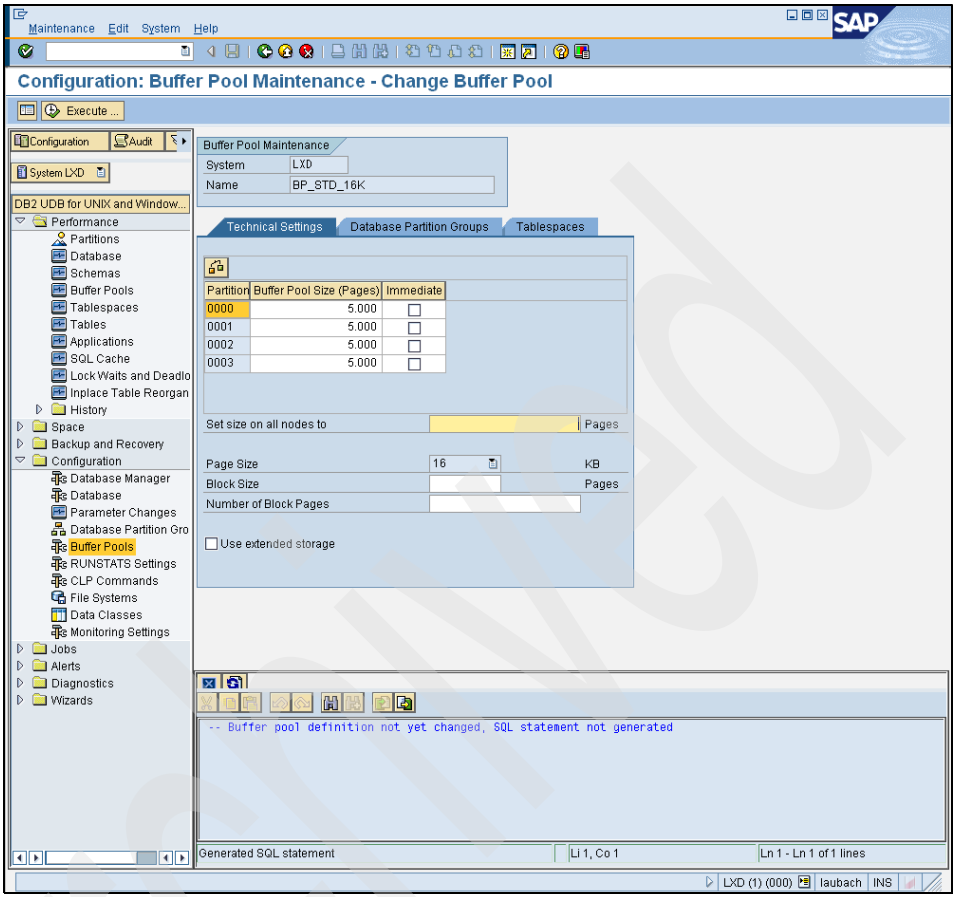


Figure 6-70 Configuration: Buffer Pools -Edit

Under the column Buffer Pool Size, you can modify the size (in pages) of the buffer pool in each of the partitions that the buffer pool relates to. Or, if you would like to modify the buffer pool to be the same size on all partitions, you can enter a value (in pages) in the *Set size on all nodes to* field.

Under the Immediate column, you can choose Immediate for a particular partition. If the check box under the Immediate column is checked, the changes to the buffer pool on that particular partition will take effect immediately.

After you have made all the necessary changes, click the **Execute ...** button to confirm the changes.

Adding buffer pools

Here are the steps to add a buffer pool:

1. In the main screen of the Buffer Pools, click the **Add** button. You see the *Add Buffer Pool* screen as shown in Figure 6-71.

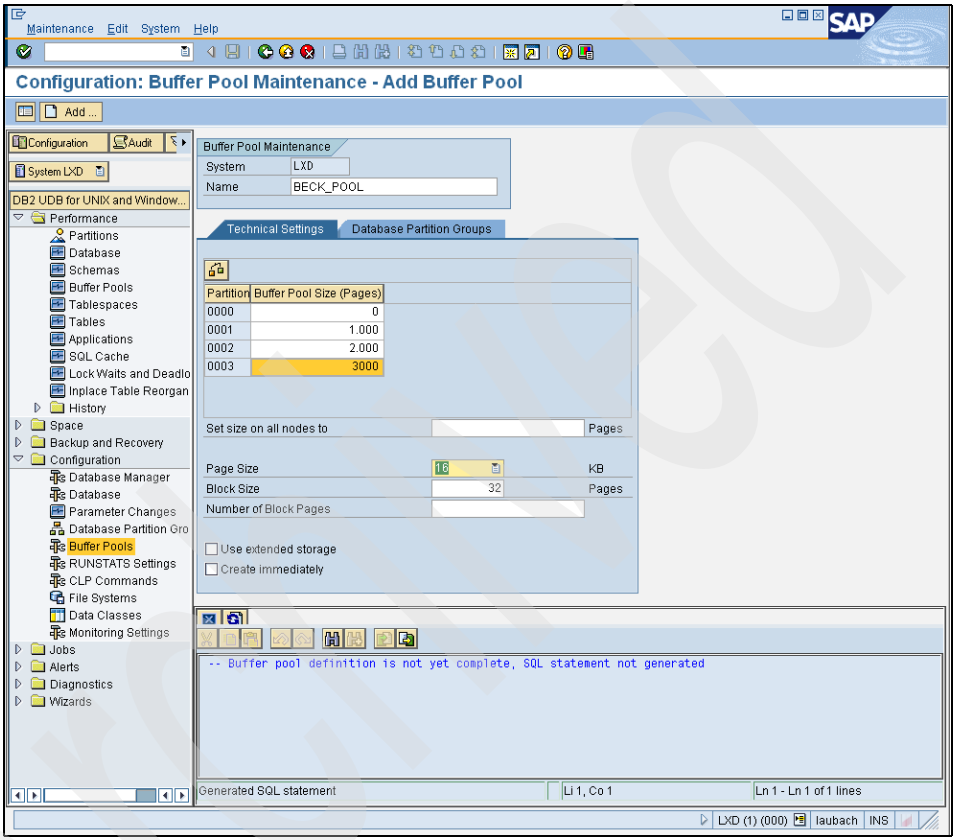


Figure 6-71 Configuration: Buffer Pools - Add Buffer Pool

2. Enter the name of the new buffer pool in the *Name* field.
3. Decide on which partition(s) you are going to create your buffer pool and how big (in pages) you want your buffer pool to be on these partitions. In Figure 6-71, we are assigning 1000 pages for partition #1, 2000 pages for partition #2, and 3000 pages for partition #3. Specify the other desired technical settings such as Page Size, Block Size and Number of Block Pages. You can choose to create the buffer pool immediate by checking the *Create immediately* check box. When you are done with the modification, click the **Add ...** button to create the new buffer pool.

6.8.6 Special RUNSTATS settings and volatile tables

In this screen (Figure 6-72), there are two kinds of special tables shown here:

- Tables that are marked as VOLATILE in the system catalogs. A volatile table is defined as a table whose contents can vary from empty to very large at run time. The statistical data is often run out-of-date and may result in a wrong access plan being chosen by the DB2 optimizer. By declaring the table volatile, the optimizer will consider using an index scan rather than a table scan. The access plans that use declared volatile tables will not depend on the existing statistics for that table.
- Tables with RUNSTATS control parameters that are not in accordance with DBA Cockpit standards. When you schedule a RUNSTATS job in transaction DB13, the default RUNSTATS settings are used. For example, for SAP products that based on SAP BASIS 640, the default setting is basic table statistics and detailed index. You can modify the RUNSTATS setting of a table by adding entries in the DBSTATC control table. Those tables with the customized RUNSTATS setting appears in the *Special RUNSTATS settings and Volatile Tables* screen.

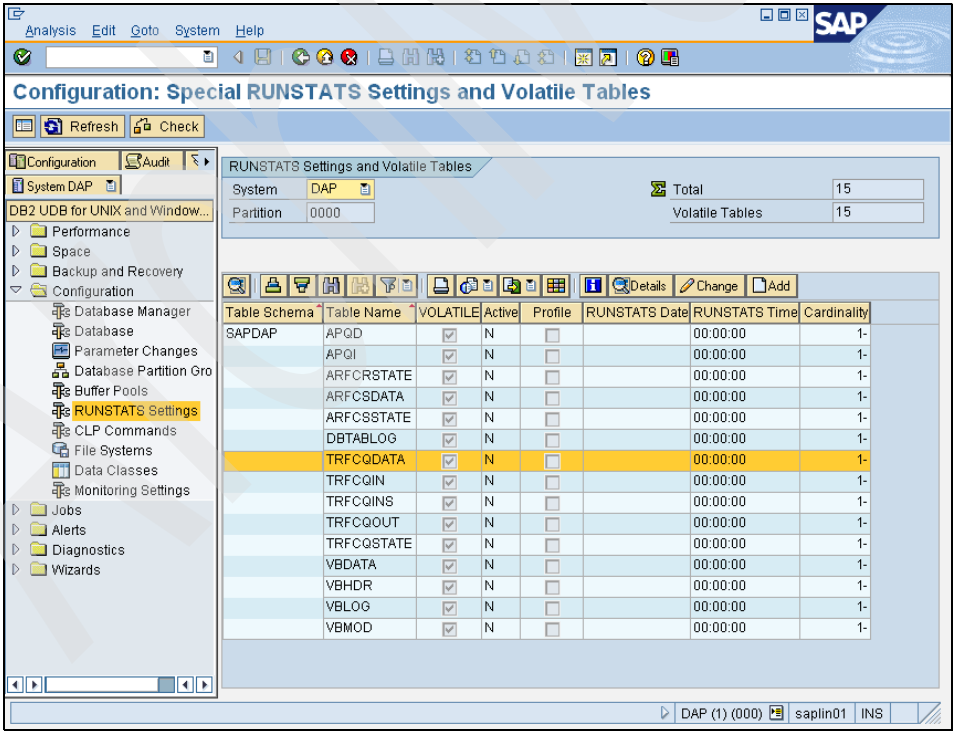


Figure 6-72 Configuration: Special RUNSTATS settings

6.8.7 CLP Command Execution

This folder contains a collection of useful “pre-packaged” CLP commands. You choose a CLP command from a list and execute to see the result of the CLP command. Here are a list the commands:

- DB2 Level
- DB2 Profile Registry
- Database Manager Configuration
- Database Configuration
- Tablespace Configuration
- Buffer Pool Configuration
- Buffer Pool to Tablespace Assignment
- Database Partition Group Configuration
- CLI Configuration
- Database Directory Configuration
- Partition Directory Configuration
- Database Manager Snapshot
- Database Snapshot
- Application Snapshot
- Buffer Pool Snapshot
- Table Snapshot
- Tablespace Snapshot
- Lock Snapshot

Some of them, such as *Buffer Pool to Tablespaces Assignment*, are particularly useful because there is no easy CLP command to find out the buffer pool to table spaces assignment. A complicated query to the system catalog table is necessary (that is, `SYSCAT.TABLESPACES`, `SYSCAT.BUFFERPOOLS`). Instead of writing the query every time, we can just go to DBA Cockpit: Configuration → CLP Commands and choose **Buffer Pool to Tablespaces Assignment** from the *Function* list.

See Figure 6-73 for an illustration of this concept.

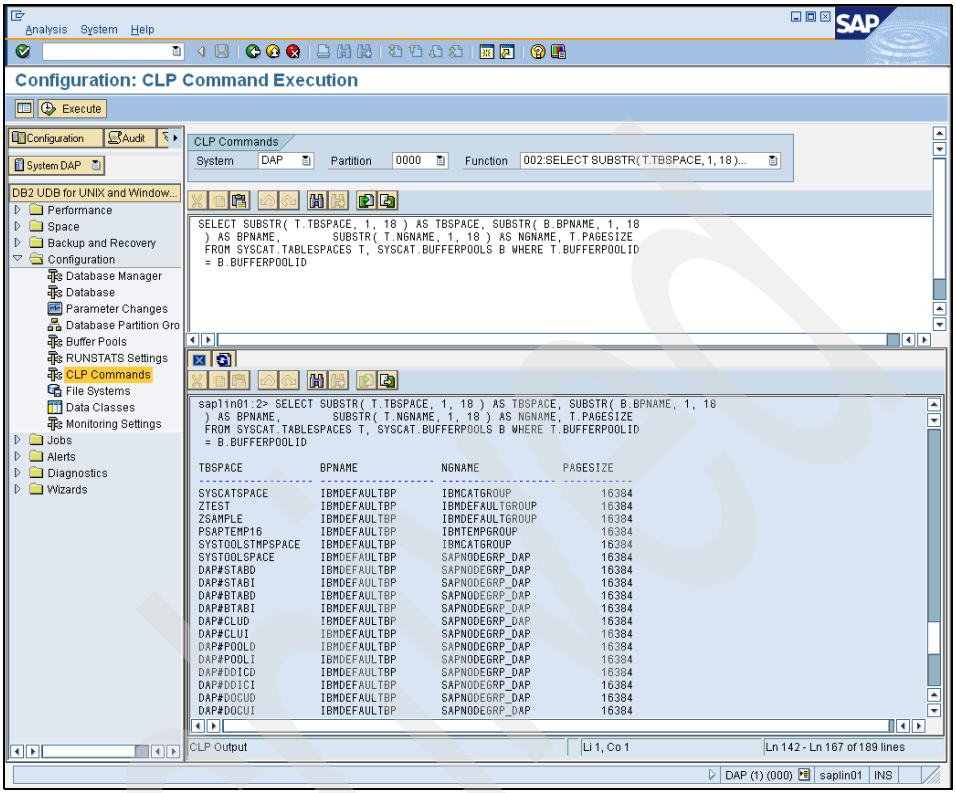


Figure 6-73 Configuration: CLP Command Execution - Buffer Pool to Tablespaces Assignment

In the latest version of DBA Cockpit, on top of running “pre-packaged” CLP commands, you can issue your own command as well (in the middle text area). This is useful when you want to run some query against the database but you just have SAP system access (no operating system level access) to a system. See Figure 6-74.

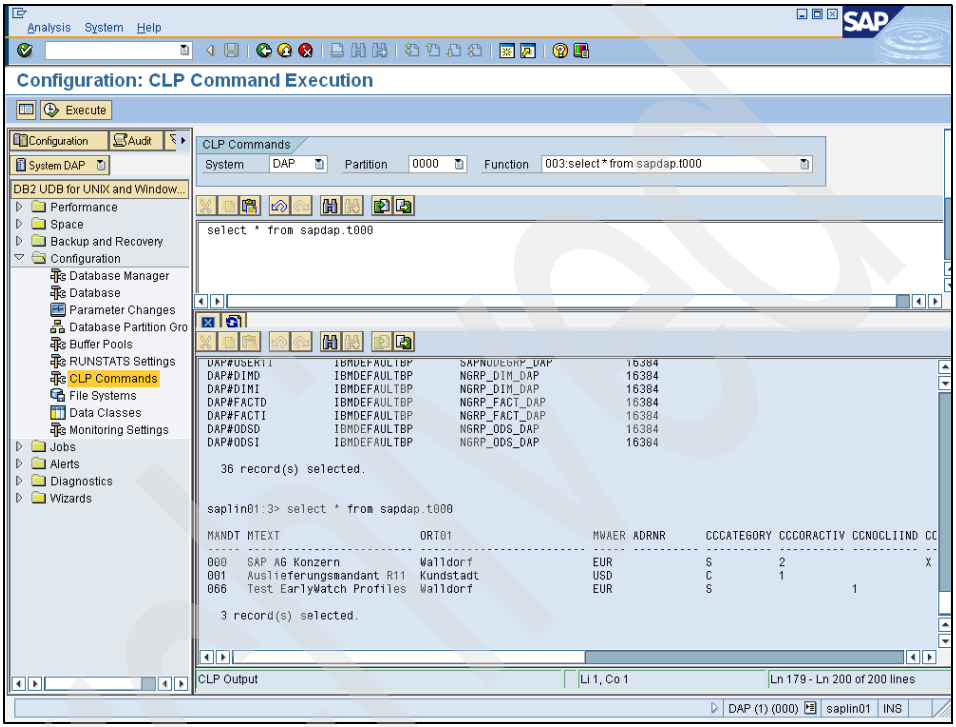


Figure 6-74 Configuration: CLP Command Execution - execute your own query

6.8.8 File Systems

This folder is helpful in determining how much space is available to the file systems. We recommend that you monitor DB2 UDB related files such as the transaction log file. The *Percent Used* or *Percent Free* columns should be noted. Figure 6-75 is the main screen of File Systems folder.

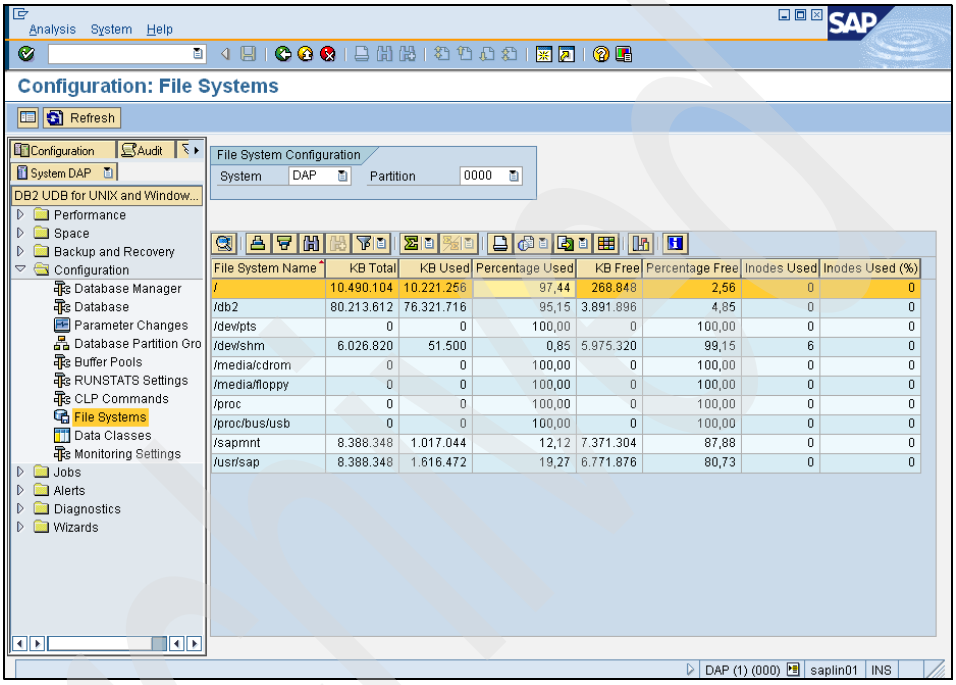


Figure 6-75 Configuration: File systems

6.8.9 Data Classes

The technical settings of SAP tables define data classes that need to be related to database table spaces. Refer to 4.3, “SAP data classes (TABARTs)” on page 100 for more information on data classes. In the main screen of Configuration: Data Classes (see Figure 6-76), you see a list of all the data classes on the system; and which data table space and index table space they are associated to. Also, you see the number of tables and indexes defined in a data class.

In Figure 6-76, the first column *State* (traffic light) checks the following considerations:

- Is there a related table space for data?
- Does the data tablespace exist in the database?
- Is there a related table space for indexes?
- Does the index table space exist in the database?
- Does the name of the table space comply with the naming conventions for the customer name space?
- Is there a description for the data class?

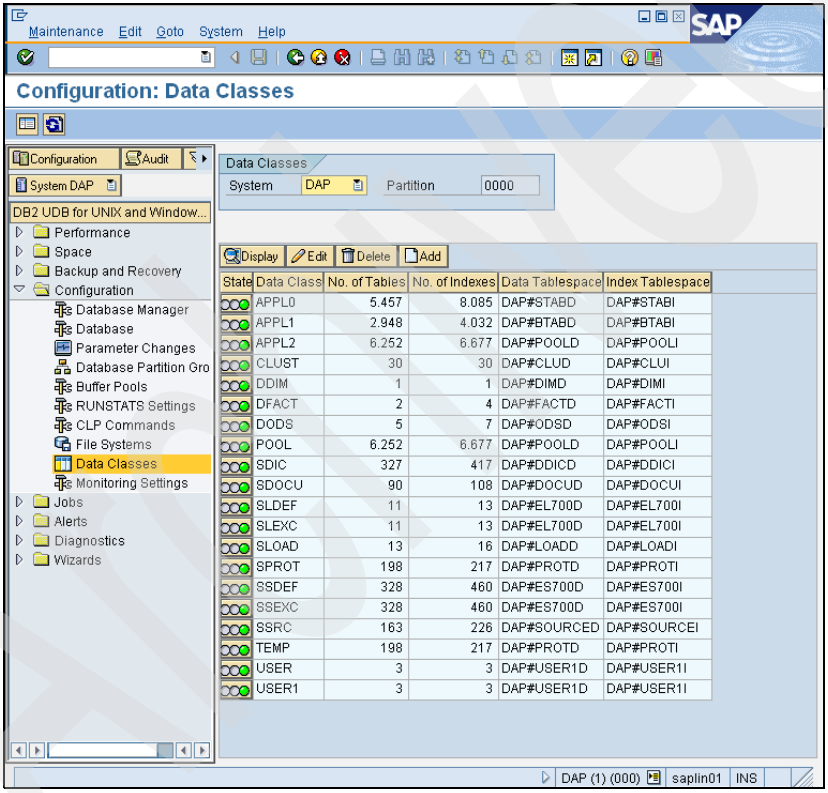


Figure 6-76 Configuration: Data Classes - main screen

Changing Data Classes

Here are the steps to change the data classes:

1. Select the data class you would like to change. Click the **Edit** button.
2. In the *Change Data Class* popup dialog (Figure 6-77), you can change the description, data table space assigned to the data class, and the index table space assigned to the data class.

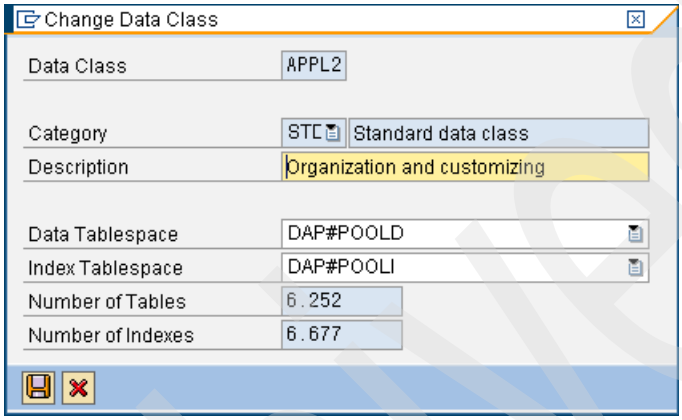


Figure 6-77 Configuration: Data Classes - changing data classes

3. Click the **Save** button to confirm the changes.

Adding Data Classes

Here are the steps to add data classes:

1. In the Data Classes main screen, click the **Add** button. You see the *Add Data Class* popup dialog (Figure 6-78). Enter the name of the new data class.

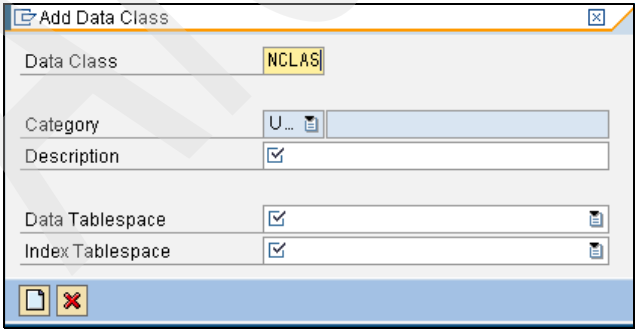


Figure 6-78 Configuration: Data Classes - adding data classes

2. Enter a description of the new data class. Enter a description that you can later tell what this new data class is used for. Select from a data table space and an index table space from the list boxes to be associated to the data class.
3. Click the **Add** button to add the new data class.

6.8.10 Monitoring Tool Settings

The Monitoring Tool Settings is used to configure the monitoring tools. This folder is discussed in 6.3.3, “Monitoring settings” on page 188.

6.9 Scheduling DBA tasks

When you need to schedule database administrative jobs such as backup or reorganization, go to Jobs in the navigation frame. You can see the following features: DBA Planning Calendar for scheduling jobs, DBA Log for checking logs related to database administration, and CLP Script Maintenance for creating your own scripts.

6.9.1 DBA Planning Calendar

When a DBA wants to schedule jobs related to a database, DBA Planning Calendar helps a lot. This may be one of the easiest ways to schedule database related jobs in SAP environment. DBA Planning Calendar provides function to schedule database related jobs periodically. The jobs scheduled from DBA Calendar are executed in background.

Overview

Figure 6-79 shows the DBA Planning Calendar. Go to **Jobs** and double-click **DBA Planning Calendar**, you see this screen.

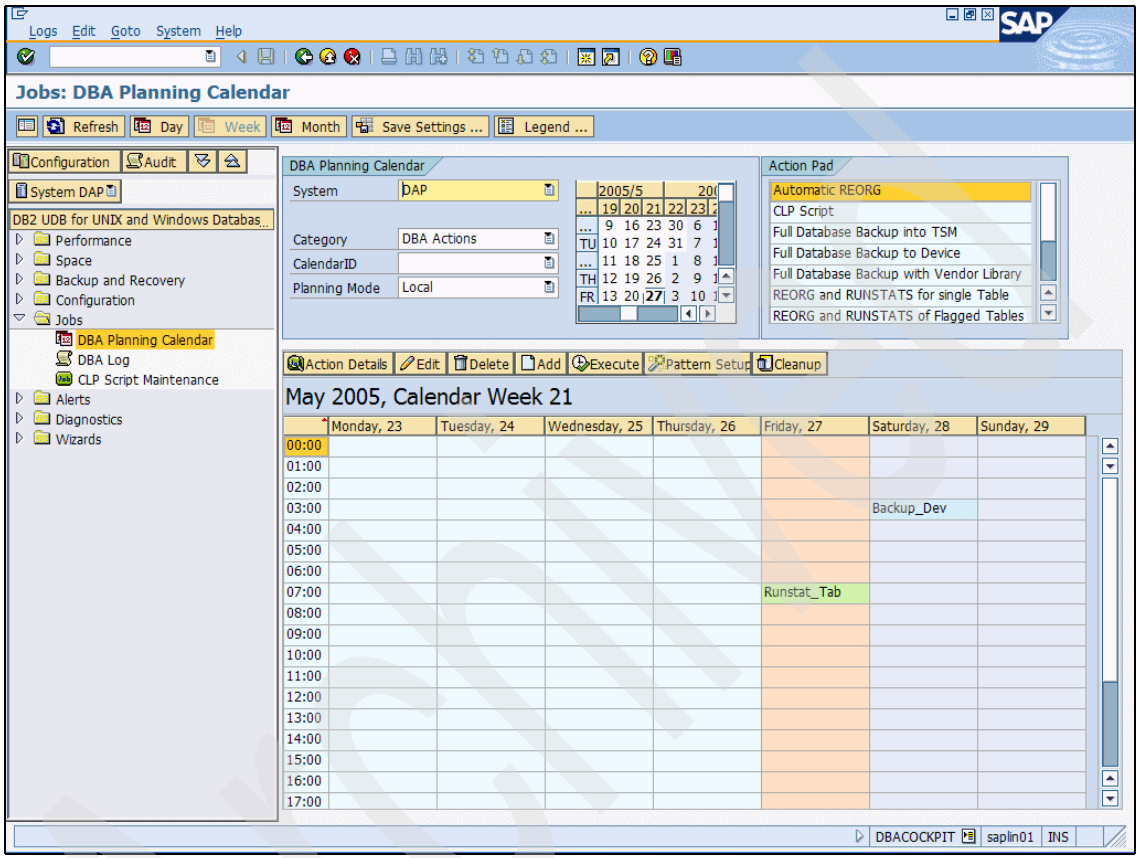


Figure 6-79 DBA Planning Calendar

You can change the job to be displayed in the calendar by using the field in the left frame in the upper half.

- Category:

  You can choose job type in this field. Two commonly used views are:

  - DBA Actions: This is the default. You can see the jobs which are scheduled by a user in the calendar.
  - DB Collectors: These jobs in the calendar are automatically scheduled by the system. Most of jobs are to collect performance/history data of the database. These jobs cannot be changed or deleted.

- Calendar ID:

  You can specify a calendar ID to set holidays for various countries. This only affects the color of the day in the lower half calendar. Holidays are in the same color as weekends.

- Calendar in the upper frame:

  This controls the day displayed in the lower half. The default displays current week. You can move to other weeks by double-clicking a date in the calendar.

In the lower calendar you can see the job scheduled. The color of the cell indicates the status of the job. If it is red, there is an error. You have to check the log. If it is yellow, the job finishes with warnings. You should check the log as well. If it is green, the job is completed successfully. If you have several jobs at same hour, you can only see one entry in the calendar which has the most severe error among those jobs.

To schedule database administration job and background job, two profiles, `S_RZL_ADMIN` and `S_BTCH_ALL`, are mandatory.

Scheduling DBA jobs

As you can see Figure 6-79 on page 267, there are pre-defined jobs such as `BACKUP`, `REORG`, or `REORGCHK` in the Action Pad. To schedule a job, you can just drag and drop the job you want to execute/schedule to the calendar, or else double-click the job in the Action Pad. Then you see a dialog box where you can specify the parameters for each job and recurrence if you want to run that job regularly.

Figure 6-80 shows an example of Full Database Backup to Device. From this window, you can specify the time the job should be run. Click **Execute** button if you want to run the job immediately.

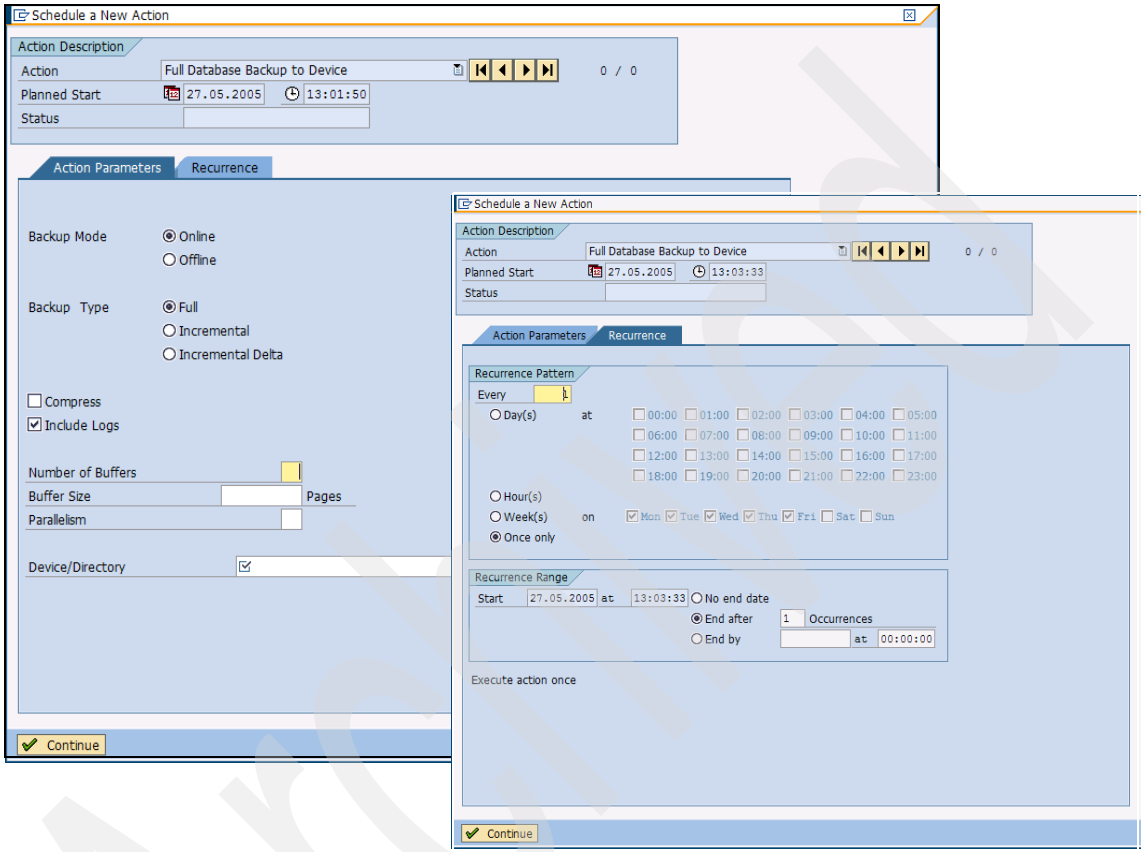


Figure 6-80 Full Database Backup to device

Pattern setup

DBA Cockpit gives some ideas for scheduling jobs by Pattern Setup. The suggested jobs under Pattern Setup are recommended to be run regularly. Figure 6-81 shows some screens from Pattern Setup. You can start scheduling recommended jobs by just clicking **Pattern Setup**. Parameters and schedule for each job can be specified as you want.

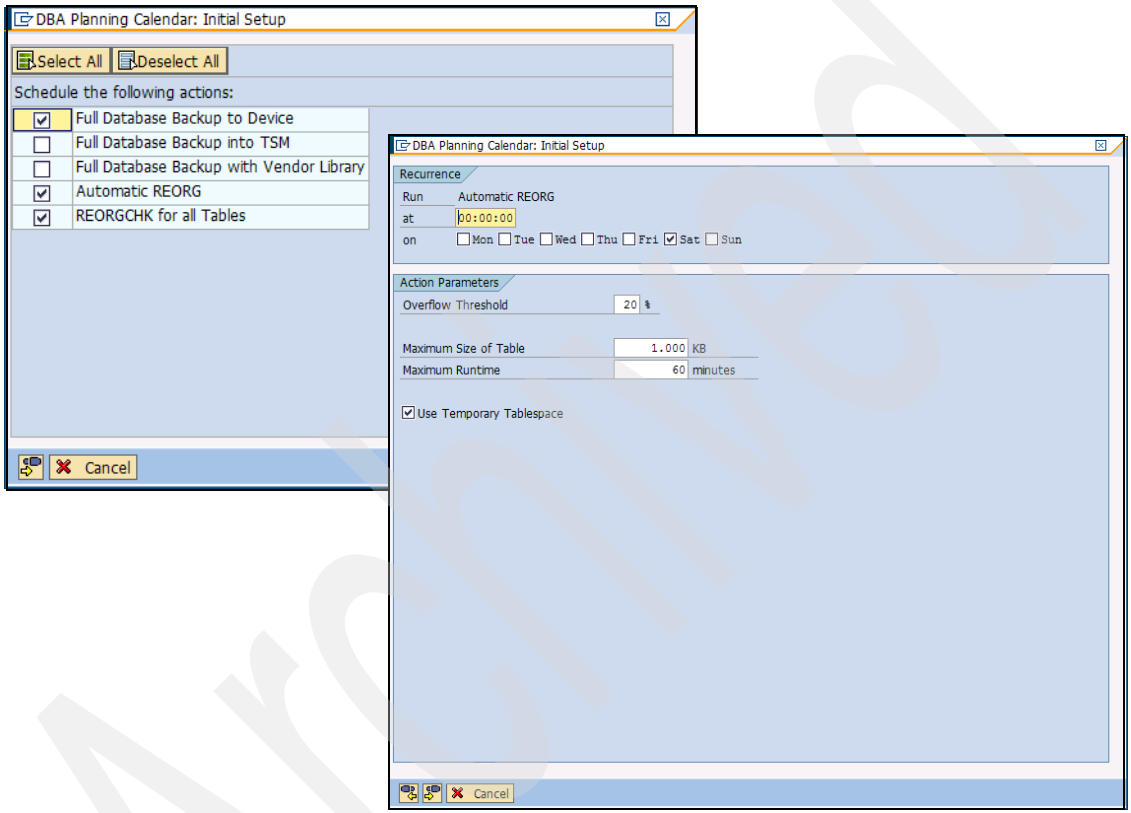


Figure 6-81 Pattern Setup

Edit and delete scheduled jobs

To edit a scheduled job (for example, changing time), select the job in the calendar and click **Edit** button. You can choose either to make the change to all occurrences or only to the current job.

To delete a scheduled job, select the job in the calendar and click **Delete** button. You can choose either to delete all occurrences or delete only the current job.

Actions

Figure 6-82 shows actions that are available when automatic statistics collection is turned on and new log file management is configured.

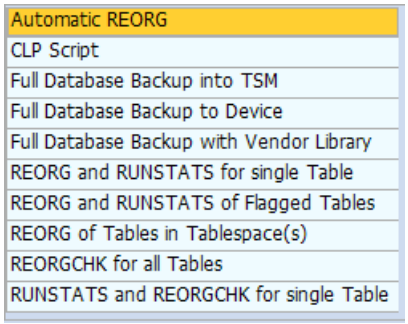


Figure 6-82 Available actions when automatic statistics collection is ON

Backup

Using backup jobs in action pad, you can take a database online/offline full backup. Incremental and incremental delta options are available too. If you do not have enough space for storing backup, the compress option may help you.

We recommend that you use online backup with the include logs option, because the job result would not be correct on DBA Cockpit, if you schedule an offline backup from DBA Cockpit. For more details about the result of offline backup from DBA Cockpit, refer to “Backup management” on page 238.

Beginning with DB2 UDB V8.2, self-tuning backup is available. If the performance parameters for backup, such as backup buffer size, number of buffers, and parallelism, are not explicitly specified, DB2 UDB automatically selects the optimal values for those parameters. Therefore we recommend that you do not set these parameters.

In multi-partitioned database system, a backup job has to be scheduled/executed for each partition. Partition 0000 should be backed up first.

For more information about DB2 backup and advanced backup solutions, refer to Chapter 7, “Protecting your SAP data” on page 313 and *DB2 UDB documentation Data Recovery and High Availability Guide and Reference V8*, SC09-4831-01.

REORG

There are four kinds of reorganization jobs that are available if DB2 automatic reorganization is not turned on. (DB2 automatic reorganization is not turned on by default.)

- Automatic REORG:

  The candidates of this reorganization job are tables which are recommended to be reorganized by REORGCHK job. This means that REORGCHK job should be run before Automatic REORG. These candidates are determined at run time of this job. You can specify some parameters to limit those candidates.

  - Overflow Threshold: Specify the percentage of overflows compared to table size. If the overflow of a table exceeds the number specified, it becomes a candidate of reorganization.
  - Maximum Size of Table: Limit the maximum table size for reorganization. This is determined by the result of last RUNSTATS and REORGCHK.
  - Maximum Runtime: Maximum runtime for this action can be specified. The runtime is checked only after reorganization of each table. If the maximum runtime you specified has been reached during a table reorganization, the reorganization is not interrupted. After the reorganization of this table, this action stops. So the value you specify is not the exact time.

  This job uses offline reorganization, and the RUNSTATS job takes place after reorganization.

  We recommend using this automatic REORG for table reorganization, and we suggest that you schedule this job once a week.

- REORG and RUNSTATS of Flagged Tables:

  The candidates of this reorganization job are tables which are recommended to be reorganized by REORGCHK job. Although there are no parameters to limit the candidate, you can select tables which you want to reorganize from the list of candidates. This list comes up when you schedule this job.

  The candidates are determined when this job is scheduled. So if this job is scheduled regularly, it continues to reorganize same tables which are determined to be reorganized at the time of scheduling. Therefore, we recommend not to schedule this job regularly.

  This job uses offline REORG.

- REORG of Tables in Tablespace(s):

  This job helps to reduce the high water mark of a table space. All tables in a table space are reorganized. This job uses offline REORG and after reorganization RUNSTATS takes place.

- REORG and RUNSTATS for Single Table:

  This reorganizes a single table specified as a parameter. Also, you can choose online/offline reorganization.

For more information about table/index reorganization, refer to 4.4, “Space reclamation strategies” on page 102 and *DB2 UDB documentation Administration Guide: Performance V8*, SC09-4821-01.

CLP Script

The CLP script created in CLP Script Maintenance under Jobs can be scheduled using this job. Also, you can directly create a CLP script using this job.

RUNSTATS/REORGCHK

If automatic statistics collection is turned on, only two RUNSTATS/REORGCHK jobs are available in the action pad, those are *REORGCHK for all tables* and *RUNSTATS and REORGCHK for single table*. Automatic statistics collection is turned on by default with SAP NetWeaver 2004s.

- REORGCHK for all tables:

  We strongly recommend that you schedule this job at least once a week, because DBA Cockpit uses special tables (DB6TREORG and DB6IREORG) to determine if a table/index is recommended to be reorganized or not. The REORGCHK job in the Action Pad updates these special tables with the result of the REORGCHK. We also recommend that you schedule this job before Automatic REORG, because Automatic REORG determines candidates from the information which is created by the REORGCHK job.

- RUNSTATS and REORGCHK for a single table:

  This job executes RUNSTATS and REORGCHK for a set of tables specified by a name with wildcard. If there are enough system resources, you can set parallelism to more than 1 as an option. By scheduling this job, you can perform RUNSTATS explicitly even though automatic statistics collection is available.

If automatic statistics collection is not turned on, the following two jobs are also available in the Action Pad:

- RUNSTATS and REORGCHK (DBSTATC):

  This job performs RUNSTATS and REORGCHK on the tables which are determined to be analyzed. DBASTATC is a special table which controls RUNSTATS and REORGCHK. If automatic statistics collection is not turned on, we recommend that you schedule this job daily.

- RUNSTATS and REORGCHK for all tables:

  This job executes RUNSTATS and REORGCHK for all tables. It may take some time and may have some effect on performance of the system. Therefore, we recommend that you do not schedule this job during business hours. To limit the runtime, you can specify the Maximum Runtime option. We recommend that you run this job once a week.

When you change automatic statistics collection from `OFF` to `ON`, there may be some `RUNSTATS` and `REORGCHK jobs` already scheduled. These jobs are not executed any more once automatic statistics collection is turned on. The job ends with errors, which tells that this job is not available because of automatic statistics collection. Figure 6-83 shows an example of a job log which ended with an error. If you do not want these errors, you need to delete jobs.

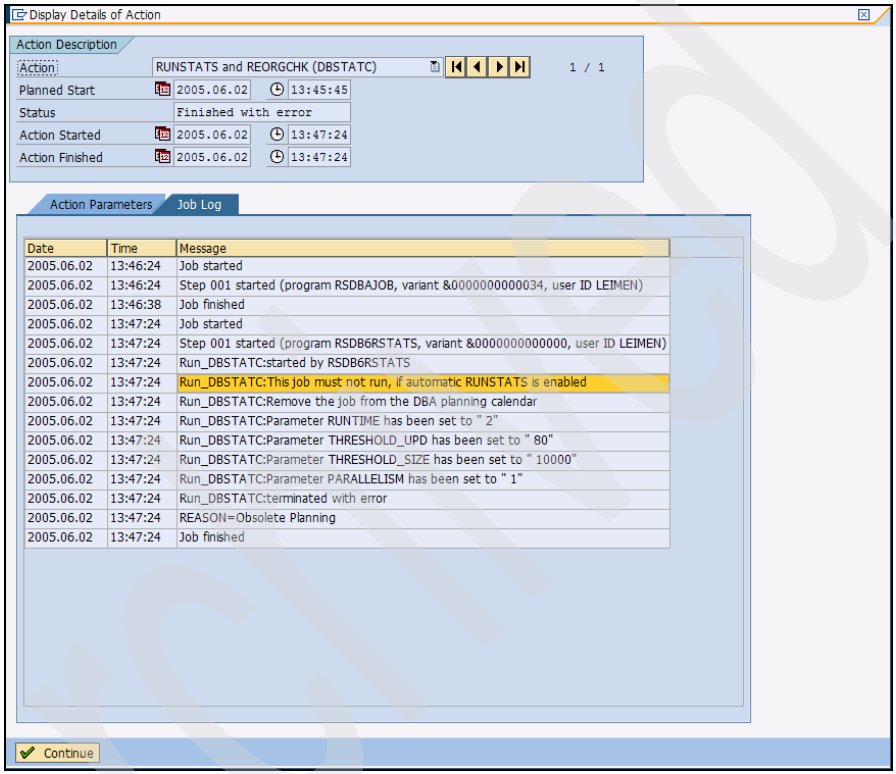


Figure 6-83 Job log ended with error

Note: Some of the jobs may have an impact on the system performance, and also, some of the jobs may affect the system availability. We recommend that you schedule jobs with careful consideration.

Note: Some database related jobs are scheduled automatically. We recommend that you consider these job schedules as well.

- Collection of database performance history data runs every 2 hours starting at 0:00.
- Monitoring database and database manager configuration parameters' changes runs daily at 8:00, 13:00, and 19:00.
- Collection of database and table space history data runs daily at 7:00 and 20:00.
- Collection of tables and indexes space history data runs weekly at 12:00 on Sunday .

You can check the schedule of these jobs from DBA Planning Calendar if you set the Category field to DB collector.

For more information about DB2 RUNSTATS and REORGCHK, refer to *DB2 UDB documentation Administration Guide: Performance V8*, SC09-4821-01.

Checking the log for scheduled jobs

Sometimes the scheduled job may have failed, so we recommend that you check DBA Planning Calendar daily to see whether scheduled jobs ran successfully or not. You can see the status of a job from the color. To see detailed information, double-click a job in the calendar or select a job and press the **Action Details** button, then the Display Details of Action screen comes up. Figure 6-84 is an example of the Job Log tab.

Tip: Even if you have several jobs at the same hour, you can only see one entry in the calendar which has the most severe error among those jobs. To see the other jobs, double-click the cell in the calendar and click the arrow button (next page or previous page).

There is a Status field in the upper frame; here you can check the status of a job. If a job is already executed, there should be a Job Log tab in the Display Details of Action window. The Job log is generated by a background processing job. You can see detailed information about the job from the Job Log tab. If there is no Job Log tab, the job might not be started. Go to transaction SM37 to check the system log. The names of jobs scheduled using DBA Planning Calendar begin with "DBA" in transaction SM37.

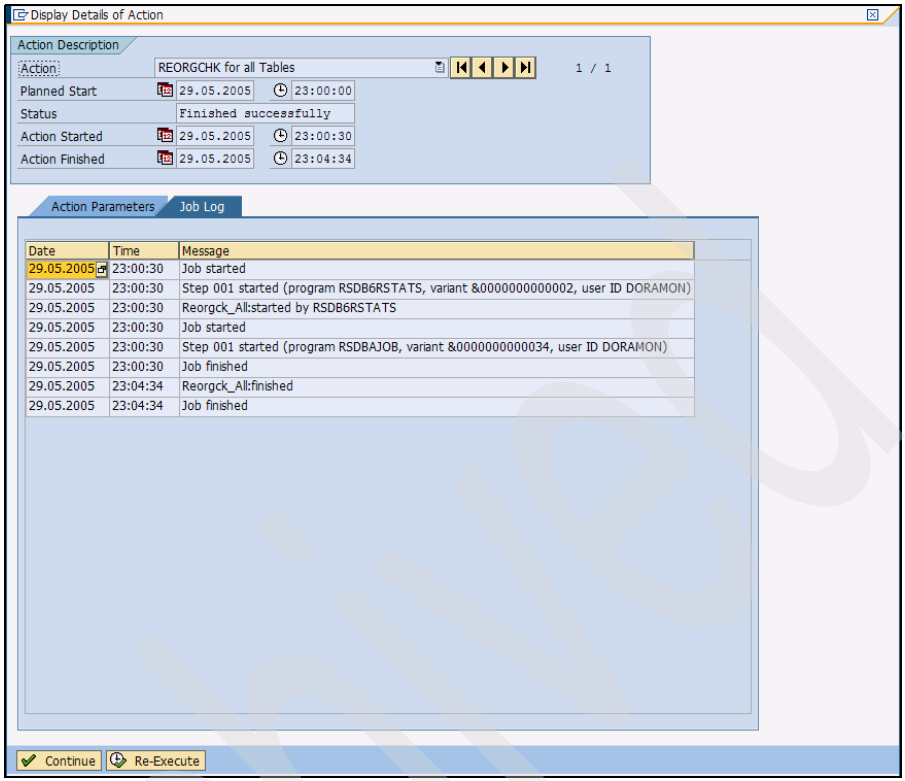


Figure 6-84 Job Log tab

6.9.2 DBA Log

Using DBA Log, you can see the list of log records which are scheduled and executed in DBA Planning Calendar. This list is very helpful to see the result of DBA jobs quickly. So we recommend that you check DBA Log daily. As this list only gives you important information about the job executed, you should check for more detailed information using DBA Planning Calendar or transaction SM37 if there is any error.

6.9.3 CLP Script Maintenance

Use this function to create your own DB2 UDB scripts. The scripts created can be scheduled/executed from DBA Planning Calendar.

To create a script, go to **Jobs** in the navigation frame, and double-click **CLP Script Maintenance**. Figure 6-85 is a screen for creating scripts.

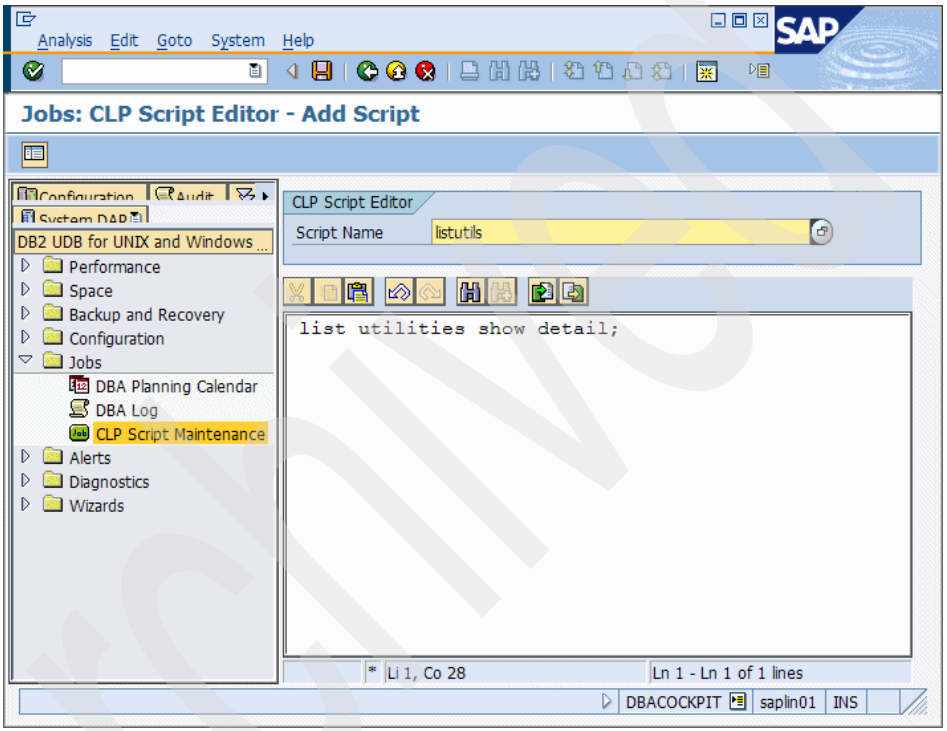


Figure 6-85 Creating a DB2 script using CLP Script Maintenance

When writing a script, use a semicolon ";" as a delimiter.

Also, it is possible to execute the script from CLP Script Maintenance once the script is saved. The result of the execution is displayed in the lower half of the screen.

6.10 Using the alert monitor

To get alerts for database system automatically, you can configure an Alert Monitor. The alert monitor in DBA Cockpit provides function to monitor DB2 UDB database system. There are pre-configured important monitoring elements and parameters in DB2 UDB database system monitoring. Here are the four categories for monitoring elements:

- Space:

  Monitoring disk space usage for table space and file systems related to the database, such as log directory

- Health:

  Monitoring availability of table spaces and containers

- Performance:

  Monitoring performance related elements such as database buffer quality, number of dead locks, number of lock escalations

- Backup:

  Monitoring availability of DB2 UDB backup and log archiving configuration

To use the alert monitor in DBA Cockpit, background monitoring has to be activated at first. To activate background monitoring, go to transaction RZ20, select **Technical Infrastructure** → **Method Execution** → **Activate Background Dispatching**.

You can use the e-mail notification function to get alert messages by e-mail. For more information about enabling the e-mail notification function, refer to SAP online help at:

http://help.sap.com

6.10.1 Alert Configuration

To configure alert monitoring parameters, go to **Alerts** → **Alert Configuration** in DBA Cockpit. Figure 6-86 shows the Alert Configuration initial screen. You can see a list of monitoring elements from here. To deactivate a monitoring element, click the Active column, then the signal is changed to red. This means that the element is deactivated. To limit the displayed elements, you can select the category of monitoring elements, status of the elements, object, or attribute by selecting an appropriate category in Current Selection frame on the upper right.

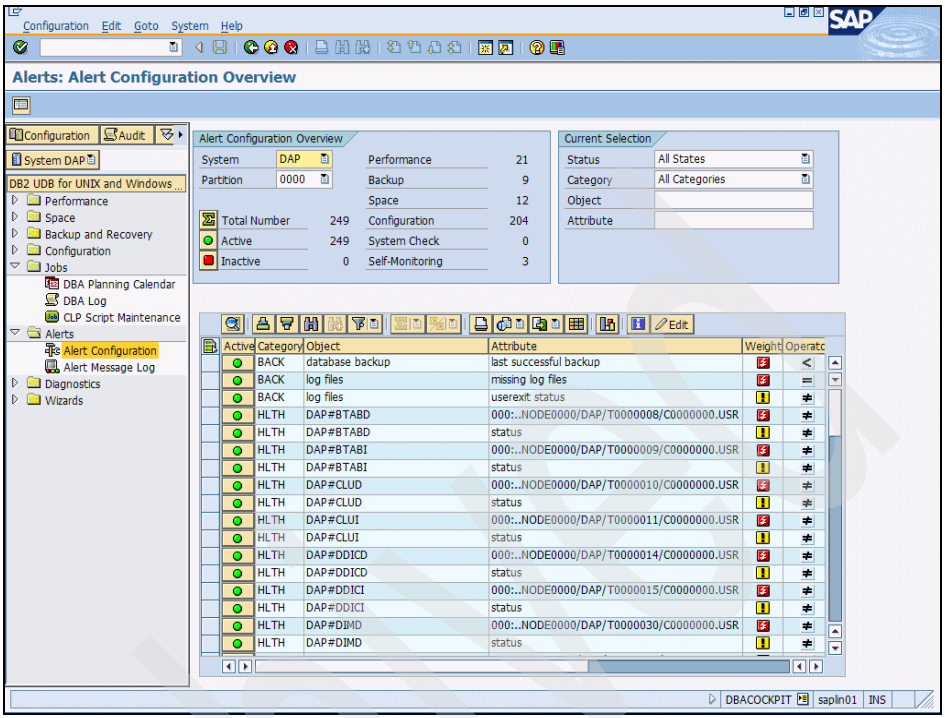


Figure 6-86 Alert Configuration

By double-clicking the element, you can see the detail configuration, such as threshold for warning or error. To change the parameters for each monitoring element, for example, changing threshold for backup availability, select an element and click **Edit**. Figure 6-87 shows the change configuration screen for database backup monitoring. There are three tabs in the screen:

- Threshold:

  Here you can change thresholds for warning, error, and normal state. You do not need to enter values for all state (warning, error, and normal state). However, all possible values have to be covered by entered values. If there is a value that is not covered, the message “There is no configuration entry for the logged value.” is displayed when the monitored value is missing.

- General (`RZ21`):

  This tab shows the schedule of collecting information about the monitored element. The value is only changed by transaction `RZ21`.

- Administration:

  This tab displays the date and time when the last change was made, and the user ID who changed the data.

Note: You have to use transaction RZ21 to change the configuration for some elements, such as Performance and Space.

Note: These configurations should be changed only by experienced users.

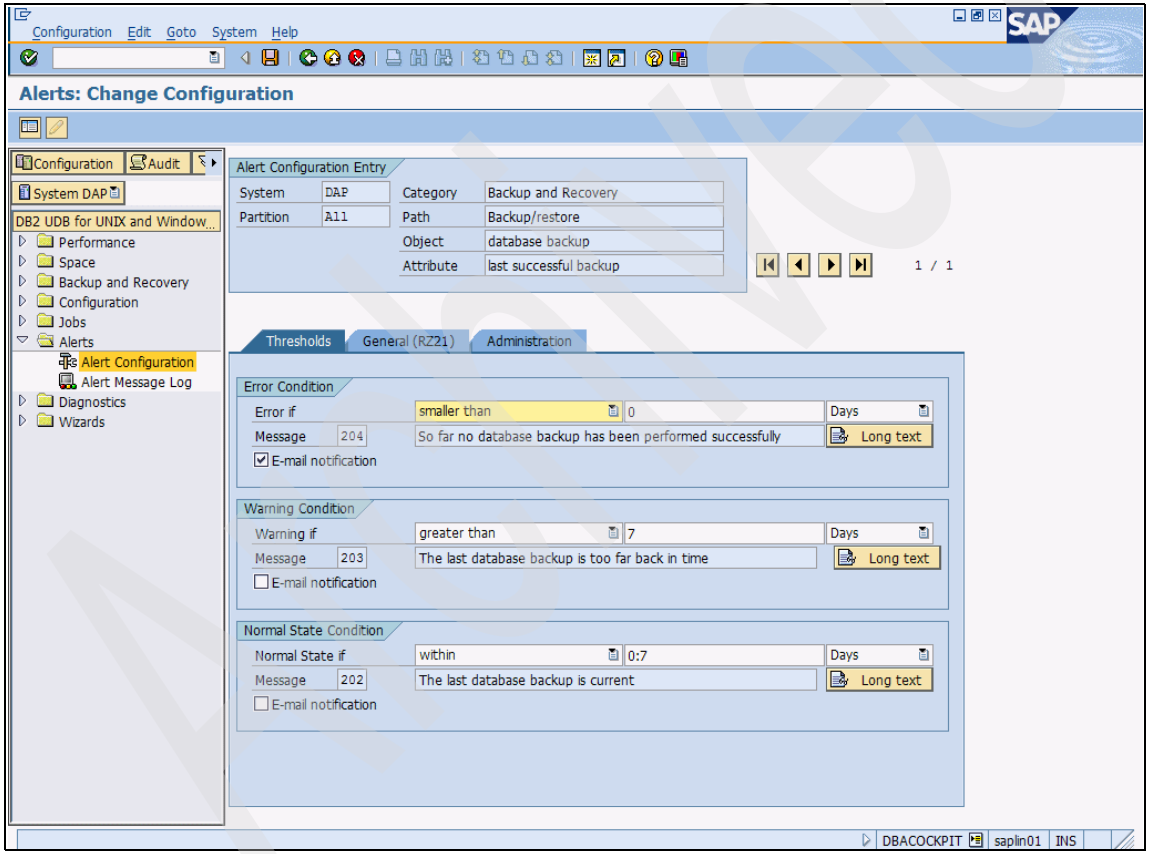


Figure 6-87 Change configuration

6.10.2 Alert Message Log

Using this function, you can easily see your alert messages. We recommend that you check alert messages daily for stable operation of your system. Go to **Alerts** and double-click **Alert Message Logs.** Figure 6-88 is the initial screen of Alert Message Logs. Here you get the list of alert messages. a message marked yellow indicates warning, and red indicates error. You can limit the displayed message by using fields in the Current Selection frame in the upper right.

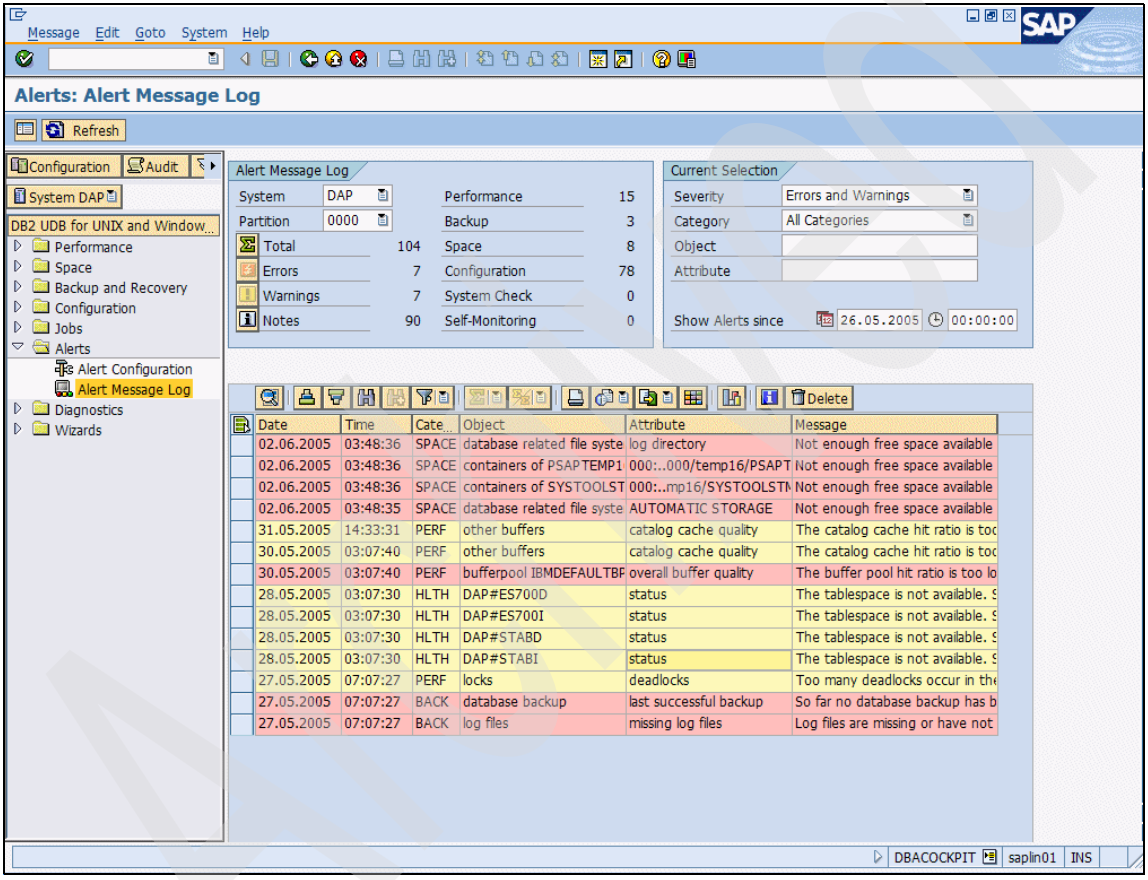


Figure 6-88 Alert Message Log

To see the detail information for these alerts, double-click an entry. The Alert Message Details screen is displayed as shown in Figure 6-89. As you can see, there is some important information you should check:

- Logged Data:

  This is the value reported by alert monitoring function. You should check this value carefully.

- Description

  The error reason and the description of configured parameters.

To see the alert messages, transaction RZ20 can be used also. Go to transaction RZ20, expand **SAP CCMS Monitoring Templates** and double-click **Database**. Then expand **DB2 Universal Database for NT/UNIX**. You can see each message by expanding the tree. The message color indicates the status. Green is normal state, yellow is warning, and red is error. The reported value and short description are displayed next to the parameter. If the analysis method is defined, you can go directly to the transaction for analyzing the problem by double-clicking the message. For example, if you double-click the entry for catalog cache quality, transaction ST04 comes up.

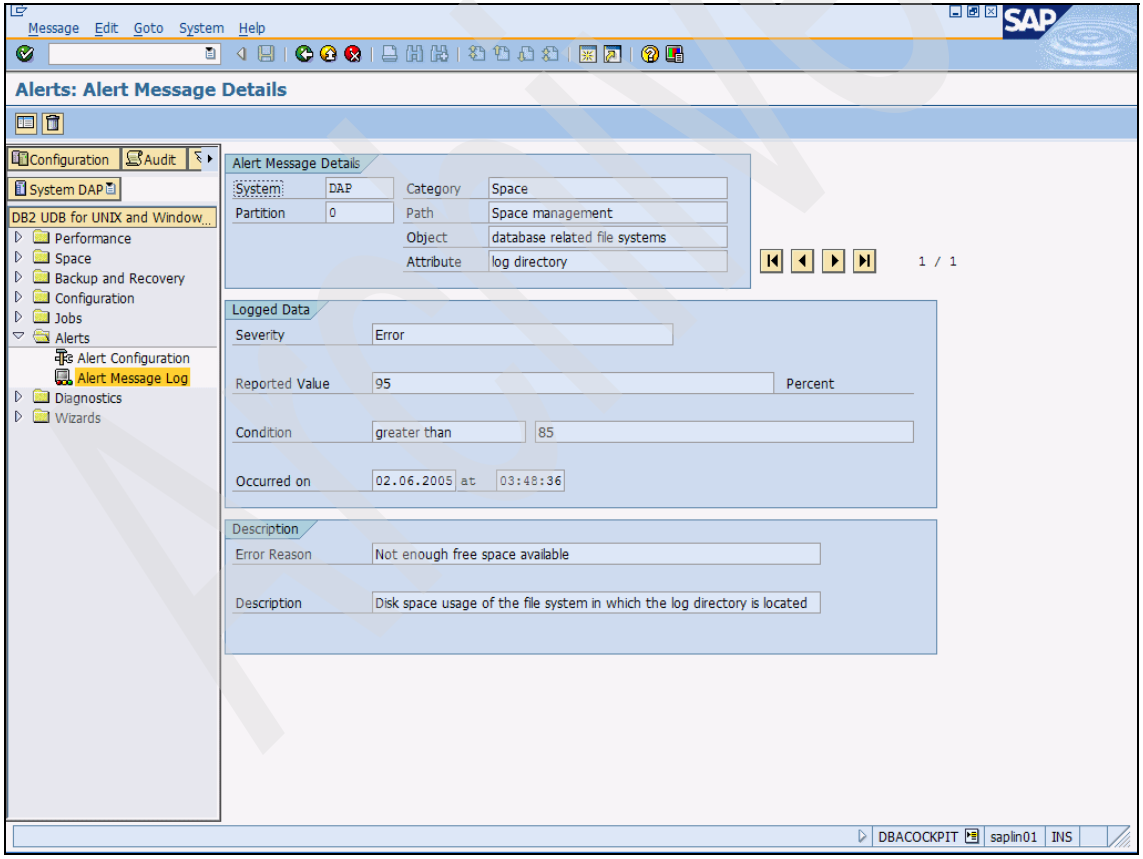


Figure 6-89 Alert Message Details

How to analyze alert messages

Here is an example of analyzing warning and errors. If you get a warning or an error, you may be able to use the following means to analyze the problem and to get resolution.

- Backup:
  - Database Backup

    Go to **Backup and Recovery** → **Backup Overview in DBA Cockpit**. You should check if backup is successfully finished or not. (If you use the advanced backup method such as suspend I/O and `db2inidb`, you can ignore these backup warnings or deactivate this monitoring element.)
  - Log files

    You should check if archiving logging is properly configured from **Backup and Recovery** → **Logging parameters**.
- Health:
  - Each table space status:

    Go to **Space** → **Tablespaces** in DBA Cockpit and check table space status.
  - Each container status:

    Go to **Space** → **Containers** in DBA Cockpit and check if the container is accessible.
- Performance:
  - Buffer pool:

    Go to **Performance** → **Buffer Pools** in DBA Cockpit. Or you can also analyze buffer pool quality from **Database** under **Performance** task area. If you need to change your buffer pool settings, you can perform configuration of your buffer pools from **Configuration** → **Buffer Pools** in DBA Cockpit.
  - Other buffers:

    Go to **Performance** → **Database** and select **Cache** tab to see the detailed information. If you need to change your cache configuration, you can use **Configuration** → **Database** → **Memory** in DBA Cockpit.
  - Locks:

    Go to **Performance** → **Database** and click **Locks and Deadlocks** tab to check the detailed information. If you need to configure lock parameters, you can use **Configuration** → **Database** → **Locks** or **Memory** in DBA Cockpit.

- Space:
  - Containers:

    At first, check the type of table space from **Space** → **Tablespaces**. If it is an auto-resize DMS table space, an SMS table space, or a table space using automatic storage with auto-resize, you should check the file system space usage from **Configuration** → **File Systems** in DBA Cockpit. To extend file systems, use OS commands.

    If you use DMS without auto-resize, you can use **Space** → **Tablespaces** to resize your table space. If the table space is reaching its maximum size, you should consider using `DB6CONV` to move some of your tables to another table space. For more information about `DB6CONV`, refer to Chapter 4, “Storage management in depth” on page 79.
  - Log directory:

    To check the current status, go to **Backup and Recovery** → **Logging Parameters**. There may be some log files that should be archived, left in the log directory. You can check the archiving log from **Backup Overview**.

  For all problems related DB2 UDB, db2diag.log contains the messages that can help you.

Deleting an Alert Message

If you solve the problem and there is no need for the alert message any more, you can delete it from **Alert Message Logs** by clicking **Delete**. Also, the system schedules a cleanup program automatically to delete the alert message older than 30 days. This program runs on Sunday.

For more information about DBA Cockpit, refer to SAP Online Help:

`http://help.sap.com`

6.11 Diagnostics

The diagnostics task area can be accessed with the transaction code `DBACOCKPIT` and then choose **Diagnostics**. It consists of a collection of tools for performing system checks, turning on / off traces, investigating explain plans for query statements, and viewer for examining database diagnostic/notification logs.

6.11.1 Missing Tables and Indexes

You can find out whether there are any inconsistencies for tables or indexes between the database level and the ABAP Dictionary (that is, table/index exists in the database level but not defined in the ABAP Dictionary or vice versa).Figure 6-90 shows the Missing Tables and Indexes initial screen.

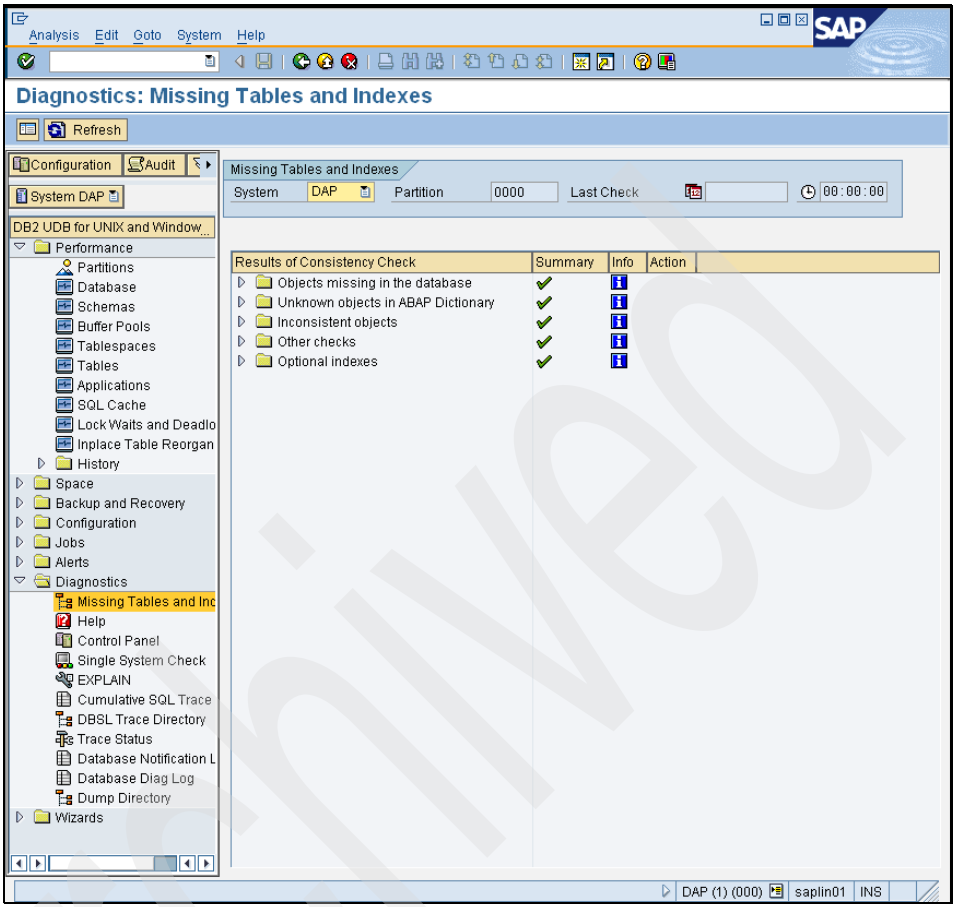


Figure 6-90 Diagnostics: Missing Tables and Indexes

- Objects missing in the database:

  For objects that are found in the ABAP Dictionary but not in the database. Possible reasons are:

  - The object was deleted from the database on purpose during development.
  - There were database errors that prevented object from creating successfully.

  Objects can be created on the database from here if there was no incomplete transport or cancelled conversion.

- Unknown objects in ABAP Dictionary:

  For objects that are found in the database but not in the ABAP dictionary. This usually means that database objects were created in the database but not using the SAP tools which bypassed the ABAP Dictionary. These are objects that cannot be accessed by ABAP programs since they are not defined in the ABAP Dictionary. In some cases, such objects were already overlooked by SAP at delivery or occurred during upgrading. To check this, search in SAP Service Marketplace for Notes with keyword *DB02* and the release level.

- Inconsistent Objects:

  A report of the comparison between the ABAP dictionary and the database.

- Other checks:

  These are the tests that are performed by Other Checks:

  – Check whether the primary index of tables defined in the ABAP Dictionary are created uniquely.

  – Check for objects that cannot be described in the ABAP Dictionary for some technical reasons.

  A report of any inconsistencies found will be displayed here along with the type of inconsistencies.

- Optional Indexes:

  Optional Indexes are indexes that are designed for a certain database system. You may see here:

  – An index that should not exist in this database system, but exists in the database.

  – An index that should exist in this database system but is missing in the database.

6.11.2 Help

You can look up DB2 UDB error codes and syntax description of DB2 Command Line Processor (CLP) commands by choosing Diagnostics Help in the navigation frame of DBA Cockpit. The option provides the equivalent of running “db2 ?” used to interpret DB2 error codes or give you the syntax of CLP commands. See Figure 6-91.

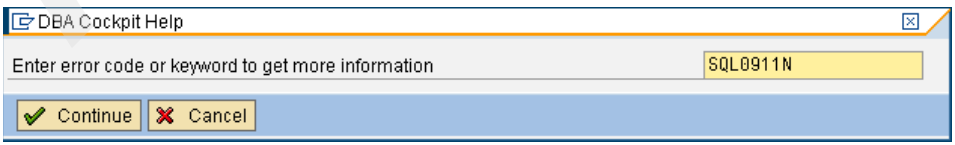


Figure 6-91 Diagnostics: Help - Enter error code

For example, suppose that you would like to find out the message text for the SQL0911N. Enter the message code and click **Continue** to see the details of the error code. See Figure 6-92.

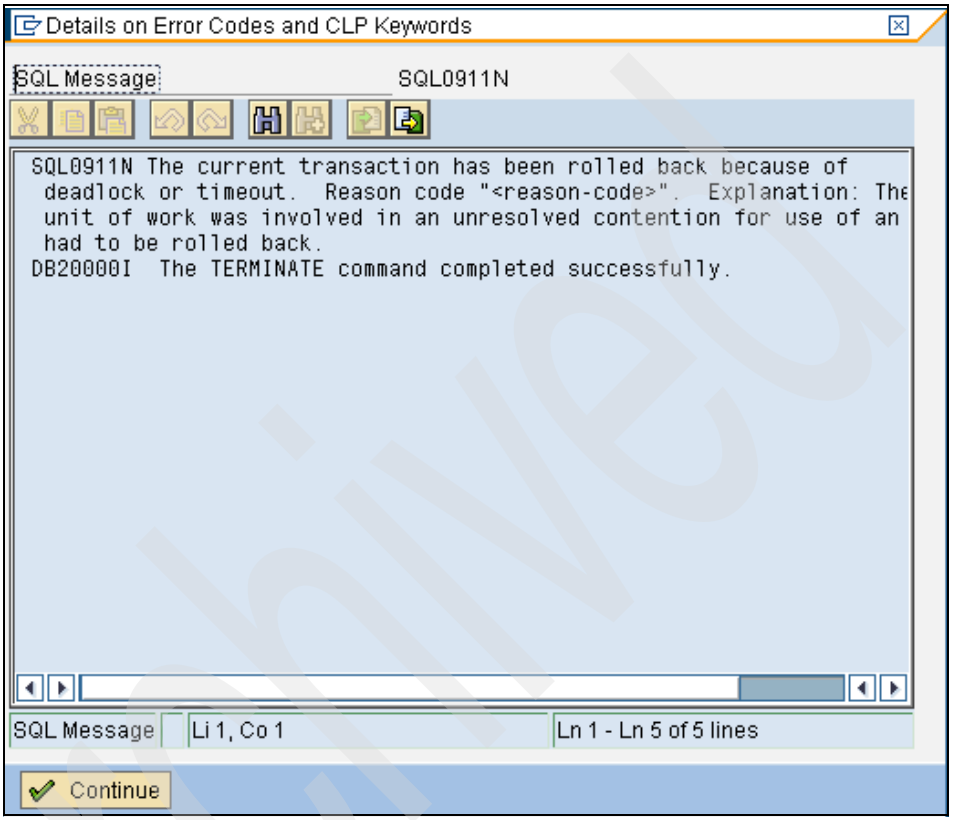


Figure 6-92 Diagnostics: Help - explanation of error code

You see the error message text for the SQL0911N error code.

6.11.3 Control Panel

The **Diagnostics: Control Panel** screen (see Figure 6-93) checks the healthiness according to a set of indicators for all of your configured systems (that is, both local and remote systems) that are switched on. You can control the frequency of the automatic refresh by setting the time period or you can choose not to refresh.

You see the following information for each system:

- *Status*: The symbol on the screen looks like a traffic light. It indicates the status of the system:
  - *Green*: No action required.
  - *Yellow*: We strongly recommend that you take immediate action to avoid serious errors.
  - *Red*: Immediate action required.
- *System*: This is the name you give when you define the systems in DBA Cockpit configuration.
- *Buffer Quality (%)*: The buffer pool hit ratio of the buffer pool that has the worst buffer quality.
- *TS with Highest Fill Level*: The name of the tablespace with the highest fill level.
- *TS Fill Level (%)*: Fill level of the tablespace with the highest fill level.
- *Last Backup*: When the last backup is taken.
- *Log Dir Fill Level (%)*: The amount of used space as a percentage in the file system where the log directory is located.

In the lower half of the screen, there is a list of messages that are written during the system check. For detailed information on the message, double-click a message log entry.

For detailed information on one of the checked objects, right-click a cell and access the following information using the context menu:

- Detailed information for the analysis of the object
- Maintenance of the object
- Alert configuration data of the object
- Alert messages related to the object

For example, if you right-click a cell under the *Buffer Quality* column, you get a dialog window pop-up with the following options:

- Bufferpool Performance
- Bufferpool Maintenance
- Alert Configuration
- Alert Message Log

By choosing for example, *Bufferpool Maintenance*, you are redirected to the *Configuration: Buffer Pool Maintenance - Change Buffer Pool screen of DBA Cockpit*.

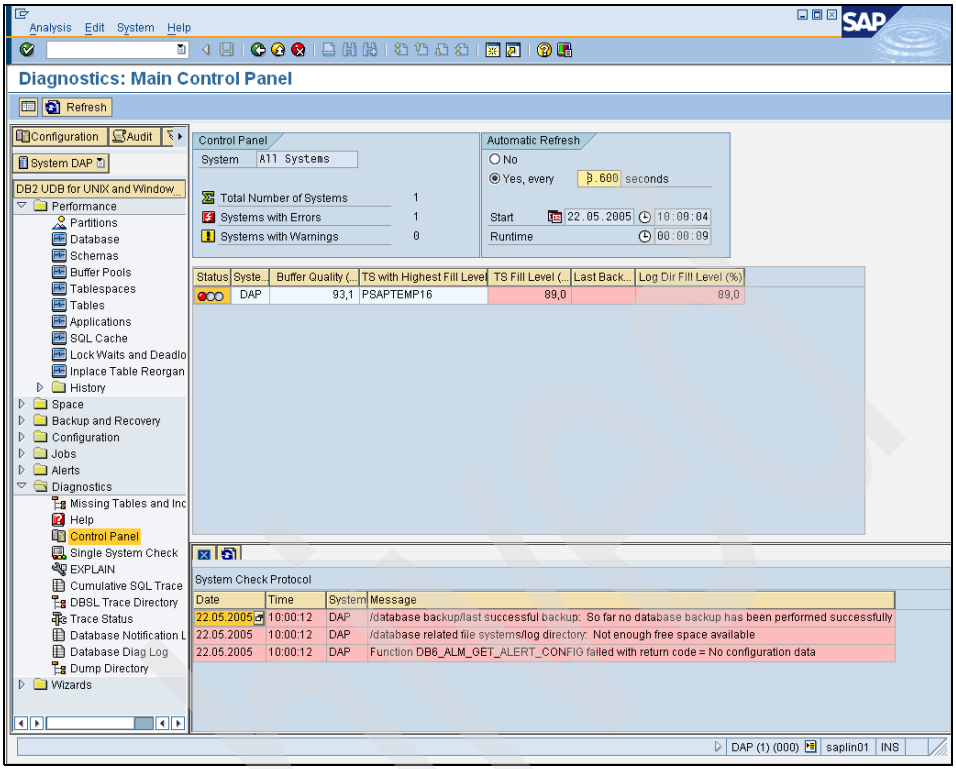


Figure 6-93 Diagnostics: Control Panel

6.11.4 Single System Check

In the **Single System Check** screen (see Figure 6-94), you can view the overall healthiness of a particular SAP system. You can choose which system to check in the *System* field. The data on the screen is automatically refreshed. You can control the frequency of the automatic refresh by setting the time period or you can choose not to refresh.

The results of the checks are indicated as follows:

- *Green*: No action required.
- *Yellow*: We strongly recommend you take immediate action to avoid serious errors.
- *Red*: Immediate action required.

Recovery Availability

Under Recovery Availability section, you see diagnostic information for:

- *Last Backup*: The date of the most recent backup.

  The older the backup, the more database transaction logs the rollforward part of a recovery has to go through.

- *User Exit*: Whether the database is enabled for *rollforward-recovery.*

  The indicator is compatible with both the SAP DB2 log file management and the new log file management.

Top Space Allocation

Under Top Space Allocation, you see diagnostic information for:

- *Log Directory*:

  This is the amount of used space as a percentage in the file system where the log directory is located. Note that if this file system gets full, no new log files can be further allocated and the database will freeze.

- *Automatic Storage*:

  The indicator only applies if your system is installed with the automatic storage feature. It shows the percentage of overall free space of all storage paths.

- *Tablespace <Tablespace Name>*:

  This displays the table space with the highest fill level (in percentage). It only checks the table spaces that were not created using automatic storage.

  For auto-resize DMS table spaces that were not created using automatic storage, the highest fill level of the file system that holds individual container is used as the fill level of the DMS table spaces. Supposed we have a auto-resize table space that has three containers and the fill level for the file systems that hold the three containers are 60%,70% and 80%. The fill level of this DMS table space will be 80%.

  For SMS table spaces, the fill level for the file system that holds the container is used as the fill level for the table space.

  For manually created DMS table spaces, fill level is the percentage ratio between the total amount of space used by all the containers against the total amount of available space for all containers.

Tip: For detailed information on one of the checked objects, double-click the corresponding traffic light and you will be forwarded to the related detail screen:

- Backup and Recovery: Overview
- Backup and Recovery: Logging Parameters
- Space: Tablespace Maintenance - Display Tablespace

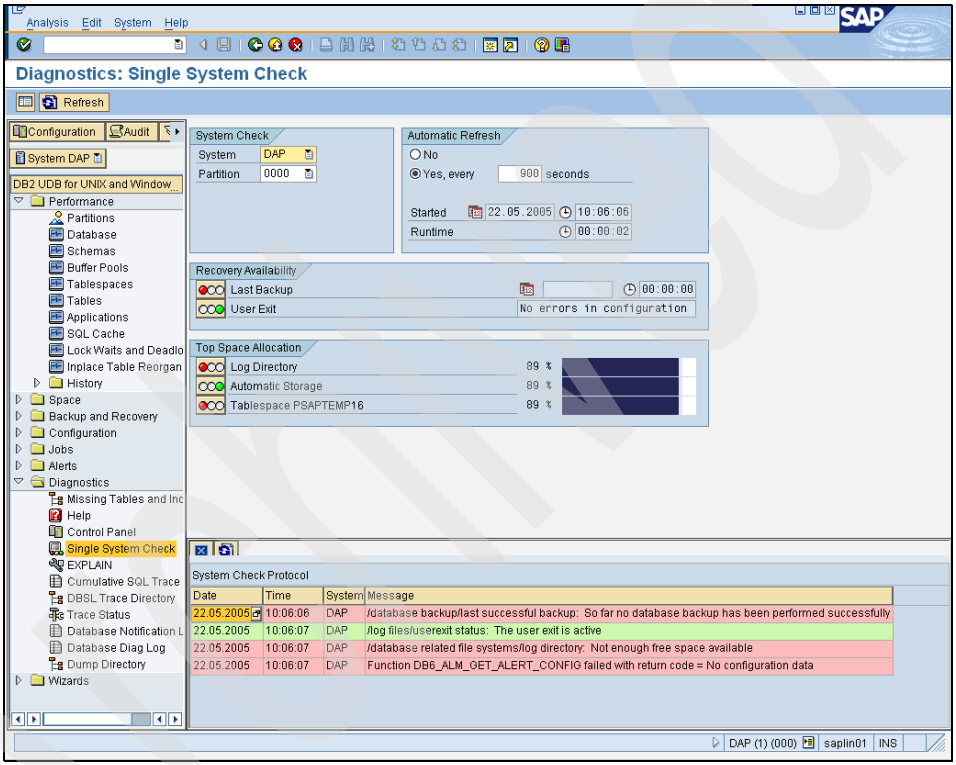


Figure 6-94 Diagnostics: Single System Check

6.11.5 Explain

In the Explain screen (see Figure 6-95), you can type in a `SELECT`, `INSERT`, `UPDATE` or `DELETE` statement and click the **Explain** button to go to the *Display Execution Plan for SQL Statement* screen.

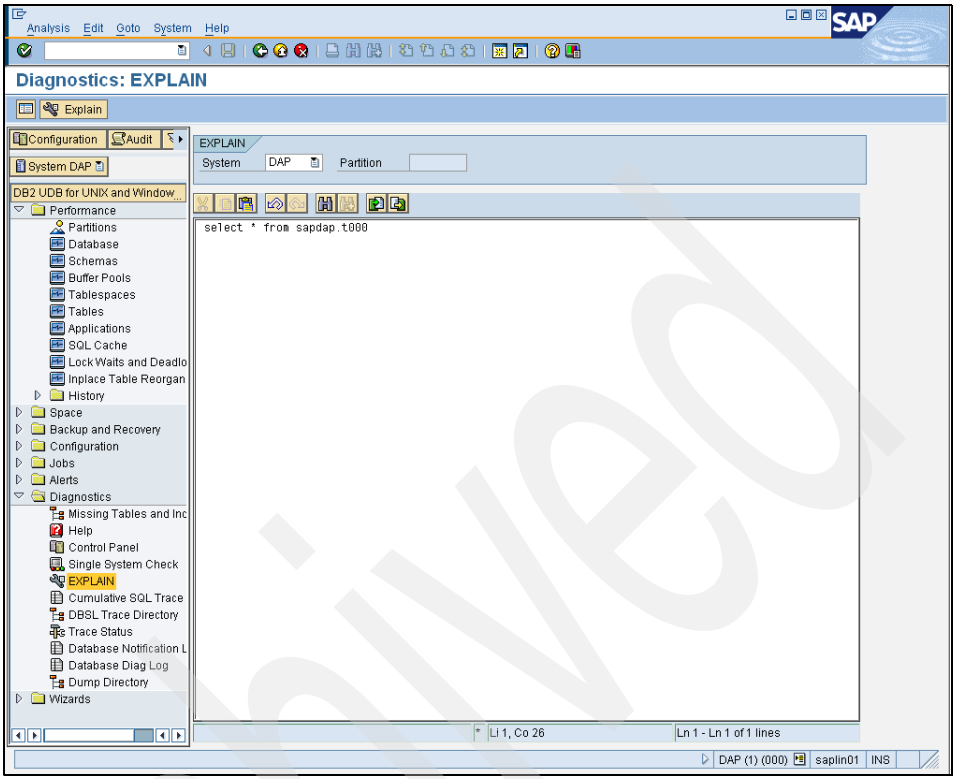


Figure 6-95 Diagnostics: Explain

The *Display Execution Plan for SQL Statement* screen (Figure 6-96) shows the access plan of this statement. In the *Display Execution Plan for SQL Statement* screen, you see the access plan of the original SQL statement and the access plan.

Tip: Note that this screen is common for all the following tools that have an Explain functionality. From all of the following screen, you access the *Display Execution Plan for SQL Statement* screen with the **Explain** button:

- DBA Cockpit: **Performance** → **SQL Cache**
- DBA Cockpit: **Diagnostics** → **Cumulative SQL Trace**
- DBA Cockpit: **Diagnostics** → **Explain**
- ST05 (SQL Trace): **Display Trace** → **Trace List**

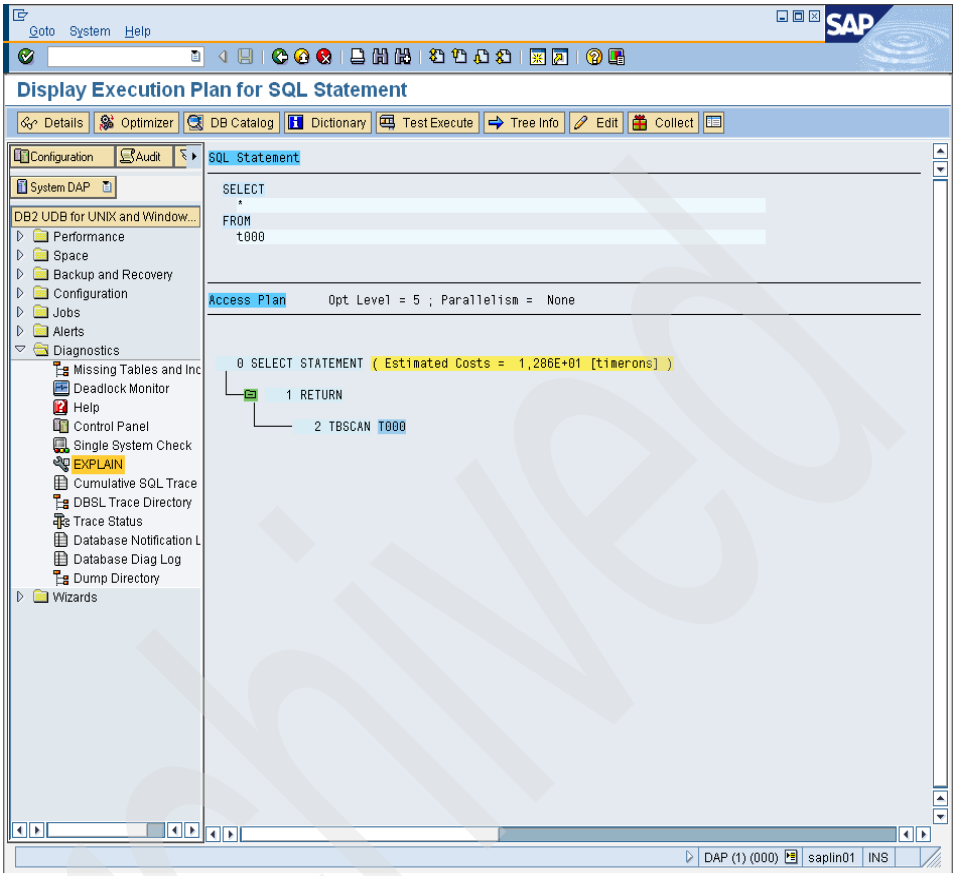


Figure 6-96 Diagnostics: Display Execution Plan for SQL Statement

At the top of the screen, you see the following buttons:

- Detail
- Optimizer
- DB Catalog
- Dictionary
- Test Execute
- Tree Info
- Edit
- Collect

In the following sections we supply an explanation of each option.

Explain - Detail

There are three cases, depending on which operator you have chosen or not chosen:

- If no operator in the access plan is highlighted when choosing this option, you see a dialog box giving detailed information on the statement and each operator. This output is similar to the one of the DB2 command line tool **db2exfmt**. See Figure 6-97.

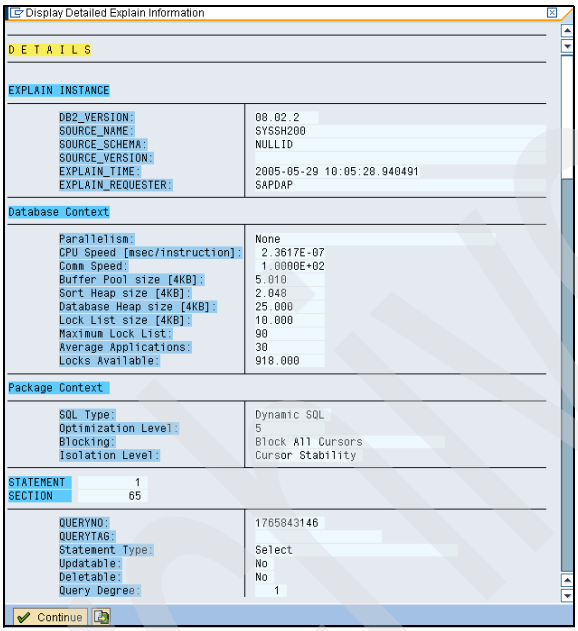


Figure 6-97 Display Execution Plan for SQL Statement - Detail (no operator)

- If operator number *0* is highlighted (in Figure 6-96, operator number 0 is the node with “0 SELECT STATEMENT”), the dialog shows the original statement and optimized statement. See Figure 6-98.

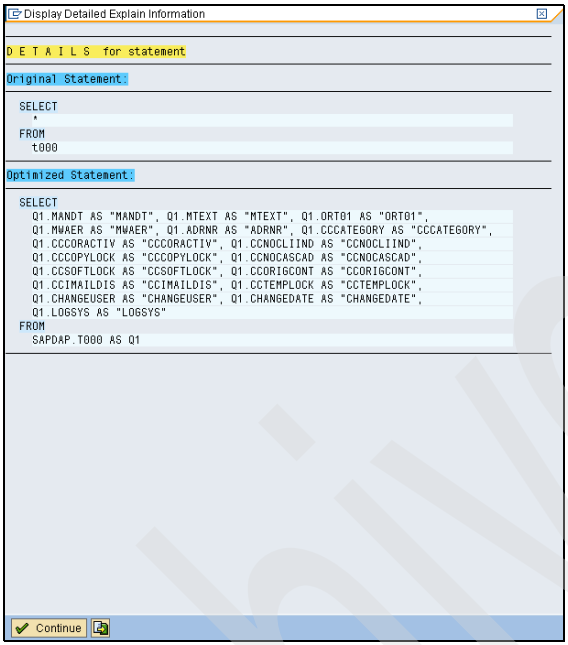


Figure 6-98 Display Execution Plan for SQL Statement - Detail (operator no. 0)

- If any other operator is highlighted, the dialog box displays detailed information on the selected operator. See Figure 6-99.

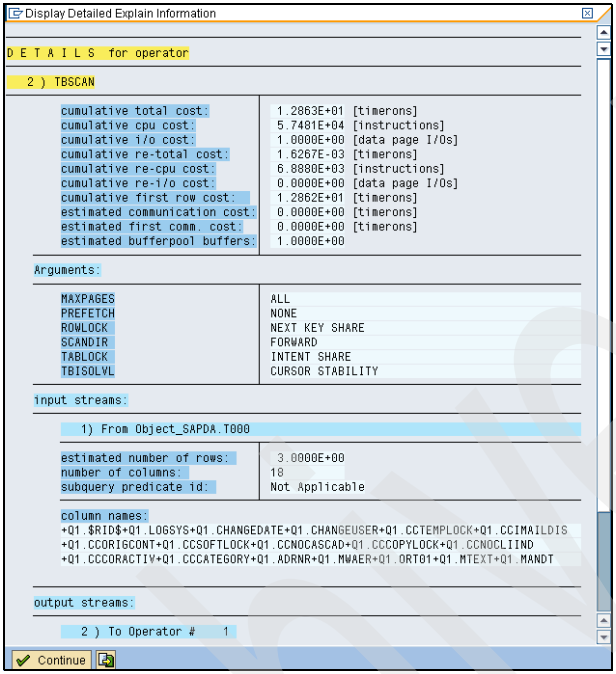


Figure 6-99 Display Execution Plan for SQL Statement - Detail (other operators)

Explain - Optimizer

The access plan that the DB2 optimizer chooses for running an SQL statement depends on the optimizer parameters (for example, optimizer level, query degree, volatile flag). In the *Change Query Optimization* dialog box (Figure 6-100), you can experiment the effect of different parameters `OPTIMIZER LEVEL`, `QUERY DEGREE` and the flag `VOLATILE` for the tables used in the access plan. After chosen the desired combination of optimizer parameters, you can explain the statement again by clicking the **Explain Again** button.

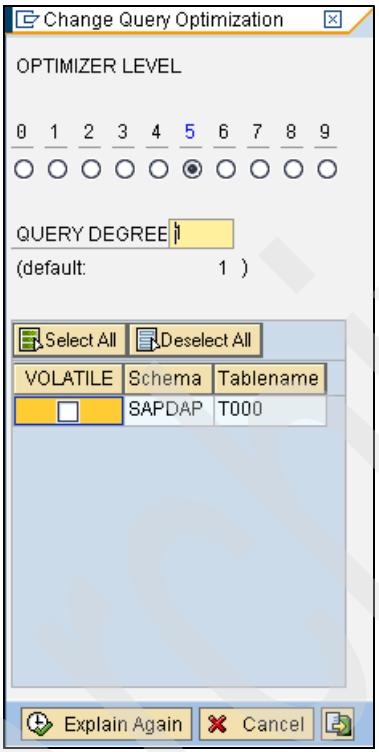


Figure 6-100 Diagnostics: Display Execution Plan for SQL Statement - Optimizer

Explain - DB Catalog

This dialog box shows the system catalog information on tables and indexes in the access plan:

- For a table (see Figure 6-101): the dialog box shows selected information from table `SYSCAT.TABLES`. Additionally, all indexes of the table are displayed with their index columns.

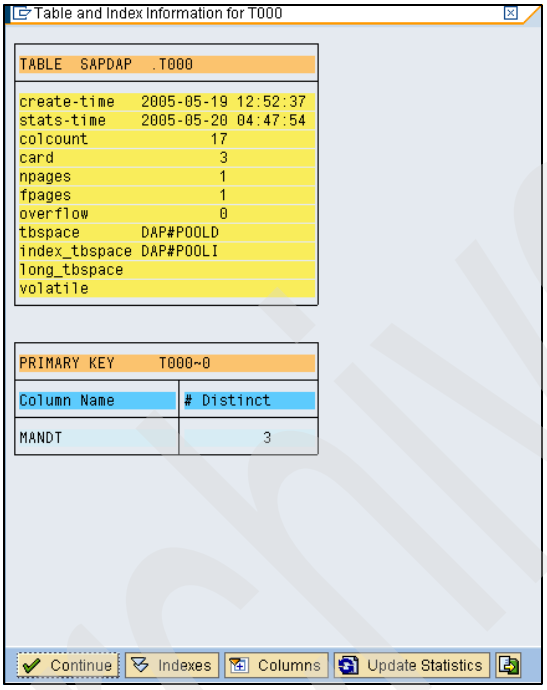


Figure 6-101 Diagnostics: Display Execution Plan for SQL Statement - DB Catalog - Table

These are the available buttons:

- *Indexes*: Switch to the DB Catalog for index dialog box (Figure 6-102) which shows selected information from table `SYSCAT.INDEXES` for this index.
- *Columns*: Displays selected information from table `SYSCAT.COLUMNS` for all table columns.
- *Update Statistics*: Updates the catalog statistics for the table. If the catalog statistics were updated successfully, the field <stats-time> is displayed in green.

- For an index (see Figure 6-102): the dialog box shows selected information from table `SYSCAT.INDEXES` for this index. Additionally, you see selected information from table `SYSCAT.COLUMNS` for all index columns.

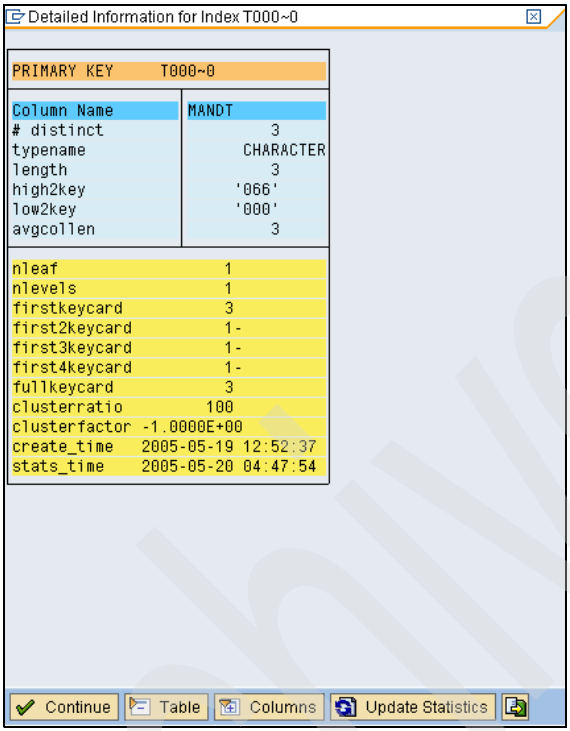


Figure 6-102 Diagnostics: Display Execution Plan for SQL Statement - DB Catalog - Index

These are the available buttons:

- *Table*: Switch to the DB Catalog for table dialog box (Figure 6-101) which selected information from table SYSCAT.TABLES. Additionally, all indexes of the table are displayed with their index columns.
- *Columns*: Displays selected information from table SYSCAT.COLUMNS for all *Update Statistics*: Updates the catalog statistics for the table. If the catalog statistics were updated successfully, the field <stats-time> is displayed in green.
- *Update Statistics:* Updates the catalog statistics for the table. If the catalog statistics were updated successfully, the field <stats-time> is displayed in green.

Explain - Dictionary

See Figure 6-103. If you have highlighted a table in the access plan, you see the ABAP Dictionary structure (definition) of the selected table.

If you do not select a table in the access plan, the ABAP Dictionary structure (definition) of the first dictionary object of the SQL statement is displayed.

With this option, you can display the structure of views, even though views never appear in the access plan.

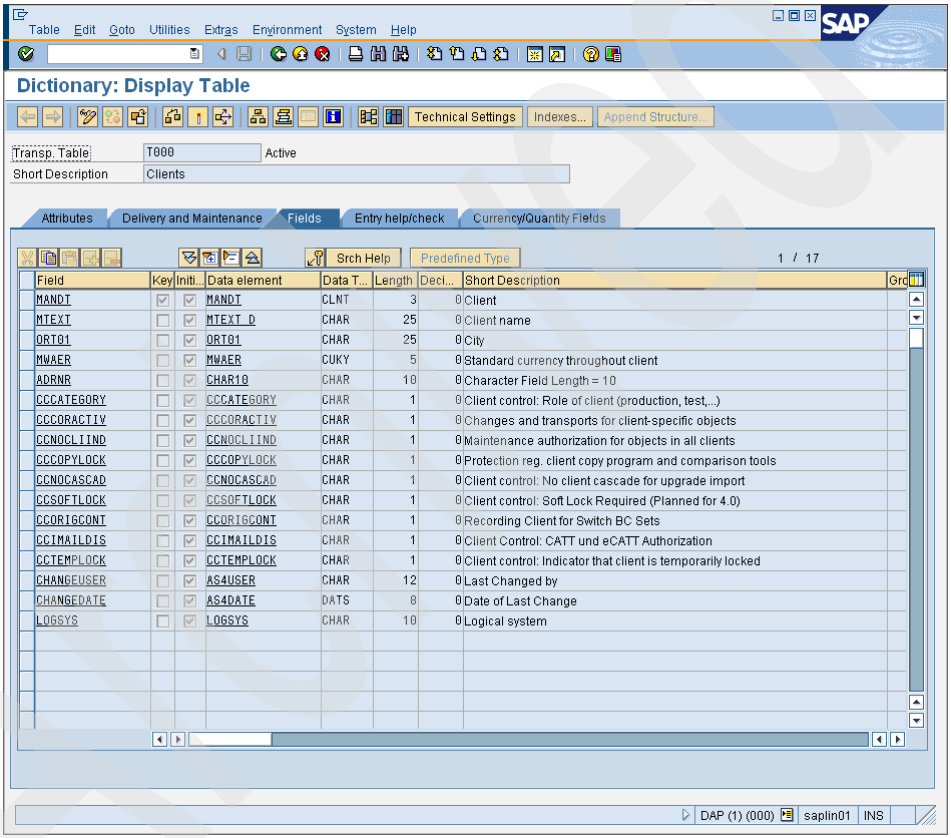


Figure 6-103 Diagnostics: Display Execution Plan for SQL Statement - Dictionary

Explain - Test Execution

Because you need run time values to execute a statement, you can perform a *Test Execution* only if:

- `A SELECT statement` is explained using transaction ST05: Trace list, the parameter values for all parameter markers of the statement are provided and the operation is not PREPARE.
- `A SELECT statement without parameter markers` (with literals) is explained.

The estimated cost for executing a statement provided by the `EXPLAIN` may not reflect the actually execution time in some conditions. When you use the `EXPLAIN` function, the entered SQL statement is only prepared and the access plan of the optimizer is chosen based on the system catalog statistics. On the basis of this information the optimizer estimates the costs for the execution of this statement. However, with bad statistics with outdated statistics, bad database layout with unfavorable layout, or problems of the optimizer itself, the estimated cost for executing a statement given by the `EXPLAIN` may not be accurate. The *Test Execution* option let you find out the “real” execution time of a statement.

The Test Execution option (see Figure 6-104.) measures the real execution time and provides other snapshot data (for example, the number of buffer pool accesses, sorts for the selected statement). When the statement is executed, the parameter markers are replaced by the actual parameter values. A dialog box appears where you can change these values to investigate the dependence of the execution time from these values.

The result of several test executions of the same statement can vary because, for example, the buffer pool may already contain data that is necessary for the execution.

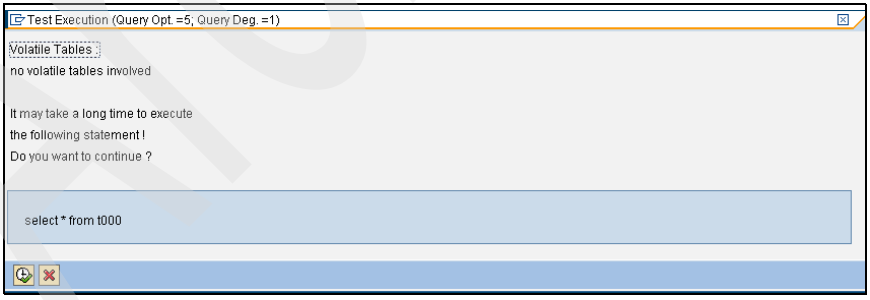


Figure 6-104 Diagnostics: Display Execution Plan for SQL Statement - Test Execution

Explain - Tree Info

The *Tree Info* option lets you toggle on/off for the display of the following information (see Figure 6-105):

- num_rows: This is the estimated number of rows in the result set.
- tot_cost: This is the estimated total cost for executing this statement.
- i/o_cost: This is the estimated I/O cost of executing this statement.

You see the same additional information when you choose the Details button in the access plan.

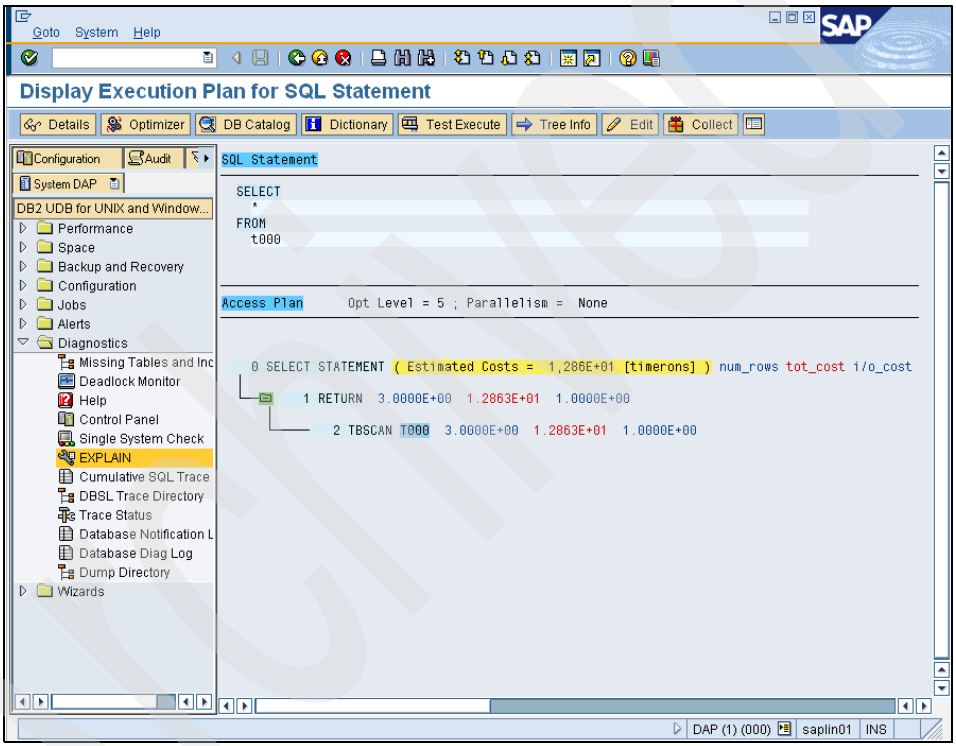


Figure 6-105 Diagnostics: Display Execution Plan for SQL Statement - Tree info

Explain - Edit

This option allows you to modify the selected SQL statement in an editor window and then you can re-explain the modified statement. See Figure 6-106.

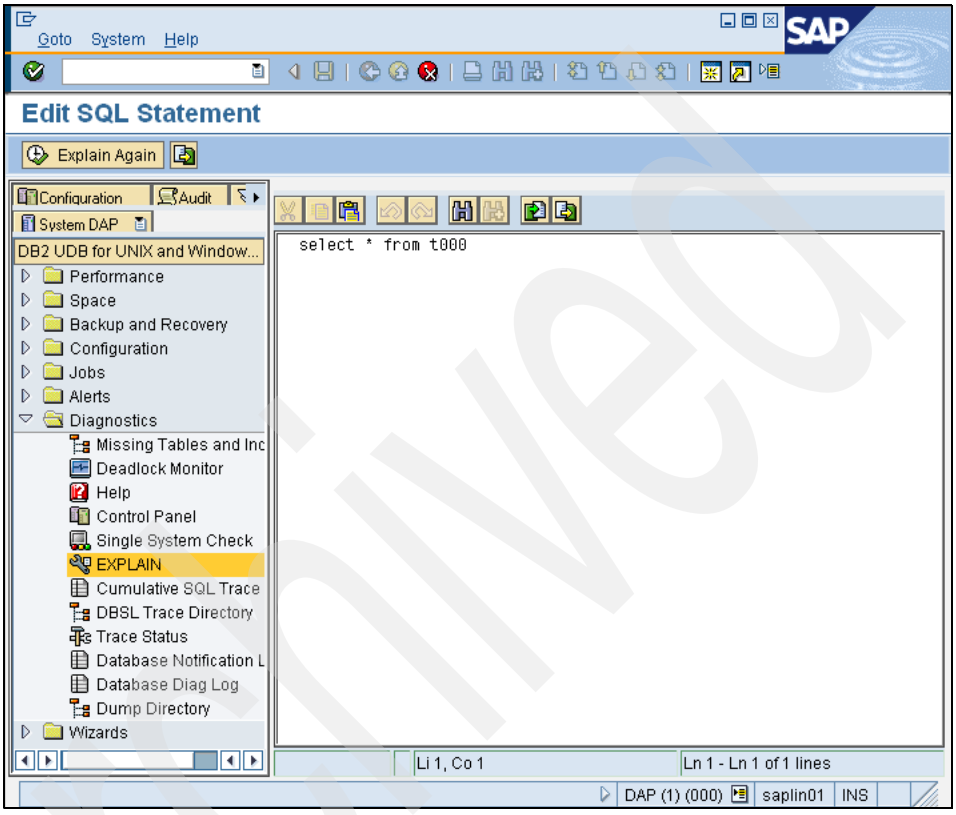


Figure 6-106 Diagnostics: Display Execution Plan for SQL Statement - Edit

Explain - Source

This option only works for the statement containing a `LOCATION` comment, for example, when you call `EXPLAIN` using transaction ST05: Trace list.

This option shows the section of the ABAP source code where this SQL statement is executed.

Explain - Collect

This is a handy option which helps you to collect diagnostic information for analyzing the query performance problem, which is sometimes requested by SAP support. You can use this option to download data to your local machine where you run your SAP GUI.

When you click the **Collect** button, you see a dialog box similar to Figure 6-107. You can select various diagnostic data to download:

- DB2 Level
- Registry Variables
- DBM Configuration: Database manager configuration
- DB Configuration: Database configuration
- Table Structure: Structures of the tables involved in the statement and all views defined in the database
- Tablespace Configuration
- Statistics: Statistics of the tables involved in the statement
- Explain: `EXPLAIN` Information on executable `db2exfmt` including the access plan

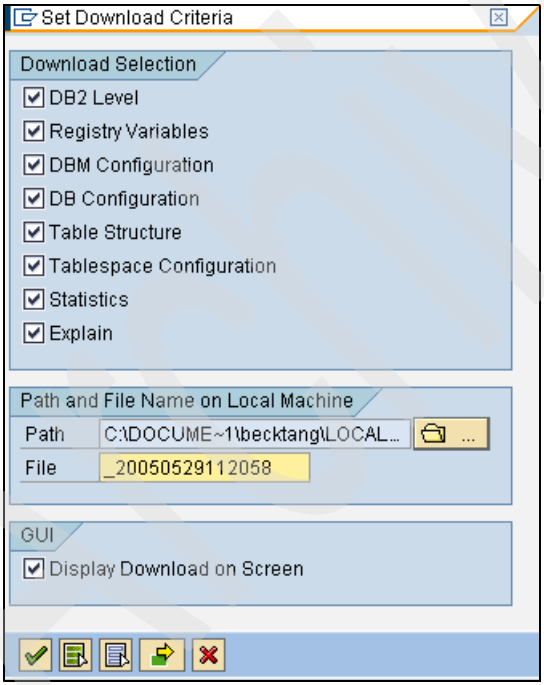


Figure 6-107 Diagnostics: Display Execution Plan for SQL Statement - Collect

6.11.6 Cumulative SQL Trace

This is where you look at the result of Cumulative SQL Trace (refer to 6.11.8, “Trace Status” on page 307 on how to turn on and turn off the Cumulative SQL Trace). See Figure 6-108.

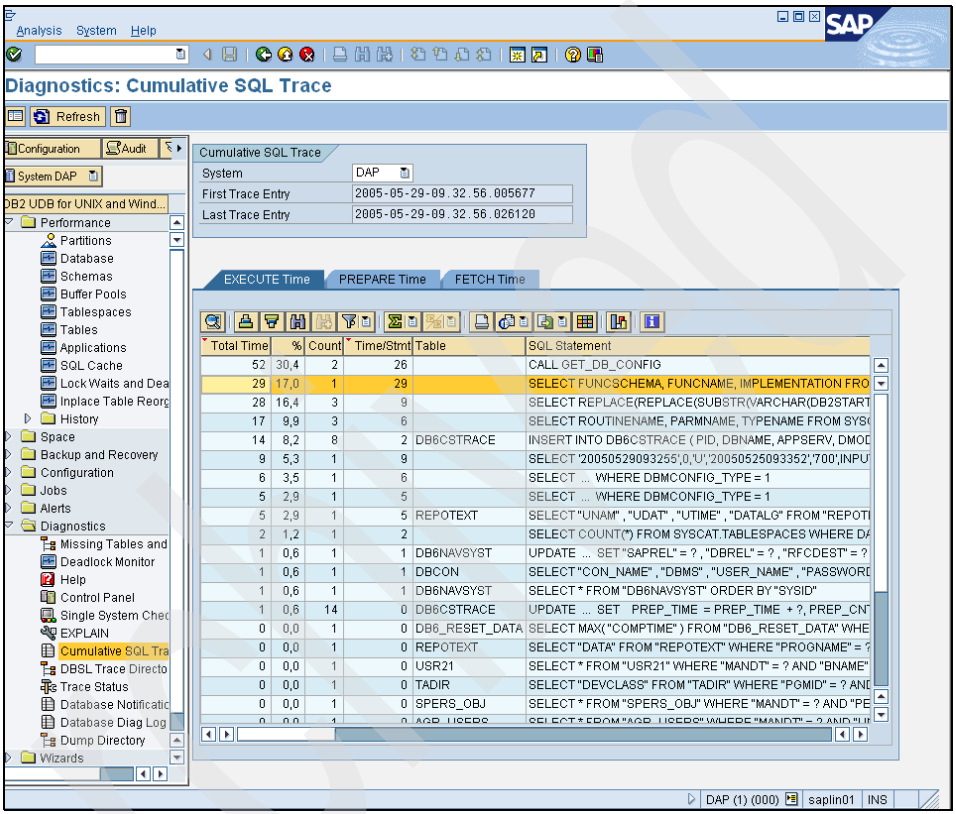


Figure 6-108 Diagnostics: Cumulative SQL Trace

Here are the means of the column on the `EXECUTE`, `PREPARE`, and `FETCH` tab:

- *Total Time*: Cumulative execution time of a statement.
- *%*: Proportional execution time of one statement with regard to all executed statements.
- *Count*: Number of executions
- *Time/Stmt*: Average execution time of one statement
- *Table*: Name of the table the SQL statements reads from. If the statement reads from more than one table, only the name of the first table will be displayed on this screen. The other names are displayed under Statement Information on the detail screen.

- *SQL Statement*: Complete SQL statement

To find out more details about a particular SQL statement, double-click that statement. See Figure 6-109.

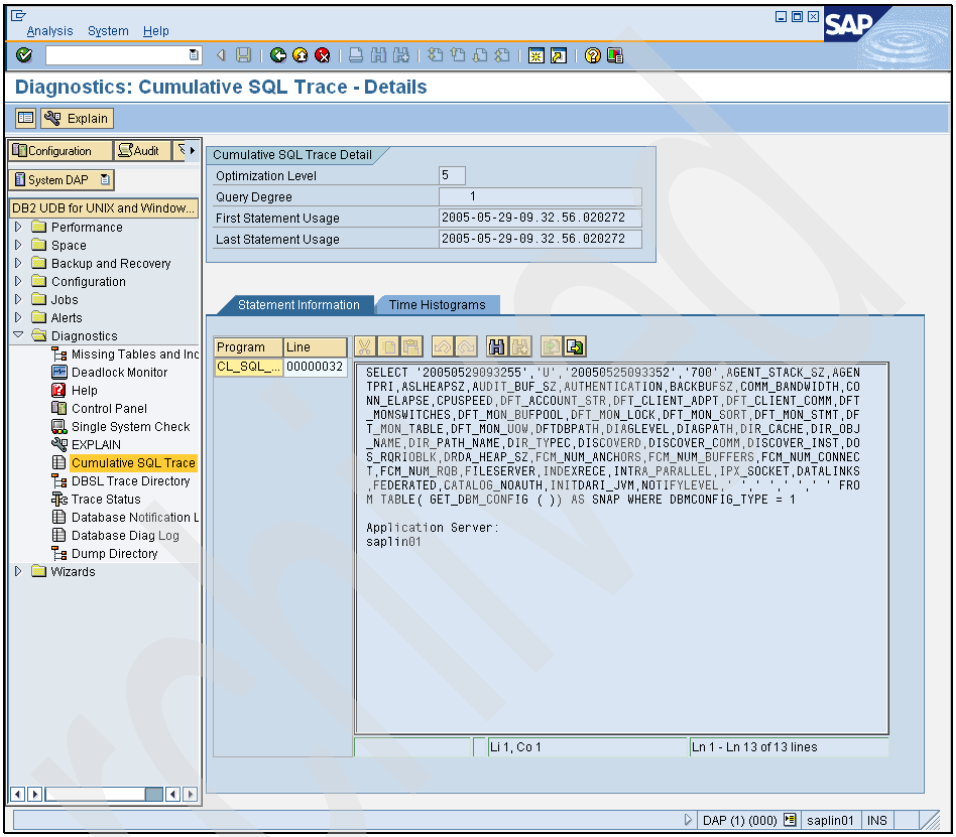


Figure 6-109 Diagnostics: Cumulative SQL Trace - Detail

In the Cumulative SQL Trace - Details screen, you find the following tabs:

- *Statement Information* tab: the entire SQL statement, the application server where the statement was executed and a list of all ABAP reports in which the statement can be found.
- *Time Histograms* tab: the distribution times of the selected SQL statement.

If you would like to analyze the access plan for the statement, click the **Explain** button.

To display the ABAP source program where the statement was defined, double-click the row in the list of ABAP reports.

6.11.7 DBSL Trace Directory

This is where you can view the trace files gathered from sequential DBSL trace and DBSL deadlock trace. To view the trace files, double-click the trace file. See Figure 6-110.

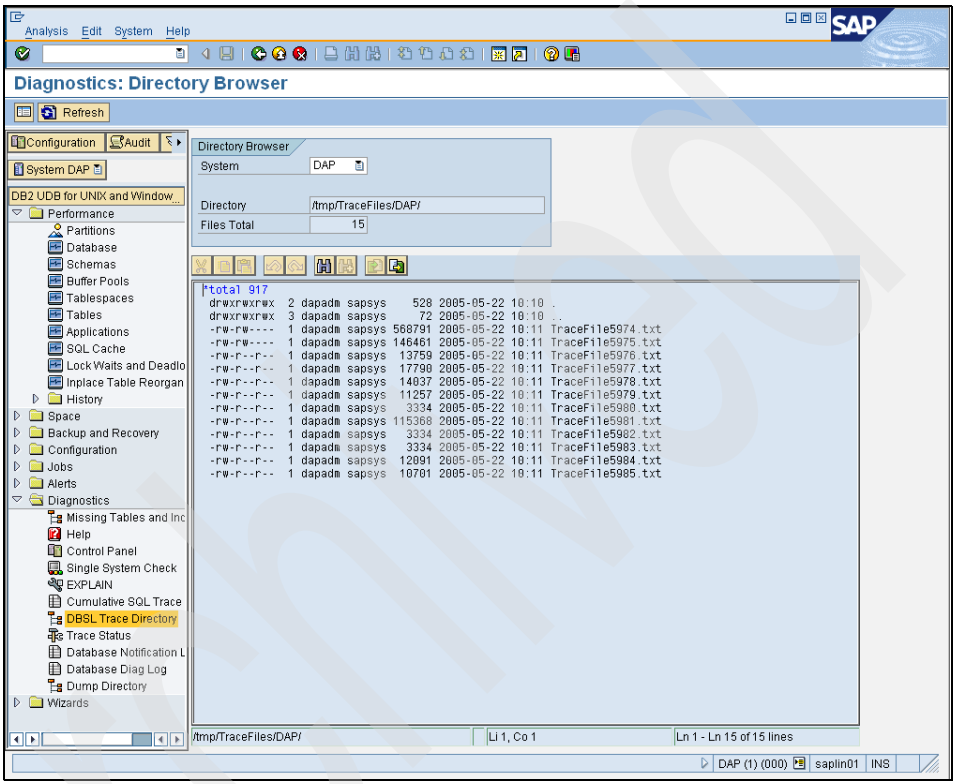


Figure 6-110 Diagnostics: DBSL Trace Directory

6.11.8 Trace Status

This allows you to turn on/off three traces: *DBSL trace*, *Cumulative Trace* (Cumulative SQL Trace), and *deadlock trace*. See Figure 6-111.

In a local system you can activate or deactivate the trace function by clicking the status icon:

- Red LED: switched off
- Green LED: switched on

You can also maintain trace parameters in a local system.

DBSL Trace

You can specify:

- *Trace Level*: Specifies the amount of data to be traced. The following trace levels are available:
  - *2*: Only statements are traced.
  - *3*: Statements and results are traced.
- *Number of I/O Records to Be Traced*: Number of result records to be traced for a statement. This value is only displayed if trace level 3 is activated.
- *Display Length for String/Raw Data*: Maximum output length.
- *DBSL Trace Search String*: If provided, only SQL statements containing this string are traced.
- *DBSL Trace Minimum Time Limit*: If provided, only SQL statements with execution times higher than this time limit are traced.

For more information on the DBSL trace, refer to “New deadlock monitoring GUI in DBA Cockpit and the enhanced DB2 UDB deadlock event monitor” on page 528.

Cumulative Trace

You can specify:

- *Trace Level*: Displays the trace level on the current application server.
  - *0*: Trace is switched off.
  - *1*: Trace is switched on.
- *First Trace Entry*: Displays the start time of this trace if trace information already exists.
- *Last Trace Entry*: Displays the end time of this trace if trace information already exists.
- *Number of Entries*: Displays the number of entries in this trace if trace information already exists.

Deadlock Trace

You can specify:

- *Detection Interval*: Only SQL statements running longer than this time are recorded for deadlock detection.

For more information on the DBSL deadlock trace, refer to section 8.2.3.8.

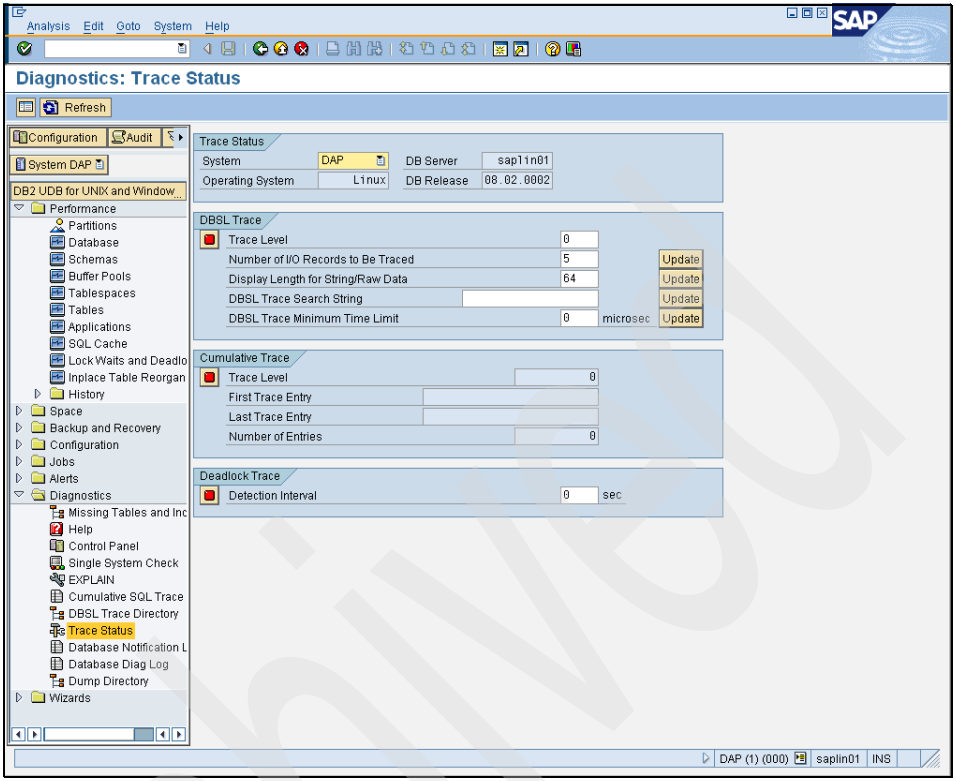


Figure 6-111 Diagnostics: Trace Status

6.11.9 Database Notification Log

The DB2 notification log file (see Figure 6-112) is an error notification issued by the system when a severe error occurs. Many notification messages provide additional information to supplement the `SQLCODE` that is provided. The type of event and the level of detail of the information gathered are determined by the `NOTIFYLEVEL` configuration parameter. However, detailed diagnostic information is not written to this log.

The `<instance_name>.nfy` file is an ASCII file that contains information logged by DB2. It is located in the directory specified by the `DIAGPATH` database manager configuration parameter.

Refer to Chapter 8, “Administration notification log and db2diag.log” on page 444 for more information on how to read the database notification log.

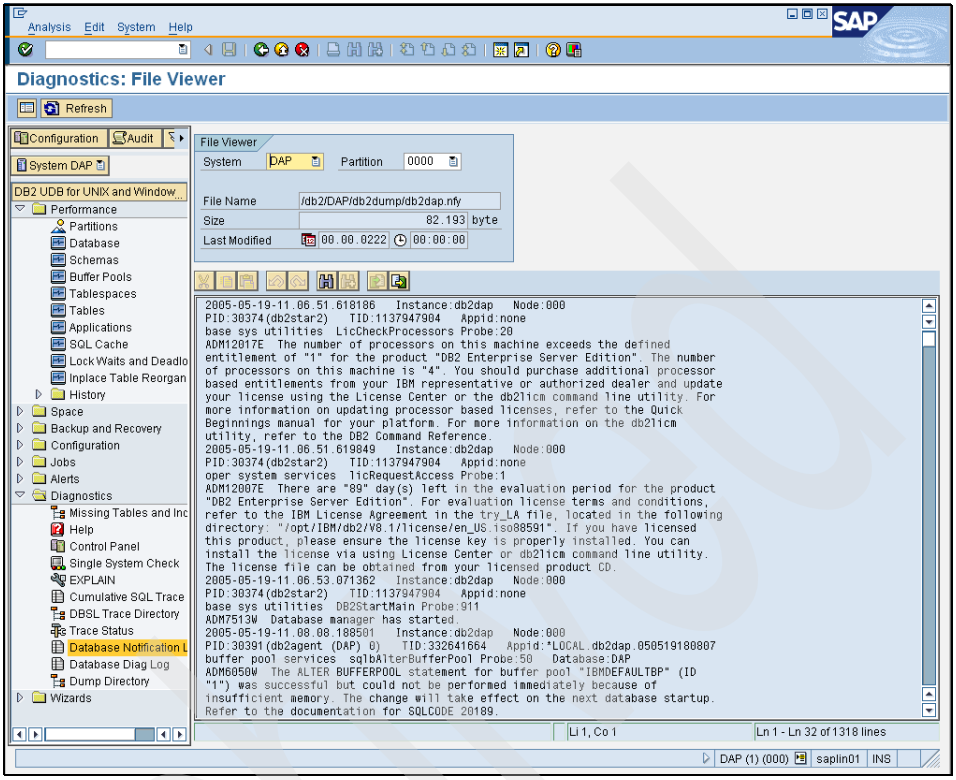


Figure 6-112 Diagnostics: Database Notification Log

6.11.10 Database Diag Log

The `db2diag.log file` (see Figure 6-113) is an ASCII file that contains diagnostic information logged by DB2 UDB. It is located in the directory specified by the `DIAGPATH` database manager configuration parameter. This information is used for problem determination and is intended for customer support.

Refer to Chapter 8, “Administration notification log and db2diag.log” on page 444 for more information on how to analyze `db2diag.log`.

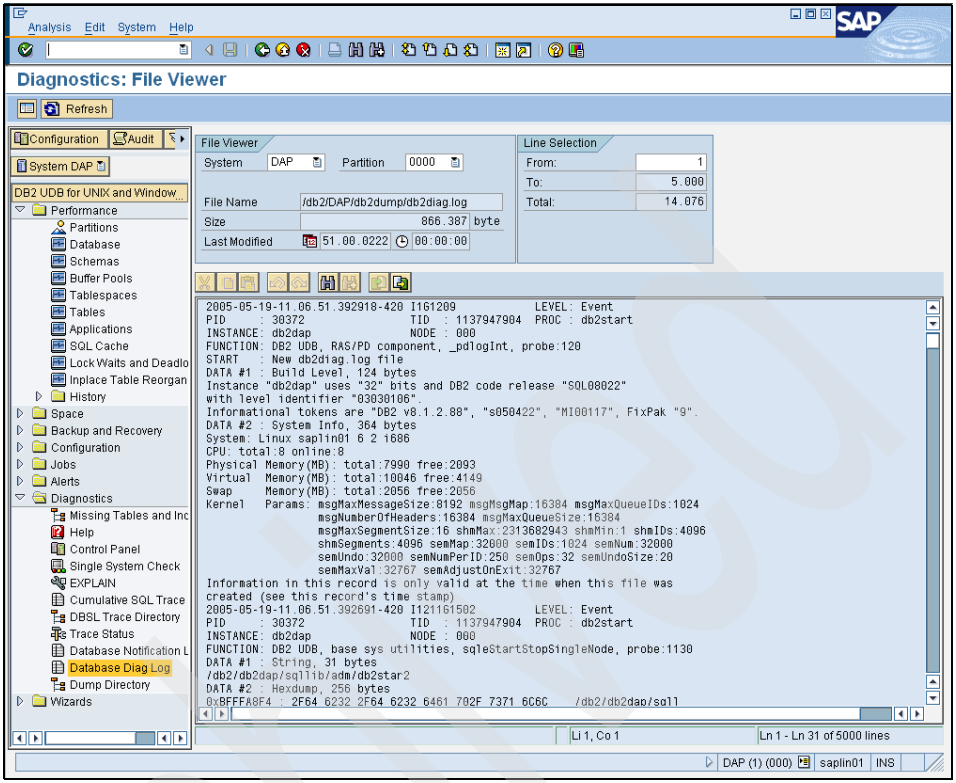


Figure 6-113 Diagnostics: Database Diag Log

6.11.11 Dump directory

This is where you can view the files in the DB2 Dump directory as specified by the DIAGPATH database manager configuration parameter (see Figure 6-114). You can find the following files in this directory:

- DB2 diag log (db2diag.log)
- DB2 notification log (<instance_name>.nfy)
- DB2 dump files
- User exit log and error files (for SAP DB2 log file management)
- Trace files

To view the content of an error log or trace file, double-click the file.

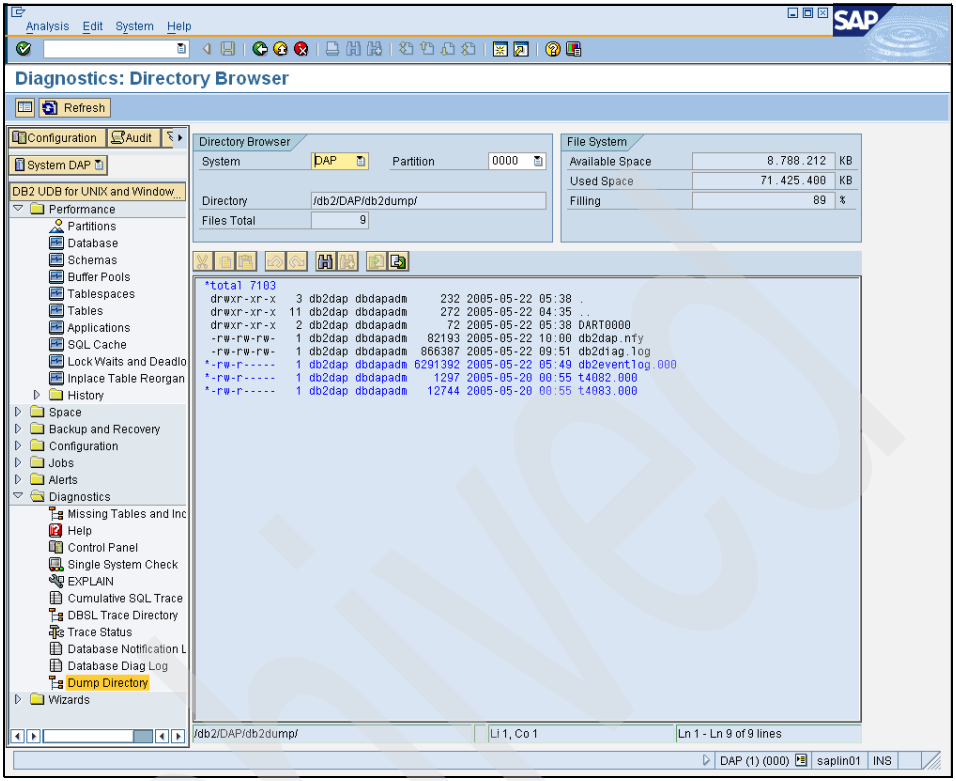


Figure 6-114 Diagnostics: Dump Directory

6.12 Partition Integration wizard

Partition Integration wizard is the second part of the process of enabling DPF environment in SAP NetWeaver 2004s. Here is a list of some of its tasks:

- Assign one or more newly added database partition(s) to one or more database partition group(s).
- Define tablespace containers on the new database partitions.
- Redistribute the data of the tablespaces if necessary.

To run the wizard, choose **Wizard** →**Partition Integration and** follow the step-by-step description in the screen.

7

Protecting your SAP data

This chapter describes the following topics:

- DB2 UDB log file management:
  - Legacy log file management
  - New log file management
- DB2 UDB backup and recovery:
  - Backup, restore, and recovery
  - Redirected restore
  - Database relocation
- Advanced backup techniques for large databases:
  - Hot standby database
  - Database clone
- Monitoring backup, restore, and recovery
- High availability:
  - HADR
  - Log file shipping
- Clustered solutions

7.1 Log file management

Beginning with V8.2, DB2 UDB has introduced a new log file management mechanism. In this section we discuss how the DB2 UDB logging feature works and describe the old and new log file management implementation.

By the end of this section, we also explain how to migrate from the legacy log management mode to the new one.

7.1.1 Basic logging concepts

A DB2 UDB database uses log files to record changes in its data. To store these changes, DB2 UDB records each data modification in log records. These log records allows DB2 UDB to know what information to process during:

- An implicit or explicit rollback operation
- Restart recovery
- Rollforward recovery

DB2 UDB log records contains information about operations done (for example, inserting a row into a table), operations undone (rolling back an insert operation), about changes in the database objects (altering a table, creating an index) and about modifications in the storage configuration (creating a table space).

The database information is stored in tables. From a physical point of view, the tables are a collection of pages on disk. For example, when an end-user wants to change a value in a row, DB2 must bring the page where the row resides from disk into memory (if the page is not already in memory, of course). The memory used is the one that is defined as the buffer pool of the table space where the table is created.

Let's suppose that the page required is in memory and the end-user executes an `UPDATE` statement. This statement has two effects. First, DB2 UDB records the operation, that means the update of information, as log records in the log buffer. Second, DB2 UDB changes the row of page in shared memory.

At a later time the information in the log buffers is written to disk by the logger process (*db2loggw*). The conditions that trigger a flush of the log buffer in memory to disk are as follows:

- A `COMMIT` statement is issued by the application
- A second has passed
- The log buffer becomes full

To sync up the changes in the buffer pool with the data in disk, the modified pages in the buffer pool are copied to disk by the page cleaners. The page cleaners are invoked when:

- The maximum amount of log space that should be read during crash recovery has been reached. This information is indicated by the SOFTMAX database configuration parameter.
- The maximum percentage of changed pages has been reached. The CHNGPGS_THRESH database configuration parameter defines this value.
- No more pages are available for an insert or update operation. In this case, one page must be copied to disk. This process is known as *victim page cleaning*, because one page from memory must be cleaned to let its space become free for another page.

As you can see, modified pages in the buffer pool are not flushed directly to disk when a change is made. However, to ensure that recovery is possible, DB2 UDB uses a strategy known as *write-ahead logging*. Using this technique, DB2 UDB ensures that log records are written to log files before the modified pages are written back to disk. This technique ensures that every data change is at least stored in the log records and makes recovery possible.

Many database configuration parameters define how DB2 UDB manages log files. To get information about how your database is configured regarding log management, use the command:

```
#db2 get db cfg for <database_name>
```

For a complete list of log file related database configuration parameters and the details, refer to the IBM DB2 UDB documentation, *Administration Guide: Implementation V8*, SC09-4820-01 and *Data Recovery and High Availability Guide and Reference V8*, SC09-4831-01.

Following are the set of database configuration parameters that define the number of log files and the log path:

```
Log file size (4KB)                         (LOGFILSIZ) = 16380
 Number of primary log files                (LOGPRIMARY) = 20
 Number of secondary log files               (LOGSECOND) = 40
 Changed path to log files                  (NEWLOGPATH) =
 Path to log files                                       =
/db2/DAP/log_dir/NODE0000/
 Overflow log path                     (OVERFLOWLOGPATH) =
 Mirror log path                         (MIRRORLOGPATH) =
 First active log file                                   = S0000017.LOG
 Block log on disk full                (BLK_LOG_DSK_FUL) = YES
```

Table 7-1 describes the meaning of each of these parameters.

Table 7-1 Database parameters define log file size, numbers, and path

Parameter	Description
LOGFILSIZ	This parameter defines the size of each primary and secondary log file created by DB2 UDB. The unit of measure for this parameter is 4 Kb.
LOGPRIMARY	This parameter specifies the number of log files that DB2 UDB can create and used for logging purposes.
LOGSECOND	This parameter defines the number of log files that DB2 UDB will create If the primary log files become full. Secondary log files are allocated, one at a time as needed, up to the maximum number specified by this parameter. If this parameter is set to -1, the database is configured with infinite active log space. In this situation, there is no limit on the total amount of log files a database can have; however, to be able to allow this, the database must be running in log retention mode.
NEWLOGPATH	This parameter is used as a placeholder for the new log file location. The new log file path becomes effective after all users disconnect from a database and the database is in consistent state. By changing this parameter you can redefine your active log directory.
LOGPATH	This parameter indicates which is the active log directory for the database. It is the place where log files are located. After database creation, log files are located under the SQLOGDIR directory, in the database directory, but this location can be changed by updating the NEWLOGPATH database configuration parameter.
OVERFLOWLOGPATH	This parameter is used to specify the directory where log files needed for a roll forward operation can be found.
MIRRORLOGPATH	This parameter defines the directory where mirrored log files are located. If this parameter is enabled, each log file in the log directory has a mirror copy in this directory. In case of a failure in the active log directory, DB2 UDB can still continue its operation using the log files in the mirror log path.

Parameter	Description
BLK_LOG_DSK_FULL	This parameter defines whether an application will hang up and wait until free space is available in the active log directory when that directory is full, or if the current transaction must be roll backed. If you want that the application waits, set this database configuration parameter to the value YES; DB2 UDB will try to create a new log file every five minutes until it succeeds. If you want to have the transaction rolled back, set this parameter to NO.

Let's see when the log space is allocated and used:

- Primary log files are created when the database is created or when a log file is moved to another location, because the NEWLOGPATH database configuration parameter was set.
- Secondary log files are created only when the primary log files are all in use and a transaction requires more space.
- If the LOGSECOND database configuration parameter is set to a value different than -1, infinite logging is not used. Therefore, it can also happen that secondary log files becomes full. In this situation, if the BLK_LOG_DSK_FULL database configuration parameter has the value NO, the application will receive an error and the transaction will be rolled back. However, if the value of this parameter is YES, DB2 UDB will try to create a new log file every five minutes until it succeeds.

The use of log files in the active log directory is also affected by the two types of logging mechanism:

- Circular logging
- Log retention mode

Circular logging is the default behavior when you create new databases. With this type of logging, once all log files are full, DB2 UDB reuses the log files starting from the first one.

If circular logging is used, you can only restore a database with full off-line backup copy, since the log file required for rollforward may not be available.

To see which logging mode is used in a database, use the command:

```
#db2 get db cfg for <database_name>
```

Then check the values of these database configuration parameters:

- In DB2 UDB V8.1 or lower, if LOGRETAIN is set to OFF and USEREXIT is set to OFF, the database is using circular logging.

- In DB2 UDB V8.2 or greater, if both LOGARCHMETH1 and LOGARCHMETH2 are set to OFF, the database is using circular logging.

Attention: In an SAP/ DB2 UDB environment, we do not recommend the use of this strategy for production environments, because recoverable facilities are limited to restoring a full off-line backup.

Remember that this is the default mode of newly created databases and must be changed to log retention mode for SAP production systems.

Log retention mode or *archive logging* is the recommended mode for production systems. In SAP/DB2 UDB environment, this mode is mandatory. The ways you can configure your database to archive log files are described in 7.1.2, "SAP log management" on page 319, if you use DB2 UDB V8.1 or lower, and in 7.1.3, "DB2 UDB V8.2 log management" on page 325, if you use DB2 UDB V8.2 or greater.

In this mode, a new log file is created to let DB2 UDB stores new log records. By archiving filled log files to a safe place, you can roll forward these log records and recover your database to a certain point in time.

To see if your database is in log retention mode, use the command:

```
#db2 get db cfg for <database_name>
```

Then look at the values of these database configuration parameters:

- In DB2 UDB V8.1 or lower, if LOGRETAIN is set to RECOVERY and USEREXIT is set to ON, the database is running in log retention mode.
- In DB2 UDB V8.2 or greater, if both LOGARCHMETH1 and LOGARCHMETH2 are set to a value different than OFF, the database is running in log retention mode.

Important: After a successful installation of SAP NetWeaver 2004s or earlier releases, or the installation of another prior SAP product, you must configure your database to run in log retention mode.

After this change, the database goes into *backup pending* state. You must then perform a full off-line database to make your database available again.

To estimate how much space you need in the active log directory, you can use the following formula:

```
(LOGPRIMARY + LOGSECOND) * (LOGFILSIZ + 2 ) * 4096 + 32768
```

Where:

- `LOGPRIMARY` is the number of primary log files, defined in the database configuration file.
- `LOGSECOND` is the number of secondary log files, defined in the database configuration file; `LOGSECOND` cannot be set to -1. (When `LOGSECOND` is set to -1, you are requesting an infinite active log space.).
- `LOGFILSIZ` is the number of pages in each log file, defined in the database configuration file.
- `2` is the number of header pages required for each log file.
- `4096` is the number of bytes in one page.
- `32768` is the number of bytes for log control files.

If you want to mirror log files, the result of this formula must be multiplied by 2.

Tip: Keep in mind that the total amount of space cannot be greater than 256 Gb for DB2 UDB V8.1 or greater.

Depending on the combination of DB2 UDB version and SAP product you are running on, there two possible ways to manage log files:

- SAP log management: log management is done using the user exit feature.
- DB2 UDB V8.2 log management: log management is done using a new component of DB2 UDB, the DB2 log manager (*db2logmgr*).

In the next section, we describe each of them in detail.

7.1.2 SAP log management

SAP releases prior to SAP NetWeaver 2004s use the user exit feature provided by DB2 UDB to manage log files. To accomplish this goal, the following components are used:

- The Admin DB database (`ADM<`*`DBSID`*`>`) created during the installation of SAP DB2 Admin Tools
- The `db2uext2` (known as the DB2 UDB logging user exit), `brarchive` and `brrestore` programs
- The log management facilities already available in DB2 UDB

This section describes the log file archive and restore methods provided by SAP DB2 Admin Tools, explains how log files are named, and discusses when log files are deleted.

More information about this subject can be found in the SAP documentation, *SAP NetWeaver '04 Database Administration Guide: SAP on IBM DB2 Universal Database for UNIX and Windows* and *IBM DB2 Universal Database for UNIX and Windows: New Log File Management*.

Components description

SAP DB2 Admin Tools complements DB2 UDB log management features. This set of tools are automatically installed as part of an SAP system installation with `R3SETUP`. If the system is installed with `SAPinst`, SAP DB2 Admin Tools should be installed afterwards, because they are not automatically installed.

Let's see what each component does:

- The `db2uext2` program is used to copy log files from the log directory to the appropriate target destination. The target destination vary depends on the kind of archive used.
- The `brarchive` program is responsible for backing up log files from the archive directory. Keep in mind that each operation done on a log file in the archive directory is recorded in the Admin DB database.
- The `brrestore` program is the counterpart of the `brarchive` program. It is used to restore log files from the backend repository. The log files are copied to the retrieve directory. Also, in this case, each operation done is recorded in the Admin DB database.

Log file archiving and restoring options

Using the user exit mechanism together with SAP DB2 Admin Tools, there are two ways to manage log files:

- Indirect log file archive and restore
- Direct log file archive and restore

When the user exit mechanism is configured, DB2 UDB automatically invokes the `db2uext2` program to archive each log file in the log directory as soon as it is full or it is closed. The target destination of the log file depends on the configuration of the user exit program.

Indirect log file archive and restore

In this scenario, the user exit program, `db2uext2`, copies log files into the archive directory. Then the `brarchive` program copies these files to the backend repository, such as disk, TSM, or tape, etc.

During a restore operation, the `brrestore` program copies the archived log files from the backend repository into the retrieve directory. The latter case occurs only if the log files needed are not already available in the archive directory.

In summary, if log files are needed for a roll forward operation, log files are available from the archive or retrieve directory.

Figure 7-1 shows how these three programs interact together to archive and restore DB2 UDB log files.

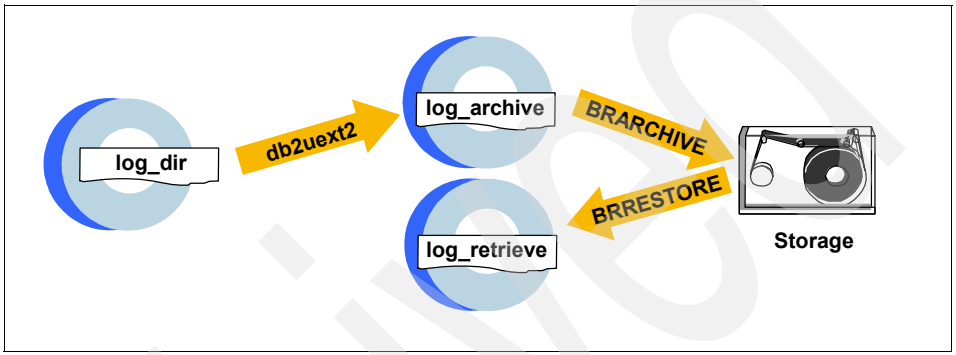


Figure 7-1 Indirect log file archive and restore

Direct log file archive and restore

In this scenario, when log files are archived, the `db2uext2` program sends them directly to the backend repository. To be able to do this, a storage management product, such as Tivoli Storage Manager or other, must be used.

In the case of a restore operation, during the roll forward phase of the restore process, log files are automatically retrieved from the back end repository and processed accordingly.

Figure 7-2 depicts this scenario.

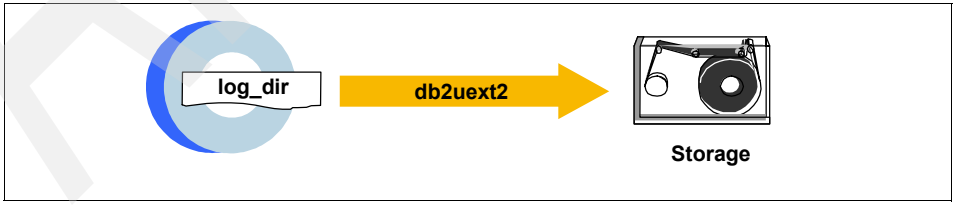


Figure 7-2 Direct log file archive and restore

Comparison of indirect and direct log file archive and restore

Direct log file archive is the recommended method to use. It is simple and more reliable than indirect archive. In this case, once log files are full or closed, they are copied directly into the storage repository. However, this method requires a storage management solution.

Only indirect log file archive and restore supports archiving to tape without the use of a storage management solution; however, you must consider the risk of failure existing with this method.

When you use indirect log file archive, log files are copied to the archive log directory before being sent to the storage repository by the `brarchive` program. If there is a disk failure on the archive directory, every log file on it not copied to the storage repository is lost.

As we described, this method also requires more administration than the direct archive and restore option because it uses the Admin DB database to store information about log files.

SAP DB2 Admin Tools configuration

The configuration of the `db2uext2`, `brarchive` and `brrestore` programs is done in the `init<DBSID>.db6` file. The location of this file depends on the version of the SAP system and on the operating system:

- For SAP system releases 4.6D or lower:
  - Linux and UNIX: `/usr/sap/<SAPSID>/SYS/global`
  - Windows: `<drive>:\usr\sap\<SAPSID>\SYS\global`
- For SAP Web AS (Application Server) 6.10 or higher:
  - Linux and UNIX: `<INSTHOME>/admintools`
  - Windows: `<INSTHOME>\admintools`

`init<DBSID>.db6` is a text file where configuration variables for `db2uext2`, `brarchive` and `brrestore` programs are defined. The values of these variables are used by the three programs during operation.

Table 7-2 shows most important variables regarding these utilities:

Table 7-2 Configuration variables for db2uext2, brarchive, and brrestore

Variable	Description
`DB2DB6_ARCHIVE_PATH`	Indicates the location of the archive directory
`DB2DB6_RETRIEVE_PATH`	Indicate the location of the retrieve directory

Variable	Description
DB2DB6_UEXIT_DIRECT	For indirect archive and recovery, this variable must be set as follows: DB2DB6_UEXIT_DIRECT=OFF If Tivoli Storage Manager is used, this variable must be set as follows: DB2DB6_UEXIT_DIRECT=TSM:<mgmt class1>[+<mgmt class2>] For archiving to disk, this variable must be set as follows: DB2DB6_UEXIT_DIRECT=DISK For using another storage manager, this variable must as follows: DB2DB6_UEXIT_DIRECT=VENDOR
DB2DB6_VENDOR_UEXIT	If DB2DB6_UEXIT_DIRECT is set to VENDOR, this variable must point to user exit provided by the vendor

Log file naming convention

DB2 UDB log files use the format Snnnnnnn.LOG, where *nnnnnnn* is a seven digit number ranging from 0000000 to 9999999. At database creation time, the number starts with 0000000 and is incremented when new log files are used. However, when you change log retention mode from enable to disable or vice versa the number is reset.

If direct archive is used, log file name is not modified except when the direct archive target destination is a disk. The pattern NODEXXXX is appended to the name of the log file to make the distinguish for multi-partitioned environments.

If indirect archive is used, the file-closure timestamp (14 digits) and the node number (NODEXXXX) are appended to the log file name when the log file is copied to the archive directory. During a roll forward operation, when log files are copied back from the archive or retrieve directory to the log directory, all the appended information is removed to bring back the name to its original form.

Log file deletion

Log files in DB2 UDB log directory should not be deleted using the rm or del command, DB2 UDB automatically deletes or reuses the log files after they are successfully archived by the user exit program.

Attention: Remember, in SAP/DB2 UDB environment, you should not delete log files manually; instead, use the utilities provided.

However, some actions must be taken for the archived log files. These actions depend on its target destinations.

Log files on disk

If the direct archive mechanism is used, no log file deletion is needed, because log files are sent directly to the backend repository.

In the other case, if the indirect archive mechanism is used, archived log files in the archive directory are only deleted if the `brachive` program is invoked with `-sd` and `-ssd` options. The same applies for the log files restored to the retrieve directory; instead of calling the program `brarchive`, you call the program `brrestore`.

Log files on tape

The Admin DB database always records the contents of the tapes. If a tape containing log files is reused, all entries in the Admin DB database are removed before the tape is overwritten with new log files. Therefore the Admin DB database does not grow.

If a tape is lost or becomes unreadable, you should use the `brrestore` program with the `-dt`, `-v` and `-out` options, to sync the contents of the Admin DB database.

Log files on Tivoli Storage Manager

When TSM is used, log file deletion under TMS differs between using an archive copy group and backup copy group.

In the case of an archive copy group, TSM automatically deletes log files. If direct archive for log file management is selected, no further action is required. If indirect archive is used, the Admin DB database has no knowledge of the automatically log file deletion done by TSM and it gets out of sync with TSM. Under this situation, you should call the `brrestore` program with `-delete` option to delete log files to update Admin DB database.

If a backup copy group is defined in TSM, log files needs to be deleted manually. If direct archive is used, use DB2 UDB command `db6adutl` to delete log files. If indirect archive is used, call the `brrestore` program with `-delete` option to delete log files.

Log files on other storage vendors

If another vendor storage manager product is used, the `brrestore` program with the `-delete` option must be called to delete log files.

7.1.3 DB2 UDB V8.2 log management

DB2 UDB V8.2 has added two components in engine architecture to manage log files. These two new components are:

- DB2 log manager (`db2logmgr`)
- DB2 tape manager (`db2tapemgr`)

DB2 log manager

DB2 log manager (`db2logmgr`) is the central component for managing log files, responsible for archiving and retrieving log files.

DB2 UDB log manager supports the backend repositories described in Table 7-3.

Table 7-3 DB2 log manager target destinations

Backend repositories	Description
`USEREXIT`	This archive media provides downward compatibility to the old user exit interface.
`TSM`	Built-in support for Tivoli Storage Manager
`DISK`	The target destination for archiving log files is a disk
`VENDOR`	A storage vendor can write its own library to allow the archive of log files

This information is configured using database configuration parameters. You can choose two backend repositories for the log files archived by DB2 UDB log manager.

To avoid the problem of log directory full, when the backend repository is not available, the DB2 UDB log manager copies log files to a local directory defined by the database parameter configuration `FAILARCHPATH`, this path is used as intermediate repository for log files. Once the backend repository becomes available, the DB2 UDB log manager moves the log files in intermediate repository to the backend.

When retrieving log files from the backend repository for a database recovery or rollforward operation, DB2 log manager copies the log files to the DB2 transaction log directory (`logpath` DB configuraiton parameter).

Figure 7-3 illustrates the DB2 log manager architecture.

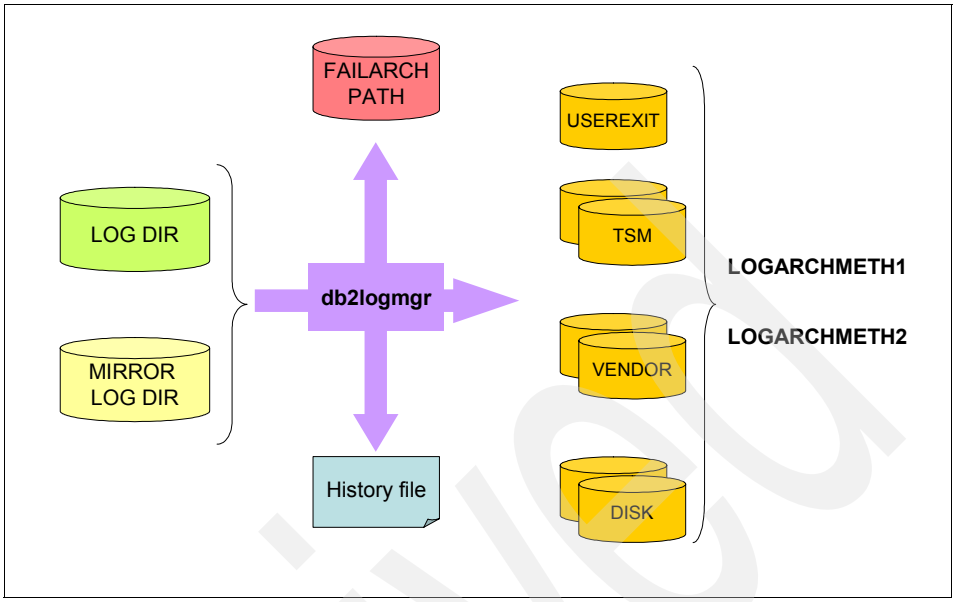


Figure 7-3 DB2 log manager architecture

Whenever a log file is written to the backend repository, its location is recorded in the DB2 history file. Using the history file DB2 log manager knows exact the set of log files that corresponds to a backup.

To show the information stored in the history file, use the command:

```
#db2 list history archive log all for <database_name>
```

DB2 log manager configuration

Table 7-4 describes the database configuration parameters used by DB2 log manager:

Table 7-4 Configuration parameters for DB2 log management

Parameter	Description
LOGARCHMETH1	This parameter specifies the media type of the primary destination for archived logs. The possible values are described in Table 7-3.
LOGARCHOPTS1	This parameter specifies the options field for the primary destination for archived logs (if required).

Parameter	Description
LOGARCHMETH2	This parameter specifies the media type of the secondary destination for archived logs. If this path is specified, log files will be archived to both this destination and the destination specified by the LOGARCHMETH1 database configuration parameter. Only DISK, TSM and vendor are allowed values for this parameter.
LOGARCHOPTS2	This parameter specifies the options field for the secondary destination for archived logs (if required).
FAILARCHPATH	This parameter specifies a path to which DB2 UDB will try to archive log files if the log files cannot be archived to either the primary or the secondary (if set) archive destinations. This specified path must reference a disk.
NUMARCHRETRY	This parameter specifies the number of times that DB2 UDB should try to archive a log file to the primary or the secondary archive directory before trying to archive log files to the failover directory. This parameter is only used if the FAILARCHPATH database configuration parameter is set. If NUMARCHRETRY is not set, DB2 will continuously retry archiving to the primary or the secondary log path.
ARCHRETRYDELAY	This parameter specifies the number of seconds to wait after a failed archive attempt before trying to archive the log file again. Subsequent retries will only take affect if the value of the NUMARCHRETRY database configuration parameter is at least 1.
OVERFLOWLOGPATH	This parameter allows you to specify a location for DB2 to find log files that are needed for a rollforward operation.

After SAP NetWeaver 2004s is installed, by default, DB2 log manager is configured as follows:

```
Overflow log path                     (OVERFLOWLOGPATH) =
First log archive method                 (LOGARCHMETH1) = OFF
Options for logarchmeth1                  (LOGARCHOPT1) =
Second log archive method                (LOGARCHMETH2) = OFF
Options for logarchmeth2                  (LOGARCHOPT2) =
Failover log archive path                (FAILARCHPATH) =
Number of log archive retries on error   (NUMARCHRETRY) = 5
Log archive retry Delay (secs)         (ARCHRETRYDELAY) = 20
Vendor options                               (VENDOROPT) =
```

As you can see, both LOGARCHMETH1 and LOGARCHMETH2 database configuration parameter are disabled; this means that the database is not running in log retention mode.

DB2 log manager log chain support

One of the most important advantages of the DB2 log manager is its log file chain support.

DB2 log files have consecutive names from `S0000000.LOG` to `S9999999.LOG`. If a log file is full, DB2 UDB creates a new log file with the next number. However, if a point in time restore is executed, log files are reused.

Before the introduction of the DB2 log manager, the DBA must correctly handle the situation of having two log files with same name but different content. Given the fact that DB2 log manager stores the location of the log file in the history file, DB2 log manager can also search the history file to know which is the exact log file it needs.

Let's consider a possible scenario where a log file chain is used (Figure 7-4).

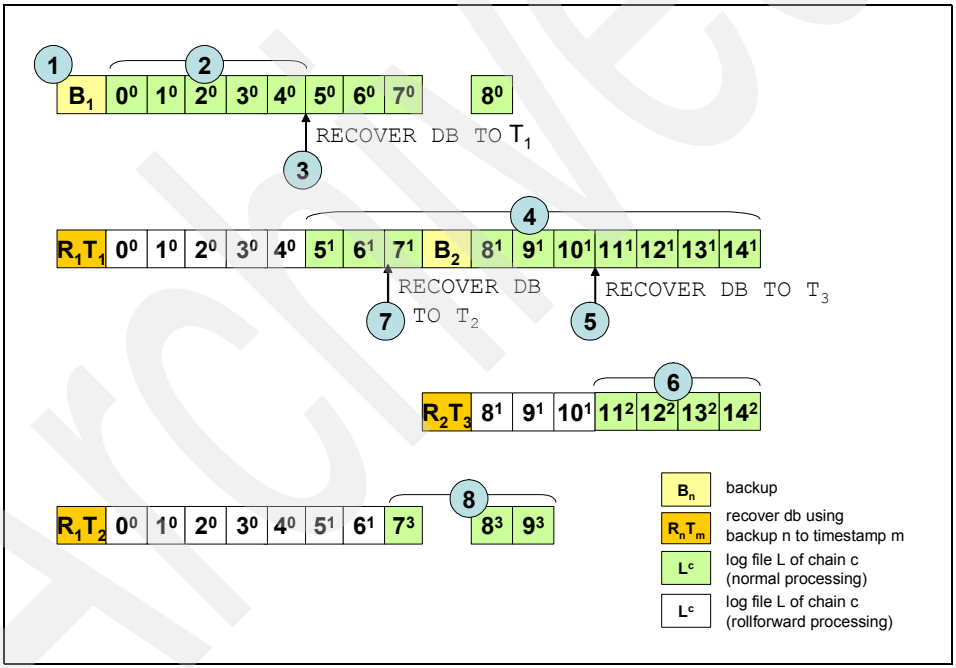


Figure 7-4 Log chain support

Figure 7-4 shows the content of the history file, if the following steps are performed:

1. An offline backup B1 is created.
2. Transactional work on the database creates log files 0 - 8 that belong to log file chain 0.

3. A database recovery to point-in time T1 is performed. This is done by using backup image 1 and applying log files 0 - 4.
4. Transactional work on the database creates log files 5 -14, which belong to log file chain 1.
5. A database recovery to point-in time T3 is performed. This is done by using backup image 2 and applying log files 8 - 10 of log file chain 1.
6. Transactional work on the database creates log files 11 - 14, which belong to log file chain 2.
7. A database recovery to point-in time T2 is performed. This is done by using backup image 1 and applying log files 0 - 4 of log file chain 0 and log files 5 - 6 of log file chain 1.
8. Transactional work on the database creates log files 7 - 9, which belong to log file chain 3.

Because all location information of the log files is stored in the history file, this ensures log file chain support and also ensures that you can recover the database to any point in time with the right set of log files.

Tip: DB2 V8.2 introduces the `RECOVERY` command which simplifies the recovery operation. Once launched, the `RECOVERY` command performs the restore and rollforward operation. The `RECOVERY` command is discussed in 7.2.3, “Recovering the database” on page 351.

DB2 log manager backend support

DB2 UDB log manager can work with different backend options. In the following sections, we briefly describe each of these options.

Disk

For active log file archiving to disk, you need to use the prefix `DISK:` in the database configuration parameter `LOGARCHMETH1` and/or `LOGARCHMETH2`:

```
#db2 update db cfg for DAP using LOGARCHMETH1 DISK:/db2/DAP/log_archive
```

Once log file archiving to disk is activated, DB2 log manager will store log files in the following directory structure:

```
<log_archive>/<instance>/<database>/NODExxxx/Cyyyyyy/Szzzzzzz.LOG
```

Table 7-5 describes each of these directories.

Table 7-5 Log chain directories

Subdirectory	Description
<log_archive>	Path specified by the database configuration parameter LOGARCHMETH1 or LOGARCHMETH2
<instance>	The name of the instance
<database>	The name of the database
NODE####	The database partition number, that is, NODE0000
Cyyyyyyy	The log file chain to which the log files belong. that is, C0000001
Szzzzzzz.LOG	The name of the log file, that is, S0000001.LOG

The entries in the history file are as follows:

```
Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log  Backup ID
-- --- ------------------ ---- --- ------------ ------------ --------------
 X  D  20050522095054      1    D  S0000000.LOG C0000000
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
   Comment:
Start Time: 20050522095054
  End Time: 20050522181103
    Status: A
--------------------------------------------------------------------------------
 EID: 40 Location:
/db2/DAP/log_archive/db2dap/DAP/NODE0000/C0000000/S0000000.LOG
```

From DB2 log manager point of view, the most important fields are:

- The Type column:

  It shows which LOGARCHMETHx is used; in this case, LOGARCHMETH1.

- The Dev(ice) column:

  It shows which kind of backend repository is used; in this case, log files are archived to DISK.

- The EID field and its location:

  The EID field is a unique identifier for an entry in the history file. The location tells where the log file is stored. In our example, the log file is: /db2/DAP/log_archive/db2dap/DAP/NODE0000/C0000000/S0000000.LOG. As you can see the convention for storing archived log files is followed.

Tivoli Storage Manager

To activate log file archiving to Tivoli Storage Manager, you need to use the prefix TSM: in the database configuration parameter LOGARCHMETH1 or LOGARCHMETH2:

```
#db2 update db cfg for DAP using LOGARCHMETH1 TSM:LOGMGMTCLS
```

If you need to provide TSM parameters, you can use the database configuration parameter LOGARCHOPTS1 or LOGARCHOPTS2 for this:

```
#db2 update db cfg for DAP using LOGARCHOPTS1 "-fromnode dapsystem"
```

The log files are stored in TSM in the file space of the database with the file name as follows:

- As high level name, the partition number is used (that is, NODE0000).
- As log level name, the log file name including the log file chain (Szzzzzzz_Cyyyyyyy.LOG) is used.

When you use TSM, in the history file the Dev(ice) column has the value A, first letter of the previous name of TSM (Adstar Storage Manager). The Type column shows which LOGARCHMETHx is used and the location field points to the storage management class configured.

Other storage vendors

If you want to use another storage vendor solution, that vendor must provide you with the required library. Vendors can consultant the documentation, *IBM DB2 UDB Administrative API Reference V8*, SC09-4824-01, to know the details of the library's implementation.

To activate log file archiving with a vendor library, you need to use the prefix VENDOR: in the database configuration parameter LOGARCHMETH1 or LOGARCHMETH2:

```
#db2 update db cfg for DAP using LOGARCHMETH1
VENDOR:d:\sqllib\bin\db2vendor.dll
```

If you need to provide parameters to the vendor library, you can set them in the database configuration parameter LOGARCHOPTS1 or LOGARCHOPTS2.

When you use another vendor storage manager, in the history file the Dev(ice) column has the value 0. The Type column shows which LOGARCHMETHx is used and the location field shows the used vendor library.

User exit

To use the user exit, you can specify the value USEREXIT for the database configuration parameter LOGARCHMETH1. In this case, you cannot set the database configuration parameter LOGARCHMETH2.

Using this mode, the user exit program db2uext2 is called to archive and retrieve log files.

Restriction: log file chain support is not available under the user exit concept.

To activate the user exit mechanism, you just need to type:

```
#db2 update db cfg for DAP using LOGARCHMETH1 USEREXIT
```

When you use the user exit mechanism, in the history file the Dev(ice) column has the value U. The Type column shows which LOGARCHMETHx is used and the location field is empty.

DB2 tape manager

DB2 tape manager (db2tapemgr) is a command line executable that can be used to archive log files to tape and retrieve log files from tape. This utility is necessary because DB2 log manager cannot access the tape directly. You have to call the DB2 tape manager explicitly from the command line.

Log files retrieved by DB2 tape manager are copied in the DB2 overflow log directory (OVERFLOWLOGPATH) and not in the transaction log directory (logpath). From that location, DB2 UDB reads the log files to perform a database recovery or rollforward operation.

If log files are archived to tape, DB2 tape manager automatically updates the history file providing needed information is necessary for database recovery.

Figure 7-5 shows DB2 tape manager architecture.

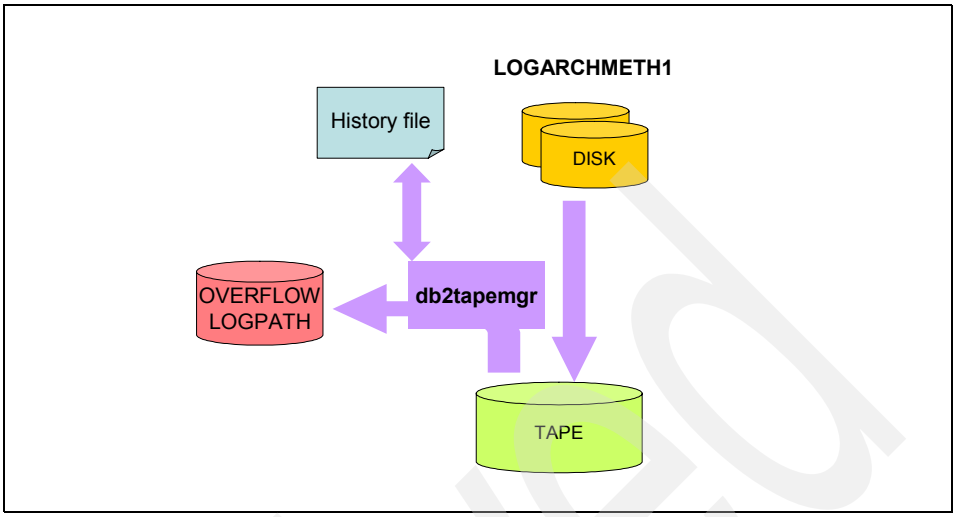


Figure 7-5 DB2 tape manager architecture

DB2 tape manager can also be used in multi-partition environments, but it must be called in each partition separately.

DB2 tape configuration

For tape support, the database manager configuration parameter `LOGARCHMETH1` must be set to a disk location. DB2 tape manager checks the history file for entries that are related to this parameter, when it is invoked to archive log files. It ignores the entries related to the database configuration parameter `LOGARCHMETH2`.

The DB2 UDB registry variable DB2_TAPEMGR_TAPE_EXPIRATION defines the number of days before the tape can be overwritten. Setting this variable avoids overwriting log files on tape that are still needed for a database recovery.

DB2 tape manager operations

Each tape used by DB2 tape manager is labeled. The label used can be generated automatically or provided as an option of this command.

This tape label can be up to 22 characters long and must be alphanumeric. If no tape label is provided, the automated label is made of the database name and the current timestamp as a 14-digit timestamp, i.e:
`<database_name><YYYYMMDDHHMMSS>`.

A tape used by DB2 tape manager has the following physical layout:

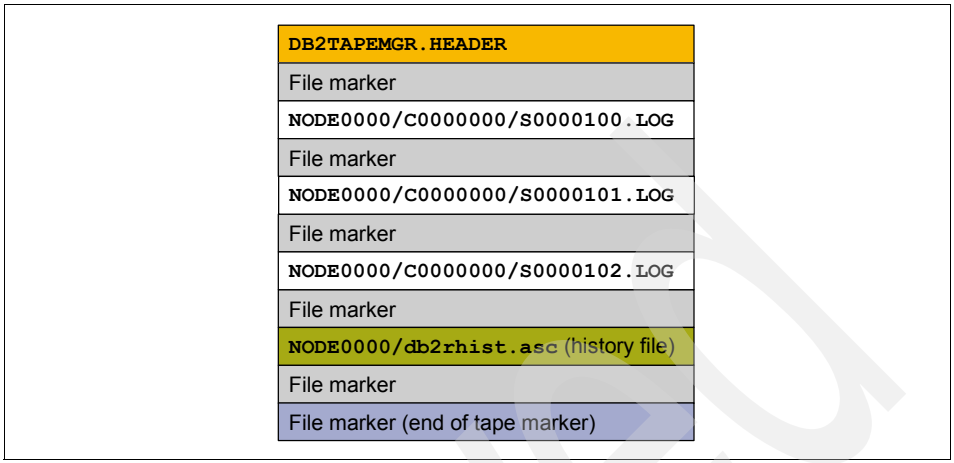


Figure 7-6 Physical tape layout

At the beginning of each tape, there is a tape header named `DB2TAPEMGR.HEADER`. The tape header contains the tape label, the host name, the instance, the database, and the partition of the log files archived. It also has a field with the list of contents in the tape.

Each log file written to tape is encapsulated in a `cpio` archive and as shown in Figure 7-6 each log file archived is delimited by a file marker.

To archive log files to tape, enter the following command:

```
db2tapemgr db DAP store on /dev/nst0
```

This command archives log files of the database `DAP` residing on disk to the tape device `/dev/st0`. As the `STORE` option is used, log files are deleted from disk afterwards. If the `DOUBLE STORE` option is used, log files are deleted from disk only if they are copied to tape the second time.

Important: The tape device used in the `ON` option must be a non-rewind tape device.

To retrieve all log files from tape, use the following command:

```
db2tapemgr db DAP retrieve for rollforward to end of logs from /dev/nst0
```

If more tape devices are required, they will be asked as needed.

To retrieve the history file stored in a tape, enter the command:

```
db2tapemgr db DAP retrieve history file to /db2/db2dap
```

To retrieve log files using a particular history file, use the command:

```
db2tapemgr db DAP retrieve history file /db2/db2dap/db2rhist.asc
```

To eject the tape from the tape drive, enter the command:

```
db2tapemgr eject tape /dev/nst0
```

To show the contents of the tape header file, enter the command:

```
db2tapemgr show tape header /dev/st0
```

To delete tape labels, enter the following command:

```
db2tapemgr delete tape label TAPE0
```

This command is used when a tape is lost or corrupted, because log file entries must be deleted from the history file to reflect that those log files stored in the tape associated with that label can no longer be retrieved.

For the complete syntax of the db2 tape manager, see DB2 UDB product documentation, *IBM DB2 UDB Command Reference V8*, SC09-4828-01.

7.1.4 Comparison of legacy and DB2 V8.2 log file management

As a summary, Table 7-6 compares most important features of both log file management types:

Table 7-6 Features comparison

Feature	Legacy log file management	DB2 V8.2 log file management
DB2 UDB version	Until DB2 V8.2 For backward compatibility in DB2 V8.2, set `LOGARCHMETH1` to `USEREXIT`	New with DB2 UDB V8.2
Multi partition support	Yes	Restricted to tape

Feature	Legacy log file management	DB2 V8.2 log file management
Log file chain support	No However, when indirect archive method it is possible to avoid a log file overwrite, because log files names are stored in the Admin DB database	Yes DB2 UDB V8.2 knows how to identify each log file correctly. It uses the information in the history file
Information about location of log files	For indirect archive method, that information is stored in the Admin DB database For direct archive method, that information does not exist	History file In DB2 UDB V8.2, all log information is stored in the history file
Backend support		
TSM	Yes in both cases For indirect archiving, use TSM to get log files from log archive directory For direct archiving, set `DB2DB6_UEXIT_DIRECT=TSM`	Yes Set `LOGARCHMETH1` or `LOGARCHMETH2` to `TSM`
Vendor	Yes in both cases For indirect archiving, copy log files from log archive directory using some vendor facility For direct archiving, set `DB2DB6_UEXIT_DIRECT=VENDOR`	Yes Set `LOGARCHMETH1` or `LOGARCHMETH2` to `VENDOR`
Tape	Only the indirect method supports the use of a tape device	Yes Use `db2tapemgr`

Feature	Legacy log file management	DB2 V8.2 log file management
Disk	Yes, in both cases For indirect archiving, use a solution like customized script. For direct archiving, set DB2DB6_UEXIT_DIRECT=DISK in init<*DBSID*>.db6	Yes Set LOGARCHMETH1 or LOGARCHMETH2 to DISK

7.1.5 Migration from legacy to DB2 log file management

We describe here the procedure you need to follow to move from legacy log management to DB2 log manager.

After the installation of DB2 UDB V8.2, the new log file management is not yet activated. The database configuration parameter LOGARCHMETH1 has the value USEREXIT and the program db2uext2 is used.

The steps to active DB2 log file management are:

- To enable display log file information in CCMS when the DB2 log file management is enabled, perform the following steps:
  a. If scheduled in transaction DB13, delete the jobs *Archive inactive log files into ADSM*, *Archive inactive log files onto device,* and *Initialize Tape*.
  b. If a job that executes the program *sddb6mir* is scheduled in transaction SM17, delete it.
  c. For SAP system Releases 4.6B, 4.6C and SAP NetWeaver 6.10 apply the appropriate SAP Support Package level according to SAP Note *300828*.
  d. For SAP system Releases 4.6B and higher, install the latest version of program dmdb6rdi. dmdb6rdi is the interface between DB2 UDB and CCMS. Refer to SAP Note *768817* for the latest patch level and other details.
- To enable the DB2 log file management, adjust the log file management database configuration parameters described in Table 7-4.
- Perform an offline backup of the database.
- If you have been using indirect archiving, use brarchive to store the remaining log files.
- De-install the SAP DB2 log file management tools using the commando sddb6ins -d. This command will remove the relevant executable, the init<*DBSID*>.db6 configuration file and the Admin DB database.

7.2 DB2 UDB backup and recovery

DB2 UDB provides different backup and recovery options that you can perform online — consistent online, offline, incremental, and delta database or table space backup. The backup data can be compressed. You can tune the backup and restore manually or use the DB2 UDB self tuning feature of DB2 UDB for backup and restore.

DB2 UDB has built-in backup and restore utilities which provide an interface to Tivoli Storage Management (TSM), libraries to 3rd party vendors, and an interface to use the X/Open Backup Services Application Programmer's Interface (XBSA).

Important: We strongly recommend that you choose a suitable backup and recovery strategy for your environment. This strategy should be tested and documented to reduce errors during the recovery process.

7.2.1 Setting up the backup media

Before performing database backups, you have to set up your backup media. When local tape drives are used, the tape drivers must be connected to the DB2 UDB server, and the tape driver should be installed. Third-party vendors products, for example, EMC Legato or Veritas NetBackup, must also be connected to the DB2 UDB server, and the necessary software must be installed according to the instructions of the third-party vendor.

DB2 UDB has a built in TSM interface; to use this, you only have to connect the TSM server to the DB2 UDB server. After installing and configuring the TSM client software, you have to configure the SAP/DB2 system to use TSM by defining the following variables for the <*sapsid*>adm and the db2<*dbsid*> users:

DSM_DIR	Installation directory TSM user interface
DSM_CONFIG	Path to the TSM option file including filename
DSM_LOG	Log directory
DSMI_DIR	Installation directory of the TSM APIs
DSMI_CONFIG	Path to the TSM option file including filename
DSMI_LOG	Log directory

More details can be found in the SAP Notes listed in Table 7-7.

Table 7-7 SAP Notes regarding TSM and DB2 UDB

SAP Note	Title
82029	DB6: TSM installation in DB2 environment
844545	DB6 on AIX: Variables DSMI_* redefined in DB2 UDB
690471	DB6: WEB sites containing related information

7.2.2 Performing backups

You can start or schedule backups within your SAP system. You can also use the command line to start backup jobs. Scripts containing the backup command could also be scheduled on the OS level.

Figure 7-7 shows the architecture of the DB2 UDB backup utility which reads the content of the DB2 UDB database containers and the DB2 UDB configuration files. The backup utility writes the content of them to a backup image and makes an update to the *recovery history file*.

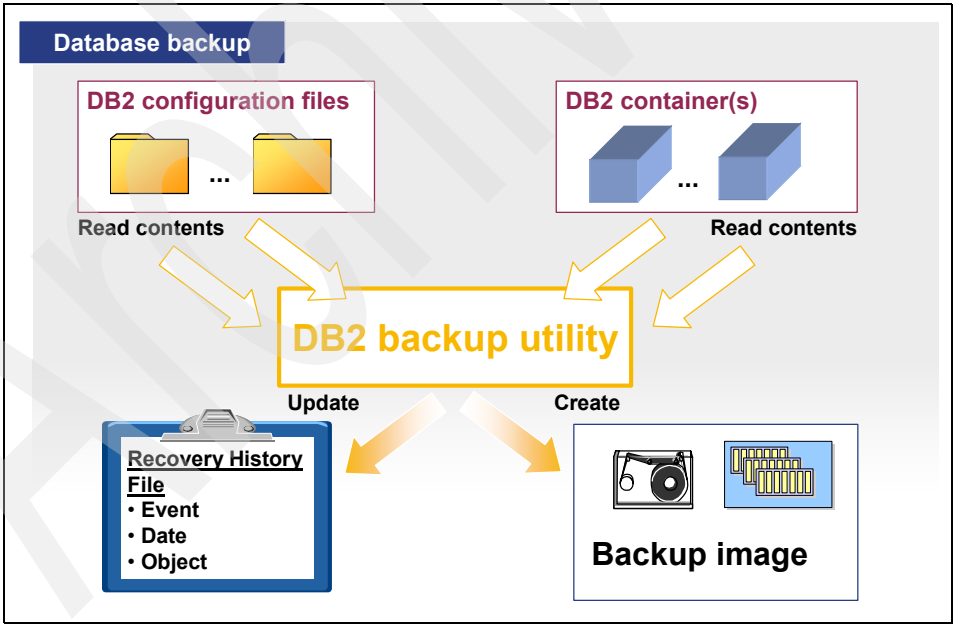


Figure 7-7 DB2 UDB backup architecture

More information about the recovery history file can be found in 7.2.3, “Recovering the database”.

SAP/DB2 database system backups can be started using the SAPGUI or by using the command line. Within the SAP system, you can only start certain types of backups with limited options. To take advantage of all options of the DB2 UDB backup utility, we recommend using the command line. Both methods are discussed in this chapter.

Performing backups using the SAPGUI

Within your SAP system, you can use transaction `DB13` (DBA Planning Calendar) to start or schedule backups. Within the DBA Planning Calendar, you can drag and drop an action from the action pad in the upper right corner to the calendar at the bottom. Figure 7-8 shows the DBA Planning Calendar. In the action pad you see the three types of backup tasks that can be started or scheduled:

- *Full Database Backup into TSM:*

  With this action you can perform a full database backup into TSM.

- *Full Database Backup to Device:*

  With this action you can perform a full database backup to a device, which could be a tape drive or a disk.

- *Full Database Backup with Vendor Library:*

  With this action you can perform a full database backup with a vendor library, for example, EMC Legato Networker Module for IBM DB2 or Veritas NetBackup for DB2.

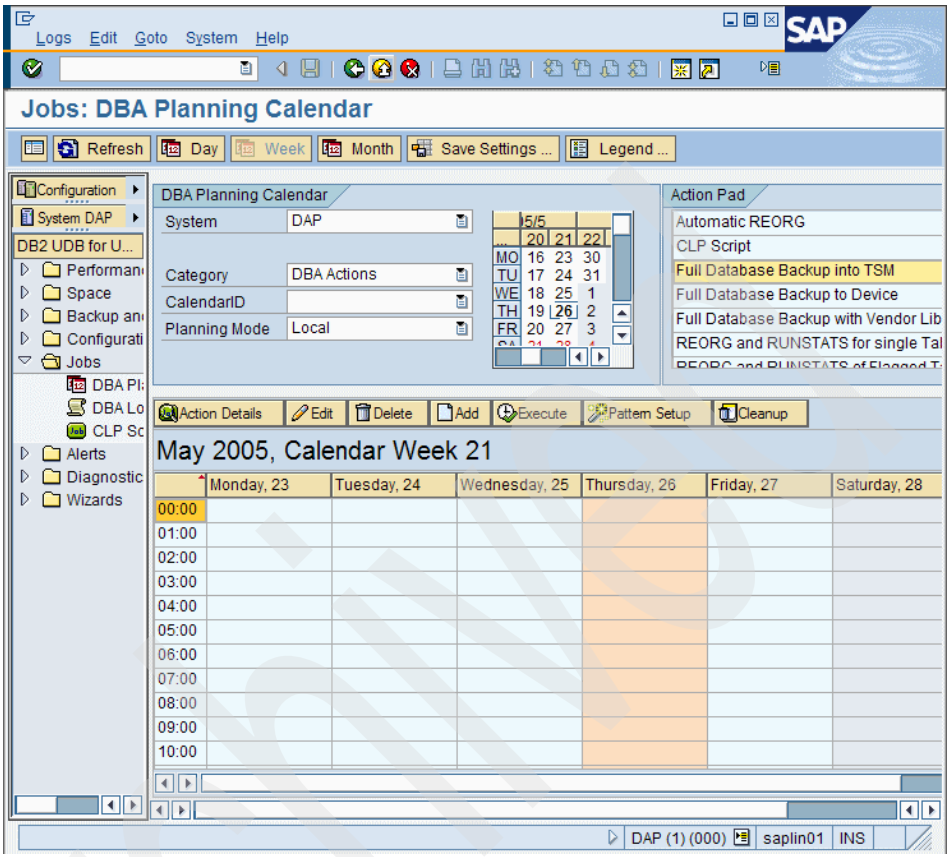


Figure 7-8 DBA Planning Calendar

When you drag and drop one of the three backup tasks from the action pad to the calendar, another window opens giving you the possibility to choose the type of backup that you want to perform and also to change parameters for the backup. Figure 7-9 shows the window that opens when you choose the *Full Database Backup into TSM*, the other two are similar.

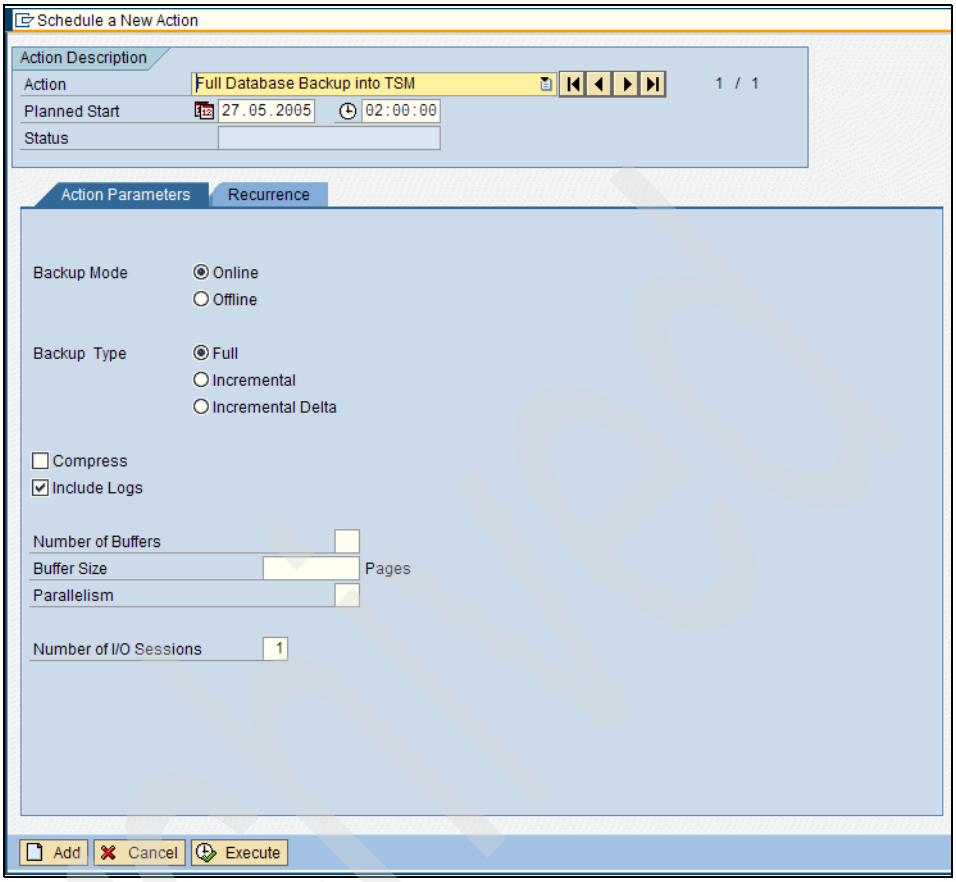


Figure 7-9 Schedule a Full Database Backup into TSM

On the *Action Parameters* tab, you can specify the options of the backup. On the *Recurrence* tab you can schedule a backup job to run on a regular basis. On the *Action Parameters* tab, you have to decide which backup mode to use:

- Offline:

  To perform an offline backup, the application must not be connected to the database. Also, during the course of the backup, DB2 UDB does not permit any new connection to the database. Therefore, an offline backup is always consistent because no active transactions are running on the database. Within an SAP environment, the SAP system may remain up and running during an offline backup to preserve the buffers of the SAP application servers. All SAP work processes are disconnected from the database and remain in a reconnect state during the offline backup. After the backup has been finished, the work processes reconnect to the database using the reconnect feature of the SAP Database Support Layer (DBSL).

- Online:

  An online backup runs in parallel to active transactions and ongoing activities on the database, because of which the backup image cannot be consistent. Therefore it is required to provide all database log files that have been written during an online backup to obtain a consistent state of the database in the case of a restore. Online backups can only be performed with a database running in log retention mode.

Furthermore, you have to decide which backup type to use:

- Full:

  This type of backup creates a backup image of the whole database, that means the contents of all table spaces and the DB2 UDB configuration files.

- Incremental:

  This type of backup creates a backup image only of those pages being changed since the last full backup.

- Incremental Delta:

  This type of backup creates a backup image only of those pages being changed since the last backup, regardless of what type the last backup is.

Tip: Incremental and incremental delta backups offer the following benefits:

- Small backup images because only modified pages will be included
- Minimum of I/O overhead for backup operations
- Recovery is possible with less rollforward time

To perform incremental and incremental delta backups, parameter `TRACKMOD` in the database configuration must be set to `ON`.

These are the other options you can change:

- Compress:

  Specifies that the backup is to be compressed to save space on the backup media.

- Include Logs:

  This option can only be used together with an online backup. It specifies that all database log files that are necessary to obtain a consistent state of the database in the case of a restore will be stored together within the backup image. This is called *consistent online backup*.

- Number of Buffers:

  Specifies the number of buffers to be used for backup.

- Buffer Size:

  Specifies the size of each backup buffer.

- Parallelism:

  Specifies the number of processes transferring data from the table spaces to the backup buffers.

- Number of I/O Sessions:

  Specifies the number of I/O sessions to be established between DB2 UDB and TSM or another backup vendor product.

Figure 7-10 shows the DB2 UDB backup model. Depending on the number of devices (tape drives, file systems, etc.) or the number of I/O sessions you specify, DB2 UDB starts media controller processes (`db2med`). The value for *Parallelism* specifies the number of buffer manipulator processes (`db2bm`). *Number of Buffers* and *Buffersize* specify the number and size of the backup buffers. With help of the prefetcher processes (`db2pfchr`), the buffer manipulators read from the table spaces and put the data to the backup buffers. The media controllers then transfer data from backup buffers to the backup media.

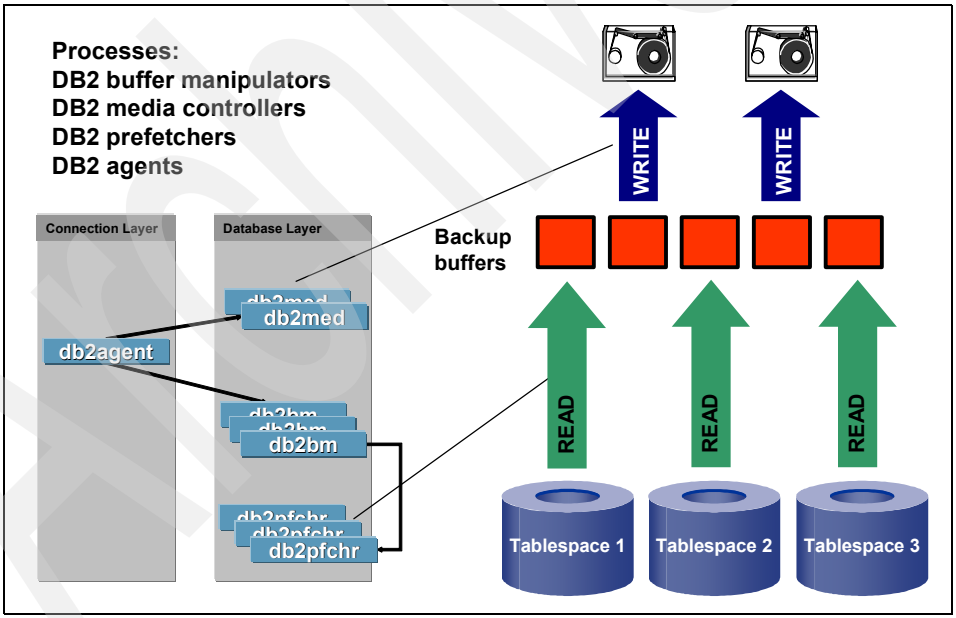


Figure 7-10 DB2 UDB backup model

The DB2 UDB V8.2 backup and restore utility is self-tuning. When not specified, DB2 UDB will automatically choose the number of buffers, the buffer size, and the parallelism, depending on the number of processors, the database configuration (`DB CFG`), and the configured utility heap memory (the `DB CFG` parameter `UTIL_HEAP_SZ`). If you want to configure the number of buffers, the buffer size, and the parallelism manually, you should keep the following considerations in mind:

- One buffer manipulator will work on a table space at a time. Because of this, you should not configure the number of parallelism higher than the number of table spaces in your SAP/DB2 database.
- The number of buffers should be high enough to keep the output devices busy. A rule of thumb is:

  ```
  #backup buffers = #output devices + parallelism + 2
  ```

- The backup buffer size should be a multiple of the largest extent size, for example, 1024 or 2048 4K pages.
- In parallel to an online backup, SAP transactions also do prefetching on the database. Because of this, you should configure enough prefetchers (`DB CFG` parameter `NUM_IOSERVERS`) when performing online backups. A rule of thumb is:

  ```
  #prefetchers = #physical disks + parallelism + 2
  ```

  Also, `#prefetchers` should not be set to more than 20.

To check whether planned backups have been finished successfully or not, you can use transaction `DB13` (DBA Planning Calender) and double-click the action in the calender for which you would like to check the status. A window like the one shown in Figure 7-11 opens. With the navigate buttons  in the upper right corner, you can select for which action you would like to check the log if more than one action has been planned for that time. On the tab *Job Log,* you see the log of the job. In this example, you see that the backup into TSM failed because of problems loading a shared library for TSM.

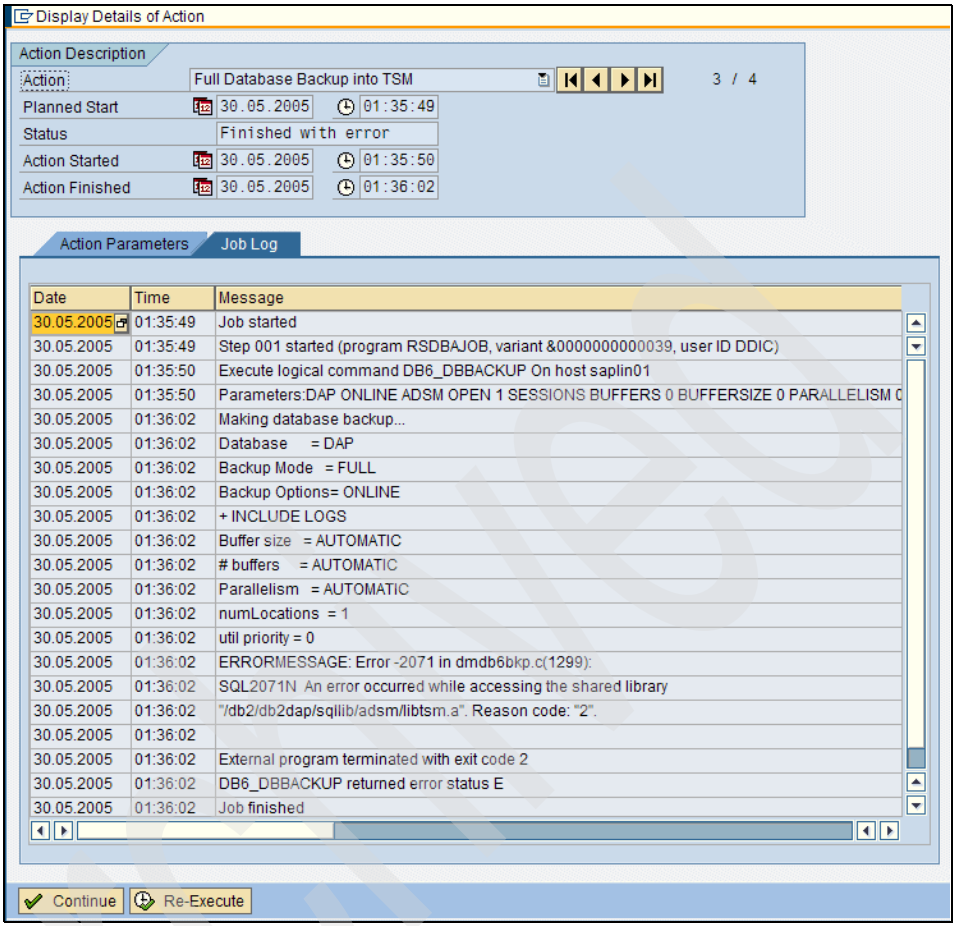


Figure 7-11 DBA Planning Calendar - Job Log

Another way in the SAP system to check the backups is the transaction DB12 (Backup and Recovery - Overview), as shown in Figure 7-12. Within this transaction you see a list with all backups. Successful backups are marked green, all others are marked red. By double-clicking a backup, the details about this backup will be shown on the right side.

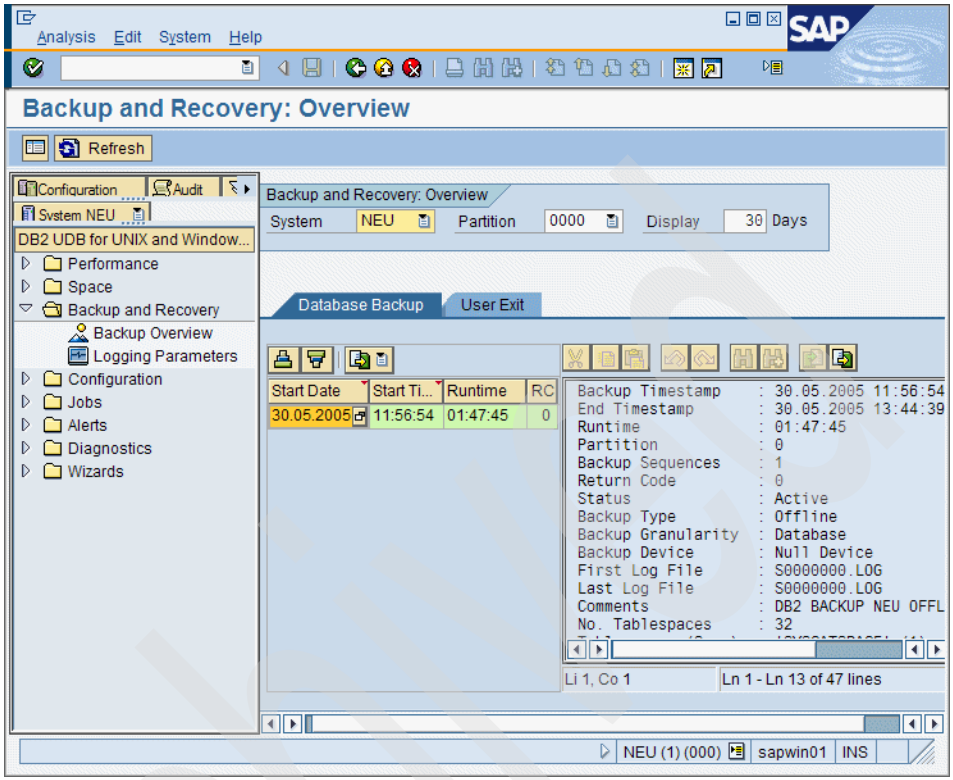


Figure 7-12 Backup and Recovery - Overview

Performing backups using the command line

As it is possible to start the basic backup functions within the SAP system, all options of the DB2 UDB backup utility can only be exploited using the command line.

Because it is not possible to provide examples for all options, we provide the syntax of the DB2 UDB BACKUP command and show some examples.

Example 7-1 shows how you can get the complete syntax of the **BACKUP command** by using the question mark ? (help) function of the DB2 UDB command line processor. This help function works for all DB2 UDB commands that must be called within the command line processor.

Example 7-1 DB2 UDB command line processor help for BACKUP command

```
> db2 "? BACKUP"
BACKUP DATABASE database-alias [USER username [USING password]]
[TABLESPACE (tblspace-name [ {,tblspace-name} ... ])] [ONLINE]
```

```
[INCREMENTAL [DELTA]] [USE {TSM | XBSA} [OPEN num-sess SESSIONS]
[OPTIONS {options-string | options-filename}] | TO dir/dev
[ {,dir/dev} ... ] | LOAD lib-name [OPEN num-sess SESSIONS]
[OPTIONS {options-string | options-filename}]]
[WITH num-buff BUFFERS] [BUFFER buffer-size] [PARALLELISM n]
[COMPRESS [COMPRLIB lib-name [EXCLUDE]] [COMPROPTS options-string]]
[UTIL_IMPACT_PRIORITY [priority]] [{INCLUDE | EXCLUDE} LOGS] [WITHOUT
PROMPTING]
```

Some options, such as the number of buffers, buffer size, parallelism, and compress have been discussed in the preceding section — performing the backup using the SAPGUI; The `UTIL_IMPACT_PRIORITY` will be discussed in section 7.2.6, “Throttling the backup utility”.

The following examples may give you some ideas about what can be done with the DB2 UDB backup utility.

Full database consistent online backup into TSM with compression

Example 7-2 shows how to invoke a consistent online backup of the whole database into TSM and use DB2 UDB’s built-in compression to save space on the backup media. With the `INCLUDE LOGS` option, at the end of the online backup, all logs needed to restore and roll forward this backup image to a consistent point in time will be added to the backup image.

Example 7-2 Full database consistent online backup to TSM with compression

```
> db2 "backup db gr4 online use tsm compress include logs"

Backup successful. The timestamp for this backup image is : 20050531121221
```

Full database backup to two devices in parallel + manual tuning

Example 7-3 shows a manual tuned backup that writes to two devices in parallel. As you can see in the **`ps -ef`** excerpts that have been done *during* the backup, DB2 UDB starts one media controller (`db2med`) process per device that you specify for the backup. Devices can be tape drives or file systems. According to the specified parallelism, DB2 UDB starts the same number of buffer manipulator (`db2bm`) processes. This is also shown in Figure 7-10 on page 344.

Example 7-3 Full database backup to two devices in parallel + manual tuning

```
> db2 backup db gr4 to /dev/rmt0, /dev/rmt1 with 8 buffers buffer 1024
parallelism 4

Backup successful. The timestamp for this backup image is : 20050531144353
```

```
> ps -ef | grep db2bm | grep -v grep
  db2gr4  438360 1040436   4 14:38:48      -  0:02 db2bm.786568.0 0
  db2gr4  692320 1040436   6 14:38:48      -  0:02 db2bm.786568.3 0
  db2gr4  766094 1040436   4 14:38:48      -  0:03 db2bm.786568.1 0
  db2gr4  843940 1040436   3 14:38:48      -  0:03 db2bm.786568.2 0

> ps -ef | grep db2med | grep -v grep
  db2gr4  909522  802954   0 14:43:53      -  0:00 db2med.786568.1 0
  db2gr4 1646644  802954   0 14:43:53      -  0:00 db2med.786568.0 0
```

Full database online incremental backup into TSM

Example 7-4 shows how to invoke an online incremental backup of the whole database into TSM. With this backup, only changed pages since the last full backup (no matter whether it was online or offline) will be written to the backup image. To be able to do incremental backups, the TRACKMOD database configuration parameter must be set to ON.

Example 7-4 Full database online incremental backup into TSM

```
> db2 backup db gr4 online incremental use tsm

Backup successful. The timestamp for this backup image is : 20050531151348
```

Online table space backup of two table spaces into TSM

Example 7-5 shows how to invoke an online backup for two table spaces.

Example 7-5 Online table space backup of two table spaces into TSM

```
> db2 "backup db gr4 tablespace (PSAPCLUD, PSAPDOCUD) online use tsm"

Backup successful. The timestamp for this backup image is : 20050531152916
```

Full database offline backup of a partitioned database into TSM

For a partitioned database, it is necessary to create backups for all partitions. The catalog partition must be backed up before other partitions can be backed up. This can be done using **db2_all**, as shown in Example 7-6.
To determine the catalog partition, use the command:

```
LIST DATABASE PARTITION GROUPS SHOW DETAIL
```

When you perform an online backup, you can backup all partitions in parallel using command

db2_all "db2 backup db <*dbsid*> **online ..."**.

Example 7-6 Full database offline backup of a partitioned database into TSM

```
> db2 list database partition groups show detail

DATABASE PARTITION GROUP     PMAP_ID DATABASE PARTITION NUMBER    IN_USE
---------------------------- ------- ---------------------------- ------
IBMCATGROUP                        0                            0 Y
IBMDEFAULTGROUP                    1                            0 Y
IBMDEFAULTGROUP                    1                            1 Y
IBMDEFAULTGROUP                    1                            2 Y
IBMDEFAULTGROUP                    1                            3 Y
NGRP_DIM_LXD                       3                            0 Y
NGRP_FACT_LXD                      4                            0 Y
NGRP_FACT_LXD                      4                            1 Y
NGRP_FACT_LXD                      4                            2 Y
NGRP_FACT_LXD                      4                            3 Y
NGRP_ODS_LXD                       5                            0 Y
NGRP_ODS_LXD                       5                            1 Y
NGRP_ODS_LXD                       5                            2 Y
NGRP_ODS_LXD                       5                            3 Y
SAPNODEGRP_LXD                     6                            0 Y

  15 record(s) selected.

> db2_all "<<+0< db2 backup database lxd to /backup/LXD"

Backup successful. The timestamp for this backup image is : 20050531111344

laubach: db2 backup database lxd to /backup/LXD completed ok

> db2_all "|<<-0< db2 backup database lxd to /backup/LXD"

rah: omitting logical node 0
rah: primary monitoring process for db2 is 9400

laubach:
laubach: Backup successful. The timestamp for this backup image is :
20050531115327
laubach:
laubach: db2 backup database lxd to /backup/LXD completed ok

laubach:
laubach: Backup successful. The timestamp for this backup image is :
20050531115328
laubach:
laubach: db2 backup database lxd to /backup/LXD completed ok

laubach:
```

```
laubach: Backup successful. The timestamp for this backup image is :
20050531115328
laubach:
laubach: db2 backup database lxd to /backup/LXD completed ok
```

7.2.3 Recovering the database

There are two different recovery techniques DB2 UDB uses: *Rollforward recovery* and *restart recovery*, as shown in Figure 7-13. Both techniques are discussed in this section.

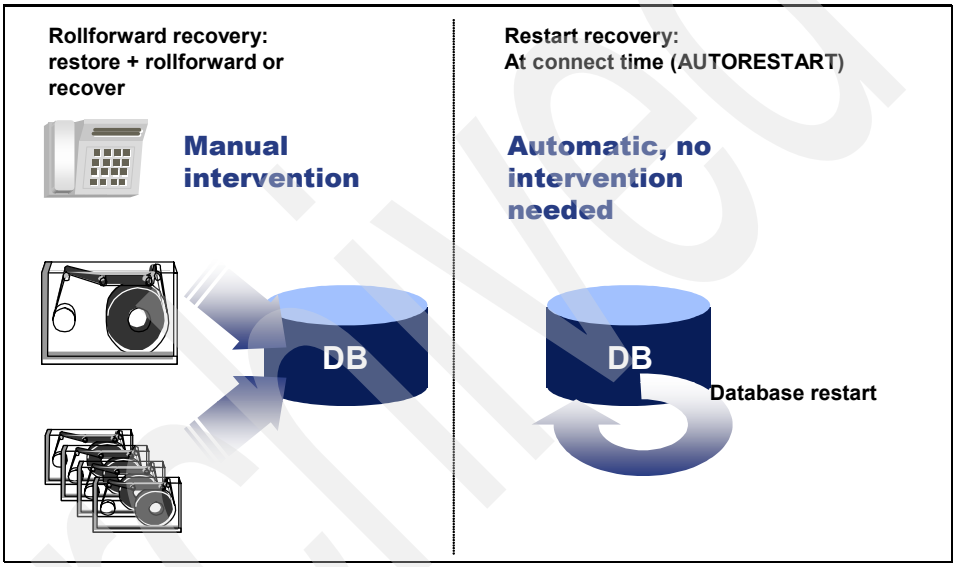


Figure 7-13 Recovery techniques

Restart recovery

Whenever the database system has not been shut down properly (for example, by a system hardware failure or power loss), DB2 UDB performs a restart recovery when it is restarted. DB2 UDB then reruns all transactions that were performed before crash and not recorded in the database files but only in the transaction log files. The DB2 UDB *log file header file* (normally located in `/db2/<DBSID>/db2<sid>/NODE0000/SQL00001/SQLOGCTL.LFH`) contains that log sequence number that DB2 UDB must use as starting point for this type of recovery. If you ever lose this file, restart recovery is not possible after an improper shut down of the database, and rollforward recovery must be invoked.

For an automatic start of the restart recovery after restarting the database, the `AUTORESTART` database configuration parameter must be set to `ON`.

Figure 7-14 shows how DB2 UDB reapplies all transactions to the database that have not been recorded in the database files but in the transaction log files. At the end of the restart recovery, all uncommitted transactions will be rolled back. The maximum amount of log space that should be read during crash recovery can be adjusted with the SOFTMAX database configuration parameter . With a small SOFTMAX, restart takes less time but there are more frequent checkpoints during normal operation. A checkpoint is a sync point for the database because all the information in the buffer pools is flushed to the database files. With a larger SOFTMAX, restart potentially has to process more log records because of fewer checkpoints during normal operation.

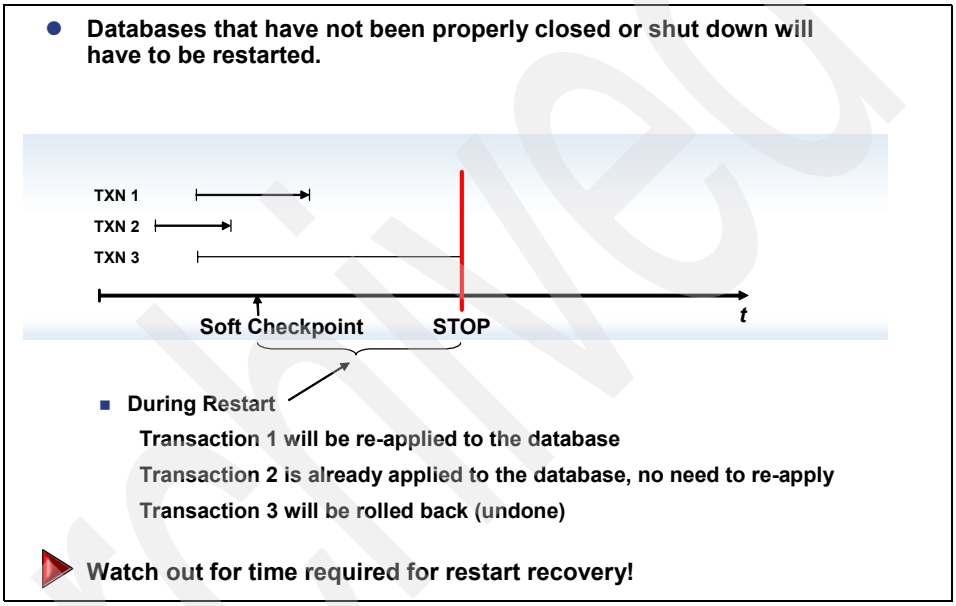


Figure 7-14 Reapplying transactions during restart recovery

The restart recovery can be seen in the db2diag.log file, which can be found in the directory specified with the DIAGPATH database manager configuration parameter; normally this is /db2/<*DBSID*>/db2dump. This is shown in Example 7-7. First DB2 UDB detects that the database has not been shut down properly (Crash Recovery is needed.), after that it invokes the redo or forward phase of the restart recovery (Crash Recovery started.). Within the redo phase DB2 UDB reapplies all changes to the database files, after that in the undo or backward phase (Forward phase of crash recovery has been completed.) it removes all uncommitted changes from the database files.

Example 7-7 db2diag.log excerpt during restart recovery

```
...
2005-06-01-15.30.35.519271+120 I2629A360          LEVEL: Warning
PID     : 725098               TID  : 1           PROC : db2agent (GR2) 0
INSTANCE: db2gr2               NODE : 000         DB   : GR2
APPHDL  : 0-8                  APPID: *LOCAL.db2gr2.050601133034
FUNCTION: DB2 UDB, base sys utilities, sqledint, probe:30
MESSAGE : Crash Recovery is needed.

2005-06-01-15.30.42.615486+120 I2990A368          LEVEL: Warning
PID     : 725098               TID  : 1           PROC : db2agent (GR2) 0
INSTANCE: db2gr2               NODE : 000         DB   : GR2
APPHDL  : 0-8                  APPID: *LOCAL.db2gr2.050601133034
FUNCTION: DB2 UDB, relation data serv, sqlrr_db_init, probe:100
MESSAGE : DB2_IMPLICIT_UNICODE enabled

2005-06-01-15.30.42.616452+120 I3359A423          LEVEL: Warning
PID     : 725098               TID  : 1           PROC : db2agent (GR2) 0
INSTANCE: db2gr2               NODE : 000         DB   : GR2
APPHDL  : 0-8                  APPID: *LOCAL.db2gr2.050601133034
FUNCTION: DB2 UDB, recovery manager, sqlpresr, probe:410
MESSAGE : Crash recovery started. LowtranLSN 00000002697F9A99 MinbuffLSN
          00000002697F9A53

2005-06-01-15.30.42.629808+120 I3783A405          LEVEL: Warning
PID     : 725098               TID  : 1           PROC : db2agent (GR2) 0
INSTANCE: db2gr2               NODE : 000         DB   : GR2
APPHDL  : 0-8                  APPID: *LOCAL.db2gr2.050601133034
FUNCTION: DB2 UDB, recovery manager, sqlprecm, probe:2000
MESSAGE : Using parallel recovery with 3 agents 36 QSets 108 queues and 64
chunk
s

2005-06-01-15.30.42.993732+120 I4189A433          LEVEL: Warning
PID     : 725098               TID  : 1           PROC : db2agent (GR2) 0
INSTANCE: db2gr2               NODE : 000         DB   : GR2
APPHDL  : 0-8                  APPID: *LOCAL.db2gr2.050601133034
FUNCTION: DB2 UDB, recovery manager, sqlprecm, probe:4000
MESSAGE : DIA2051W Forward phase of crash recovery has completed.  Next LSN is
          "00000002697F9A99".

2005-06-01-15.30.43.011321+120 I4623A388          LEVEL: Warning
PID     : 725098               TID  : 1           PROC : db2agent (GR2) 0
INSTANCE: db2gr2               NODE : 000         DB   : GR2
APPHDL  : 0-8                  APPID: *LOCAL.db2gr2.050601133034
FUNCTION: DB2 UDB, recovery manager, sqlpresr, probe:3170
MESSAGE : Crash recovery completed. Next LSN is 00000002697F9A99
...
```

You can also watch the progress of the restart recovery process by using the **LIST UTILITIES SHOW DETAIL** command, as shown in Example 7-8.

Example 7-8 Restart recovery progress

```
> db2 list utilities show detail

ID                                   = 1
Type                                 = CRASH RECOVERY
Database Name                        = GR2
Partition Number                     = 0
Description                          = Crash Recovery
Start Time                           = 06/01/2005 15:30:42.630425
Progress Monitoring:
   Estimated Percentage Complete = 0
   Phase Number [Current]        = 1
      Description                = Forward
      Total Work                 = 18446743821935274316 bytes
      Completed Work             = 42 bytes
      Start Time                 = 06/01/2005 15:30:42.630436
   Phase Number                  = 2
      Description                = Backward
      Total Work                 = 18446743821935274316 bytes
      Completed Work             = 0 bytes
      Start Time                 = Not Started
```

Rollforward recovery

Within a rollforward recovery, an online or offline backup image and all the transaction log files created during and after the backup are used to recover the database to a consistent state. When developing a backup strategy, you should consider how much time you can spend for recovery. A more current backup allows you to perform a faster recovery because fewer transactions log files needs to be reapplied to the database. Figure 7-15 shows the rollforward recovery process using the legacy log file management and indirect archiving.

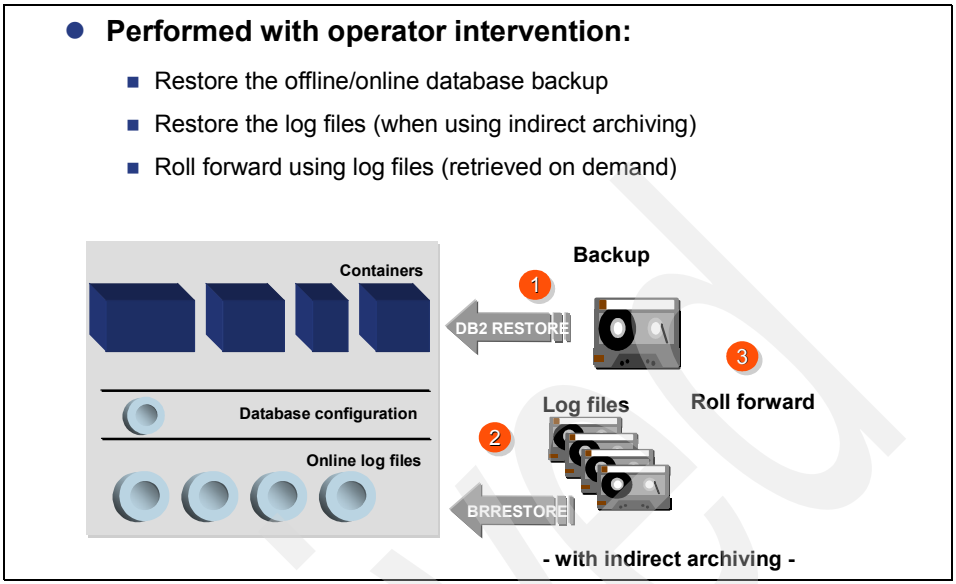


Figure 7-15 Rollforward recovery process overview

Figure 7-16 illustrates the rollforward recovery process using the legacy log file management, which consists of the following steps. We discuss these steps in detail, with examples given in the following sections.

1. Determine the timestamp of the (most recent) backup.

   First you have to find out the timestamp of the backup that you want to use for your restore. For this you can use the command **LIST HISTORY BACKUP ALL FOR** <db*sid*>, which shows the content of the recovery history file regarding backups. If you performed table space backups instead of full backups, you have to search for multiple backup entries in the recovery history file.

2. Restore the database.

   Use the **RESTORE** command to restore the database.

3. Retrieve the necessary transaction log files.

   You have to retrieve all the necessary transaction log files for rollforward. For online backups you have to retrieve at least those transaction log files being written during the backup to bring the database to a consistent state.

4. Reapply the transactions to the database using the transaction log files.

   With the **ROLLFORWARD** command, you re-apply all the transactions to the database using the transaction log files. The rollforward can be done in more than one steps, for example, when you're not able to retrieve all necessary transaction log files at once because of less free space on your disk.

In this case, retrieve as much transaction log files as you can, apply their transactions using the **ROLLFORWARD command**, and then retrieve the next ones.

5. Make the database consistent.

   At the end of the rollforward, there are some open transactions within the database still waiting for a commit. With the **ROLLFORWARD ... STOP** command you perform a roll back of those open transactions and bring the database to a consistent state.

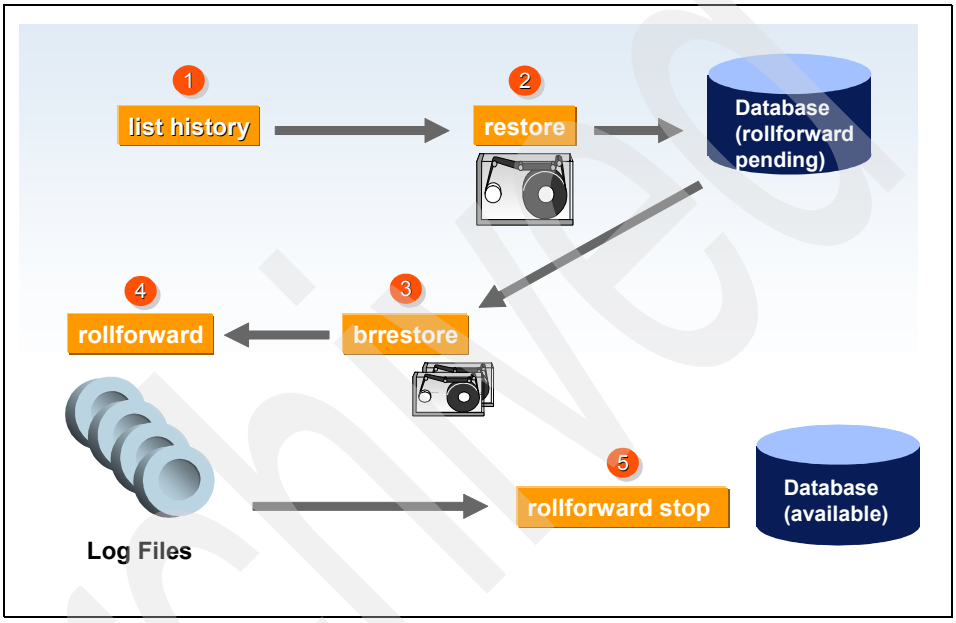


Figure 7-16 Rollforward recovery process

The preceding described steps are valid for the legacy log file management using indirect archiving. When you have implemented the legacy log file management using direct archiving, you should not perform step number 3 (retrieve the necessary transaction log files). During the reapplying of the transactions to the database, the necessary transaction log files will be automatically retrieved when needed.

When you have implemented the new DB2 UDB V8.2 log file management, you can also use the **RECOVER** instead of the **RESTORE** command. The **RECOVER** command combines the steps 2 to 4 in one, because it restores the backup image, does the reapplying of the transactions and also retrieves the necessary transaction log files when needed.

Figure 7-17 shows the DB2 UDB restore model, which is equivalent to the DB2 UDB backup model. As mentioned in the section about the backup, DB2 UDB starts media controller processes (`db2med`) depending on the number of devices (tape drives, file systems, etc.) or the number of IO sessions you specify. The value for parallelism specifies the number of buffer manipulator processes (`db2bm`). The number and size of the restore buffers can also be specified. The media controllers transfer data from the backup media to the restore buffers. With help of the prefetcher processes (`db2pfchr`), the buffer manipulators read from the restore buffers and put the data into the table spaces.

DB2 UDB V8.2.2 backup and restore utilities are self-tuning. When not specified, DB2 UDB will automatically choose the number of buffers, the buffer size, and the parallelism depending on the number of processors, the database configuration (DB CFG) and the configured utility heap memory (DB CFG parameter `UTIL_HEAP_SZ`). We recommend using the self-tuning feature.

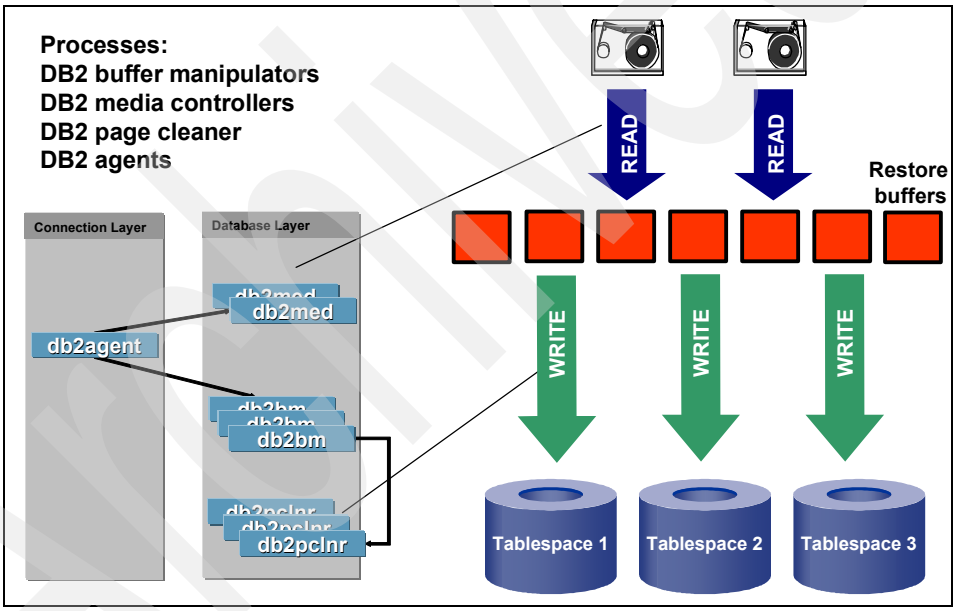


Figure 7-17 DB2 UDB restore model

The recovery history file

The recovery history file, which contains information about actions like backup, restore, rollforward, and `REORG` can be queried with the **`LIST HISTORY`** command. The complete syntax can be queried with the help functionality of the DB2 UDB command line processor, as shown in Example 7-9.

Example 7-9 Syntax of the LIST HISTORY command

```
> db2 "? LIST HISTORY"
LIST HISTORY {BACKUP | ROLLFORWARD | REORG |
CREATE TABLESPACE | ALTER TABLESPACE | DROPPED TABLE | LOAD |
RENAME TABLESPACE | ARCHIVE LOG}
{ALL | SINCE timestamp |CONTAINING {schema.object_name | object_name}}
FOR [DATABASE] database-alias
```

To get information about usable backups, use the **LIST HISTORY BACKUP ALL FOR** <*dbsid*>**.** Use **LIST HISTORY BACKUP SINCE *timestamp* FOR <*dbsid*>** to restrict the list to a specific timeframe, as shown in Example 7-10.

Example 7-10 Example for LIST BACKUP SINCE

```
> db2 list backup since 200505270900 for gr2

                    List History File for gr2

Number of matching file entries = 1


 Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log  Backup ID
 -- --- ------------------ ---- --- ------------ ------------ --------------
  B  D  20050527104331001   F    A  S0000044.LOG S0000044.LOG
 ----------------------------------------------------------------------------
  Contains 31 tablespace(s):

  00001 SYSCATSPACE
  00002 ZSAPBTAB2D
...
  00031 SYSTOOLSPACE
 ----------------------------------------------------------------------------
    Comment: DB2 BACKUP GR2 OFFLINE
 Start Time: 20050527104331
   End Time: 20050527124146
     Status: A
 ----------------------------------------------------------------------------
  EID: 1560 Location: adsm/libtsm.a
```

Table 7-8 shows the different symbols for the backup types used in the recovery history file in the column Type. An explanation of all symbols used in the output of the **LIST HISTORY** command can be found in the DB2 UDB documentation, *Command Reference V8.2*, SC09-4828-01.

Table 7-8 Symbols used for the backup types in the recovery history file

Symbol	Explanation
F	Offline
N	Online
I	Incremental offline
O	Incremental online
D	Delta offline
E	Delta online

Within each backup entry, you find all necessary information, the timestamp, the backup type, the names of the table spaces the backup images contains and also the names of the transaction log files belonging to this backup image. In the case of a full offline backup, the earliest log is the first transaction log file which you need when you want to recovery the database using this backup image. For an full online backup, the transaction logs covering the period between the earliest log and the current log is the minimum set of logs required to bring the database to a consistent state.

If you lose the recovery history file, you'll have to restore the latest version of the recovery history file to determine the necessary information out of that file.

To restore the recovery history file from the latest backup image, you also use the **RESTORE DATABASE** *<dbsid>* **HISTORY FILE** command, as shown in Example 7-11.

Example 7-11 Restore of a recovery history file from TSM

```
> db2 restore db gr2 history file use tsm
DB20000I  The RESTORE DATABASE command completed successfully.
```

More options of the **RESTORE** command can be queried with the help functionality of the DB2 UDB command line processor using **? RESTORE** or can be found in the DB2 UDB documentation, *Command Reference V8.2*, SC09-4828-01.

Rollforward recovery using restore and rollforward commands

Because it is impossible to discuss all scenarios in this chapter, we provide you here two common scenarios using DB2 UDB's **RESTORE** and **ROLLFORWARD** commands. How to retrieve the necessary transaction log files for a recovery depends on the log file management used.

Rollforward recovery scenario 1

In the first scenario we want to recover our database to a specified point in time where one of the users in the SAP/DB2 system accidentally deleted data in the system after Friday, June 3 17:15:00 DFT 2005. To minimize the recovery time, we use the latest full online and the latest subsequent incremental and delta backups as shown in Figure 7-18.

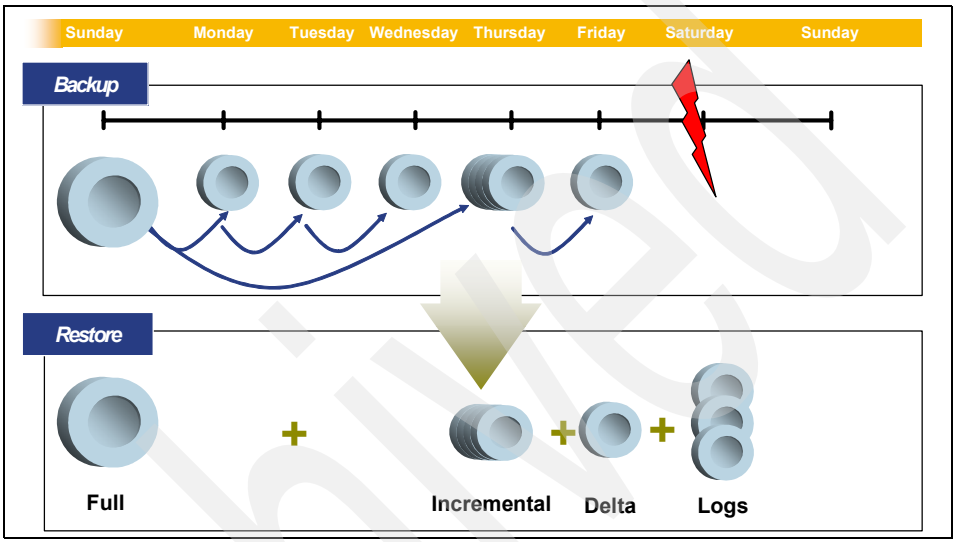


Figure 7-18 Incremental restore

The timestamps of all necessary backups images can be manually retrieved from the recovery history file by **LIST HISTORY BACKUP** or by using the **db2ckrst** command, which generates a list of timestamps for the backup images needed for the incremental restore. Example 7-12 shows how we determine the timestamp of the incremental target image and the first needed transaction log file from the recovery history file, afterwards we check the list of timestamps of the backup images needed.

Example 7-12 Example of db2ckrst

```
> db2 list backup since 20050603 for gr2
...
 Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log  Backup ID
 -- --- ------------------ ---- --- ------------ ------------ --------------
  B  D  20050603171056001   E    A  S0000070.LOG S0000071.LOG
 ------------------------------------------------------------------------------
  Contains 31 tablespace(s):

  00001 SYSCATSPACE
...
```

```
  00031 SYSTOOLSPACE
 ------------------------------------------------------------------------
    Comment: DB2 BACKUP GR2 ONLINE
 Start Time: 20050603171056
   End Time: 20050603171354
     Status: A
 ------------------------------------------------------------------------
  EID: 1616 Location: adsm/libtsm.a

> db2ckrst -d gr2 -t 20050603171056

Suggested restore order of images using timestamp 20050603171056 for
database gr2.
======================================================================
 restore db gr2 incremental taken at 20050603171056
 restore db gr2 incremental taken at 20050603164434
 restore db gr2 incremental taken at 20050603170308
 restore db gr2 incremental taken at 20050603171056
======================================================================
```

As you can easily see, the db2ckrst command suggests to restore the incremental backup image twice, in the beginning and in the end of the incremental restore. In the beginning of an incremental restore, this is necessary to ensure that the database is initially configured with the correct history, database configuration, and table space definitions for the database that will be created during the restore operation. In cases where a table space has been dropped since the initial full database backup image was taken, the table space data for that table space will be read from the backup images but ignored during incremental restore processing.

The restore can be done using the generated, simplified restore commands or with the **RESTORE DATABASE** <*dbsid*> **INCREMENTAL AUTOMATIC** command as shown in Example 7-13. The database backup image specified for the INCREMENTAL AUTOMATIC option is the incremental image of the date you want to restore. The warning that you have to confirm before the restore starts can be avoided by specifying the **WITHOUT PROMPTING** clause at the end of the **RESTORE** command.

Example 7-13 RESTORE DATABASE ... INCREMENTAL AUTOMATIC

```
> db2 restore database gr2 incremental automatic use tsm taken at
20050603171056
SQL2539W  Warning!  Restoring to an existing database that is the same as the
backup image database.  The database files will be deleted.
Do you want to continue ? (y/n) y
DB20000I  The RESTORE DATABASE command completed successfully.
```

Now we have restored all necessary backup images. The next step is to bring the database to the point in time state using the **ROLLFORWARD** command. In this example, we have indirect archiving and the logs have been restored to the directory /large_fs. This allows us to demonstrate more options of the **ROLLFORWARD** command. All options of the ROLLFORWARD command can be queried with the help functionality of the DB2 UDB command line processor using ? ROLLFORWARD or can be found in the DB2 UDB documentation, *Command Reference V8.2*, SC09-4828-01.

Example 7-14 shows first how to check the current rollforward state of the database. As you can see, the DB pending state tells you that the whole database is in the rollforward state and that the next needed transaction log file is the S0000070.LOG, as we already determined in the beginning from the recovery history file. The USING LOCAL TIME clause tells the **ROLLFORWARD** command to interpret all timestamps as local time of the DB2 UDB client where the **ROLLFORWARD** command has been invoked from.

We now want to recover our database to Friday, June 3 17:15:00 DFT 2005 using the transaction log files that we already restored to the directory /large_fs. We specify this directory with the OVERFLOW LOG PATH (...) option.

In the last step we must bring the database to a consistent state and roll back all uncommitted transactions using the STOP clause of the **ROLLFORWARD** command. Since the USING LOCAL TIME clause cannot be used with the STOP clause, the timestamps returned by DB2 UDB is the coordinated universal time (UTC). Once the OVERFLOW LOG PATH (...) option is used during the rollforward, it is necessary to use the same option with the STOP clause so that DB2 UDB can access the transaction log files again for the rollback of the uncommitted transactions.

Example 7-14 Recovery of a database

```
> db2 "rollforward db gr2 query status using local time"

                                Rollforward Status

 Input database alias                   = gr2
 Number of nodes have returned status   = 1

 Node number                            = 0
 Rollforward status                     = DB  pending
 Next log file to be read               = S0000070.LOG
 Log files processed                    =  -
 Last committed transaction             = 2005-06-01-15.29.46.000000
```

```
> db2 "rollforward db gr2 to 2005-06-03-17.15.00 using local time overflow log
path (/large_fs)"

                                Rollforward Status

 Input database alias                   = gr2
 Number of nodes have returned status   = 1

 Node number                            = 0
 Rollforward status                     = DB  working
 Next log file to be read               = S0000077.LOG
 Log files processed                    = S0000070.LOG - S0000075.LOG
 Last committed transaction             = 2005-06-03-17.14.59.000000

DB20000I  The ROLLFORWARD command completed successfully.


> db2 "rollforward db gr2 stop overflow log path (/large_fs)"

                                Rollforward Status

 Input database alias                   = gr2
 Number of nodes have returned status   = 1

 Node number                            = 0
 Rollforward status                     = not pending
 Next log file to be read               =
 Log files processed                    = S0000070.LOG - S0000076.LOG
 Last committed transaction             = 2005-06-03-15.14.59.000000

DB20000I  The ROLLFORWARD command completed successfully.
```

Tip: Legacy log file management using indirect archiving is the only case where you have to care about the location of the restored transaction log files. If the log files are restored to the directory where they have been originally located (normally this is /db2/<*DBSID*>/log_archive/<DB*SID*>), the files must have their timestamp and node as suffix, for example, S0000106.LOG.20050605154400.NODE0000. If the log files are restored to a different directory because of free space reasons and will be used in the **OVERFLOW LOG PATH (...)** clause during recovery, they must only have .LOG as suffix, for example, S0000106.LOG.

Rollforward recovery scenario 2

In the second scenario we want to recover our database using a consistent online TSM database backup. We first check if our backup is a consistent online backup. From the recovery history file we have determined 20050605210224 as

the timestamp of the backup and we need the transaction log files S0000077.LOG-S0000092.LOG to bring this backup to a consistent state.

With the **db2adutl** command it is possible to dump the header of a backup image stored on TSM as shown in Example 7-15. From the header we can see if a backup copy is a consistent online backup or not. In this case the `Include Logs -- 1` indicates that the backup image also includes the transaction log files needed for bringing the database to a consistent state. For backup images stored on other devices, use the **db2ckbkp** command to check this information. For more information about **db2adutl** and **db2ckbkp,** refer to the IBM DB2 UDB documentation, *Command Reference V8.2*, SC09-4828-01.

Example 7-15 Dump header of backup image stored on TSM

```
> db2adutl verify headeronly taken at 20050605210224 verbose without prompting

Query for database GR2


Retrieving FULL DATABASE BACKUP information.  Please wait.

   FULL DATABASE BACKUP image:
     ./GR2.0.db2gr2.NODE0000.CATN0000.20050605210224.001, DB Partition Number:
0  Mgmt Class: BACKUP_IBB

   Verifying file: ./GR2.0.db2gr2.NODE0000.CATN0000.20050605210224.001


======================
MEDIA HEADER REACHED:
======================
        Server Database Name           -- GR2
        Server Database Alias          -- GR2
        Client Database Alias          -- GR2
        Timestamp                      -- 20050605210224
        Database Partition Number      -- 0
        Instance                       -- db2gr2
        Sequence Number                -- 1
        Release ID                     -- A00
        Database Seed                  -- 53E25BFE
        DB Comment's Codepage (Volume) -- 0
        DB Comment (Volume)            --
        DB Comment's Codepage (System) -- 0
        DB Comment (System)            -- SAP R3 database GR2
        Authentication Value           -- 255
        Backup Mode                    -- 1
        Includes Logs                  -- 1
        Compression                    -- 0
        Backup Type                    -- 0
```

```
        Backup Gran.                    -- 0
        Status Flags                    -- 30
        System Cats inc                 -- 1
        Catalog Partition Number        -- 0
        DB Codeset                      -- ISO8859-1
        DB Territory                    --
        LogID                           -- 1117996020
        LogPath                         -- /db2/GR2/log_dir/NODE0000/
        Backup Buffer Size              -- 9965568
        Number of Sessions              -- 1
        Platform                        -- 14

 The proper image file name would be:
GR2.0.db2gr2.NODE0000.CATN0000.20050605210224.001


Image header dumped -- NO VERIFICATION PERFORMED.
...
```

With a consistent online backup image, you can specify where the **RESTORE** utility should save the transaction log files included in the backup image using the **LOGTARGET** ***directory*** clause as shown in Example 7-16. Without this clause, only the backup image without the transaction log files will be restored.

Example 7-16 Restore of a consistent online backup including the logs

```
> db2 "restore db gr2 use tsm taken at 20050605210224 logtarget /large_fs"
SQL2539W  Warning!  Restoring to an existing database that is the same as the
backup image database.  The database files will be deleted.
Do you want to continue ? (y/n) y
DB20000I  The RESTORE DATABASE command completed successfully.

> ls /large_fs
S0000077.LOG  S0000081.LOG  S0000085.LOG  S0000089.LOG
S0000078.LOG  S0000082.LOG  S0000086.LOG  S0000090.LOG
S0000079.LOG  S0000083.LOG  S0000087.LOG  S0000091.LOG
S0000080.LOG  S0000084.LOG  S0000088.LOG  S0000092.LOG
```

Now having the database restored and the transaction log files needed in place, we use the **ROLLFORWARD** command to recover the database to a consistent state, as show in Example 7-17. Please note that using the **TO END OF LOGS** clause always stops with an error that the next log file cannot be retrieved. This is because DB2 UDB always tries to apply the next transaction log file including the one not exist yet.

Example 7-17 Recover to end of logs

```
> db2 "rollforward db gr2 to end of logs overflow log path (/large_fs)"
SQL1268N  Roll-forward recovery stopped due to error "30" while retrieving log
file "S0000093.LOG" for database "GR2" on node "0".


> db2 "rollforward db gr2 query status"

                                 Rollforward Status

Input database alias                   = gr2
 Number of nodes have returned status   = 1

Node number                            = 0
Rollforward status                     = DB  working
 Next log file to be read               = S0000093.LOG
Log files processed                    = S0000077.LOG - S0000088.LOG
 Last committed transaction             = 2005-06-05-19.12.24.000000


> db2 "rollforward db gr2 stop overflow log path (/large_fs)"

                                 Rollforward Status

 Input database alias                   = gr2
 Number of nodes have returned status   = 1

 Node number                            = 0
 Rollforward status                     = not pending
 Next log file to be read               =
 Log files processed                    = S0000077.LOG - S0000093.LOG
 Last committed transaction             = 2005-06-05-19.39.19.000000

DB20000I  The ROLLFORWARD command completed successfully.
```

Tip: When restoring a partitioned database, you have to ensure that you first restore the catalog partition and after that all other partitions. Assuming that you have four partitions and partition 0 is the catalog partition, this can be done by:

```
db2_all '<<+0< db2 restore database ....'
db2_all '<<+1< db2 restore database ....'
db2_all '<<+2< db2 restore database ....'
db2_all '<<+3< db2 restore database ....'
```

The log file directory /db2/<*DBSID*>/log_archive/<DB*SID*> should be mounted on all partitions. You should restore transaction log files into this directory and use the **ROLLFORWARD command** with the **ON ALL DBPARTITIONNUMS** to recover your database.

We recommend that you test and document this procedure or use the new log file management of DB2 UDB V8.2 together with the **RECOVER** command for partitioned databases, as discussed in the following section.

Rollforward recovery using RECOVER command

The **RECOVER** command allows you to restore the database and roll forward transaction logs in one step. The command will retrieve the necessary transaction log file automatically. To be able to use this command, the new log file management of DB2 UDB V8.2 must be configured properly and an up-to-date recovery history file must be available. To keep an up-to-date recovery history file, we recommend to create an empty table space and back up this empty table space on a frequently regular base. Each time the table space is backed up, the recovery history file will be updated. Backing up an empty table space does not create much overhead on the DB2 UDB server and backup media (for example, TSM server).

Example 7-18 shows the point in time (PIT) recovery of a database using the **RECOVER** command.

Example 7-18 Recover database

```
> db2 "recover db gr4 to 2005-06-05-23.16.00 using local time"

                                Rollforward Status

Input database alias                   = gr4
 Number of nodes have returned status   = 1

Node number                            = 0
Rollforward status                     = not pending
 Next log file to be read               =
```

```
Log files processed                    = S0000063.LOG - S0000072.LOG
 Last committed transaction             = 2005-06-05-23.15.38.000000

DB20000I  The RECOVER DATABASE command completed successfully.
```

If no up-to-date recovery history file is available, we recommend that you restore the database and retrieve the transaction log files and use the **ROLLFORWARD command** to recover the database as described in “Rollforward recovery scenario 1” on page 360. For example, if you use TSM to save your transaction log files, they can be queried and retrieved using the `db2adutl` command as shown in Example 7-19.

Example 7-19 db2adutl to query and retrieve transaction log files

```
> db2adutl query logs

Query for database GR4

Retrieving LOG ARCHIVE information.
   Log file: S0000062.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-01-18.21.06
   Log file: S0000062.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-01-18.21.07
   Log file: S0000063.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-04-09.56.19
   Log file: S0000063.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-04-09.56.28
   Log file: S0000064.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-22.56.05
   Log file: S0000064.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-22.56.08
   Log file: S0000065.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-22.59.55
   Log file: S0000065.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-22.59.55
   Log file: S0000066.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-23.10.52
   Log file: S0000066.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-23.10.52
   Log file: S0000067.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-23.15.09
   Log file: S0000067.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-23.15.10
   Log file: S0000068.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-23.15.15
   Log file: S0000068.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-23.15.15
```

```
   Log file: S0000069.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-23.15.16
   Log file: S0000069.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-23.15.16
   Log file: S0000070.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-23.15.17
   Log file: S0000070.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-23.15.17
   Log file: S0000071.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-23.15.49
   Log file: S0000071.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-23.15.49
   Log file: S0000072.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-23.16.49
   Log file: S0000072.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-23.16.49
   Log file: S0000073.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-23.16.52
   Log file: S0000073.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-23.16.52
   Log file: S0000072.LOG, Chain Num: 4, DB Partition Number: 0, Taken at:
2005-06-05-23.59.59
   Log file: S0000072.LOG, Chain Num: 4, DB Partition Number: 0, Taken at:
2005-06-05-23.59.59


> db2adutl extract logs between S0000070.LOG and S0000072.LOG CHAIN 3

Query for database GR4

Retrieving LOG ARCHIVE information.

   LOG ARCHIVE image:
      Log file: S0000070.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-23.15.17
      Do you want to extract this log image (Y/N)? y
        Writing to file:
           ./NODE0000/GR4/C0000003/S0000070.LOG

   LOG ARCHIVE image:
      Log file: S0000070.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-23.15.17
      Do you want to extract this log image (Y/N)? y
      NODE0000/GR4/C0000003/S0000070.LOG
      Warning: log archive file already exists, renaming to
NODE0000/GR4/C0000003/S0000070.LOG.1         Writing to file:
           ./NODE0000/GR4/C0000003/S0000070.LOG.1

   LOG ARCHIVE image:
```

```
      Log file: S0000071.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-23.15.49
      Do you want to extract this log image (Y/N)? y
         Writing to file:
            ./NODE0000/GR4/C0000003/S0000071.LOG

   LOG ARCHIVE image:
      Log file: S0000071.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-23.15.49
      Do you want to extract this log image (Y/N)? y
      NODE0000/GR4/C0000003/S0000071.LOG
      Warning: log archive file already exists, renaming to
NODE0000/GR4/C0000003/S0000071.LOG.1         Writing to file:
            ./NODE0000/GR4/C0000003/S0000071.LOG.1

   LOG ARCHIVE image:
      Log file: S0000072.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-23.16.49
      Do you want to extract this log image (Y/N)? y
         Writing to file:
            ./NODE0000/GR4/C0000003/S0000072.LOG

   LOG ARCHIVE image:
      Log file: S0000072.LOG, Chain Num: 3, DB Partition Number: 0, Taken at:
2005-06-05-23.16.49
      Do you want to extract this log image (Y/N)? y
      NODE0000/GR4/C0000003/S0000072.LOG
      Warning: log archive file already exists, renaming to
NODE0000/GR4/C0000003/S0000072.LOG.1         Writing to file:
            ./NODE0000/GR4/C0000003/S0000072.LOG.1
```

Tip: For partitioned databases, you also can use the **RECOVER** command, but this must be invoked from the catalog node. This command automatically covers all partitions in the `db2nodes.cfg` file of your database.

7.2.4 Redirected restore of the database

During a restore operation, DB2 UDB can add, change, or remove table space containers, which is called a *redirected restore*. This can be very helpful when performing system copies or when you want to improve the physical layout of your database, for example, the distribution of the containers over your disks.

With the redirected restore, it is possible to change:

- The number and size of containers per table space.
- The name and location of table space containers.
- The database path for an automatic storage management enabled database.

- The number of storage paths for an automatic storage enabled database.

But it is not possible to add and remove table spaces, change the page size of a table space, or change the type of your table space (from SMS to DMS and vice versa). Furthermore, the auto-resize of DMS feature cannot be switched on or off during an redirected restore.

The tool `brdb6brt` from SAP can be used to generate a script for a redirected restore. `brdb6brt` uses the physical layout of the database to generate a script which can be easily modified. Within the generated script there are also a lot of information lines telling you the High Water Mark (HWM) of a table space, etc.

The minimum option you need for calling the `brdb6brt` is "`-bm RETRIEVE`"; more options can be specified. `REPLACE` is another useful option for homogeneous system copy. When you call the tool without any options, a help page will be displayed.

Redirected restore of a non-automatic storage enabled DB

Example 7-20 shows an excerpt of a redirected restore script generated by `brdb6brt`. In the Part 1 section, the general redirected restore procedure, you can modify the general restore parameter. In the Part 2 section, all table spaces of the database must be redefined. For those table spaces which you don't want to change, specify the table space definition as was before. In the last section, Part 3, the restore of the data to the newly defined table spaces will be done.

The redefining of the table spaces can be done as required. You can change number, size, and location of the containers but can't change the type (SMS or DMS). As a help, the actual size, filling degree, and the HWM are included in the information lines. With the **`IGNORE ROLLFORWARD CONTAINER OPERATIONS`** clause you can avoid running into errors during a subsequent recover of the redirected restored database when DB2 UDB tries to apply container operations like resize or add to the redefined table space.

Example 7-20 Generated redirected restore script (excerpt)

```
UPDATE COMMAND OPTIONS USING S ON Z ON GR4_NODE0000.out V ON;
SET CLIENT ATTACH_NODE  0;
SET CLIENT CONNECT_NODE 0;
ECHO @./GR4_NODE0000.scr@;

-- *****************************************************************************
-- **  DB2-CLP script
-- **  created for database GR4
-- **
-- **  Usage notes:
-- **
-- **  > In general all lines beginning with '--' are only comment lines
```

```
-- **    and will be ignored by the DB2 Command Line Processor
-- **  > Lines beginning with '-- **' are just comments and only for
-- **    your information
-- **  > Lines beginning with '-- S#' mark the beginning of a section
-- **    that can be edited
-- **  > Lines beginning with '-- E#' mark the end of a section that
-- **    can be edited
-- **
-- **  The script is subdivided into three major parts:
-- **  1. The general redirected restore procedure
-- **     (for further information about this part see the
-- **      'RESTORE DATABASE' command in the Command
-- **      Reference of DB2)
-- **  2. One redefining part for each tablespace
-- **     (for further information about this part see the
-- **      'SET TABLESPACE CONTAINERS' command in the Command
-- **      Reference of DB2)
-- **  3. The closing part
-- **     (this part completes the restore procedure.
-- **      See also the 'ROLLFORWARD DATABASE' command in
-- **      Command Reference of DB2)
-- **
-- ******************************************************************************



-- ******************************************************************************
-- ** Part 1 : General redirected restore procedure
-- ******************************************************************************

RESTORE DATABASE GR4
-- ** Path or device where the backup image is stored
-- S#############################
USE TSM OPEN 1 SESSIONS
-- ** Timestamp (when was the backup image taken? )
-- ** use the given format: YYYYMMDDhhmmss
-- S#############################
TAKEN AT 20050606113513
-- E#############################
-- ** New Database name
-- S#############################
INTO GR4
-- E#############################
-- ** If you want the logfiles to be be written to a new
-- ** directory, uncomment the following line and specify
-- ** the path name where the new primary logfiles should be written
-- ** Replace the <new_Log_Path>  a full quallified path name
-- S#############################
-- NEWLOGPATH /db2/GR4/log_dir/NODE0000/
```

```
-- E############################
-- ** Specify the number of buffers to be used for the restore procedure
-- S############################
WITH 2 BUFFERS
-- E############################
-- ** Specify the size of the buffers used for the restore
-- S############################
BUFFER 1024
-- E############################
REDIRECT
-- ** Specify the degree of parallelism used for restore
-- S############################
-- PARALLELISM 1
-- E############################
-- ** If the database should not be set to the 'rollforward pending' state
-- ** after the restore action, the following line has has to be uncommented.
-- S############################
-- WITHOUT ROLLING FORWARD
-- E############################
-- *****************************************************************************
-- ** LOGPATH information
-- *****************************************************************************
-- ** current LOGFILSIZ (4KB)       : 16380
-- ** current LOGPRIMARY            : 20
-- ** current LOGSECOND             : 40
-- *****************************************************************************
;


-- *****************************************************************************
-- ** Part 2 : Redefining of the tablespace containers for
-- **          each tablespace
-- *****************************************************************************


SET TABLESPACE CONTAINERS FOR 0
-- *****************************************************************************
-- ** 'IGNORE ROLLFORWARD' would specify that ALTER TABLESPACE operations
-- ** in the log are to be ignored when performing a roll forward.
-- S############################
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
-- E############################
USING (
-- ** Container information for DMS tablespace [0] SYSCATSPACE
-- *****************************************************************************
-- ** current total pages           : 50000
-- ** currently used pages          : 42592
-- ** current high water mark       : 42592
-- ** current page size (bytes)     : 4096
```

```
-- ** current extent size (pages)   : 32
-- ********************************************************************************
-- ** Container information
-- ** Type of containers can be changed. Valid modifications
-- ** are the both types FILE and DEVICE
-- ** If you want to add a container separate the new
-- ** container line by a comma.
-- **
-- ** type | name                                                        |       size
-- S############################
   FILE      /db2/GR4/sapdata1/SYSCATSPACE.container000                    25000
  ,FILE      /db2/GR4/sapdata2/SYSCATSPACE.container001                    25000
--,FILE      <new file>                                                   <size>
-- E############################
);

...
...

SET TABLESPACE CONTAINERS FOR 30
-- ********************************************************************************
-- ** 'IGNORE ROLLFORWARD' would specify that ALTER TABLESPACE operations
-- ** in the log are to be ignored when performing a roll forward.
-- S############################
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
-- E############################
USING (
-- ** Container information for SMS tablespace [30] SYSTOOLSTMPSPACE
-- ********************************************************************************
-- ** current total pages          : 1
-- ** currently used pages         : 1
-- ** current high water mark      : 0
-- ** current page size (bytes)    : 16384
-- ** current extent size (pages)   : 8
-- ********************************************************************************
-- ** Container information
-- ** Don't change the type of the container(s).
-- ** If you want to add a container separate the new
-- ** container line by a comma.
-- **
-- ** type | name
-- S############################
   PATH      /db2/GR4/sapdatat/SYSTOOLSTMPSPACE
--,PATH      <new directory>
-- E############################
);


-- ********************************************************************************
```

```
-- ** Part 3 : Complete the restore (and rollforward the database)
-- ***************************************************************************


RESTORE DATABASE GR4 CONTINUE ;

-- ***************************************************************************
-- ** If you want to rollforward the database you have to
-- ** uncomment the 'ROLLFORWARD DATABASE ...' line(s) below.
-- ** For more information about the rollforward process see
-- ** the documentation for BRDB6BRT-Tool or the
-- ** 'ROLLFORWARD DATABASE' command in the Command Reference of DB2
-- ***************************************************************************
-- S############################
-- ROLLFORWARD DATABASE GR4 TO END OF LOGS;
-- E############################
ECHO ********************************************************;
ECHO **  THE RESTORE PROCEDURE HAS NOW FINISHED SUCCESSFULLY  **;
ECHO ********************************************************;
```

After you have adapted the generated scripts to your needs, simply start the restore by issuing **db2 -tvf *scriptname***, as shown in Example 7-21.

Example 7-21 Redirected restore example

```
> db2 -tvf "./GR4_NODE0000.scr"
UPDATE COMMAND OPTIONS USING S ON Z ON GR4_NODE0000.out V ON
DB20000I  The UPDATE COMMAND OPTIONS command completed successfully.

SET CLIENT ATTACH_NODE  0
DB20000I  The SET CLIENT command completed successfully.

SET CLIENT CONNECT_NODE 0
DB20000I  The SET CLIENT command completed successfully.

@./GR4_NODE0000.scr@
RESTORE DATABASE GR4 USE TSM OPEN 1 SESSIONS TAKEN AT 20050606113513 INTO GR4
REDIRECT
SQL2539W  Warning!  Restoring to an existing database that is the same as the
backup image database.  The database files will be deleted.
Do you want to continue ? (y/n) y
SQL1277N  Restore has detected that one or more table space containers are
inaccessible, or has set their state to 'storage must be defined'.
DB20000I  The RESTORE DATABASE command completed successfully.

SET TABLESPACE CONTAINERS FOR 0 USING ( FILE
/db2/GR4/sapdata1/SYSCATSPACE.container000                   25000 ,FILE
/db2/GR4/sapdata2/SYSCATSPACE.container001                   25000 )
DB20000I  The SET TABLESPACE CONTAINERS command completed successfully.
```

```
SET TABLESPACE CONTAINERS FOR 1 USING ( PATH      /db2/GR4/sapdatat/TEMP16 )
DB20000I  The SET TABLESPACE CONTAINERS command completed successfully.

...
...

SET TABLESPACE CONTAINERS FOR 30 USING ( PATH
/db2/GR4/sapdatat/SYSTOOLSTMPSPACE )
DB20000I  The SET TABLESPACE CONTAINERS command completed successfully.

RESTORE DATABASE GR4 CONTINUE
DB20000I  The RESTORE DATABASE command completed successfully.

************************************************************
**  THE RESTORE PROCEDURE HAS NOW FINISHED SUCCESSFULLY  **
************************************************************
```

Tip: The redirected restore of a non-automatic storage enabled database can be used to switch the database from file-based DMS to device-based DMS. When doing so, you must first create the necessary partitions or logical volumes and use the **DEVICE** clause instead of the **FILE** clause when specifying the containers of the table spaces, for example:

```
SET TABLESPACE CONTAINERS FOR 5
USING (
   DEVICE    /dev/rGR2POOLI_000 49152
  ,DEVICE    /dev/rGR2POOLI_001 49152
);
```

In an SAP environment, SAP recommends that you do not create the temporary table spaces, SYSCATSPACE, and SYSTOOLSPACE as device-based DMS table spaces.

Redirected restore of an automatic storage enabled database

Redirected restore can also be performed on an automatic storage enabled database. Example 7-22 shows the generated script for redirected restore. In the Part 2 section, you can only adapt those table spaces that are not created with automatic storage management. The generated table space statements have been removed since they are same as Example 7-21. All other modifications must be done in the Part 1 section where you can modify the database path and also the storage paths of the database.

Example 7-22 Generated redirected restore script (excerpt) for automatic storage

```
UPDATE COMMAND OPTIONS USING S ON Z ON LEI_NODE0000.out V ON;
```

```
SET CLIENT ATTACH_NODE  0;
SET CLIENT CONNECT_NODE 0;
ECHO @./LEI_NODE0000.scr@;

-- *****************************************************************************
-- **  DB2-CLP script
-- **  created for database LEI
-- **
-- **  Usage notes:
-- **
-- **  > In general all lines beginning with '--' are only comment lines
-- **    and will be ignored by the DB2 Command Line Processor
-- **  > Lines beginning with '-- **' are just comments and only for
-- **    your information
-- **  > Lines beginning with '-- S#' mark the beginning of a section
-- **    that can be edited
-- **  > Lines beginning with '-- E#' mark the end of a section that
-- **    can be edited
-- **
-- **  The script is subdivided into three major parts:
-- **  1. The general redirected restore procedure
-- **     (for further information about this part see the
-- **      'RESTORE DATABASE' command in the Command
-- **      Reference of DB2)
-- **  2. One redefining part for each tablespace
-- **     (for further information about this part see the
-- **      'SET TABLESPACE CONTAINERS' command in the Command
-- **      Reference of DB2)
-- **  3. The closing part
-- **     (this part completes the restore procedure.
-- **      See also the 'ROLLFORWARD DATABASE' command in
-- **      Command Reference of DB2)
-- **
-- *****************************************************************************

-- *****************************************************************************
-- ** Part 1 : General redirected restore procedure
-- *****************************************************************************

RESTORE DATABASE LEI
-- ** Path or device where the backup image is stored
-- S#############################
FROM /db2/LEI/backup
-- ** Timestamp (when was the backup image taken? )
-- ** use the given format: YYYYMMDDhhmmss
-- S#############################
TAKEN AT 20050606113819
-- E#############################
-- ** Specify the automatic storage path list
```

```
-- ** one or more drives/paths separated by commas
-- S#############################
ON /db2/LEI/sapdata1
-- E#############################
-- ** Specify the database path
-- S#############################
DBPATH ON /db2/LEI
-- E#############################
-- ** New Database name
-- S#############################
INTO LEI
-- E#############################
-- ** If you want the logfiles to be be written to a new
-- ** directory, uncomment the following line and specify
-- ** the path name where the new primary logfiles should be written
-- ** Replace the <new_Log_Path>  a full quallified path name
-- S#############################
-- NEWLOGPATH /db2/LEI/log_dir/NODE0000/
-- E#############################
-- ** Specify the number of buffers to be used for the restore procedure
-- S#############################
WITH 2 BUFFERS
-- E#############################
-- ** Specify the size of the buffers used for the restore
-- S#############################
BUFFER 1024
-- E#############################
REDIRECT
-- ** Specify the degree of parallelism used for restore
-- S#############################
-- PARALLELISM 1
-- E#############################
-- ** If the database should not be set to the 'rollforward pending' state
-- ** after the restore action, the following line has has to be uncommented.
-- S#############################
-- WITHOUT ROLLING FORWARD
-- E#############################
-- ******************************************************************************
-- ******************************************************************************
-- ** Database Size
-- ******************************************************************************
-- ** AUTOMATIC STORAGE    :     8321 MB or 8726216704 Bytes
-- ** Additional Container :       98 MB or 103743488 Bytes
-- ******************************************************************************
-- ** LOGPATH information
-- ******************************************************************************
-- ** current LOGFILSIZ (4KB)       : 16380
-- ** current LOGPRIMARY            : 20
-- ** current LOGSECOND             : 40
```

```
-- ************************************************************************
;

-- ************************************************************************
-- ** Part 2 : Redefining of the tablespace containers for
-- **          each tablespace
-- ************************************************************************

...
...

-- ************************************************************************
-- ** Part 3 : Complete the restore (and rollforward the database)
-- ************************************************************************


RESTORE DATABASE LEI CONTINUE ;

-- ************************************************************************
-- ** If you want to rollforward the database you have to
-- ** uncomment the 'ROLLFORWARD DATABASE ...' line(s) below.
-- ** For more information about the rollforward process see
-- ** the documentation for BRDB6BRT-Tool or the
-- ** 'ROLLFORWARD DATABASE' command in the Command Reference of DB2
-- ************************************************************************
-- S############################
-- ROLLFORWARD DATABASE LEI TO END OF LOGS;
-- E############################
ECHO ********************************************************;
ECHO **  THE RESTORE PROCEDURE HAS NOW FINISHED SUCCESSFULLY  **;
ECHO ********************************************************;
```

7.2.5 Database relocation

If you have the need to rename a database, or relocate a database or parts of a database (for example, the containers or the log directory), you can do this by using the **db2relocatedb** command. All the changes must be done using OS commands when the database has been stopped. These changes must be entered in a configuration file, which will be used by the **db2relocatedb** command to make the necessary changes to the DB2 UDB instance and database support files. In the SAP environment, you can generate a relocate script using the following SAP utility:

```
brdb6brt -bm RETRIEVE_RELOCATE
```

The configuration file has the following format. The explanation of the variables in the configuration file can be found in Table 7-9.

```
DB_NAME=oldName,newName
DB_PATH=oldPath,newPath
INSTANCE=oldInst,newInst
NODENUM=nodeNumber
LOG_DIR=oldDirPath,newDirPath
CONT_PATH=oldContPath1,newContPath1
CONT_PATH=oldContPath2,newContPath2
...
STORAGE_PATH=oldStoragePath1,newStoragePath1
STORAGE_PATH=oldStoragePath2,newStoragePath2
...
```

Table 7-9 Explanation of the variables for db2relocatedb

Variable	Explanation
DB_NAME	Specifies the name of the database to be relocated. If the database name is to be changed, both the old name and the new name must be specified. This is a required field.
DB_PATH	Specifies the original path of the database to be relocated. If the database path is to be changed, both the old path and new path must be specified. This is a required field.
INSTANCE	Specifies the instance where the database exists. If the database is to be moved to a new instance, both the old instance and new instance must be specified. This is a required field.
NODENUM	Specifies the node number for the database node to be changed. The default is 0.
LOG_DIR	Specifies a change in the location of the log path. If the log path is to be changed, both the old path and new path must be specified. This specification is optional if the log path resides under the database path, in which case the path is updated automatically.
CONT_PATH	Specifies a change in the location of table space containers. Both the old and new container path must be specified. Multiple CONT_PATH lines can be provided if there are multiple container path changes to be made. This specification is optional if the container paths reside under the database path, in which case the paths are updated automatically. If you are making changes to more than one container where the same old path is being replaced by a common new path, a single CONT_PATH entry can be used. In such a case, an asterisk (*) could be used both in the old and new paths as a wildcard.
STORAGE_PATH	This is only applicable to databases with automatic storage enabled. It specifies a change in the location of one of the storage paths for the database. Both the old storage path and the new storage path must be specified. Multiple STORAGE_PATH lines can be given if there are several storage path changes to be made.

Example 7-23 shows how we move the container PSAPBTABD.container001 from sapdata4 to sapdata1 because there is an I/O bottleneck in the disk where sapdata4 is located. After creating the configuration file, we stop the DB2 UDB instance. We use the OS command **mv** to move the container to the other sapdata directory. Before starting the DB2 UDB instance again, we call the **db2relocatedb** command to adapt the DB2 UDB instance and database support files.

Example 7-23 Relocating one container

```
> cat relocate.cfg
DB_NAME=GR4
DB_PATH=/db2/GR4
INSTANCE=db2gr4
CONT_PATH=/db2/GR4/sapdata4/PSAPBTABD.container001,/db2/GR4/sapdata1/PSAPBTABD.
container001


> db2stop
06/06/2005 10:51:33     0   0   SQL1064N  DB2STOP processing was successful.
SQL1064N  DB2STOP processing was successful.


> mv /db2/GR4/sapdata4/PSAPBTABD.container001
/db2/GR4/sapdata1/PSAPBTABD.container001


> db2relocatedb -f relocate.cfg
Files and control structures were changed successfully.
DBT1000I  The tool completed successfully.


> db2start
06/06/2005 10:51:30     0   0   SQL1063N  DB2START processing was successful.
SQL1063N  DB2START processing was successful.
```

7.2.6 Throttling the backup utility

When you run online utilities such as the online backup during critical production periods, it is possible for DB2 UDB to guarantee that the performance impact on the production workload will be within acceptable limits.

To do so, first of all you have to define an impact policy in your database system, which will be defined by setting the UTIL_IMPACT_LIM database manager configuration parameter:

`UTIL_IMPACT_LIM=100`

With this default value for `UTIL_IMPACT_LIM`, no utility invocations will be throttled and the running utilities and production workload can compete for all the resources.

`UTIL_IMPACT_LIM<100`

With this value for `UTIL_IMPACT_LIM`, utility invocations can be throttled and then you can expect that all the utilities running in throttled mode will not impact the workload by more then `UTIL_IMPACT_LIM` percent.

After having defined an impact policy, you can start your online backups with the `UTIL_IMPACT_PRIORITY [priority]` clause, for example, **`BACKUP DB`** *<dbsid>* **`ONLINE ... UTIL_IMPACT_PRIORITY`** [*priority*]. The following options regarding throttling are possible:

- `UTIL_IMPACT_PRIORITY` clause not specified:

  The backup runs in an unthrottled mode.

- `UTIL_IMPACT_PRIORITY`:

  The backup runs in a throttled mode with a default priority of 50.

- `UTIL_IMPACT_PRIORITY [priority]`:

  The backup runs in a throttled mode with the specified priority, which can be a number between 1 and 100. By specifying a value for priority, it is possible to prioritize one utility over the other. The `UTIL_IMPACT_PRIORITY [priority]` can be set for backups taken with transaction `DB13` and can be configured using the `DB2DB6_DMDB6BKP_UTIL_IMPACT_PRIORITY` variable in `init<SAPSID>.db6`.

To check whether a utility like backup has been started in a throttled mode, you can use the **`LIST UTILITIES SHOW DETAIL`** command, as shown in Example 7-24.

Example 7-24 LIST UTILITIES SHOW DETAIL

```
> db2 list utilities show detail

ID                                   = 2
Type                                 = BACKUP
Database Name                        = GR5
Partition Number                     = 0
Description                          = online db
Start Time                           = 05/31/2005 08:32:45.667491
Throttling:
   Priority                          = Unthrottled
Progress Monitoring:
   Estimated Percentage Complete = 0
```

```
      Total Work                = 25253023817 bytes
      Completed Work            = 2821169 bytes
      Start Time                = 05/31/2005 08:32:45.668633
```

To change the UTIL_IMPACT_PRIORITY for a running utility (that means to throttle a utility or vice versa, or only to reprioritize a running utility), you can use the command:

SET UTIL_IMPACT_PRORITY FOR *utility_id* **TO** *priority*

Where:

priority = 0	Forces a utility to continue running in an unthrottled mode.
priority > 0	Forces a utility to continue running with a given priority, which can be a number between 1 and 100. With the priority, it's possible to prioritize some utilities over others.

Example 7-25 shows how to throttle the running backup, assuming that an impact policy has been defined within the database system by setting the UTIL_IMPACT_LIM database manager configuration parameter.

Example 7-25 SET UTIL_IMPACT_PRIORITY

```
> db2 set util_impact_priority for 2 to 50
DB20000I  The SET UTIL_IMPACT_PRIORITY command completed successfully.

> db2 list utilities show detail

ID                                = 2
Type                              = BACKUP
Database Name                     = GR5
Partition Number                  = 0
Description                       = online db
Start Time                        = 05/31/2005 08:32:45.667491
Throttling:
   Priority                       = 50
Progress Monitoring:
   Estimated Percentage Complete = 4
      Total Work                  = 25253089353 bytes
      Completed Work              = 1028909617 bytes
      Start Time                  = 05/31/2005 08:32:45.668633
```

7.3 Advanced Backup technology for a large database

These days, the data volume required for a customer's business is rapidly growing. It means that their database becomes much larger than before. In addition, customers need to run their business 24x7. Customers can no longer stop their production system for several hours in order to take a backup every day. Therefore, it is essential to take backup of a large database system quickly, without having a large impact on the performance of the customer's online operation, in order to protect their system from disaster, hardware failures, and so on. Quick recovery is also needed since their data includes critical information for their business.

To achieve this, DB2 UDB provides functions to support taking backup of a large DB2 UDB database quickly with advanced storage technology such as IBM ESS, EMC Timefinder, etc. With those storage systems, a split mirror image of a disk system can be created instantly. And those split mirror images can be mounted on another server or on a different directory or a file system of the same server.

You can also build a database clone or hot standby system with these features.

Note: These DB2 UDB features can also be used without advanced storage systems. For example, you can take backup of your file systems with the **cp** command, although it may take some time to take backup of your file system without advanced storage systems.

7.3.1 DB2 UDB functions for a large database backup/restore

This section describes DB2 UDB functions provided for taking a split mirror image of a database instantly and to prepare the image for use.

set write suspend

Syntax:

```
>>-SET--WRITE--+-SUSPEND-+--FOR--+-DATABASE-+------------------><
                                  '-DB-------'
```

This command suspends all database write operations to disk for table spaces and log files. (This is known as suspend I/O.) After the write operation is suspended by this command, you can create a split mirror image of your database that has integrity.

This command does not suspend write operations to memory areas such as the buffer pool and log buffer. You can execute SQL statements which do not require write operations to disk, that is, log files and table space containers. If an SQL statement you execute requires a write operation to disk, it waits until the write

operation is resumed. Even SELECT statements may not be executed, if they require write operations to temporary table spaces or if they require dirty pages in the buffer pool to be flushed to disk.

Once SET WRITE SUSPEND is executed, DB2 UDB flushes log records in the log buffer to the log file (disk). But DB2 UDB does not flush the contents of the buffer pools. As the log records in the log buffer are written to disk, DB2 UDB can apply data changes stored in the buffer pools by roll forward (or crash recovery) and the database can be consistent.

The active log file being used is truncated when SET WRITE SUSPEND is issued.

When SET WRITE SUSPEND is issued, some entries are written in db2diag.log. Example 7-26 shows an excerpt of the db2diag.log. This example is for the case in which the database configuration (DB CFG) parameter DIAGLEVEL is set to 3.

Example 7-26 SET WRITE SUSPEND message in db2diag.log

```
2005-05-19-21.22.24.799502+120 E13033A544      LEVEL: Warning
PID     : 368830               TID  : 1          PROC : db2agent (RED) 0
INSTANCE: db2red               NODE : 000        DB   : RED
APPHDL  : 0-23                 APPID: *LOCAL.db2red.050519192153
FUNCTION: DB2 UDB, buffer pool services, sqlbSuspendWrite, probe:80
MESSAGE : ADM6075W  The table space "SYSCATSPACE" (ID "0") has been placed in
the WRITE_SUSPEND state.  All write I/O for this table space will be suspended
until a WRITE RESUME is issued.
```

set write resume

Syntax:

```
>>-SET--WRITE--+-RESUME-+--FOR--+-DATABASE-+------------------><
                                '-DB-------'
```

This command resumes a suspended write operation for the database. The database is placed in its normal state and any SQL statements can be executed. The SQL statements waiting while the write operation was suspended are now executed. When SET WRITE SUSPEND is issued, some entries are written in db2diag.log.

Example 7-27 shows an excerpt of the db2diag.log. This example is for the case in which a DB CFG parameter DIAGLEVEL is set to 3. And the message shown is written for each table space that was set to write suspend state.

Example 7-27 SET WRITE RESUME message in db2diag.log

```
2005-05-19-21.22.46.066748+120 E14666A540       LEVEL: Warning
PID     : 368830               TID  : 1         PROC : db2agent (RED) 0
INSTANCE: db2red               NODE : 000       DB   : RED
APPHDL  : 0-23                 APPID: *LOCAL.db2red.050519192153
FUNCTION: DB2 UDB, buffer pool services, sqlbUnsuspendWrite, probe:20
MESSAGE : ADM6076W  The table space "SYSCATSPACE" (ID "0") which was previously
in the WRITE_SUSPEND state is no longer in that state.  Write I/O has been
resumed to the table space.
```

Tip: We recommend that you use the same connection for executing the `SET WRITE RESUME` command as for the `SET WRITE SUSPEND` command. If the connection in which you execute `SET WRITE SUSPEND` hangs, you can use the `RESTART DATABASE` command with the `WRITE RESUME` option. This option resumes suspended write operations without executing crash recovery.

If your database terminates abnormally while a write operation is suspended, you can also use the `RESTART DATABASE` command with the `WRITE RESUME` option. In this case, DB2 UDB runs crash recovery and resumes the write operation.

db2inidb

Syntax:

```
>>-db2inidb--database_alias--AS--+-SNAPSHOT-+------------------->
                                 +-STANDBY--+
                                 '-MIRROR---'
>--+---------------------------+-------------------------------><
   '-RELOCATE USING--configFile-'
```

This command initializes a split mirror image of a database which is taken while the database write operation is suspended. **`db2inidb`** has three options: `AS MIRROR`, `AS STANDBY`, and `AS SNAPSHOT`:

- If you need quick backup and recovery, you may need to use `AS MIRROR`.
- If you want to create a hot standby database, you may need to use `AS STANDBY`.
- If you need to take a DB2 UDB backup from the split mirror image, you may need to use `AS STANDBY` as well.
- If you need to build a cloned database, you may need to use `AS SNASPHOT`.

Here we describe the three options for `db2inidb`:

- `AS MIRROR`

  You can use this option to use your split mirror image of a database as a backup image. This provides you a quick recovery of your database from file system level backup. Use this option after copying back or mounting your split mirror image to your primary database server. This option sets a database to roll forward pending state without performing crash recovery, so roll forward recovery is possible after executing this command. To use this option, archive logging on the database is mandatory.

  Figure 7-19 shows an image of this option.

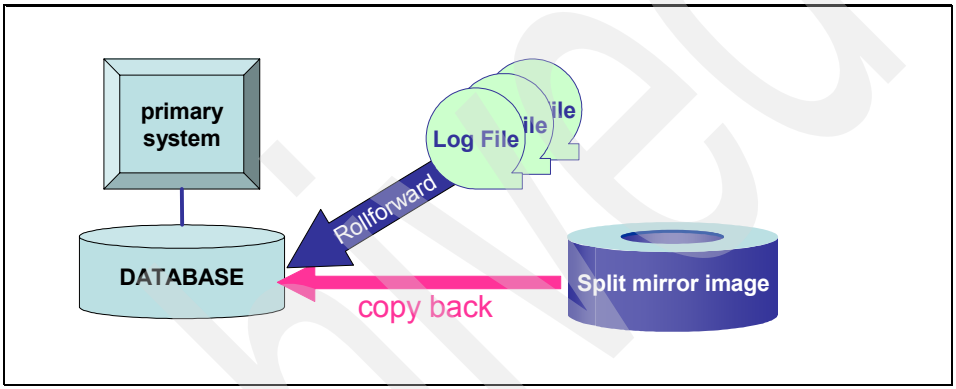


Figure 7-19 Using a split mirror image as backup image with db2inidb as mirror

- `AS STANDBY`

  A hot standby database system can be created with this option. A hot standby database system can be synchronized by transferring and applying (rolling forward) log files from the primary server. This standby database system also protects your system from logical errors by applying transferred log files with some delay.

  Also, a normal DB2 UDB backup is possible for a database initialized with this option.

  This option sets a database roll forward pending state without crash recovery. To use this option, archive logging on the primary database system is mandatory.

  Figure 7-20 gives you an image of how to create a standby database with `db2inidb`.

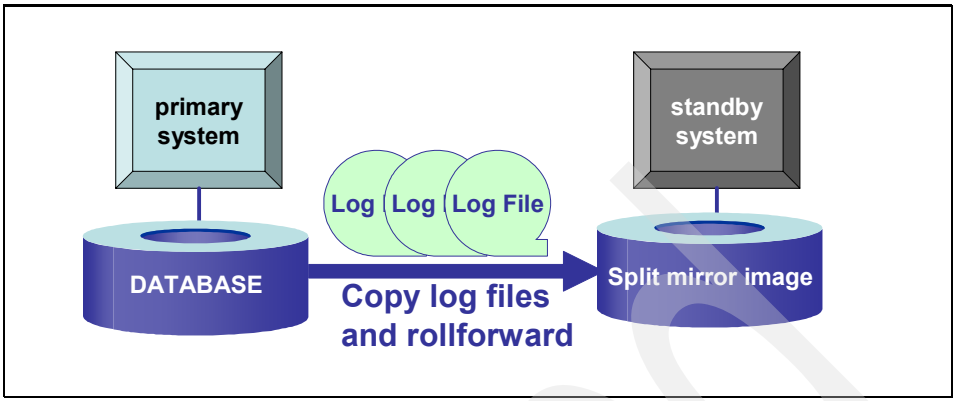


Figure 7-20 Creating a standby database with db2inidb as standby option

Figure 7-21 shows a simple picture how you can take a DB2 backup using `DB2INIDB AS STANDBY`.

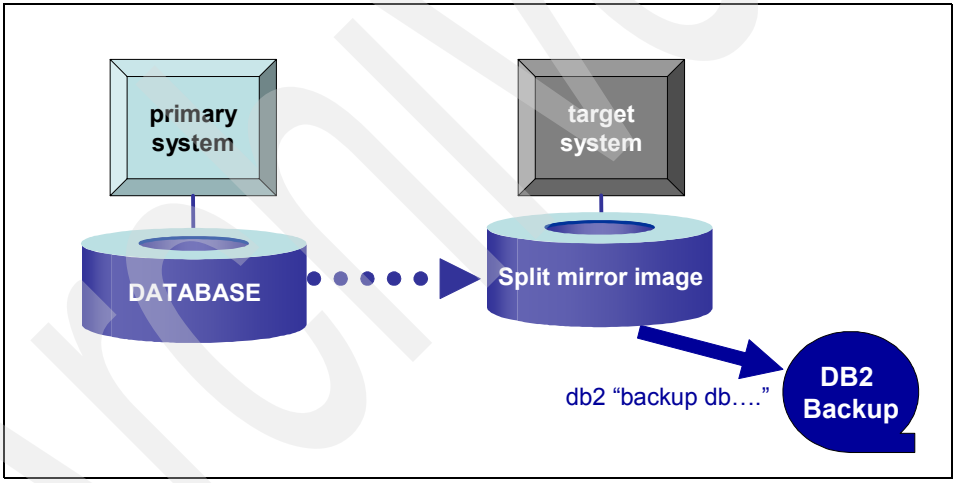


Figure 7-21 Taking a DB2 UDB backup from a split mirror image using db2inidb as standby

- `AS SNAPSHOT`

  You can create a database clone with this option. A database clone can be used for test systems or quality assurance systems. This option can be used even if circular logging is enabled. This option runs crash recovery and does not set a database to roll forward pending state. You cannot roll forward the cloned database using log files from the source system. Figure 7-22 shows a simple picture of how the system would be. It also tells you that a cloned

database will not have any relationship with the source database after splitting and initializing a image.

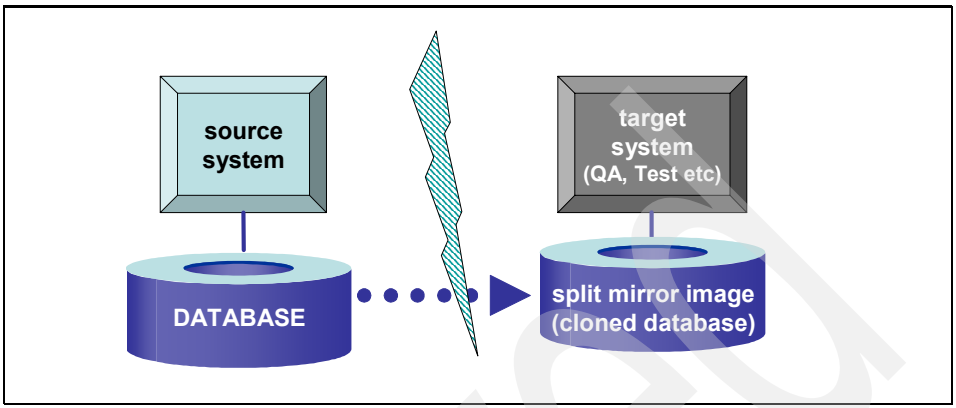


Figure 7-22 Creating a cloned database using db2inidb as snapshot

Note: You can issue **db2inidb** only against the database images which are taken while the write operation is suspended using the `SET WRITE SUSPEND` command. You have to execute **db2inidb** before using a split mirror image of your database.

If the **db2inidb** command is not issued against a split mirror image, you may get a `DBT1008N` error.

Example:

```
$ db2inidb RED as mirror
   DB2 UDB -- Initialize a Mirrored Database
   Usage: db2inidb database_alias as {SNAPSHOT | STANDBY | MIRROR} [relocate
using <configFile>]
   Notes: one of: SNAPSHOT, STANDBY or MIRROR must be specified.
DBT1008N  Database "RED" is not a split mirror image.
```

7.3.2 Back up and restore a large database

For a large database, an efficient way to create a database backup image which can also be restored efficiently is using the DB2 UDB suspend I/O function and **db2inidb as mirror**. In this section, we provide the backup and restore procedures and examples.

Note: We recommend that you use DB2 UDB new log file management or direct archiving with `db2inidb as mirror`.

Procedure for creating a backup image

For creating a backup image, the following steps have to be performed on the primary database server.

1. Log in to the primary database as user db2<*dbsid*>.
2. Suspend I/O on the primary database using command:

   db2 “set write suspend for db”

   Example:

   ```
   $ db2 set write suspend for db
   DB20000I  The SET WRITE command completed successfully.
   ```

3. Split a mirror from the primary database by an appropriate procedure for your storage system, such as FlashCopy® on IBM ESS.
4. Issue SET WRITE RESUME FOR DB on the primary database to resume I/O.

   db2 “set write resume for db”

   Example:

   ```
   $ db2 set write resume for db
   DB20000I  The SET WRITE command completed successfully.
   ```

Procedure for database restore

For restoring a backup image and recovery the database, the following steps have to be performed on the primary server:

1. Shut down the primary database instance using the command:

   db2stop

2. Copy back or mount the split mirror image which was taken at step 3 above by an appropriate procedure for your storage system, such as FlashCopy on IBM ESS.
3. Log in to the primary system as user db2<*dbsid*> and start the database instance using the command:

   db2start

4. Initialize the database using the command:

   db2inidb <database> **as mirror**

   Example:

   ```
   $ db2inidb BLU as mirror
   DBT1000I  The tool completed successfully.
   ```

5. Prepare log files for roll forward. To determine which log file you should prepare, the “rollforward db query status” command may help you.

Example:

```
$ db2 "rollforward db BLU query status"
                 Rollforward Status
Input database alias                   = BLU
Number of nodes have returned status   = 1
Node number                            = 0
Rollforward status                     = DB  pending
Next log file to be read               = S0000000.LOG
Log files processed                    =  -
Last committed transaction             = 1970-01-01-00.00.00.000000
```

This example indicates that S0000000.LOG is necessary to start roll forward. (See the field “Next log file to be read”.) For rolling forward further, subsequent log files are necessary.

6. Roll forward the database to the end of logs or to a point in time and complete roll forward using these commands:

   db2 “rollforward db <database> **to end of logs**

   or

   db2 “rollforward db <database> **to** <point in time>

   then

   db2 rollforward db <database> **complete**

   Example:

```
$ db2 "rollforward db BLU to end of logs"
Rollforward Status
Input database alias                   = BLU
Number of nodes have returned status   = 1
Node number                            = 0
Rollforward status                     = DB  working
Next log file to be read               = S0000002.LOG
Log files processed                    = S0000000.LOG -S0000002.LOG
Last committed transaction             = 2005-05-20-13.45.55.000000
DB20000I  The ROLLFORWARD command completed successfully.

$ db2 "rollforward db BLU complete"
Rollforward Status
Input database alias                   = BLU
Number of nodes have returned status   = 1
Node number                            = 0
Rollforward status                     = not pending
Next log file to be read               =
Log files processed                    = S0000000.LOG -S0000002.LOG
Last committed transaction             = 2005-05-20-13.45.55.000000
DB20000I  The ROLLFORWARD command completed successfully.
```

Tip: You can check the roll forward status by issuing the `ROLL FORWARD` command with the `QUERY` option. We recommend that you check the status before completing the roll forward.

What you need to include in the split mirror image

To make a consistent backup image, the following data must be included in your split mirror image:

- All database containers
- Database directory:
  - UNIX: `/db2/<DBSID>/db2<dbsid>`
  - Windows: `<drive>:\db2\<DBSID>\db2<dbsid>`

Note: Do not include the following items in your split mirror image:

- Log directory (`log_dir`)
- Archive log directory
- Retrieve log directory
- DB2 UDB instance directory

The log directory, archive log directory, and retrieve log directory should not be overwritten with the old image included in the split mirror image, because the current log might be necessary for recovery.

The DB2 UDB instance directory should not be included in the split mirror image, because write operations to the instance directory cannot be suspended by the `SET WRITE SUSPEND` command.

7.3.3 Creating a hot standby database

For high availability or disaster recovery solutions, you may need to prepare a hot standby database. To make a hot standby database efficiently, the DB2 UDB suspend I/O and **`db2inidb as standby`** commands can be used.

Note: We recommend that you use DB2 UDB new log file management or direct archiving with **`db2inidb as standby`** for creating a hot standby database.

Procedure on the primary server

At first, you need to take a split mirror image of your database on your primary server. Here is the procedure:

1. Log in to the primary database as user `db2<dbsid>`

2. Suspend I/O on the primary database with the command:

   db2 “set write suspend for db”

3. Split a mirror from the primary database by an appropriate procedure for your storage system, such as FlashCopy on IBM ESS.

4. Issue `SET WRITE RESUME FOR DB` on the primary database to resume I/O:

   db2 “set write resume for db”

Procedure on the standby server

A hot standby database can be created efficiently with a split mirror image taken on the primary server using db2inidb as standby command:

1. Install an SAP system by means of standard homogeneous system copy on the standby server.

2. Mount or copy the split mirror image which was taken at the step 3 above by an appropriate procedure for your storage system, such as IBM ESS FlashCopy.

3. Log in to the standby system as user DB2 UDB instance owner and start the db2 instance:

   db2start

4. Initialize the database using the command:

   db2inidb <database> as standby

   Example:

```
$ db2inidb BLU as standby
DBT1000I  The tool completed successfully.
```

Procedure for synchronizing the hot standby database

It is necessary to synchronize your hot standby database with the primary database regularly. Here is a procedure.

1. Copy offline log files from the primary server to the standby server regularly.

2. Roll forward the hot standby database to the end of logs or to a point in time. The commands to be used are:

   db2 “rollforward db <database> **to end of logs”**

   or

   db2 “rollforward db <database> **to** <point in time>**”**

   Example:

```
$ db2 "rollforward db BLU to end of logs"
Rollforward Status
Input database alias                   = BLU
```

```
Number of nodes have returned status   = 1
Node number                            = 0
Rollforward status                     = DB  working
Next log file to be read               = S0000008.LOG
Log files processed                    = S0000007.LOG -S0000007.LOG
Last committed transaction             =2005-05-24-13.00.17.000000
DB20000I  The ROLLFORWARD command completed successfully.
```

3. Repeat the log shipping and rollforward of the shipped logs on the standby database on a regular basis. When you need to access the hot standby database (for example, when the primary system fails), complete the rollforward:

   db2 “rollforward db <database> complete”

   Example:

```
$ db2 "rollforward db BLU complete"
Rollforward Status
Input database alias                   = BLU
Number of nodes have returned status   = 1
Node number                            = 0
Rollforward status                     = not pending
Next log file to be read               =
Log files processed                    = S0000007.LOG-S0000007.LOG
Last committed transaction             = 2005-05-24-13.00.17.000000
DB20000I  The ROLLFORWARD command completed successfully.
```

What you need to include in the split mirror image

To get a consistent image and prepare for synchronization with the primary database, the following data must be included in your split mirror image:

- All database containers
- Database directory:
  - UNIX: /db2/<*DBSID*>/db2<*dbsid*>
  - Windows: <drive>:\db2\<*DBSID*>\db2<*dbsid*>

Note: Beginning with Version 8.2, DB2 UDB provides a log record shipping function called *High Availability Disaster Recovery* (HADR). A standby database can be created with HADR easily. For more information about HADR, refer to 7.5.1, “High Availability Disaster Recovery (HADR)” on page 404

7.3.4 Taking a normal DB2 UDB backup

To take a normal DB2 backup for a large database takes some time. In case you do not want to take a normal DB2 backup from your primary database because of some performance impact, you can consider taking a DB2 backup from your

secondary server. Here, DB2 suspend I/O and db2inidb as standby commands can be used to create a secondary database for taking a DB2 backup.

Procedure for preparing for taking DB2 UDB backup

These steps describe how to get an image of the primary database. They should be performed on the primary database:

1. Log in to the primary database as user db2<*dbsid*>.
2. Suspend I/O on the primary database:

   ```
   db2 “set write suspend for db”
   ```

3. Split a mirror from the primary database by an appropriate procedure for your storage system, such as FlashCopy on IBM ESS.
4. Issue SET WRITE RESUME FOR DB on the primary database to resume I/O:

   ```
   db2 “set write resume for db”
   ```

Taking DB2 UDB backup from split mirror image

Using this procedure, a database image can be ready for a DB2 backup. This procedure should be performed on the secondary server:

1. Create a DB2 instance:

   ```
   db2icrt <instance_name>
   ```

2. Mount or copy the split mirror image which was taken at step 3. above by using an appropriate procedure for your storage system, such as FlashCopy on IBM ESS.
3. Log in to the system as the DB2 UDB instance owner and start the newly created instance:

   ```
   db2start
   ```

4. Initialize the database:

   ```
   db2inidb <database> as standby
   ```

   Example:

   ```
   $ db2inidb BLU as standby
   DBT1000I  The tool completed successfully.
   ```

5. Take the DB2 UDB backup:

   ```
   db2 “backup database <database> to <destination>”
   ```

   Example:

   ```
   $ db2 "backup db BLU to /db2/backup"
   Backup successful. The timestamp for this backup image is : 20050520163841
   ```

   Some entries are written in db2diag.log about the backup.

Example:

(This example is for the case in which a DIAGLEVEL database configuration parameter is set to 3.)

```
2005-05-20-16.38.41.410110+120 I110598C359 LEVEL: Warning
PID     : 405574               TID  : 1   PROC : db2agent (BLU) 0
INSTANCE: db2blu               NODE : 000 DB   : BLU
APPHDL  : 0-10                 APPID: *LOCAL.db2blu.050520143841
FUNCTION: DB2 UDB, data protection, sqlpgint, probe:5110
MESSAGE : Backup after split mirror.

2005-05-20-16.38.41.497009+120 I110958C378  LEVEL: Warning
PID     : 405574               TID  : 1    PROC : db2agent (BLU) 0
INSTANCE: db2blu               NODE : 000  DB   : BLU
APPHDL  : 0-10                 APPID: *LOCAL.db2blu.050520143841
FUNCTION: DB2 UDB, database utilities, sqlubSetupJobControl, probe:2025
MESSAGE : Starting an offline db backup.

2005-05-20-16.39.02.223890+120 I111337C351        LEVEL: Warning
PID     : 405574               TID  : 1    PROC : db2agent (BLU) 0
INSTANCE: db2blu               NODE : 000  DB   : BLU
APPHDL  : 0-10                 APPID: *LOCAL.db2blu.050520143841
FUNCTION: DB2 UDB, database utilities, sqlubcka, probe:130
MESSAGE : Backup Complete.
```

Note: If the database consists of DMS table spaces and SMS temporary table spaces only, you can take a DB2 UDB backup. If your database has SMS table spaces except for temporary table spaces, you cannot take a DB2 UDB backup from the split mirror image. In the SAP environment, only temporary table spaces are created as SMS table spaces by default.

Note: After taking backup, you can roll forward the database with log files from the source system. Once you start rolling forward, you cannot take a DB2 UDB backup any more.

What you need to include in the split mirror image

To make a consistent image of your database on the secondary server, the following data must be included in your split image:

- All database containers
- Database directory:
  - UNIX: /db2/*<DBSID>*/db2*<dbsid>*
  - Windows: *<drive>*:\db2*<DBSID>*\db2*<dbsid>*

7.3.5 Creating a database clone

You can quickly create a database clone for test purposes or for the quality assurance system by using the `AS SNAPSHOT` option of **db2inidb**, especially with advanced storage systems such as IBM ESS, EMC Timefinder.

Procedure on the source system

This procedure is for creating a split mirror image of your database. These steps should be performed on the source server:

1. Log in to the source database as user `db2`<*dbsid*>.
2. Suspend I/O on the source database using the command:

   db2 “set write suspend for db”
3. Split a mirror from the source database by an appropriate procedure for your storage system, such as FlashCopy on IBM ESS.
4. Issue `SET WRITE RESUME FOR DB` on the source database to resume I/O:

   db2 “set write resume for db”

Procedure on the target system

This procedure describes what to do on your target system to create a database clone:

1. Prepare an SAP system by means of standard homogeneous system copy.
2. Mount or copy the split mirror image which was taken at step 3 above by an appropriate procedure for your storage system, such as FlashCopy on IBM ESS.
3. Log in to the clone system as user `db2`<*dbsid*> and start the database instance:

   db2start
4. Initialize the cloned database using the command:

   db2inidb <DBSID> **as snapshot**

   Example:

```
$ db2inidb BLU as snapshot
DBT1000I  The tool completed successfully
```

Tip: If you want to change your container layout, use `RELOCATE USING` option with `db2inidb` command. The SAP tool `brdb6brt` helps you create a configuration file for the `RELOCATE USING` option. You can also change the database name, log path, and so on by the `RELOCATE USING` option. For more information about `brdb6brt`, refer to 7.2, “DB2 UDB backup and recovery” on page 338.

Note: The command `DB2INIDB AS SNAPSHOT` rolls back all transactions which are in flight when the split mirror image is taken on the source system. (Crash recovery runs.)

Entries for crash recovery may be written in `db2diag.log`.

```
Example:(This example is in case that a DB CFG parameter DIAGLEVEL is set
to 3.)
2005-05-20-16.21.56.385661+120 I97608C388     LEVEL: Warning
PID     : 352480               TID  : 1        PROC : db2agent (BLU) 0
INSTANCE: db2blu               NODE : 000      DB   : BLU
APPHDL  : 0-8                  APPID: *LOCAL.db2blu.050520142156
FUNCTION: DB2 UDB, recovery manager, sqlpresr, probe:3170

MESSAGE : Crash recovery completed. Next LSN is 000000000138800C
```

Note: As this command starts a new log file chain, log files created in the source system cannot be applied to the new cloned database.

5. If you use indirect archiving in your source system, you are required to delete the database `ADM`<*DBSID*> on the cloned database system:
   a. Delete the Admin DB:

      db2 “drop db ADM<*DBSID*>”

   b. Prune DB2 UDB backup history file:

      db2 “prune history force”

      Tip: DB2 UDB backup history file can be useful only for the source database system.

   c. If you need the log archiving feature on your cloned system, configure log file management accordingly (by DB2 UDB new log file management or using SAP Admin tools).

What you need to include in the split mirror image

To make a complete database on the target server, the following data must be included in your split mirror image:

- All database containers
- Database directory
  - UNIX: `/db2/<DBSID>/db2<dbsid>`
  - Windows: <drive>:\db2\<*DBSID*>\db2<*dbsid*>
- Log files in `log_dir`

  Crash recovery needs log files in `log_dir`.

7.3.6 Special considerations for IBM ESS Flash Copy on AIX platform

IBM ESS TotalStorage® FlashCopy can be used for creating a split mirror image of a DB2 UDB database in conjunction with suspend I/O and `db2inidb` feature of DB2 UDB. If any writes are occurring to the file systems on the ESS source volumes when the FlashCopy is being established, an inconsistency between the file system data and the file system log could occur.

To ensure the integrity of the target database, we recommend that you use the AIX *freeze* and *thaw* features if applicable. The AIX `freeze` and `thaw` features are option for the **`chfs`** command. For more information about AIX commands, refer to the AIX manual:

`http://publib.boulder.ibm.com/infocenter/pseries/index.jsp`

The AIX `freeze` and `thaw` feature is only available to JFS2 file systems. AIX 5.2 with APAR IY66043, or the combination of IY59928 and IY59770, provides this function. Also, AIX 5.3 with APAR IY59929 provides this function.

Here is a summary of procedures for implementing an online copy of DB2 UDB with FlashCopy:

1. Suspend write operations on DB2 UDB on the primary server:

   `db2 “set write suspend for database”`
2. Issue the AIX **`chfs`** command with the **`freeze`** option.
3. Establish the FlashCopy.
4. Issue the AIX **`chfs`** command with the **`thaw`** option.
5. Resume the write operation on DB2 UDB on the primary server.

   `db2 “set write resume for database”`

Note: Don't include the DB2 UDB instance directory and archive log files in the FlashCopy'd volume. You should take a backup of some important files under the instance directory, such as db2nodes.cfg or db2systm, by other means.

If you cannot use the AIX `freeze/thaw` feature and you use Copy Service V1, consider having a DMS temporary table space instead of a SMS temporary table space. Because a write operation for the SMS temporary table space occurs asynchronously, the AIX cache could be flushed to the file system when FlashCopy is established. This may result in unexpected errors.

To avoid i-node contention, we recommend that you create several containers for the DMS temporary table space. If you might have many sort operations in parallel, you should have more containers (for example, 10 containers) for a DMS temporary table space.

For more information, refer to IBM technote #1191417 and DB2 UDB documentation, *Data Recovery and High Availability Guide and Reference V8*, SC09-4831-01.

7.3.7 Offline backup image and recovery

When you need consistent backup and do not require 24x7 availability, you can create an offline split mirror image of your database. When recovery is necessary, the **db2rfpen** command can set the restored database image to roll forward pending state. After **db2rfpen** is successfully issued, the roll forward operation can be done to recover your database.

Here is a syntax of **db2rfpen**:

```
>>-db2rfpen--ON--database_alias-- -log--logfile_path----------><
```

Summary of the procedure for creating backup image

Here are the steps in this procedure:

1. Stop the SAP system and DB2 UDB on the primary server.
2. Unmount the file systems related to the database.
3. Create a split mirror image of the file systems.
4. Mount the file systems.
5. Restart the SAP system and DB2 UDB on the primary server.

Summary of the procedure for restore and recovery

Here are the steps in this procedure:

1. Mount or copy back the split mirror image taken at step 3 above:

   d. Issue **db2rfpen**.

   Example:

```
$ db2rfpen on BLU

-------------------------------------------------------------------------------
                ____    D B 2 R F P E N    ____
            IBM - Reset ROLLFORWARD Pending State
  The db2rfpen tool is a utility to switch on the database
rollforward pending state.
It will also reset the database role to STANDARD.

This tool should be used under the advisement of DB2 Service

SYNTAX: db2rfpen  on  < database_alias | -log logfile_path>

-------------------------------------------------------------------------------

Original Rollforward Pending state is Off.
Setting rollforward pending State to On.
```

2. Prepare log files for rolling forward.

   The DB2 rollforward command with the QUERY STATUS option can be used to determine which log file is necessary to start roll forward. The following example indicates that log file S0000008.LOG is necessary to begin roll forward. If further roll forward is necessary, subsequent log files are required.

   Example:

```
$ db2 "rollforward db BLU query status"
                     Rollforward Status
Input database alias                 = BLU
Number of nodes have returned status = 1
Node number                          = 0
Rollforward status                   = DB  pending
Next log file to be read             = S0000008.LOG
Log files processed                  = S0000007.LOG - S0000007.LOG
Last committed transaction
```

3. Roll forward the database and complete roll forward.

What you need to include in a split mirror image

Here is what you need to include:

- All database containers
- Database directory:
  - UNIX: /db2/<*DBSID*>/db2<*dbsid*>
  - Windows: <drive>:\db2\<*DBSID*>\db2<*dbsid*>

Note: The database must be configured in log archiving mode. Otherwise **db2rfpen** fails, because roll forward recovery cannot be done with circular logging. For more information about db2rfpen, refer to the IBM DB2 UDB documentation, *Command Reference V8*, SC09-4828-01.

7.4 Monitoring backup, restore and recovery progress

Since DB2 UDB V8.2, you can monitor the progress of backup, restore, and recovery operations using the LIST UTILITY command with the SHOW DETAIL option.

The LIST UTILITIES command displays, to the standard output device, the list of active utilities on the instance. The description of each utility can include attributes such as start time, description, throttling priority, and progress monitoring. The display of information regarding throttling priority and progress monitoring depends on the utility itself.

Tip: In a partitioned environment, you must issued this command on each partition you want to monitor, because the LIST UTILITIES command only shows the information from the partition it is executed on.

Let's look at an example for an offline backup:

```
$db2 list utilities show detail
ID                                   = 1
Type                                 = BACKUP
Database Name                        = GR4
Partition Number                     = 0
Description                          = online db
Start Time                           = 05/31/2005 09:29:45.411871
Throttling:
   Priority                          = Unthrottled
Progress Monitoring:
   Estimated Percentage Complete = 41
      Total Work                     = 4532126457 bytes
      Completed Work                 = 1845120737 bytes
      Start Time                     = 05/31/2005 09:29:45.413244
```

The information listed by the LIST UTILITIES command may be different depending on the utility.

The output shown for a backup operation indicates an initial estimate of the number of bytes to be processed. As the backup operation progresses, the number of bytes to be processed is updated. However, the bytes shown does not correspond to the size of the backup image. The actual image might be different depending on whether it is an incremental or compressed backup.

For restore operations, no initial estimate is given. The value UNKNOWN appears in this field. As each buffer is read from the image, the actual amount of bytes read is updated. For automatic incremental restore operations where multiple images might be restored, the progress is tracked using phases. Each phase represents an image to be restored from the incremental chain. At first, the output only indicates one phase. After the first image is restored, the total number of phases is shown. After the restore of each image, the number of phases completed and the number of bytes processed is updated.

For crash recovery and rollforward recovery, there are two phases of progress monitoring:

- FORWARD
- BACKWARD

Log files are read and then log records are applied to the database during FORWARD phase. In the case of crash recovery, the total amount of work is estimated using the starting log sequence number up to the end of the last log file. For rollforward recovery, when FORWARD phase begins, UNKNOWN is specified for the total work estimate; however, as the operation progresses the amount of work processed in bytes is updated.

During the BACKWARD phase, any uncommitted changes applied during the FORWARD phase are rolled back. In this case also an estimate, in bytes, is provided for the amount of log data to be processed. This estimate is updated as the operation progresses.

DB2 UDB BACKUP DATABASE is a throttling utility. You can manage the backup job so it will not compete the resources with other database applications. you can throttle it using the SET UTIL_IMPACT_PRIORITY command. You use the ID given in the output of the LIST UTILITIES command, as input parameter for the SET UTIL_IMPACT_PRIORITY command.

You can find more information about the use of the SET UTIL_IMPACT_PRIORITY command in 7.2.6, “Throttling the backup utility” on page 381.

7.5 High availability

High availability is the term used to describe systems that run almost all the time. Systems running using a high availability solution must be able to recover quickly in case of a hardware or software failure. The idea behind this concept is to minimize the impact of the failure to the end users.

DB2 UDB offers different options regarding high availability. There are two features introduced In DB2 UDB V8.2 to improve high availability operations. These features are High Availability and Disaster Recovery (HADR) and automatic client reroute.

Another option available in DB2 UDB V8.2, coming from previous versions, is to have a copy of the database in another machine and to send log files to that other machine on regular base. This solution is known as log file shipping.

Additionally, other high availability solutions are available depending on the platform your system is running. These solutions usually require another software product, that work together with DB2 UDB.

7.5.1 High Availability Disaster Recovery (HADR)

High Availability Disaster Recovery (HADR) was introduced in DB2 UDB V8.2. This is a database replication feature that provides a high availability solution for partial and complete site failures. It protects database against data loss by replicating data changes from one database server to another one.

In a HADR environment, the source database is called *primary database* and the target database is called *standby database*. When the configuration is not enabled, the database's role is referred as *standard database*.

The *primary database* is the database on which all the activity is carried on. Users and applications connect to this database and execute all their work. From an end user's point of view, it is like a standard database without HADR configured.

The *standby database* is an replication of the primary database. The database in standby system is updated by rolling forward the log files generated on the primary database. Log records of log files are sent from the primary database to the standby database by an internal DB2 UDB process using TCP/IP communication. These log files are replayed and the tables on standby systems are modified accordingly. Since the data changes between the primary and standby databases are via the log files, sometimes HADR is also referred as *log record replication*.

Figure 7-23 shows a diagram of the HADR architecture.

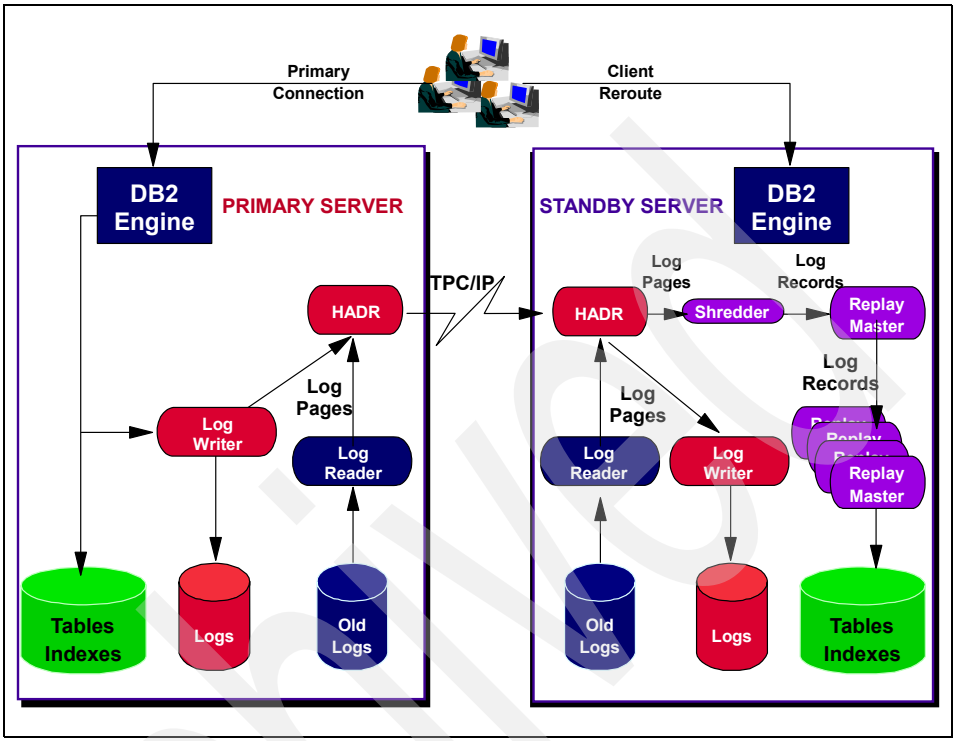


Figure 7-23 HADR implementation

The client connection to the standby database shown in Figure 7-23 only takes places if the primary database fails. This connection to the standby database is managed by the DB2 UDB automatic client reroute feature.

The *standby database* is usually located on a database server in different location. The users or applications will not access the standby database. It is a database that will be used when the primary system is not available.

In an HADR environment, the standby database can take over the operation in seconds if primary system encounters partial site failure (such as hardware, network or disk) or a complete site failure (that is, when the entire site is lost because of a major disaster — fire, earthquake, etc.). The standby system also can be used if system upgrade is required on the primary system.

Synchronization modes

When you configure HADR, you can choose between the following synchronization modes:

- Synchronous (`SYNC`)
- Near synchronous (`NEARSYNC`)
- Asynchronous (`ASYNC`)

Depending on the mode you choose, there is a trade-off between the protection against data loss and the transaction response time.

Synchronous

This mode provides the greatest protection level against data loss. With this mode a log write on the primary database is only considered to be successful when the primary database has received an acknowledgement from the standby database indicating that the log write to the disk was successful.

Near synchronous

This mode provides less protection level against data loss in comparison to the synchronous mode, but the response time is better. In this mode, a log write on the primary database is considered to be successful when the log records have been written to disk on the primary database and the primary database has received acknowledgement from the standby database that the logs have also been received.

Asynchronous

This mode is considered to provide the lowest protection level against data loss and the best transaction response time. In this mode, log writes are considered to be successful only when the log records have been written to the log files on the primary database and sent to the standby database by the TCP communication layer; however, there is no wait time for acknowledgement from the standby database.

To choose a mode, you use the command:

```
#db2 update db cfg for <database> using HADR_SYNCMODE <mode>
```

Where <mode> can have any of the values mentioned above.

HADR replicated operations

The following operations are replicated from the primary to the standby database:

- Data definition language (that is, `CREATE TABLE`, `CREATE INDEX`, `ALTER TABLE`, etc.) statements

- Data manipulation language (INSERT, DELETE, UPDATE) statements
- Buffer pool operations
- Table space operations
- Online reorganization
- Offline reorganization
- Metadata for stored procedures and user defined functions (UDFs)

HADR non-replicated operations

Some of the database operations, which are not replicated from the primary database to the standby database, are:

- Tables created with the NOT LOGGED INITIALLY option.
- BLOBs and CLOBS are not replicated; however, the space for them is allocated on the standby database.
- Datalinks are not supported in HADR databases.
- Changes to the database configuration parameters and database manager configuration parameters.
- The contents of the history file and its changes are not shipped from the primary database to the standby database. However, the RESTORE command can use the REPLACE HISTORY FILE and HISTORY FILE options to use another history file.

General recommendations

Whenever you plan to use HADR, it is important to consider these requirements:

- A high-speed network should be present in your environment. Remember that TCP/IP is the communication method used by HADR.
- DB2 UDB versions should be identical in both machines. The exception will be system upgrading. The upgrade window should be kept as small as possible. Keep in mind that DB2 UDB version in the primary database cannot be lower than the one in the standby database.
- Table spaces (type and size) and containers (path, size, and type) must be identical on both databases.
- Available memory in both machines should be the same, because buffer pool operations are also replayed on the standby database.
- Database configuration parameters and database manager configuration parameters should be identical on both systems to avoid possible errors during HADR operation. Changes on the primary database configuration parameters must be manually replicated to the standby database, because those kind of changes are not logged.

- The LOGINDEXBUILD database configuration parameter is used to decide whether index creation, recreation, and reorganization operations are fully logged or not. Depending on the value set, DB2 UDB configured with HADR takes special actions when an index is accessed in standby database. *IBM DB2 UDB Data Recovery and High Availability Guide and Reference V8*, SC09-4831-01 provides more information regarding this behavior.
- The INDEXREC database configuration parameter set to RESTART on the primary and standby databases causes invalid indexes to be rebuilt after a takeover operation is complete. See *IBM DB2 UDB Data Recovery and High Availability Guide and Reference V8*, SC09-4831-01 for more information about what are the effects of this parameter in an HADR environment.

Automatic client reroute

The automatic client reroute feature allows a client to continue working if it losses the communication with the primary server. DB2 UDB routes the transaction to another system and continue the work. From the client application's point of view, there is only a minimal communication interruption.

If automatic client reroute is not enabled, client applications will receive error message SQL30081, and no further attempts will be made to establish a connection with the server.

To enable automatic client reroute, you use the command:

```
db2 update alternate server for database <database_name> using hostname
<host_name> port <port_number>
```

Where:

- database_name: Specifies the alias of the database of the alternate server.
- host_name: Is the machine where the alternate database resides.
- port_number: Is the port number of the alternate database manager instance.

For example, this command sets the alternate server for the DAP database to be in the hosts *leimen*, listening at port *5800*.

```
#db2 update alternate server for db DAP using hostname leimen port 5800
```

To see if an alternate server is configured in your system, execute the command:

```
#db2 list db directory

 System Database Directory

 Number of entries in the directory = 3

Database 1 entry:
```

```
Database alias                       = DAP
Database name                        = DAP
Local database directory             = /db2/DAP
Database release level               = a.00
Comment                              = SAP database DAPINO
Directory entry type                 = Indirect
Catalog database partition number    = 0
Alternate server hostname            = leimen
Alternate server port number         = 5800
```

The alternate server fields list the configuration for each entry. For a complete description of this command, see *IBM DB2 UDB Command Reference V8*, SC09-4828-01.

HADR restrictions

The following list is a summary of HADR restrictions:

- HADR is supported in DB2 UDB Enterprise Server Edition (ESE) as a no-charge option and on DB2 Express and Workgroup Edition as a separately charged option.
- Both machines must have the same architecture and have installed the same operating system version, including patch levels.
- The DB2 UDB release on primary and standby databases must be the same bit size (32 or 64 bit).
- The standby database cannot be used for reading.
- Log archiving can only be performed by the primary database.
- Non-logged operations (that is, changes in the database configuration file) are not replicated to the standby database.
- Load operations with the `COPY NO` option specified are not supported.
- Use of Data Links is not supported.

Using HADR in SAP environments

In this section we provide an example of how to configure the new DB2 UDB V8.2 feature HADR in an SAP environment. We explain each of the steps you need to follow to get your system up and running, and we show you which commands you need to issue to do it. By the end of this section, we simulate a failure on the primary database and a takeover of the operation by the standby database.

Configuring HADR

To configure High Availability Disaster Recovery, you should follow these steps:

1. Identify the hostname, IP address and communication port to be used in the HADR environment. This information is required the machines holding the primary and database instance.

   – Hostname:

     This value is the name of the machine in which the primary and database would reside.

     In our example, the primary database resides in a Linux machine, whose name is leimen, and the standby database resides in another Linux machine, whose name is saplin01. Both machines runs Linux SUSE 9.

   – IP Address:

     This value is the IP address of the machine in which the primary and database would reside.

     To validate if this information is correct, use the ping command for both machines.

```
db2dap@saplin01:/db2/HADR_test> ping leimen
PING LEIMEN (9.153.164.148) 56(84) bytes of data.
64 bytes from LEIMEN (9.153.164.148): icmp_seq=1 ttl=64 time=0.204 ms
64 bytes from LEIMEN (9.153.164.148): icmp_seq=2 ttl=64 time=0.196 ms

db2lei@leimen:/db2/HADR_test> ping saplin01
PING saplin01 (9.153.164.155) 56(84) bytes of data.
64 bytes from saplin01 (9.153.164.155): icmp_seq=1 ttl=64 time=0.174 ms
64 bytes from saplin01 (9.153.164.155): icmp_seq=2 ttl=64 time=0.207 ms
```

     If the ping does not respond or the configuration is not right, contact your network administrator and let him know what are the changes you need.

     If you are using a Linux or UNIX operating system, you can modify the */etc/hosts* file to update it with the correct information (if you have the privileges to do it).

   – Communication port:

     This value is the port number used by the HADR processes to communicate with each other.

     You should add two entries in the services file of your operating system.

     For our example, we add these values in both machines:

```
HADR_saplin01   70000/tcp
HADR_leimen     70001/tcp
```

2. Create the standby database by restoring a backup image from the primary database.

   In our example, we create an offline backup of the database and restore it using the `RESTORE` command.

Tip: You can also create the standby database using the `db2inidb` command with the standby option.

Important: Do not rollforward the database, when the standby database catch-up with the primary database, it will automatically rollforward the transactions from the primary database.

3. Set the HADR database configuration parameters.

   The parameters to set are described in Table 7-10:

Table 7-10 HADR configuration parameters

Parameter	Description
`HADR_LOCAL_HOST`	This parameter specifies the local host for HADR TCP communication. Either a host name or an IP address can be used.
`HADR_LOCAL_SVC`	This parameter specifies the TCP service name or port number for which the local HADR process accepts connections.
`HADR_REMOTE_HOST`	This parameter specifies the TCP/IP host name or IP address of the remote HADR node.
`HADR_REMOTE_SVC`	This parameter specifies the TCP service name or port number that will be used by the remote HADR node.
`HADR_REMOTE_INST`	This parameter specifies the instance name of the remote server.
`HADR_TIMEOUT`	This parameter specifies the time (in seconds) that the HADR process waits before considering a communication attempt to have failed.
`HADR_SYNCMODE`	This parameter specifies the synchronization mode, which determines how primary log writes are synchronized with the standby when the systems are in peer state. The possible values are: `SYNC`, `NEARSYNC`, `ASYNC`. `NEARSYNC` is the default value.

In our example, we use these commands to change the database configuration parameters of the primary and standby database.

– For the primary database, we execute

```
db2 update db cfg for LEI using HADR_LOCAL_HOST leimen
db2 update db cfg for LEI using HADR_LOCAL_SVC HADR_leimen
db2 update db cfg for LEI using HADR_REMOTE_HOST saplin01
db2 update db cfg for LEI using HADR_REMOTE_SVC HADR_saplin01
db2 update db cfg for LEI using HADR_INST_HOST db2lei
db2 update db cfg for LEI using HADR_SYNCMODE NEARSYNC
```

– For the standby database, we execute

```
db2 update db cfg for LEI using HADR_LOCAL_HOST saplin01
db2 update db cfg for LEI using HADR_LOCAL_SVC HADR_saplin01
db2 update db cfg for LEI using HADR_REMOTE_HOST leimen
db2 update db cfg for LEI using HADR_REMOTE_SVC HADR_leimen
db2 update db cfg for LEI using HADR_INST_HOST db2lei
db2 update db cfg for LEI using HADR_SYNCMODE NEARSYNC
```

Starting HADR

To start High Availability Disaster Recovery, you should follow these steps:

1. Start HADR on the standby database by using the command `START HADR`.

   In our example, we start HADR in *saplin01* this way:

   ```
   # db2 start hadr on db LEI as standby
   DB20000I The START HADR ON DATABASE command completed successfully.
   ```

2. Verifiy that HADR in the standby database is waiting for HADR to start in the primary database.

   To be sure that HADR in the standby database has been started up correctly, look for the following information in DB2 UDB diagnostic log:

   ```
   MESSAGE : Info: HADR Startup has begun.
   MESSAGE : Info: Replaymaster Starting...
   MESSAGE : Info: HADR Startup has begun.
   MESSAGE : Starting Replay Master on standby.
   MESSAGE : Info: Standby Started.
   MESSAGE : ADM1602W  Rollforward recovery has been initiated.
   MESSAGE : ADM1603I  DB2 is invoking the forward phase of the database
             rollforward recovery.
   ```

   As the primary database is not started up to now, you should see a message like this, also:

   ```
   MESSAGE : Failed to connect to primary. rc:
   ```

   Then other messages like the following ones should appear:

   ```
   MESSAGE : Info: HADR Startup has completed.
   CHANGE  : HADR state set to S-RemoteCatchupPending (was S-LocalCatchup)
   ```

If the standby database is in this state, it means that it is waiting for HADR to start in the primary database and get in sync with them.

3. Start HADR on the primary database by using the command `START HADR`.

   In our example, we start HADR in *leimen* this way:

   ```
   # db2 start hadr on db LEI as primary
   DB20000I  The START HADR ON DATABASE command completed successfully.
   ```

4. Verifiy that HADR in the primary database is started and that both databases, the primary database and the standby database, try to get in sync.

   Look for the following information in the DB2 UDB diagnostic log of the primary database:

   ```
   MESSAGE : Info: HADR Startup has begun.
   MESSAGE : Info: Primary Started.
   MESSAGE : Info: HADR Startup has completed.
   CHANGE  : HADR state set to P-RemoteCatchupPending (was P-Boot)
   MESSAGE : Info: HADR Startup has completed.
   MESSAGE : remote catchup starts at 00000001A67B700C
   MESSAGE : near peer catchup starts at 00000001A67B7CBE
   CHANGE  : HADR state set to P-NearlyPeer (was P-RemoteCatchup)
   CHANGE  : HADR state set to P-Peer (was P-NearlyPeer)
   ```

   And verify that the standby database change its state. To do this, consult the DB2 UDB diagnostic log of the standby database:

   ```
   CHANGE  : HADR state set to S-RemoteCatchupPending (was
   S-RemoteCatchupPending)
   CHANGE  : HADR state set to S-RemoteCatchup (was S-RemoteCatchupPending)
   CHANGE  : HADR state set to S-NearlyPeer (was S-RemoteCatchup)
   CHANGE  : HADR state set to S-Peer (was S-NearlyPeer)
   ```

5. Verifiy that process responsible for rolling forward the transactional information is started in the standby database.

   Using the command `list applications`, you must find a process running whose name is *db2replay*.

6. Verifiy that the standby database is in peer with the primary database:

   You can do it, as follows:

   ```
   # db2 get snapshot for database on lei

   Database Snapshot

   Database name                              = LEI
   ...
   Database status                            = Rollforward
   ...
   HADR Status
   ```

```
  Role                   = Standby
  State                  = Peer
  Synchronization mode   = Nearsync
  Connection status      = Connected, 06/02/2005 06:55:34.823590
  Heartbeats missed      = 0
  Local host             = saplin01
  Local service          = HADR_saplin01
  Remote host            = leimen
  Remote service         = HADR_leimen
  Remote instance        = db2lei
  timeout(seconds)       = 120
  Primary log position(file, page, LSN) = S0000035.LOG, 3625,
00000001A67DD856
  Standby log position(file, page, LSN) = S0000035.LOG, 3625,
00000001A67DD856
  Log gap running average(bytes) = 8092
```

7. Verify that the primary database is in peer with the standby database:

   You can do it, as follows:

```
              Database Snapshot

Database name                              = LEI
...
Database status                            = Active

HADR Status
  Role                   = Primary
  State                  = Peer
  Synchronization mode   = Nearsync
  Connection status      = Connected, 06/02/2005 15:10:59.852553
  Heartbeats missed      = 0
  Local host             = leimen
  Local service          = HADR_leimen
  Remote host            = saplin01
  Remote service         = HADR_saplin01
  Remote instance        = db2dap
  timeout(seconds)       = 120
  Primary log position(file, page, LSN) = S0000035.LOG, 3626,
00000001A67DE220
  Standby log position(file, page, LSN) = S0000035.LOG, 3625,
00000001A67DDFF1
  Log gap running average(bytes) = 109
```

Stoping HADR

To stop High Availability Disaster Recovery, you should follow these steps:

1. Stop HADR on the primary database by using the command `STOP HADR`.

   In our example, we stop HADR in *leimen* this way:

   ```
   # db2 stop hadr on db lei
   DB20000I The STOP HADR ON DATABASE command completed successfully.
   ```

2. Stop HADR on the standby database by using the command `STOP HADR`.

   In our example, we stop HADR in *saplin01* this way:

   ```
   #db2 deactivate db lei
   DB20000I  The DEACTIVATE DATABASE command completed successfully.
   # db2 stop hadr on db lei
   DB20000I  The STOP HADR ON DATABASE command completed successfully.
   ```

7.5.2 Log file shipping

Log file shipping is the process of copying a whole log file to a standby machine. In this high availability scenario, the standby database receives the log files from the primary machine and rolls forward these log files to redo the transactions generated in the primary machine. This copy process is usually done by a user exit program configured in the DB2 UDB environment. This technique was commonly used in DB2 UDB V8.1 and prior version.

The hardware of the standby machine does not need to match exactly the primary one. The number and speed of the processors, number and size of disks can be different. However, the physical (containers) and logical (table spaces) definitions of both machines must be equal.

To set up the log file shipping environment, first we initialize the standby system. There are two ways to initialize a standby machine:

- Use the `RESTORE` command.

  By using the `RESTORE` command, you restore a backup image from the primary database into the standby machine. For a description of this command, see 7.2.3, “Recovering the database” on page 351.

- Use the `db2inidb` command with the `STANDBY` option.

  This option is typical for large databases, using the `db2inidb` command you split a mirror copy of the production database to set up the standby database. For a lists of tasks to do when using this command, see “7.3, “Advanced Backup technology for a large database” on page 384.

The second step is to prepare the primary machine to transfer log files to the standby machine. Two options are available:

- Archived log files in the primary machine are accessible for the standby machine by some kind of shared access. In this case, there is no log file transfer, but as log files are available to the standby database, it can roll them forward.
- Log files in the primary machine are copied to the standby machine by using some kind of transfer program, that is, `ftp`, `rsh`, etc. This option does require the transfer of the logs to the other machine.

If the primary system is not available, to bring up the standby system for applications:

1. Transfer the remaining log files which haven't been transferred yet to the standby machine.
2. In the standby machine, rolls forward log files to the end of the logs and stop.
3. The clients reconnect to the standby machine and resume operations.

The main step in bringing up the standby system in case of failure is rolling forward log files to its end. If log files are copied to the standby machine but not rolled forward at all; in case of failure, the time needed to bring up the standby system is longer. In addition, the amount of disk space needed in the standby machine to hold all log files will be more. To shorten the wait time until the standby system is productive, roll forward the standby system in appropriate intervals.

To avoid having problems during the operation of the standby machine, it is highly recommended to ensure that both machines are configured as equally as possible. DB2 UDB is one of the software components in the environment, but not the only one. If you need to use the standby machine in case of the primary machine fails, all other software components must be configured correctly to minimize operation downtime.

7.5.3 Clustered solutions

Failover strategies are usually based on clusters of systems. A cluster is a group of connected systems that work together as a single system. Each physical machine within a cluster contains one or more logical nodes. Clustering allows servers to back each other up when failures occur, by taking over the workload of the failed server.

Failover software may use heartbeat monitoring or keepalive packets between systems to confirm availability. Heartbeat monitoring involves system services that maintain constant communication between all the nodes in a cluster. If a

heartbeat is not detected, failover to a backup system starts. End users are usually not aware that a system has failed.

The two most common failover strategies on the market are known as *idle standby* and *mutual takeover*, although the configurations associated with these terms may also be associated with different terms that depend on the vendor:

- Idle Standby:

  In this configuration, one system is used to run a DB2 UDB instance, and the second system is “idle”, or in standby mode, ready to take over if there is an operating system or hardware failure on the first system. Overall system performance is not impacted, because the standby system is idle until needed.

- Mutual Takeover:

  In this configuration, each system is the designated backup for another system. Overall system performance may be impacted because the backup system then must do extra work following a failover: it must do its own work plus the work that was being done by the failed system.

Failover strategies can be used to failover a database, an instance, or a database partition.

In this section, we discuss the following clustered solutions:

- Tivoli System Automation for Linux
- High Availability Cluster Multi-Processing, Enhanced Scalability, for AIX
- Microsoft® Cluster Server, for Windows operating systems
- Sun Cluster, for the Solaris Operating Environment
- Veritas Cluster Server, for the Solaris Operating Environment
- Multi-Computer/ServiceGuard, for Hewlett-Packard

We provide an introduction to each of these options and give you references to other sources of information useful for configuring them.

High availability using Tivoli System Automation for Linux

IBM Tivoli System Automation (TSA) for Linux is a product that provides high availability by automating the control of resources such as processes, file systems, IP addresses, and other arbitrary resources in Linux-based clusters.

It facilitates the automatic switching of users, applications, and data from one system to another in the cluster after a hardware or software failure. A complete High Availability setup includes many parts, one of which is the high availability software. As well as tangible items such as hardware and software, a good high availability solution includes planning, design, customizing, and change control.

For more information, see the following URL:

http://www.software.ibm.com/tivoli/products/sys-auto-linux

High availability on AIX using enhanced scalability

Enhanced scalability (ES) is a feature of High Availability Cluster Multi-Processing (HACMP) for AIX. This feature provides the same failover recovery and has the same event structure as HACMP. For more specific information about HACMP in AIX systems, browse the following URL:

http://www-1.ibm.com/servers/eserver/pseries/library/hacmp_docs.html

Enhanced scalability also has other provisions, such as:

- Larger clusters support.
- Additional error coverage through user-defined events.
- Use of monitored areas; monitored areas can trigger user-defined events, which can be as diverse as the death of a process or the fact that paging space is nearing capacity. Such events include pre- and post-events that can be added to the failover recovery process, if needed. Extra functions that are specific to the different implementations can be placed within the HACMP pre-event and post-event streams.
- A rules file (`/usr/sbin/cluster/events/rules.hacmprd`) contains the HACMP events. User-defined events are added to this file. The script files that are to be run when events occur are part of this definition.
- HACMP client utilities for monitoring and detecting status changes (in one or more clusters) from AIX physical server outside of the HACMP cluster.

The servers in HACMP ES clusters exchange messages called heartbeats or keep alive packets, by which each server informs the other server about its availability. A server that has stopped responding causes the remaining servers in the cluster to invoke recovery. The recovery process is called a server-down-event and may also be referred to as failover. The completion of the recovery process is followed by the reintegration of the server into the cluster. This is called a server-up-event.

There are two types of events:

- Standard:

  Standard events are anticipated within the operations of HACMP ES.

- User-defined:

  User-defined events are associated with the monitoring of parameters in hardware and software components.

For example, one of the standard events is the server-down-event. When planning what should be done as part of the recovery process, HACMP allows two failover options: hot (or idle) standby and mutual takeover.

In a hot standby configuration, the AIX processor node that is the takeover node is not running any other workload. In a mutual takeover configuration, the AIX processor node that is the takeover node is running other workloads.

For a more complete introduction to how DB2 UDB interacts with Enhanced Scalability in AIX operating systems, see the DB2 UDB product documentation, *IBM DB2 UDB Data Recovery and High Availability Guide and Reference V8*, SC09-4831-01

For detailed information on the implementation and design of highly available IBM DB2 Universal Database environments on AIX see the following white papers

- IBM DB2 Universal Database Enterprise Edition for AIX and HACMP/ES
- IBM DB2 Universal Database Enterprise - Extended Edition for AIX and HACMP/ES

The paper can be found at the DB2 UDB and DB2 Connect™ Support Web site

`http://www.ibm.com/software/data/pubs/papers/`

High availability on the Windows operating system

High availability on the Windows operating system environment can be achieved using Microsoft Cluster Server (MSCS).

MSCS is a feature of Windows NT® Server, Windows 2000 Server, and Windows Server 2003 operating systems. It is the software that supports the connection of two servers (up to four servers in DataCenter Server) into a cluster for high availability and easier management of data and applications.

MSCS can also automatically detect and recover from server or application failures. It can be used to move server workloads to balance machine utilization and to provide for planned maintenance without downtime.

The following products in DB2 UDB family have support for MSCS:

- DB2 Universal Database Workgroup Server Edition
- DB2 Universal Database Enterprise Server Edition (DB2 ESE)
- DB2 Universal Database Connect Enterprise Edition (DB2 CEE)

For a more complete introduction to how DB2 UDB interacts with MSCS, see the DB2 UDB product documentation, *IBM DB2 UDB Data Recovery and High Availability Guide and Reference V8*, SC09-4831-01.

For detailed information on the implementation and design of highly available DB2 Universal Database environments on the Windows Operating System, see the following white papers, which are available from the DB2 UDB and DB2 Connect Support Web site:

http://www.ibm.com/software/data/pubs/papers/

- Implementing IBM DB2 Universal Database Enterprise - Extended Edition with Microsoft Cluster Server
- Implementing IBM DB2 Universal Database Enterprise Edition with Microsoft Cluster Server
- DB2 Universal Database for Windows: High Availability Support Using Microsoft Cluster Server - Overview

High availability in the Solaris operating system environment

High availability in the Solaris operating environment can be achieved through the use of DB2 UDB working with Sun Cluster, or Veritas Cluster Server (VCS).

Sun Cluster 3.0 provides high availability by enabling application failover. Each node is periodically monitored and the cluster software automatically relocates a cluster-aware application from a failed primary node to a designated secondary node. When a failover occurs, clients may experience a brief interruption in service and may have to reconnect to the server. However, they will not be aware of the physical server from which they are accessing the application and the data. By allowing other nodes in a cluster to automatically host workloads when the primary node fails, Sun Cluster 3.0 can significantly reduce downtime and increase productivity.

VERITAS Cluster Server can be used to eliminate both planned and unplanned downtime. It can facilitate server consolidation and effectively manage a wide range of applications in heterogeneous environments. VERITAS Cluster Server supports up to 32 node clusters in both storage area network (SAN) and traditional client/server environments, VERITAS Cluster Server can protect everything from a single critical database instance, to very large multi-application clusters in networked storage environments.

For a more complete introduction on high availability concepts for DB2 UDB on the Solaris operating system see the DB2 UDB product documentation, *IBM DB2 UDB Data Recovery and High Availability Guide and Reference V8*, SC09-4831-01.

For information about Sun Cluster, see the white paper entitled “DB2 Universal Database and High Availability on Sun Cluster 3.X”, which is available from the “DB2 UDB and DB2 Connect Online Support” Web site:

http://www.ibm.com/software/data/pubs/papers/

For information about VERITAS Cluster Server, see the white paper entitled “DB2 and High Availability on VERITAS Cluster Server”, which is available from the “IBM Support and downloads” Web site:

http://www.ibm.com/support/docview.wss?uid=swg21045033

High Availability on HP/UX

HP MC/ServiceGuard monitors the health of each server and quickly responds to failures in a way that minimizes or eliminates application downtime. MC/ServiceGuard is able to detect and respond automatically to failures in the following components:

- System processors
- System memory
- LAN media and adapters
- System processes
- Application processes

Application packages

With HP MC/ServiceGuard, application services and all the resources needed to support the application are bundled into special entities called application packages. These application packages are the basic units that are managed and moved within an enterprise cluster. Packages simplify the creation and management of HA services and provide outstanding levels of flexibility for workload balancing.

Fast detection of failure, fast restoration of applications

Within an enterprise cluster, HP MC/ServiceGuard monitors hardware and software components, detects failures, and responds by promptly allocating new resources to support mission-critical applications. The process of detecting the failure and restoring the application service is completely automated—no operator intervention is needed.

Recovery times for failures requiring the switch of an application to an alternate server will vary, depending on the software services being used by the application. For example, a database application that is using a logging facility would need to perform transaction rollbacks as part of the recovery process. The time needed to perform this transaction rollback would be part of the total time to recover the application. MC/ServiceGuard will detect the server failure, reconfigure the cluster, and begin executing the startup script for the application package on an alternate server in a short period of time.

For detailed information about HP MC/ServiceGuard, see the white paper which discusses IBM DB2 ESE V8.1 with HP MC/ServiceGuard High Availability Software, which is available from the “IBM DB2 Information Management Products for HP” Web site:

http://www-306.ibm.com/software/data/hp/

8

Database problem diagnostics

In this chapter we first provide an overview of the Problem Description / Problem Source Identification (PD/PSI) methodology, and discuss the various approaches for troubleshooting common problems based on problem types, such as performance, hang, deadlock, crash, and database corruption.

We also discuss a wide set of DB2 UDB and SAP diagnostic data required for problem analysis via a variety of available DB2 and SAP diagnostic tools.

In a later section, we describe a seamless technical support engagement model jointly developed by SAP and IBM.

At the end of this chapter, we provide a list of knowledge base resources which are available to assist you in using DB2 UDB and SAP products, as well as troubleshooting problems in the DB2 UDB and SAP environments.

8.1 Introduction to PD/PSI

Problem Description / Problem Source Identification (PD/PSI) is a methodology we commonly use for the problem analysis.

8.1.1 What is PD/PSI?

A good problem description is essential in order to understand what is happening on your system. Once you understand the problem and the circumstances that led up to it, you can try to identify the source of the problem, leading to a faster resolution.

Without a problem description, you will not know where to start investigating the cause of the problem. This step includes asking yourself such basic questions as these:

- What is the problem?
- Where is the problem happening?
- When does the problem happen?
- Under which conditions does the problem happen?
- Is the problem reproducible?

Answering these and other questions will lead to a good description to most problems, and is the best way to start down the path of problem resolution.

What is the problem?

When starting to describe a problem, the most obvious question is "*what is the problem?*". This may seem like a straightforward question; however it can be broken down into several other questions to create a more descriptive picture of the problem. These questions can include:

- Who or what operation is reporting the problem?
- What are the error codes, error messages, and reason codes if applicable?
- What are the actions that preceded the failed operation?
- What is the business impact?

Here are examples of some good and bad descriptions:

- I can't connect to the database. (Not good.)
- I'm getting an SQL1034C error code when I try to connect to the database AXB. Prior to that, since my database filesystem on the server is running out of space, I've been moving some of the files over to other filesystems. (Good.)

After you have an initial overview of the problem with an error code to begin with, you need to know what the error code or message means. DB2 UDB informational messages are always returned in the form of *CCCnnnnnS*. The *CCC* identifies the DB2 UDB component returning the message, the *nnnnn* is a four or five digit error code, and the *S* is a severity indicator. You can completely interpret any DB2 UDB return code message using the DB2 UDB command line by separating the **db2** command and the error code with a question mark. For example, you can identify the error message for the SQL1034C error code:

```
% db2 "? sql1034c"

SQL1034C The database is damaged.  The application has been
          disconnected from the database.  All applications
          processing the database have been stopped.
...
```

Most error messages contain further information than what is shown above, including an explanation of the error and potential user responses, which are not shown in the above example to conserve space. The DB2 UDB documentation, *Message Reference, Volumes 1 & 2* , lists all DB2 messages.

What is the environment and configuration?

Determining where the problem originates is not always easy, but is one of the most important steps in resolving a problem. Many layers of technology almost always exist between the reporting and failing components. Networks, disks, and drivers are only a few components to be considered when you are investigating problems.

- Is the problem platform specific, or common to multiple platforms?
- Is the current environment and configuration supported?
- Is the application running local on the database server or on a remote server?
- Is there a gateway involved?

These types of questions will help you isolate the problem layer, and are necessary to determine the problem source. Remember that just because one layer is reporting a problem, it does not always mean the root cause exists there.

Part of identifying where a problem is occurring is understanding the environment in which it exists. You should always take some time to completely describe the problem environment, including the operating system, its version, all corresponding software and versions, and hardware information. Confirm you are running within an environment that is a supported configuration, as many problems can be explained by discovering software levels that are not meant to run together, or have not been fully tested together.

When does the problem happen?

Developing a detailed time line of events leading up to a failure is another necessary step in problem analysis, especially for those cases that are one-time occurrences. You can most easily do this by working backwards - start at the time an error was reported (as exact as possible, even down to milliseconds), and work backwards through available logs and information. Usually you only have to look as far as the first suspicious event that you find in an diagnostic log, however this is not always easy to do and will only come with practice. Knowing when to stop is especially difficult when there are multiple layers of technology each with its own diagnostic information.

- Does the problem only happen at a certain time of day or night?
- How often does this happen?
- What sequence of events lead up to the time the problem is reported?
- Does the problem happen after an environment change such as upgrading existing or installing new software or hardware?

Responding to questions like this will help you create a detailed time line of events, and will provide you with a frame of reference in which to investigate.

Under which conditions does the problem happen?

Knowing what else is running at the time of a problem is important for any complete description. If a problem occurs in a certain environment or under certain conditions, that can be a key indicator of the problem cause.

- Does the problem always occur when performing the same task?
- Does a certain sequence of events need to occur for the problem to surface?
- Do other application fail at the same time?

Answering these types of questions will help you explain the environment in which the problem occurs, and correlate any dependencies. Remember that just because multiple problems may have occurred around the same time, it does not necessarily mean that they are always related.

Is the problem reproducible?

From a problem description and investigation standpoint, the “ideal” problem is one that is reproducible. With reproducible problems you almost always have a larger set of tools or procedures available to use to help your investigation. Consequently reproducible problems are usually easier to debug and solve. However, reproducible problems can have a disadvantage: if the problem is of significant business impact, you don’t want it recurring. If possible, recreating the problem in a test or development environment is often preferable in this case.

- Can the problem be recreated on a test machine?
- Are multiple users or applications encountering the same type of problem?

- Can the problem be recreated by running a single command, a set of commands, or a particular application, or a standalone application?
- Can the problem be recreated by entering the equivalent command/query from a DB2 command line?

Recreating a single incident problem in a test or development environment is often preferable, as there is usually much more flexibility and control when investigating.

Conclusion

Describing a problem accurately and completely may be easy for some problems, but very difficult for others. (The difficulty usually increases as the environmental complexity increases). However, the questions that you need to ask are usually the same: who, what, where, when, and how.

Many of the questions and problems you have can be cleared or resolved by searching and reviewing the information from knowledge resources listed in 8.4, “Problem research resources” on page 554.

As a last resource, you can consider to engage the technical support from the SAP or IBM DB2 support centers. The detailed engagement model is discussed in 8.3, “Support Integration of SAP and DB2” on page 552.

8.1.2 Problem types

The approach to debug problems differs by problem type. Here we address several common problem types in SAP and DB2 environments:

- Performance
- Hangs
- Deadlock
- Crash
- Database or data corruption

Performance

Performance problems cover a wide range of scenarios:

- Identifiable query performing slower than expected
- Workload or batch job not completing as soon as expected, reduction in transaction rate or throughput
- Overall system slowdown
- Suspected bottleneck in some type of system resource such as CPU, memory, and I/O

- Query or workload consuming more resource than expected or available
- Comparison is being made between one system and another

There are some subtleties in the scenarios depicted above. For problem diagnosis purposes, it is important to clarify whether something is not meeting expectations or is exceeding resource capacity. Sometimes it is both.

To many people, troubleshooting performance problems is like throwing darts in the dark. We'd like to introduce you a more thoughtful and enlightened strategy via a structured and methodical "decision tree" approach shown in Figure 8-1.

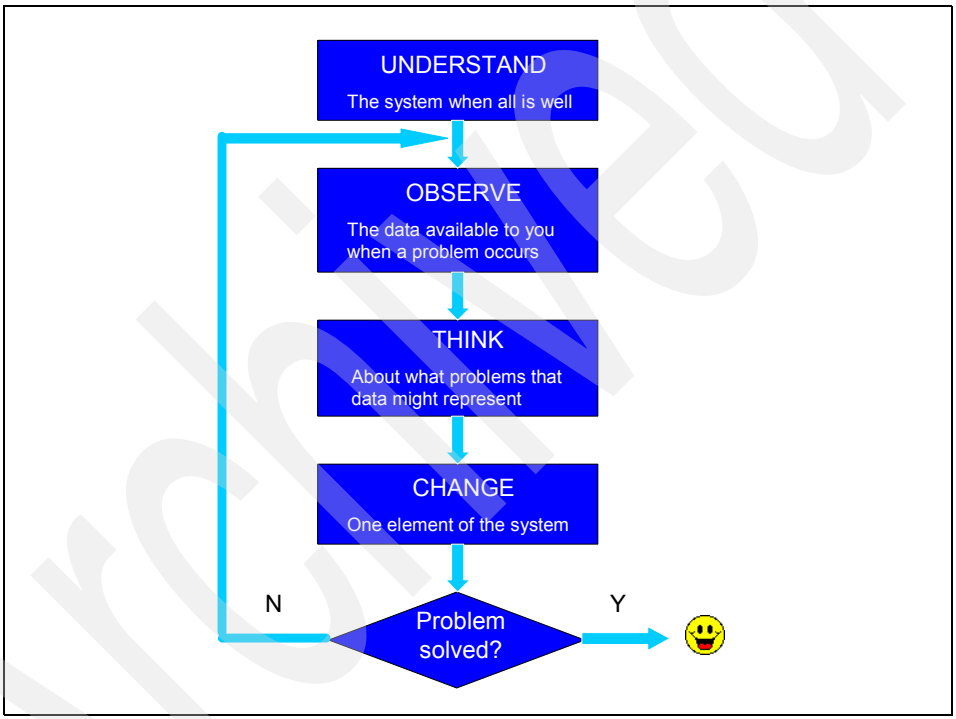


Figure 8-1 "Decision Tree" approach for troubleshooting performance problem

It is important to be prepared and understand how the system is supposed to work. Properly diagnosing a performance problem will require data on system behavior. We need to have a baseline to compare that with. Having relevant baseline data available makes problem diagnosis easier and faster, and also is useful for trend analysis as the system grows.

Here is the common performance data you need to look at:

- Configuration data:
  - Database and database manager configuration parameters
  - DB2 registry variables
  - Schema definition with `db2look`
  - Disk configuration
- Runtime data collected at both average and peak times:
  - Application throughput/response time
  - DB2 snapshots (all monitoring switches enabled - `bufferpool`, `locks`, `sort`, `statement`, `table`, `UOW`, and `timestamp`)
  - Event monitor data (`statement` and `deadlock event monitors`)
  - Query access plans via `db2exfmt` if it is query slow-down
  - Operating system data (`vmstat`, `iostat`, `sar`, `perfmon`, `truss`, `strace`, ...)

When a performance problem has developed, the most basic question to ask is whether anything has been changed since when performance met expectations:

- New database applications?
- Other non-database loads on the system?
- More users?
- More data?
- Hardware configuration changes?
- Software configuration changes? that is, DB/DBM configuration, registry variables, schema changes, etc.

If you can identify a change at this stage, you should then consider whether it can or should be undone. If this is something you have to adapt to, you need to follow the decision tree approach to understand what type of bottleneck you are dealing with, to rule out many possibilities. The bottlenecks refer to:

- System resource bottlenecks: disk, CPU, memory, or network.
- Non-resource bottlenecks (“lazy system”): locking, or external factors

SAP provides a series of monitoring tools which allow you to identify the bottleneck when performance problem surfaces:

- Monitors for technical analysis:
  - ST06 - operating system monitor.
    Monitor the load on CPU and RAM.
  - ST04 - database monitor.
    Check load on the database buffers, locks, and other wait situations; read and write accesses to hard disks; monitor SQL statements.

  - ST02 - SAP memory monitor.
    Monitor load on the SAP buffer and other memory areas, and SAP work processes.
  - SM50 - work process overview.
    Monitor load on the SAP work processes.
- Monitors for application analysis:
  - ST03/ST03N/ST03G - workload monitor.
    Overview of load distribution in the SAP system. In an application analysis, transactions, programs, and users that place a heavy load on the system, can be identified and analyzed.
  - ST07/ST14 - application monitors.
    Monitor the use of resources according to SAP modules.
  - ST05 - SQL Trace and SE30 - ABAP trace.
    Trace functions for a detailed analysis of ABAP program.
- CCMS alert monitor (RZ20) - continuous monitoring
  - Complete and detailed monitoring via performance indicators of SAP software components, servers, databases, and 3rd party components.
  - Status indicators (alerts) if threshold values are exceeded or not meet.
  - Counters, such as average response times, throughput statistics, degree of process workload or the fill level of memory areas.
  - Text information, such as error messages.

Once the monitoring data available to you when a performance problem occurs, you can then think about what problems that data might represent as follows:

Note: We only cover the problem scenarios from the DB2 UDB perspective in each of the following problem categories.

Disk bottleneck

After a disk bottleneck is identified, the further problem search approach differs by database objects identified with high I/O activities:

- Data table space:

  For high reads on a data table space, you can check whether you are getting an unwanted table scan, which can be confirmed when *rows read* is much greater than *number of executions* shown in the snapshot or the table scan lists in the query access plan generated by **`db2exfmt`** etc. Then you should consider whether the table statistics are out of date, or whether the table is sufficiently indexed. If a table scan is unavoidable, then check whether it is failing to be prefetched.

In a table space snapshot, compare *asynchronous pool data page reads* with *buffer pool data physical reads*. Ideally, almost all table scan physical reads will be asynchronous. You can possibly increase NUM_IOSERVERS database parameter. You can also potentially use materialized query table to eliminate aggregate re-calculation, or multi-dimensional clustering to reduce scan size.

For high writes on a data table space, most considerations are the same as those for reads. Some additional considerations are whether LONG VARCHARS or LOBs are present in the "write-hot" statement or table. SMS or DMS file containers my help in such case by taking advantage of file system cache for LOB table spaces. Also, check whether DB2 is cleaning too aggressively, which can potentially cause cleaning and recleaning the same page as updates are applied to it. It is indicated by an "excessive" dirty page threshold and LSN gap triggers. You can then consider lowering the values of SOFTMAX and/or CHNGPGS_THRES database parameters.

- Index table space:

  For high I/O activities with index table spaces, the problem identification approach is much in common with data table space approach, but more difficult to detect and affect. Your DB2 Configuration Advisor may be able to help in such a case.

- Temporary table space:

  The I/O activities with temporary table spaces are typically driven by spilled sorts and intermediate results. When the snapshot shows many sorts spilling to disk via the snapshot indicators *sort overflows* and *sort time statistics,* it means that SORTHEAP may be too low, or SHEAPTHRES is improperly set.

  Sometimes sorts and intermediate result sets also may be the result of poor query access plans, which can be verified via the sort and runtime statistics for individual statements shown in the snapshot. The table statistics out-of-date may be one possibility. Also, additional indexes recommended by the DB2 Configuration Advisor may reduce the problem.

- Logs:

  The placement of logs can be very performance sensitive, especially in an OLTP environment. It is recommended to place logs on your best hardware, such as dedicated and fast disks, RAID parallelization with small stripe size, and fast controller with writing caching.

CPU bottleneck

CPU bottleneck is often shown by high CPU system time or user time. For high CPU user time, you can leverage DB2 application and dynamic SQL snapshots and statement event monitor to track down the CPU consumers on the database server via the indicators of *CPU by application*, *CPU by dynamic SQL statement*, and *CPU by static SQL statement* respectively.

Several common scenarios associated with high CPU user time usage are:

- High CPU selects with high rows read but low physical reads. Frequent in-memory table scans can consume significant CPU. The DB2 Configuration Advisor may be able to help in this case.
- Repeated dynamic SQL statements use literals instead of parameter markers. If at all possible, use parameter markers to avoid recompiling cost.
- Dynamic SQL statements being re-prepared unnecessarily, that is, dynamic SQL snapshot shows many compilations for some statements. If there are package cache inserts occurring, consider increasing the database package cache size. The best practice for repeated dynamic SQL is to prepare once, save the statement handle, and re-execute with new data.
- Connections are apparently short-lived. The snapshots show a small number of commits even though the system is quite active, and connect time is always very recent. You need to avoid frequent connects or disconnects.
- A subset of CPUs are saturated. For example, your four-way system can appear to be only 25% busy, and still be stuck if one of the four CPUs is 100% used. You need to have the workload to be parallelized to use more CPUs.
- A utility is saturating the system. Many DB2 utilities are highly parallel and designed to exploit the system's resources. There are mechanisms available to throttle DB2 UDB utilities and free up resources for applications, such as the `UTIL_IMPACT_PRIORITY` option on **`RUNSTATS`** and **`BACKUP`**, and the `CPU_PARALLELISM` option on **`LOAD`**.

The high CPU system time can be caused by process scheduling and management. Here are some possible DB2 UDB related problem scenarios associated with such cases:

- The system has a high number of context switches, such as the *cs* column in **`vmstat`** greater than 75K or 100K per second when there is very high number of connections, or short transaction with very frequent commits. In such case enabling connection concentrator or lengthening transactions (if possible) may help.
- DB2 UDB processes are coming and going frequently, such as DB2 UDB agents or subagents frequently appearing and disappearing in the `ps` output or Windows task manager. Frequent process or thread creation is very expensive. You can then possibly increase `NUM_POOL_AGENTS` closer to `MAX_AGENTS`.

Memory bottleneck

The high swap activity shown in **`vmstat`** or performance monitor indicates that system memory is over-allocated. You should consider reducing the memory consumption by tuning down some of the major DB2 memory consumers with caution, such as buffer pools and sort heap, etc.

Non-resource bottleneck ("lazy system")

"Lazy system" normally refers to the system which is running slow but with no obvious system resource stress symptoms. It is generally much more difficult to find and solve the performance problems on such systems. The common DB2 culprits causing lazy systems are:

- Locking problems:

  The typical examples are lock escalation, lock contention, or deadlocks.

- Agent I/O:

  DB2 prefetchers and page cleaners are much more efficient for I/O than DB2 agents.

- Application issues:

  Sometimes the application is driving the database "hard enough". The application snapshot shows that many or most DB2 agents are waiting for work (status "UOW waiting"), and the event monitor shows that more time is being spent on the application side than when the system was "healthy". You need to examine the application and client side for bottlenecks, and then possibly increase application parallelism to have more connections and more work in parallel.

There are many possibilities which can result in a performance problem. The decision tree approach we introduced earlier lets us rule out many possibilities at each stage, narrowing down the problem very quickly. Combining this with some familiarity with the stem being analyzed and the basics of SAP, DB2 and operating system monitor tools, you will have a very efficient and effective way to tackle database performance problems.

Hangs

For problem determination purposes, in many cases hangs can be lumped together with performance because many investigative strategies apply to both. In addition, it may not be possible at first to define the problem as a hang versus a performance problem. To a user waiting for a response, a long-running job can look like a hang even if in fact much activity can be taking place on behalf of the application on the database server. There can also be a significant buildup of activity during a severe system slowdown such that all or most commands appear to hang on a system.

In addition to characterizing the problem correctly in terms of where the symptom is observed (query/application/system resource) and what is wrong with it (slowness or too much resource used), you require many other pieces of information to put the problem in context. The following questions serve to quickly determine the best place to start looking for a potential cause.

- *When did the problem first occur?*

  If the problem has been occurring for some time, and if a database monitor schedule has been implemented, you can use historical data to find differences. This will allow you to focus on changes in system behavior and then focus on why these changes were introduced. It is also important to consider whether any recent changes occurred, such as hardware or software upgrades, a new application rollout, additional users, etc.

- *Does the hanging situation appear to be system-wide or isolated to DB2 and its applications*?

  System-wide hanging problems suggest an issue outside of DB2. It is something at the operating system level needs to be addressed.

- *If isolated to one application, is there a particular query that appear to be problematic?*

  If one application is problematic, then you can further examine if users are reporting a query or set of queries that are experiencing a slowdown. You might be able to isolate the issue to one application and a potential group of queries.

- *If further isolated down to DB2, is it instance level, database-wide, a specific DB2 utility, or a particular query hanging?*

  It is possible that after your initial review, you identify that there is a strong possibility that DB2 is the source of the hang issues. In such case, you can first confirm whether basic DB2 commands (that is, commands that do not actually run SQL queries) hang in the instance. If all basic DB2 commands do not response in a reasonable time manner, it can be considered as an instance-level hang situation. You can use the similar approach to further identify whether it is a database-wide, a specific DB2 utility, or a particular query hanging.

The hang problem normally takes long time to debug, and also requires extensive skills across the entire stack of the operating system and the application products (that is, SAP, and DB2 product knowledge). So after you have done the initial review about the hang situation with the answers to the above questions, you can then consider to engage SAP & DB2 support center for further assistance. The technical support analyst will be able to provide you further instructions on collecting additional diagnostic data to identify the root cause to your hang problem.

Deadlock

With multiple applications working with data from the database, there are opportunities for a deadlock to occur between two or more applications.

A lock-wait situation is created if one application tries to set a lock on a database object where an incompatible lock is already owned by another application. In this case the application has to wait for the other application to release its lock. Situations may occur where applications are waiting in a cyclic chain for each other (for example,: application 1 waiting for application 2 waiting for application 1). Those situations are called *deadlocks*.

Deadlock situations cannot be resolved by the applications itself because all of them are caught in a lock-wait situation within a database call. A database deadlock detector process is required to break up cyclic lock chains and allow application processing to continue. As its name suggests, the deadlock detector monitors the information about agents waiting on locks. If a deadlock is detected, the deadlock detector arbitrarily selects one of the applications, its locks are released and the data required by other waiting applications is made available for use. The waiting applications can then access the data required to complete their transactions. The deadlock detector process runs periodically. Its frequency is controlled by the DB2 UDB database configuration parameter `DLCHKTIME` (Time interval for checking deadlock). The default value is 300000 ms (5 minutes). This means that a deadlock can exist between 0 and 5 minutes from the time it occurs.

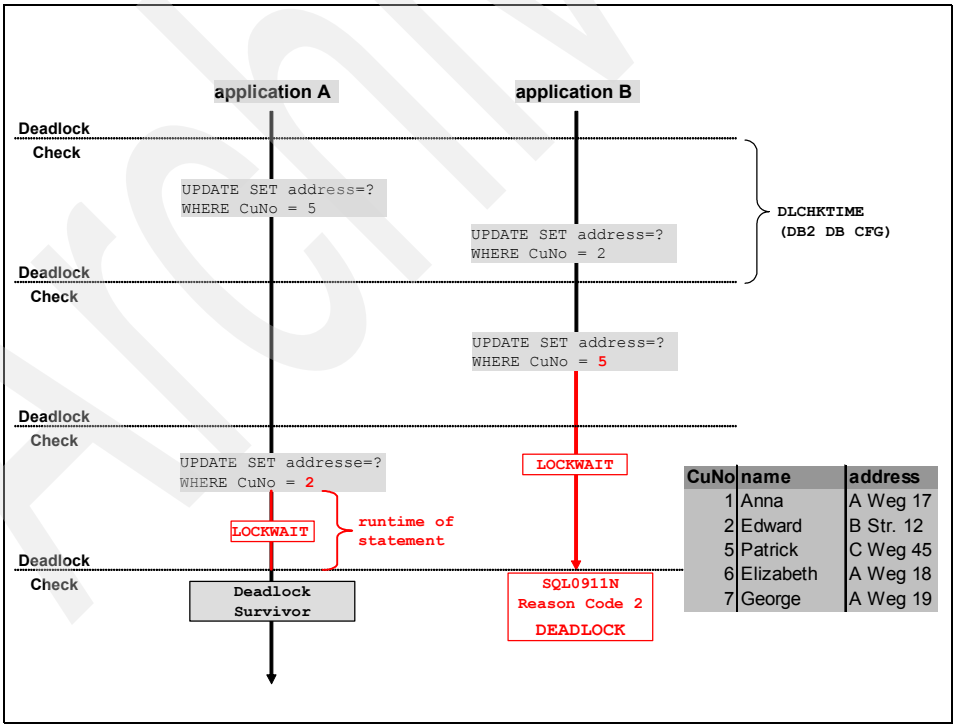


Figure 8-2 A 'classical' deadlock scenario

In the preceding example shown in Figure 8-2, Application A is holding a row lock with `CuNo = 5` and Application B is holding a row lock with `CuNo = 2`. Then Application B attempts to update the row with `CuNo=5` but at his point Application A already has a row lock on this row. So, Application B has to go into lock wait and wait until Application A finishes updating the row with `CuNo=5` before it can perform the update.

On the other side, Application A attempts to update `CuNo=2` now; but since Application B has a row lock on this row, Application A has to wait till Application B finishes updating the row with `CuNo=2`; not knowing that Application B is at the same time waiting for Application A to finish its update on the row with `CuNo=5`. At this stage, both Application A and Application B are in a lockwait state waiting for each other to release the row locks; and neither of them can proceed further with their transaction: A deadlock is formed.

In an SAP environment, when you hit a deadlock, the “victimized” application causes an ABAP short dump. In the transaction *SM21: System Log*, you may get an error message similar to what is shown in Figure 8-3.

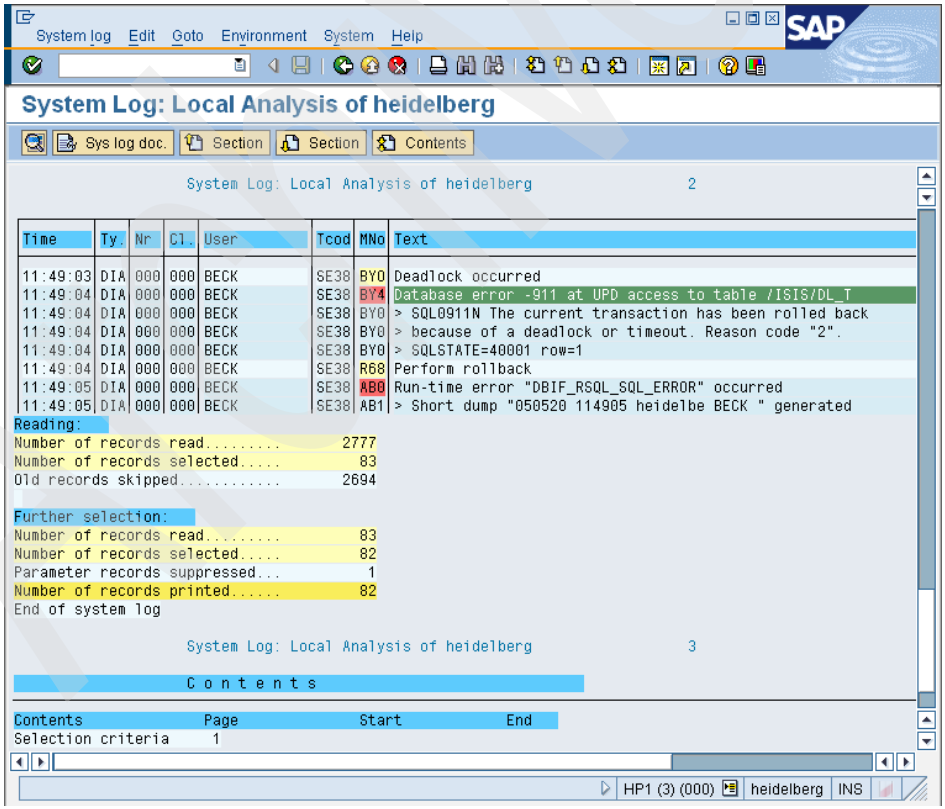


Figure 8-3 SM21 deadlock error message: SQL0911N(-911) rc=2

DB2 UDB SQL0911N with reason code 2 means that the transaction is rolled back due to deadlock. Normally, you need to restart the cancelled application to redo the transaction that has been rolled back.

Deadlocks can be caused by several different reasons. Here are some of them:

- Lock escalations:

  Lock escalation occurs when the total number of locks held by an application reaches the maximum amount of lock list space available to the application, or the lock list space consumed by all applications is approaching the total lock list space. The amount of lock list space available is determined by the MAXLOCKS and LOCKLIST configuration parameters. When lock escalation happens, row locks are escalated into a table lock. In db2diag.log file, you find the entries about lock escalations. A database snapshot show the number of lock escalations that occurred in your system. To resolve the situation, you can consider to increase the LOCKLIST configuration parameter value.

- Improperly written applications:

  One of the causes of deadlock from the application side is that application does not commit transactions often enough, it holds more locks than it needs. These long running transactions reduce the level of concurrency and the chance of hitting deadlocks rises. In the preceding example as shown in Figure 8-2, a COMMIT after the first UPDATE statement in either (or both) applications will prevent this deadlock from happening (provided that the application logic allows this change).

  A deadlock can also occur if two applications access database rows in different order. Let's consider the following example:

  a. Application 1 updates row A and puts a row lock on row A.
  b. Application 2 update row B and puts a row lock on row B.
  c. Application 1 then attempts to update row B but row B is locked by instance 2 at the moment. Application 1 goes into lock wait.
  d. Application 2 attempts to update row A but row A is locked by application 1 at the moment. application 2 goes into lock wait. A deadlock occurs.

  To avoid this kind of deadlock, applications need in the same order. Imagine that each application uses an internal working table of rows in one table that are going to be updated. Deadlocks can be avoided by sorting each working table before processing it.

- Inappropriate DB2 profile parameter setting:

  The other kind of deadlock is more subtle. In the following example, we are going to show how a missing DB2 profile parameter can lead to a deadlock situation.

For SAP systems, the DB2 registry variable DB2_WORKLOAD=SAP is mandatory with DB2 V8.2.2 and higher. Under the covers, this will activate other registry settings like:

- DB2_EVALUNCOMMITTED=YES_DEFERISCANFETCH
- DB2_SKIPDELETED=YES
- DB2_SKIPINSERTED=YES

These allow a greater level of concurrency.

In the following example, the system does not have the DB2 profile parameter DB2_EVALUNCOMMITTED activated. If DB2_EVALUNCOMMIT=YES_DEFERISCANFETCH is set, predicates in an SQL statement are evaluated in an uncommitted way. Locks are only acquired if those predicates qualify. This avoids lock-wait situations caused by locking of non-qualifying rows. Without this DB2 profile parameter, the following scenario could happen (see Figure 8-4):

a. Application A issues an UPDATE statement against the CUSTOMER table where CuNo = 2. After that, Application A holds a lock on the row that qualifies the predicate CuNo = 2 in the CUSTOMER table.

b. Application B issues an UPDATE statement against the ITEM table. Application B now holds a lock on the row where ItemNo = 101 for the ITEM table.

c. Application B issues an UPDATE statement against the CUSTOMER table where CuNo = 5. Let's assume that the DB2 optimizer decides to use a table scan to access the CUSTOMER table. Without having DB2_EVALUNCOMMITTED active, DB2 attempts to lock temporarily every rows in the CUSTOMER table that it tries to evaluate (it releases the lock after the evaluation if the row does not qualify the predicate CuNo = 5). In this case, when the table scan reaches the row where CuNo = 2, it goes into lock wait because at the moment this row is locked by Application A in step a.

d. Application A issues an UPDATE statement against the ITEM table where ItemNo = 101. This row is already locked by Application B. This completes the deadlock.

In step c, if we have DB2_EVALUNCOMMITTED=YES_DEFERISCANFETCH activated, the deadlock will not occur. Instead of waiting for Application A to release the lock, the table scan of Application B will read all rows in uncommitted mode and only acquire a lock if the predicate CuNo = 5 qualifies.

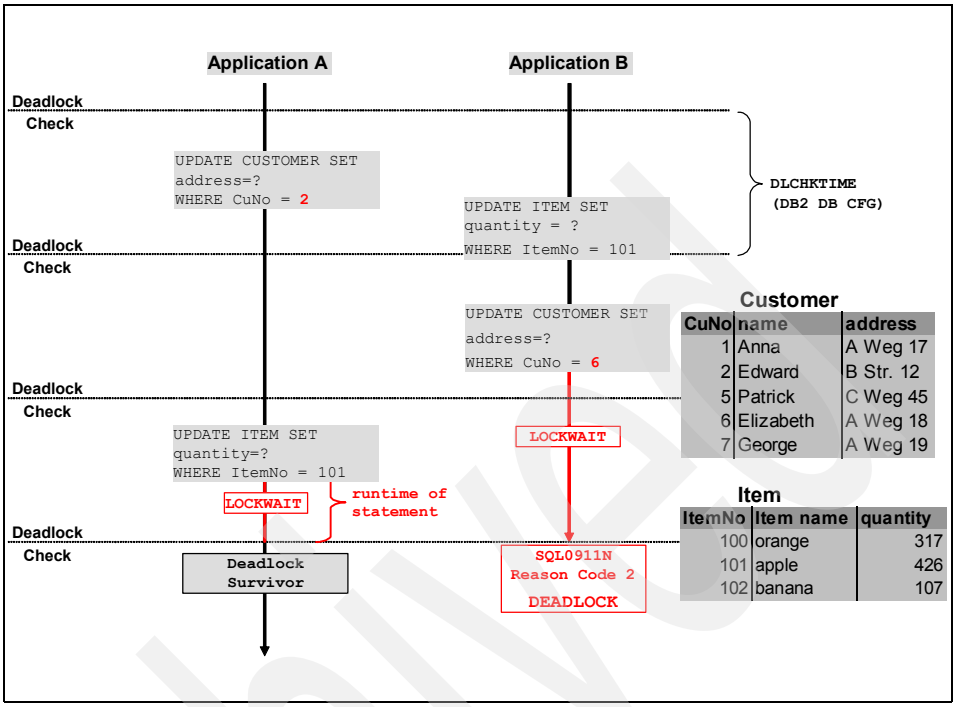


Figure 8-4 deadlock cause by missing DB2 profile parameter DB2_EVALUNCOMMITTED

There are several improvements in DB2 UDB V8.2.2 that allows greater level of concurrency which help avoiding deadlocks:

- Evaluate Uncommitted (the DB2 profile parameter: DB2_EVALUNCOMMITTED=YES_DEFERISCANFETCH)
- Skip Inserted (the DB2 profile parameter: DB2_SKIPINSERTED=YES)

Both DB2 profile parameters are automatically set by turning on DB2 workload optimization for SAP environment (DB2_WORKLOAD=SAP).

Crash

As a database administrator, when you receive the user report saying that there is a crash (or “abend” or “trap”), find out if it is just an application or GUI that is disappearing, or whether it is indeed the instance itself that is stopping.

In order to better understand the database manager crash, you should be aware of several basic terms:

- *signal* (UNIX and Linux platforms):

  A software interrupt. Signals are a mechanism by which a process can be notified that some even has occurred or some condition has been encountered that may alter its path of execution (including termination).

- *exception* (Windows systems):

  Similar to signals on UNIX and Linux. It is used to alert a program that something has gone wrong or something significant has occurred which it needs to react to.

- *signal / exception handler*:

  Code which will be executed when the process/thread receives a signal, or an exception is "raised". For the common case (that is, "serious" signals or exceptions), DB2 signal/exception handlers will dump diagnostic information (including DB2 trap files), after which the process will usually terminate.

- *SIGCHLD* (DB2 child signal handler):

  DB2 UDB processes have a child signal handler installed that will dump the child process's process ID in hex-decimal at Probe 20 of the function `sqloEDUSIGCHLDHandler()` and its exit status at Probe 30 of that function. The Probe information is recorded in the *db2diag.log file*.

- *trap file*:

  A file created by DB2 UDB which includes the name of the instance the process belongs to, the name of the process, the time the trap occurred, the signal/exception received, the registers, a stack traceback, etc. In "Trap files" on page 447 we have more detailed discussion on trap files.

- *stack traceback*:

  A listing of functions calls. The "trapping" function is the one at the stack top.

- *Dr. watson*:

  A Windows debugger which intercepts an exception received by a process for the purposes of logging an entry to the Dr. watson log file.

- *APAR:*

  IBM resolves defects discovered by customers in Authorized Program Analysis Reports (APARs). Seldom will you hear the term "Authorized Program Analysis Report", as the acronym is very common. An APAR is simply an externalized view of an IBM defect. APARs have unique identifiers. DB2 APARs are always specific to a particular version, but may affect multiple products in the DB2 family running on multiple platforms. The 8.4, "Problem research resources" on page 554 provides the URL to the online DB2 UDB APAR library.

In our context, trap means that the process/thread did something illegal (that is, the operating system cannot allow), and the process must terminate (after handling signal/exception). Examples of the external symptoms you may encounter are:

- Database connection receives an `SQL1224` error code
- A pop-up window on Windows with the error message of the exception C0000005 in db2syscs.exe
- The database manager's *diag_path* directory fills with trap files
- File system fills with large core dumps
- Sometimes trap can cause the symptom of a hang
- Database manager is no longer active

Instance crash incident does not necessarily result from a DB2 product defect. For example, when the instance crashes with signal #9, it is always caused either by a user manually (or through a script) killing a DB2 process or sometimes by the operating system (that is, when the system is running out of swap space. The operating system starts to kill processes when the swap space is running out.). Per the current DB2's design, when a DB2 instance is up and running, none of DB2 processes is allowed to be killed with signal #9. Otherwise, an instance crash is expected. Another typical example is that DB2 intentionally brings down the DB2 engine with the signal of `SIGABRT` if severe error is identified. The root problem should be investigated with the entries before the engine `abort()` in the db2diag.log file. The third example is if the trapping function of the "died" DB2 process is not a DB2 function, then typically this is not a DB2 problem. Instead the related product vendor who owns that trapping function code needs to be engaged.

When you exclude any of the above as the cause to your instance crash incident, you can start to think whether your instance crash is actually caused by a known DB2 defect. You would use the DB2 trapping function shown in the trap file along with the DB2 version information to search the online DB2 APAR library. If matched DB2 APAR is identified, you can then resolve your problem by either using the workaround provided or applying the related DB2 fixpack. However, if no matched APAR is found or if you are not certain about which DB2 APAR is the real cause, you should engage the SAP or IBM DB2 support center with the output of the *db2support* utility. The following basic invocation is usually sufficient for collecting most of the information required to debug a problem including instance crash:

```
db2support <output path> -d <database name> -c
```

A thorough review of the *db2support* utility is given in 8.2.1, "DB2 data collection and diagnostic tools" on page 443.

Database corruption

Normally, the first time you notice a database/data corruption problem is when you fail to connect to a certain database or fail to access a certain database object. Corruption is a very general term, as various factors could potentially cause corruptions to database subjects. Many of the database corruptions result from the physical disk problems. And it is generally difficult to identify what exactly caused the corruption in the first place. So here we focus on understanding possible database corruption scenarios and how to recover from them. Some typical examples are as follows:

- *Log control header file (SQLOGCTL.LFH) corruption:*

  The database log control header file (*SQLOGCTL.LFH*) is used by DB2 to govern the overall logging process and is located under the database directory <db_path>/<instance_name>/NODEXXXX/SQLYYYYY, where XXXX is the partition number and YYYYY is the database sequence number. When the SQLOGCTL.LFH file is missing or corrupted, the database connection command would fail with errors.

- *Transaction log file(s) corruption:*

  Crash recovery or rollforward processing of database log files will not be able to complete successfully when a bad log page is encountered. A bad log page can be that the structure of a log page is corrupted, or that a log page header gets overwritten or missing etc.

- *Tablespace file(s) (SQLSPCS.1 & SQLSPCS.2) corruption:*

  There are two table space files (*SQLSPCS.1* & *SQLSPCS.2*) which contain the information regarding table spaces and their containers. There is a single entry in the *SQLSPCS* file for each table space in the database. The files are in the database directory <db_path>/<instance_name>/NODEXXXX/SQLYYYYY, where XXXX is the partition number and YYYYY is the database sequence number. Those two files are identical to each other. If DB2 detects a problem with one of them it will use the other file to "fix" it. This is done by overwriting a bad table space entry with the same entry from the other file if entry corruption is encountered during database startup. If both files get corrupted in some way, you may not be able to connect to the database, or have problem to access tables in certain table space(s).

- *Data/Index objects corruption:*

  There are different types of corruptions which can occur within a data or index page. Any of the physical page corruptions will cause DB2 to mark the database as damaged, and stop all applications processing the database. DB2 UDB ships the *db2dart tool* with repair function enabled. The *db2dart* tool can be used to not only inspect the entire database or specified data/index object to find out the extent of the corruption, but also eliminate any or all of these errors. Besides such type of physical corruption with

corrupted data or index page, there is also a type of logical corruption, that is, data and index page mismatch. Two types of data and index mismatch could happen:

- The index key entry in the index points to a record which does not exist in the data page. It would cause the database to be marked bad when you attempt to select or update the record using an index scan.
- A record in the data page exists for which there is no corresponding index key entry in the index. It would cause the database to be marked bad when you update or attempt to delete the record.

The logical corruption of the data and index mismatch will not be detected by the *db2dart tool*.

No matter what type of database corruption it is, the best recovery option is to restore a known good backup and roll forward the log files to an appropriate point of time. If there is no available backup, depending on the type of the corruption, we can then attempt to "correct" the corruption by patching the log control header file or the table space control file, or fixing corrupted data/index page with the db2dart tool. Generally speaking, the patching or fixing method is not recommended and does come with some requirements. In 8.2.1, "DB2 data collection and diagnostic tools" on page 443, we provide a detailed introduction to the *db2dart* tool.

8.2 Tools and diagnostic data collection

Understanding what problem you have is the first step to begin your investigation. The knowledge of collecting the right diagnostic data via available tools is essential for helping you to further the problem determination effort.

8.2.1 DB2 data collection and diagnostic tools

This section introduces you to the basic set of DB2 diagnostic data and common troubleshooting tools.

DB2 data collection

Each database manager maintains a first failure data capture (FFDC) directory to keep the important diagnostic files generated by DB2. FFDC is a general term applied to the set of diagnostic information that DB2 captures automatically when errors occur. This information reduces the need to reproduce errors to get diagnostic information. Such diagnostic information is contained in a single location.

The location of the FFDC directory can be configured via the DIAGPATH database manager configuration parameter. The default value for DIAGPATH is a null string. To change the DIAGPATH value, use the following command:

```
db2 update dbm cfg using DIAGPATH <new_path>
```

By default, the DB2 FFDC information is placed in the following locations:

- On UNIX and Linux systems:

  $HOME/sqllib/db2dump

  Where $HOME is the home directory of the instance id.

- On Windows systems:
  - If the DB2INSTPROF environment variable is not set:

    <db2path>\<db2instance>

    where db2path is the path referenced in the DB2PATH environment variable, and DB2INSTANCE is the name of the instance id.
  - If the DB2INSTPROF environment variable is set:

    x:\db2instprof\db2instance

    Where x is the drive referenced in the DB2PATH environment variable. DB2INSTPROF is the instance profile directory, and DB2INSTANCE is the environment variable containing the instance ID.

The information captured by DB2 FFDC includes:

- Administration notification log
- db2diag.log file
- Trap files
- Dump files

We recommend that you archive the FFDC directory periodically to keep it from becoming too large.

Administration notification log and db2diag.log

When significant events occur, DB2 writes information to the administration notification log. The information is intended for use by database and system administrators. Many notification messages provide additional information to supplement the SQL error code that is provided. The type of event and the level of detail of the information gathered are determined by the NOTIFYLEVEL database manager configuration parameter. However, detailed diagnostic information is not written to this log, but to the db2diag.log file instead.

On UNIX platforms, the administration notification log is a text file called *<instance_name>.nfy*. On Windows, all administration notification messages are written to the event log and can be reviewed through the Windows Event Viewer. The errors can be written by DB2, the Health Monitor, the Capture and Apply programs, and user applications.

The NOTIFYLEVEL parameter specifies the type of administration notification messages that are written to the administration notification log. Valid values for the NOTIFYLEVEL parameter are:

- 0 - No administration notification messages captured
- 1 - Fatal or unrecoverable errors
- 2 - Immediate action required
- 3 - Important information, no immediate action required
- 4 - Informational messages

The administration notification log includes messages having values up to and including the value of the NOTIFYLEVEL parameter. For example, setting NOTIFYLEVEL to 3 will have the administration notification log to include messages applicable to levels 1, 2, and 3.

To check the current setting of NOTIFYLEVEL, issue:

```
db2 get dbm cfg
```

Look for the following parameter:

```
Notify Level                            (NOTIFYLEVEL) = 3
```

To alter the parameter, use the command:

```
db2 update dbm cfg using NOTIFYLEVEL X
```

where X is the desired notification level.

Example 8-1 shows an administration notification log entry.

Example 8-1 Administration notification log entry

```
2005-05-22-11.39.35.252967 [1] Instance:axb [2]   Node:000 [3]
PID:21637(db2agent (SAMPLE) 0) [4] TID:1 [5] Appid:*LOCAL.axb.050522163916 [6]
relation data serv [7] sqlrr_dump_ffdc [8] Probe:10  [9] Database:SAMPLE [10]

ADM00001C  A severe error has occurred.  Examine the administration notification
log and contact IBM Support if necessary. [11]
```

Example 8-1 shows that each notification entry may contain the following parts:

- Timestamp, noted by [1].

- Instance name, noted by [2].
- Partition number, noted by [3].
- Process ID and name, noted by [4].
- Thread ID, noted by [5].
- Application ID, noted by [6].
- Component Identifier, noted by [7].
- Function identifier, noted by [8].
- Unique error identifier (probe point within the function), noted by [9].
- Database name, noted by [10].
- Error description and/or error code, noted by [11].

More detailed diagnostic information about errors is recorded in the db2diag.log file. This information is used for problem determination and is intended for SAP or IBM DB2 support personnel.

The type of diagnostic errors recorded in the db2diag.log file is determined by the DIAGLEVEL database manager parameter. Valid values are:

- 0 - No diagnostic data captured
- 1 - Severe errors only
- 2 - All errors
- 3 - All errors and warnings
- 4 - All errors, warnings and informational messages

The default value 3 can be used as the recommended value for DIAGLEVEL. However, you should consider to use DIAGLEVEL 4 during the stage of initial installation/confirmation time, times of configuration changes, or times when you are experiencing errors. DIAGLEVEL 4 logs more information than lower levels. It will cause DB2 to run a little slower. So you need to balance out the extra data provided with the decreased response time to determine the best setting or the error source for your environment.

An example of the db2diag.log file entry is shown in Example 8-2.

Example 8-2 .db2diag.log entry

```
2005-05-22-11.39.36.224670-300 [1]  I205857C369  [2]     LEVEL: Severe [3]
PID     : 21637          [4]  TID  : 1     [5]     PROC : db2agent (idle) 0 [6]
INSTANCE: axb            [7]  NODE : 000   [8]     DB   : SAMPLE    [9]
APPHDL  : 0-7            [10] APPID: *LOCAL.axb.050522163916        [11]
FUNCTION: DB2 UDB, base sys utilities, sqleTermDbConnect, probe:110 [12]
RETCODE : ZRC=0xFFFFFBF6=-1034          .... [13]
```

Similar to the notification entry, each diagnostic entry contains various parts:

- [1]: Timestamp
- [2]: Record ID.
  It specifies the file offset at which the current message is being logged (that is, 205857) and the message length (that is, 369) for the platform where the DB2 diagnostic log was created.
- [3]: Diagnostic level associated with the message. Possible value is `Info`, `Warning`, `Error`, `Severe`, or `Event`.
- [4]: Process ID
- [5]: Thread ID
- [6]: Process name
- [7]: Instance name
- [8]: Partition number
- [9]: Database name
- [10]: Application handle.
  It aligns with that used in db2pd output and lock dump files. It consists of the coordinator partition number followed by the coordinator index number, separated by a dash.
- [11]: Application ID.
  It aligns with that used in the `db2 list applications` command on a DB2 UDB server, the `db2 list dcs applications` command on a DB2 UDB Connect gateway, the `db2 get snapshot for application` command, or the `db2pd -applications -db db_name` command.
- [12]: Product name (`DB2 UDB`), component name (`base sys utilities`), and function name (`sqleTermDbConnect`) as well as the probe point (`110`) within the function.
- [13]: Return code from the calling function.
  It consists of the message type (`ZRC`, that is, DB2 internal errors), the return code value (`0xFFFFFBF6=-1034`), and the corresponding error description if there is any.

Trap files

DB2 generates a trap file if its process cannot continue processing because of an illegal instruction, segmentation violation, or exception.

All signals or exceptions received by DB2 are recorded in the trap file. The trap file contains the function sequence that was running when the error occurred. This sequence is also referred to as “stack trackback”. The trapping function is at the stack top. The trap file also contains additional information when the signal or

exception was caught, such as the name of the instance, the name of the process, the time the trap occurred, and the registers etc.

On UNIX and Linux systems, the trap file is named *txxxxx.yyy*, where *xxxxx* is the process identifier (PID), and *yyy* is the partition number (000 on single partition database). For example, `t21331.005` is the trap file for the process with PID `21331` running on partition `5`.

On Windows systems, each trap file is named *DBXXXXYYYY.TRP*, where *xxxx* is the PID and *yyyy* is the thread ID. For example, `DB11201244.TRP` is the trap file for the process with PID `1120` and thread id `1244`.

On UNIX and Linux, trap files are in text format, and need not be formatted. However, on Windows, the trap file is in binary format, which contains a raw stack dump, a dump of the instruction stream, and currently loaded DLLs etc. You need the *db2xprt.exe* tool to convert it to readable text, and the tool can be found under the `SQLLIB/BIN` directory. The command syntax is:

```
>>-db2xprt--+----------+--+----+--+----+----infile--+---------+---><
            +-/p--path-+  '-/m-'  '-/n-'           '-outfile-'
             '-/v-------'
```

Where these are the parameters:

`/p path`	A semicolon (;) separated path that points to the location or locations where DB2 symbol files (*.dbg) are located.
`/v`	Display version information.
`/m`	Format a memory dump along with the rest of the trap file.
`/n`	Format data without regard to line number information.
`infile`	Specify the input file.
`outfile`	Specify the output file.

For example, when you are working on the same machine where the trap file (`DB11201244.TRP`) was generated, you can format it as follows:

```
db2xprt DB11201244.TRP DB11201244.FMT
```

If you are formatting the trap file on a machine other than where it was generated, make sure that either the working machine is running the same level of the DB2 code, or you format the trap file with the DB2 symbol files (`*.dbg`) with the correct level of code as follows:

```
db2xprt -p d:\db2dbg_dir DB11201244.TRP DB11201244.FMT
```

Where `d:\db2dbg_dir` contains the right level of DB2 symbol files which you can retrieve from the DB2 fixpack file.

Signal/exception number and the trapping function are two key elements when comes to the trap file analysis. Examples of signal/exception information contained in the trap files on AIX and Windows are as follows.

In Example 8-3, `t453942.000` captured the signal #11 with instance `db2inst1`. The related section within `t453942.000` looks like the one shown in Example 8-3.

Example 8-3 Signal #11 trap on AIX

```
db2 build information: DB2 v8.1.0.64 s040812 SQL08020
timestamp: 2005-05-17-11.02.29.471745
instance name: db2inst1.000
EDU name     : db2agent (SAMPLE) 0
Signal #11
uname: S:AIX R:2 V:5 M:0020EC5D4C00 N:umecir15
thread id : 1 (0x1)
...
```

`DB11201244.TRP` captured the exception C0000005. The related section within `DB11201244.TRP` looks Example 8-4.

Example 8-4 Exception C0000005 trap on Windows

```
...
The following information is for pid <1120> tid <1244>

Exception C0000005 Occurred
Exception Address = 6C771022
Invalid linear address 0000005C
...
```

Here are the common signals/exceptions that you may see from the trap files:

- **Invalid memory accesses**

  It is indicated by exception `C000005` on Windows platforms, and `Signal #11` (`SIGSEGV`) on UNIX. It occurs when memory is not mapped (doesn't exist) or when a process does not have permission to perform the desired action (for example, write to protected memory).

- **Illegal instructions**

  It is indicated by exception `C000001C` on Windows platforms, and `Signal #4` (`SIGILL`) on UNIX. It is caused by a bad (often NULL) function pointer, stack overwrites (overwriting the "saved" instruction address).

- **Stack overflow exception**

  It is indicated by exception `C00000fd` on Windows platforms. It occurs when the process stack definition is too small or the bad user programming has excessive use of stack or unrestricted recursion. Such software interrupt is rarely seen on UNIX.

- **SIGABRT**

  On UNIX platforms, sometimes DB2 self-triggers `SIGABRT` (`Signal #6`) via an *abort()* call to intentionally bring down the DB2 instance if severe error is identified. Most `SIGABRT` (`Signal #6`) traps on UNIX need not to be investigated because the stack traceback of the process is DB2 standard without additional useful information. The *abort()* itself and the resulting trap files and error entries in the db2diag.log file are not a direct cause to the problem and should not be analyzed. Instead the root problem should be investigated with the entries before the engine *abort()* in the `db2diag.log` file.

- **SIGKILL**

  The `Signal #9` can be caused by someone manually (or through a script) killing a DB2 process, in which case, no trap file is generated for the killed process except that its parent processes will generate `SIGABRT` trap files while shutting down the entire DB2 instance.

- **Diagnostic trap**

  There are cases where DB2 will force the stack traceback of a process to be dumped into a trap file for diagnostic purposes by sending the DB2 process a diagnostic signal. The diagnostic signal will not terminate the process, that is, it doesn't affect the instance running. The diagnostic signal varies from platform to platform:

  - `Signal #36` on AIX
  - `Signal #21` on Solaris
  - `Signal #29` on HP-UX
  - `Signal #23` on Linux
  - Windows will have a trap file with an entry stating that no exception record is present.

Once you identify the signal or exception, the process is received, you are then required to find out the trapping function. The trapping function is normally at the stack top. Examples of the trapping function and the stack traceback of the "died" process shown in the trap files on AIX and Windows are as follows.

`t453942.000` shows the trapping function, and the top part of the stack traceback is shown in Example 8-5.

Example 8-5 Stack traceback of the trap on AIX

```
*** Start stack traceback ***

0xD2B93D18 bcopy + 0x138
0xD3135658 sqlno_copy_setN__FP13sqlno_globalsP9sqlno_sdbP10sqlno_apcb9sqlno_set
P9sqlno_set_43_4 + 0x24
0xD31332C8 sqlno_union_setsN__FP13sqlno_globalsP9sqlno_sdbP10sqlno_apcb9sqlno_s
etT4P9sqlno_set + 0x70
0xD35FD9FC sqlno_prop_union_adjusts__FP13sqlno_globalsP10sqlno_apcbP9sqlno_qtb9
sqlno_setT4P9sqlno_set + 0x350
0xD35FECA4 sqlno_prop_nljn__FP13sqlno_globalsP21sqlno_plan_propertiesP19sqlno_p
lan_functionP18sqlno_join_context + 0xA80
0xD31564A8 NLJN__FP13sqlno_globalsP13sqlno_contextP19sqlno_plan_operatorT39sqln
o_setT5UlP10sqlno_part13SQLNN_BOOLEANPP19sqlno_plan_operator + 0x134
...
```

DB11201244.TRP shows the trapping function (sqlzint3), and the top part of the stack traceback is shown in Example 8-6.

Example 8-6 Stack traceback of the trap on Windows

```
...
++++++++++++++++++++++++++++++++++++++++++++
Failing instruction at 6C771022 offset: 00000002 in <_sqlzint3>
<NT/sqlzint3.asm:55>
++++++++++++++++++++++++++++++++++++++++++++
...
Stack calling chain:
--EBP -----EIP-----ARGS--
00D7FB34 6C771022 00000000 00000001 03000002 00000001 offset: 00000002 in
<_sqlzint3> <NT/sqlzint3.asm:55>
00D7FB64 0040AF0E 00000000 00000000 505C3A47 52474F52
00D7FD98 00402949 00000000 8003A0E5 00000000 00000000
00D7FDB0 00401BCE 00C23A40 00000000 6E267400 00C38040
00D7FDF8 6D14808E 00401B60 00402ED0 00000000 00C3864C offset: 00000092 in
<?sqlra_event_user_priv@@YAXPAUsqlrr_cb@@PAUsqlr_rpc_apm_request@@@Z>
<sqlra_events.C:802>
...
```

After you extract the key elements from the trap file, you can follow the analysis methodology provided for crash problems in “Crash” on page 439 to reach a resolution.

Dump files

Dump files are created when an error occurs for which there is additional information that would be useful in diagnosing a problem (such as internal control blocks). Every data item written to the dump files has a timestamp associated

with it to help with problem determination. Dump files are in binary format, and are intended for IBM DB2 support personnel.

When a dump file is created or appended, an entry is made in the db2diag.log indicating the time and the type of data written. An example is:

```
2005-05-22-20.59.14.596371-300 I407843C173        LEVEL: Severe
PID:15358 TID:1 NODE:000 Title: RDS INVOCATION CB
Dump File:/home/axb//sqllib/db2dump/153581.000
```

db2support - Data collection utility for hotline support

The *db2support* utility as a problem analysis and environment collection tool is designed to automatically collect both DB2 UDB and operating system diagnostic information about either a client or server machine, and place all diagnostic information into a compressed file containing either the file name specified by the user, or the default file name, *db2support.zip*.

A good practice is to use `db2support` to help convey information when you engage SAP or IBM support center for DB2 problems. `db2support` collects a comprehensive set of operating system and DB2 diagnostic information a support representative requires for the initial problem analysis. Since some of the information (that is, operating system or DB2 snapshot type of information) collected by `db2support` is transient, it is recommended to run `db2support` while the system is experiencing the problem.

Using `db2support` avoids back-and-forth information request interaction between you and the support personnel, which can shorten the problem determination process. Also, it alleviates the burden on you to understand and know which commands to run or what files to collect.

You can execute **`db2support -h`** to get the command syntax. For syntax details, please refer to the DB2 UDB documentation, *Command Reference V8*, SC09-4828-01. The type of information that `db2support` captures depends on the way the command is invoked, whether or not the database manager has been started, and whether it is possible to connect to the database.

The `db2support` archive contains the following output files:

- Flat ASCII file output (database related information)
- `db2support.html` (database related information in HTML)
- `userResponse.xml` (with `-q` option)
- db2diag.log
- `db2supp_system.zip`:

  Which contains:

  - Flat ASCII file output (system related information)
  - `detailed_system_info.html` (with `-m` option)

Depending on the options specified, db2support may collect the following additional command output in db2support.zip:

- All trap files, dump files, lock list files in the DIAGPATH directory
- Database configuration (with -d option)
- Database manager configuration
- db2level output
- CLI configuration
- Action log files (with -d -l options)
- Database/lock snapshot output
- Admin node, DCS and database directory output
- db2set -all output
- List command options output

Also, depending on the options specified, db2support will collect the following physical files within db2supp_system.zip:

- System and environment related information
- Common OS commands that gather network, disk, hardware, software, kernel, CPU, and memory related information
- Output to either ASCII files or detailed_system_info.html (with -m option)
- Log file header (with -d option)
- Buffer pool and table space files (with -d option)
- Recovery history file
- Report log output (containing a record of all system files that were archived successfully to db2supp_system.zip)
- *db2cli.ini*

A common invocation of db2support which is usually sufficient for collecting most of the information required to debug a problem is:

db2support <output path> **-d** <database name> **-c**

<output path> specifies the path where the archived library is to be created. This is the directory where user created files must be placed for inclusion in the archive.

-d <database name> specifies the name of the database for which data is being collected.

-c specifies that the utility will establish a connection to the database.

db2support V8.2.2 enhancement for optimizer data collection

Optimizer or sub-optimal access plan problem is one of the most frustrating problems to diagnose and identify the root cause. We often see multiple iterations of back and forth between the customer and the support personnel to gather the required diagnostic data for proper problem determination. An enhancement of `db2support` is available in DB2 UDB V8.2.2 to automate and centralize diagnostic data collection for optimizer problems.

A compact syntax diagram of the `db2support` command with the options for optimizer problems is as follows:

```
>>-db2support--output path--+--------------------+--+---------------+->
                            '--st--SQL statement-'  '--sf--SQL file-'

>--+----------------------+-------------------------------------------->
   '-se--embedded sql file-'

>--+-------------------------------------------------------------------+--->
   '--d--database name--+---------------------------------------+-'
                        '--c--+-------------------------------+-'
                              '--u--userid--+---------------+-'
                                            '--p--password--'

>--+----------------+--+----------------+--+------------------+------->
   '-cd--cur degree-'  '--ol--opt level-'  '--ra--refresh age-'

>--+----------------+--+------------------+--+--------------------+--->
   '--fp--func path-'  '--op--opt profile-'  '--cs--current schema-'

>--+------------------+--+----------------------+--+----------------+--->
   '--ot--opt tables-'  '--il--isolation level-'  '--td--delimiter-'

>--+-----+--+--------------------+--+-----+-------------------------><
   '--ro-'  '--cl--collect level-'  '--co-'
```

Note: Only optimizer problem related options are shown here.

`db2support` will collect the optimizer data for a problematic query when `-st, -sf` or `-se` option is specified. In case of error or trap during optimization, the "`-cl 0`" option should be used to collect database catalog tables and db2look table definitions without trying to explain a problematic query. There are four collect levels:

- 0 = collect only catalogs, `db2look`, DB CFG,DBM CFG, and `db2set`
- 1 = collect 0 plus `db2exfmt`
- 2 = collect 1 plus `db2service` (it is the default value of collect level.)
- 3 = collect 2 plus `db2batch`

In case special registers have been set to values other than default during the query execution, it is very important for correct problem analysis that the same values are passed as parameters to the db2support utility. When db2support is invoked to collect optimizer data, an additional output file (`db2support_opt.zip`) will be archived to `db2support.zip`.

Below are scenarios of invoking `db2support` in the optimizer mode via the four different scenarios mentioned above:

- Scenario 1: Via `-st` option

  ```
  db2support <output path> -d <database name> -st <sql_statement>
  ```

  Where SQL statement is input from the command line. For example,

  ```
  db2support . -d sample -st "select * from employee"
  ```

- Scenario 2: Via `-sf` option

  ```
  db2support <output path> -d <database name> -sf <sql_file>
  ```

  Where SQL statement stored in a file. For example,

  ```
  db2support . -d sample -sf badquery.sql
  ```

- Scenario 3: viViaa `-se` option

  ```
  db2support <output path> -d <database name> -se <embedded_sql_file>
  ```

  Where file containing embedded static SQL statement with the problematic query. For example

  ```
  db2support . -d sample -se badquery.sqc
  ```

- Scenario 4: Via `-cl 0` option

  ```
  db2support <output path> -d <database name> -cl 0
  ```

  Where collect optimizer related information when SQL statement is not provided. For example:

  ```
  db2support . -d sample -cl 0
  ```

In an SAP environment, `db2support` called with options "`-c -g -s`" is commonly used, where `-c` = connect, `-g` = all files from `db2dump`, and `-s` = system information. However, if there are hang situations while connecting to the database, option `-c` should be omitted.

Please note that when submitting the SQL with `db2support` through the command line, you need to make sure that the statement is submitted as *one* argument by using quotes. Also, make sure, especially on UNIX, that the quotes that are necessary in the SQL are passed through properly.

For example:

```
db2support . -d <sid> <some options> -cs <sap<sid>> -st select indname,
tabname from syscat. indexes where tabname = 'TADIR' order by indname
```

This will not work because the statement provided with out proper quotes will be treated as different parameters and the single quote (') surrounding TADIR will not be passed by the shell.

```
db2support . -d <sid> <some options> -cs <sap<sid>> -st 'select indname,
tabname from syscat.indexes where tabname = 'TADIR' order by indname'
```

This will not work, since again the apostrophes will not be passed by the shell. The correct command would be

```
db2support . -d <sid> <some options> -cs <sap<sid>> -st "select indname,
tabname from syscat.indexes where tabname = 'TADIR' order by indname"
```

Refer to SAP Note *83819* for further information.

DB2 UDB diagnostic tools

There are a wide range of DB2 UDB diagnostic tools available. Some typical tools are introduced in the following sections.

db2diag - db2diag.log analysis tool

Starting from DB2 UDB Version 8, the primary log file intended for use by database and system administrators is the administration notification log. The db2diag.log file, on the other side, is intended for use by the support personnel for troubleshooting purposes.

The *db2diag* tool serves to filter and format the volume of information available in the db2diag.log file. You can execute **db2diag -h** to get an initial help on the command syntax. The complete syntax of the db2diag command can be found in the DB2 UDB documentation, *Command Reference V8*, SC09-4828-01.

Following are some common db2diag usage scenarios.

Filtering the db2diag.log by database name using the command

```
db2diag -g db=SAMPLE
```

You would only see *db2diag.log* entries which contained "DB:SAMPLE" as shown in Example 8-7.

Example 8-7 Filtering the db2diag.log by database name

```
2005-05-22-20.59.14.571570-300 I404385C479       LEVEL: Severe
PID     : 15358                TID  : 1          PROC : db2agent (SAMPLE) 0
INSTANCE: axb                  NODE : 000        DB   : SAMPLE
```

```
APPHDL  : 0-7                  APPID: *LOCAL.axb.050523015904
MESSAGE : RDS UCINTFC: pCurrentPID->rdbname =
DATA #1 : Hexdump, 18 bytes
0x026149A0 : 5341 4D50 4C45 2020 2020 2020 2020 2020    SAMPLE
0x026149B0 : 2020
```

Filtering the db2diag.log by process id (display all severe error messages produced by process with process ID 7823) using command

```
db2diag -g level=Severe, pid=13077
```

it would successfully retrieve db2diag.log entries which meet the requirements, such as shown in Example 8-8.

Example 8-8 Filtering the db2diag.log by process id

```
2005-05-10-22.38.02.656105-300 I6836C354         LEVEL: Severe
PID     : 13077                TID  : 1          PROC : db2hmon 0
INSTANCE: axb                  NODE : 000
FUNCTION: DB2 UDB, routine_infrastructure, sqlerFmp0neTimeInit, probe:100
MESSAGE : DiagData
DATA #1 : Hexdump, 4 bytes
0xFFBFF824 : FFFF FBEE                                   ....
```

The following command filters all entries occurring after May 12, 2005 containing non-severe and severe errors logged on partition 0 or 1. It outputs the matched entries such that the time stamp, partition number and level appear on the first line, PID, TID and instance name on the second line, and the error message follows thereafter:

```
db2diag -time 2005-05-12 -node "0,1" -level "Severe, Error" -fmt
"Time: %timestamp Partition: %node Message Level: %{level} \nPid:
%{pid} Tid: %{tid} Instance: %{instance}\nMessage: %message\n"
```

An example of the output produced is shown in Example 8-9.

Example 8-9 Formatting the db2diag tool output

```
Time: 2005-05-26-11.31.17.373948 Partition: 000 Message Level: Severe
Pid: 15315 Tid: 1 Instance: axb
Message: DiagData

Time: 2005-05-26-11.31.21.007615 Partition: 000 Message Level: Error
Pid: 8506 Tid: 1 Instance: axb
Message: ADM7006E  The SVCENAME DBM configuration parameter was not
          configured.  Update the SVCENAME configuration parameter using the
          service name defined in the TCP/IP services file.
```

db2trc - trace utility

In problem determination it is sometimes very useful to have information on what was occurring within DB2 during the actual time of failure. For reproducible problems, you can use the *db2trc* utility to capture such information on:

- What DB2 UDB functional calls were made (most recent is at bottom of output)
- The actual code path used
- Sometimes even the data being manipulated at each point within the function

By default the DB2 trace information is kept in a memory buffer called the *trace buffer*. Otherwise it can be written to a file if the "-f" option is specified. When the DB2 trace is invoked, trace points within the DB2 source will "fire" during runtime. The firing of each trace point causes information such as the location within the code, error codes, return codes, and certain variables to be written to the trace buffer.

A DB2 trace can be initiated by issuing the command `db2trc on`. On UNIX and Linux systems, the default trace buffer size is 4 megabytes with 32-bit DB2 engine and 32 megabytes with 64-bit DB2 engine, while on Windows the default size is 8 megabytes. The trace buffer is circular, meaning that once the trace utility has reached the bottom of the buffer it will wrap back up to the top. If the buffer is too small, earlier trace information can be overwritten by the later one. So you will get a "wrapped" trace. If so, you can use "-l" option to specify a larger buffer than the default one when you start the trace.

Starting from DB2 version 8, it is crucial that the trace buffer size is a power of two (that is, 1, 2, 4, 8 etc. megabytes), which is required by the new very fast slot reservation scheme used in the implementation of the `db2trc` facility to achieve high performance. There is now a warning that indicates that the specified trace buffer has been rounded down if it is not a valid power of two. Trace will still function properly but with a reduced buffer size.

```
% db2trc on -l 8000000
Warning: The requested buffer size is not a power of 2.  The buffer has
been rounded down to 4 megabytes.
Trace is turned on
```

To simplify specifying the trace buffer size, you can now specify the number of megabytes of trace buffer using a "M" or "m" after the value. For example, to specify a 4 megabyte trace buffer:

```
db2trc on -l 4M.
```

On UNIX, If the trace is off when the instance is started, DB2 will allocate the trace buffer from a shared memory area with the default pre-defined size. And in such case for any trace enabled after the instance is started, you cannot use the

trace buffer size larger than the pre-allocated size. For example, for 32-bit DB2 on AIX platform, the default trace buffer is 4 megabytes. The only way to be able to use a bigger trace buffer is to specify a new trace buffer size before the DB2 instance is started as follows:

```
% db2stop
% db2trc on -l 8M
% db2start
% db2trc off
```

After the foregoing steps, you can start DB2 trace at any later time with a buffer up to the size specified previously (8 megabytes in this example).

Once the trace facility has been enabled using the "on" option, all subsequent work done by the instance will be traced.

While the trace is running, you can use the "clr" or "clear" option to clear out the trace buffer. All existing information in the trace buffer will be removed.

```
% db2trc clr
Trace has been cleared
```

Once the operation being traced has finished, use the "dmp" or "dump" option followed by a trace file name to dump the memory buffer to disk. For example:

```
% db2trc dmp trace.dmp
Trace has been dumped to file
```

The trace facility will continue to run after dumping the trace buffer to disk. To turn tracing off, use the "off" option:

```
% db2trc off
Trace is turned off
```

The trace dump file created above is in a binary format that is not readable. It is recommended that you format the trace dump file before sending them to the support personnel. This is accomplished via the "flw" and "fmt" options. You must provide the binary dump file along with the name of the ASCII file that you want to create:

```
% db2trc flw trace.dmp trace.flw
% db2trc fmt trace.dmp trace.fmt
```

If the format output indicates "Trace wrapped" is "YES", then it means that the trace buffer was not large enough to contain all the information collected during the trace period. So there is a chance that some trace information related to the problem determination may be overwritten. And you might consider to redo the operation tracing with a larger trace buffer.

There are a few more things to be aware of about the DB2 trace:

- In a multi-partitioned database, only one DB2 trace is required for all logical partitions on that physical server as the trace information for all logical partitions on the same server goes into the same trace buffer, and each trace entry has a partition number associated with it.
- On UNIX, DB2 UDB will automatically dump the trace buffer to disk when the instance shuts down (crashes). If tracing is enabled when an instance ends abnormally, a file will be created in the `DIAGPATH` directory and its name will be `db2trdmp.xxx`, where `xxx` is the partition number. This does not occur on Windows platform, and you have to dump the trace manually instead.

To summarize, Example 8-10 shows a common use of `db2trc`.

Example 8-10 Using db2trc

```
% db2trc on -l 8M
% db2trc clr

execute the failed command
After received the error, then

% db2trc dmp trace.dmp
% db2trc off
% db2trc flw trace.dmp trace.flw
% db2trc fmt trace.dmp trace.fmt
```

CLI trace

The CLI trace captures information about applications that access the DB2 CLI driver. When enabled, the CLI trace facility generates one or more log files whenever an application accesses the driver. These log files provide detailed information about:

- The order in which CLI functions were called by the application.
- The contents of input and output parameters passed to and received from CLI functions.
- The return codes and any error or warning messages generated by CLI.

The CLI trace is applicable for situations where a problem is encountered in:

- CLI applications
- ODBC applications
- DB2 CLI stored procedures
- Legacy Type 2 JDBC™ applications and stored procedures

There are no ODBC applications in the SAP environment and SAP does not support the legacy type 2 JDBC driver. In the SAP environment, usually CLI trace is taken only when it is suggested by SAP support.

The CLI trace is enabled by adding specific entries to the DB2 CLI/ODBC configuration keyword file (`db2cli.ini`). By default, the `db2cli.ini` file resides under the `sqllib` directory on Windows platforms, and under the `sqllib/cfg` directory on UNIX platforms. The environment variable `DB2CLIINIPATH` can be used to override the default and specify a different location for the file.

The typical steps to take the CLI trace are as follows:

1. Create a path for the trace files.

   It is important to create a path that every user can write to.

2. Update the CLI configuration keywords.

   It can be done by either manually editing the `db2cli.ini` file or using the `UPDATE CLI CFG` command. The latter is highly recommended, as it is less problem prone.

   ```
   db2 UPDATE CLI CFG FOR SECTION COMMON USING Trace 1
   db2 UPDATE CLI CFG FOR SECTION COMMON USING TracePathName <path>
   db2 UPDATE CLI CFG FOR SECTION COMMON USING TraceComm 1
   db2 UPDATE CLI CFG FOR SECTION COMMON USING TraceFlush 1
   db2 UPDATE CLI CFG FOR SECTION COMMON USING TraceTimeStamp 3
   ```

3. Verify the `db2cli.ini` configuration.

   Issue the following command to verify that the correct keywords are set:

   ```
   db2 GET CLI CFG FOR SECTION COMMON
   ```

4. Restart the application.

   The `db2cli.ini` file is only read on application start. So for any changes to take effect, the application must be restarted. If tracing a CLI stored procedure, this means restarting the DB2 instance.

5. Capture the error.

   Run the application until the error is generated, and then terminate the application.

6. Disable the CLI trace.

   Issue:

   ```
   db2 UPDATE CLI CFG FOR SECTION COMMON USING Trace 0
   ```

   Then restart applications that may be tracing.

7. Collect the trace information.

   The CLI trace files are written to the path specified in the TracePathName keyword. The name of the trace output files looks like p<pid>t<tid>.cli where <*pid*> is the process ID, and <*tid*> is the thread ID.

Keep in mind that enabling CLI trace has the performance impact on all running applications. So it is recommended to turn off the trace after the trace information has been collected. Restarting the whole SAP system to activate the CLI trace makes all SAP work processes write to the trace file and the CLI trace will become big in a short time period. This should be avoided unless is required. In most cases, it is much better to restart only the SAP work process where the problem is expected to occur or to use TRACEREFESHINTERVAL to activate the trace just before the problem is reproduced.

In SAP systems with kernel releases 6.10 and higher, you can also dynamically activate the DB2 CLI trace by setting the dbs/db6/dbsl_cli_trace SAP profile parameter in the transaction RZ11. By default, the CLI trace files are then generated in \\<hostname>\sapmnt\TraceFiles\<SAPSID> directory on Windows platforms, and in /tmp/TraceFiles/<SID> on UNIX platforms.

For more detailed information about the CLI trace, you can refer to the IBM DB2 UDB documentation, *Call Level Interface Guide and Reference, Volume 1, V8,* SC09-4849-01 and SAP Note *486951*.

db2dart - database inspection and repair tool

The *db2dart tool* has the ability to inspect table spaces and tables in a DB2 UDB database for their architectural integrity, and check the pages of the database for page consistency. Also, it can repair some of the database/data corruption if identified. db2dart must be run with no users connected to the database.

To display all of the possible options, simply execute the command **db2dart** without any parameters. For syntax details, you can refer to the DB2 UDB documentation, *Command Reference V8*, SC09-4828-01.

By default, db2dart will create a report file named <database_name>.RPT under a subdirectory in the DIAGPATH directory. The subdirectory is called DARTXXXX, where XXXX is the partition number.

db2dart accesses the data and metadata in a database by reading them directly from disk. Because of that, db2dart should never be run against a database that still has connections. Otherwise, since *db2dart* will not know about pages in the buffer pools and control structures in memory etc., it may report false errors as a result. Similarly, if you run db2dart against a database that requires crash recovery or that has not completed roll-forward recovery, due to the inconsistent nature of the data on disk, db2dart may report false errors as well.

You can use db2dart to inspect a database, a table space, or a table. If the database size is relatively small or you are able to schedule a long enough window, it is recommended to inspect the entire database. Sometimes the first time you notice a database corruption is when you fail to access a certain table or table space, and there may be corruptions with other database objects as well.

- *Inspect an entire database.*

  When inspecting the entire database, the only option specified in the db2dart command is the database name as follows:

```
% db2dart sample

        The requested DB2DART processing has completed successfully!
                        Complete DB2DART report found in:
/home/axb/sqllib/db2dump/DART0000/SAMPLE.RPT
```

If a database is very large or you are only interested at one table space or one table, you can follow the examples below to use db2dart to inspect a table space or table accordingly:

- *Inspect a table space.*

  db2dart database_name **/ts**

  You need to specify the /TS option and the table space ID in order to inspect a table space. You can provide the table space ID within the same command via the /TSI option or you can let db2dart prompt you for it. If you do not know the table space ID, you can obtain it via the **db2 list tablespaces** command. For example, to inspect the USERSPACE1 table space which has the table space ID of 2 in SAMPLE database.

  Method 1:

```
% db2dart sample /ts /tsi 2

        The requested DB2DART processing has completed successfully!
                        Complete DB2DART report found in:
/home/axb/sqllib/db2dump/DART0000/SAMPLE.RPT
```

  Method 2:

```
%db2dart sample /ts

Please enter tablespace ID:
2

        The requested DB2DART processing has completed successfully!
                        Complete DB2DART report found in:
/home/axb/sqllib/db2dump/DART0000/SAMPLE.RPT
```

- *Inspect a table.*

  db2dart database_name **/t**

  A table and its associated objects (LOBs and indexes etc.) can be inspected using the /T option. When using this option, you must provide either the table name or object ID, and the ID of the table space in which the table resides. To determine the object ID and table space ID for a table, you can query the TABLEID and TBSPACEID columns of the SYSCAT.TABLES catalog view. For example, determine the object ID and table space ID for the STAFF table in SAMPLE database by executing the following query:

```
% db2 connect to sample
% db2 "select tableid, tbspaceid from syscat.tables where tabname =
'STAFF'"

TABLEID TBSPACEID
------- ---------
      3         2

  1 record(s) selected.
```

  To inspect this table, execute one of the following db2dart commands:

  Method 1:

```
% db2dart sample /t /tsi 2 /oi 3
```

  Method 2:

```
% db2dart sample /t /tsi 2 /tn STAFF
```

  Method 3:

```
% db2dart sample /t

Please enter Table ID or name, and tablespace ID:
3,2

        The requested DB2DART processing has completed successfully!
                     Complete DB2DART report found in:
/home/axb/sqllib/db2dump/DART0000/SAMPLE.RPT
```

  Method 4:

```
% db2dart sample /t

Please enter Table ID or name, and tablespace ID:
STAFF,2

        The requested DB2DART processing has completed successfully!
                     Complete DB2DART report found in:
/home/axb/sqllib/db2dump/DART0000/SAMPLE.RPT
```

When DB2 UDB identifies a database corruption caused by data or index page corruptions, it will mark the database as bad and doesn't allow users to access the database any more. In such case, the best option is to restore a known good backup and roll forward the database to an appropriate point of time. However, if there is no available backup, we can attempt to "fix" the corrupted page with `db2dart`.

For "fixing" the data page corruption, `db2dart` can be used to initialize the corrupted data page and make it look as though there are no data records on it. You must use *db2dart* to retrieve the available records from the corrupted data page(s) before initializing the page(s), or the data is lost. The retrieved records are written to a delimited ASCII file which can be imported into the table at a later time. After you initialize the corrupted data page(s), it may introduce potential inconsistency into the database. So once this is done, you must export all tables and recreate the database.

To summarize, typical steps in using `db2dart` to fix a data page corruption problem are:

1. A data page corruption is detected by DB2 UDB.
2. Use `db2dart` to examine the database to determine which page(s) are affected.
3. Use `db2dart` to "dump" the available data on any corrupted page(s) to delimited ASCII format file.
4. Use `db2dart` to initialize the corrupted page(s) to make the database accessible.
5. Export all tables in the database, drop the database, recreate the database and tables, import the data dumped by `db2dart` and exported earlier.

In this book we intentionally skip the steps on how to use `db2dart` to dump and initialize the corrupted data pages. We would like you to perform such steps under the close guidance of IBM DB2 support personnel to avoid any possible mistake which will put your data in danger.

For "fixing" index page corruption, we do not repair index page corruptions. Instead we completely rebuild the index object for the table. To do this we can use `db2dart` to mark the indexes for a table as bad. The next time when any index defined on the table is used, all indexes on the table will be rebuilt.

Please keep in mind if you would like to determine the root cause of the index page damage, please engage the support personnel to collect the necessary diagnostic information first before you take the steps to mark the index as bad and rebuild the index. The required diagnostic information could be the dump of the damaged index pages and/or the pages around the damaged pages, and the related transactional log files etc.

Typical steps in fixing an index page corruption are as follows:

1. An index corruption is detected by DB2 UDB.
2. Use `db2dart` to examine the database to determine which indexes are affected.
3. Use `db2dart` to mark the indexes as invalid.
4. The first use of any of the indexes on the table will trigger DB2 UDB to rebuild all of the table's indexes.

Example of fixing an index page corruption

When querying the database, you receive:

```
% db2 "select count(*) from emp_photo"
SQL1034C  The database is damaged.  All applications processing the
database
have been stopped.  SQLSTATE=58031
```

`db2diag.log file` shows:

```
2005-05-22-11.39.34.295925-300 I35282C464        LEVEL: Error
PID     : 21637                TID  : 1           PROC : db2agent (SAMPLE)
0
INSTANCE: axb                  NODE : 000         DB   : SAMPLE
APPHDL  : 0-7                  APPID: *LOCAL.axb.050522163916
FUNCTION: DB2 UDB, buffer pool services, sqlbrdpg, probe:1146
RETCODE : ZRC=0x87020036=-2029912010=SQLB_BADHDR "Bad Page Header"
          DIA8547C An error occurred in a database page header.

2005-05-22-11.39.34.308585-300 I35747C438        LEVEL: Error
PID     : 21637                TID  : 1           PROC : db2agent (SAMPLE)
0
INSTANCE: axb                  NODE : 000         DB   : SAMPLE
APPHDL  : 0-7                  APPID: *LOCAL.axb.050522163916
FUNCTION: DB2 UDB, buffer pool services, sqlbrdpg, probe:1146
DATA #1 : String, 77 bytes
 Obj={pool:2;obj:8;type:1} State=x5 Page=0 Cont=0 Offset=0 BlkSize=12
BadPage
```

Note the following entries in the db2diag.log file:

- pool:2 — Indicates table space id for the corrupted object is 2.
- obj:8 — The object id of the corrupted object is 8.
- type:1 — The type of object is 1, indicating an index object. Type = 0 indicates a data object.
- Page=0 -— The page in the object with the error is page 0.

Let's run db2dart to see what the error really is:

```
$ db2dart sample

                  DB2DART Processing completed with error!
                      Complete DB2DART report found in:
/home/axb/sqllib/db2dump/DART0000/SAMPLE.RPT
```

Examine the db2dart report SAMPLE.RPT for more details:

```
...
Table inspection start: AXB.EMP_PHOTO

Data inspection phase start. Data obj: 8  In pool: 2
Data inspection phase end.

Index inspection phase start. Index obj: 8  In pool: 2
Error: BPS check read error for pool page 1, from object ID 8, pool 2,
Error: BPS check read error for pool page 1, from object ID 8, pool 2,
Error: BPS Header object ID incorrect. Expecting 8, Found 9.
Error: BPS Header problems found
Error: in page 1, pool page 1, of obj 8, in tablespace 2.
Error: Page data will be dumped to report.

000       *0030 0FD0 0000 0172 0400 0000 007E 114F*   *.0.....r.......O*
010       *0000 0000 0009 010A 9B62 AA90 0000 0000*   *.........b......*
020       *0000 0002 0000 0000 0000 0000 0000 0000*   *................*
030       *                                       **
030  000  *0050 4652 0100 0210 0000 0002 0000 0002*   *.PFR............*
040  010  *0000 0000 0000 0000 0000 0000 007E 12FD*   *................*
050  020  *0000 0000 0000 0000 0000 0000 0000 0000*   *................*
060  030  *0000 0000 0000 0000 0000 0000 0000 0000*   *................*
...
```

Now mark the indexes on the table EMP_PHOTO as *bad*.

```
% db2dart sample /MI

Please enter index object ID and tablespace ID:
8,2

Attempting to mark index (p=2;o=8) as bad.

Modification for page (obj rel 0, pool rel 0) of pool ID (2) obj ID (8),
written out to disk successfully.
        The requested DB2DART processing has completed successfully!
                      Complete DB2DART report found in:
/home/axb/sqllib/db2dump/DART0000/SAMPLE.RPT
```

Now submit the query again and it will trigger the index to be rebuilt:

```
% db2 connect to sample
% db2 "select count(*) from emp_photo"
1
-----------
         12

  1 record(s) selected.
```

db2mtrk - memory tracker

The memory tracker tool (db2mtrk) provides a complete report of memory status, for instance, databases, and agents. It is a lightweight version of DB2 snapshot monitor by providing the snapshot information of memory allocation only.

db2mtrk outputs the following memory pool allocation information:

- Current size
- Maximum size (hard limit)
- Largest size (high water mark)
- Type (identifier indicating how memory will be used)
- Agent which allocated the pool (if the memory pool is private)

Example 8-11 shows using db2mtrk with option -i to see instance level memory.

Example 8-11 db2mtrk - instance level memory

```
% db2mtrk -i -v
Tracking Memory on: 2005/05/25 at 14:27:35

Memory for instance

   Database Monitor Heap is of size 278528 bytes
   Other Memory is of size 1589248 bytes
   Total: 1867776 bytes
```

Example 8-12 shows using command **db2mtrk -d** to see database level memory for all active databases within the instance

Example 8-12 db2mtrk - database level memory for all active databases in a instance

```
% db2mtrk -d -v
Tracking Memory on: 2005/05/25 at 14:29:33

Memory for database: SAMPLE

   Backup/Restore/Util Heap is of size 16384 bytes
   Package Cache is of size 163840 bytes
   Catalog Cache Heap is of size 65536 bytes
```

```
   Buffer Pool Heap is of size 4456448 bytes
   Buffer Pool Heap is of size 688128 bytes
   Buffer Pool Heap is of size 425984 bytes
   Buffer Pool Heap is of size 294912 bytes
   Buffer Pool Heap is of size 229376 bytes
   Lock Manager Heap is of size 589824 bytes
   Database Heap is of size 3948544 bytes
   Other Memory is of size 0 bytes
   Total: 10878976 bytes

Memory for database: HP3

   Backup/Restore/Util Heap is of size 16384 bytes
   Package Cache is of size 7290880 bytes
   Catalog Cache Heap is of size 1376256 bytes
   Buffer Pool Heap is of size 83181568 bytes
   Buffer Pool Heap is of size 16760832 bytes
   Buffer Pool Heap is of size 688128 bytes
   Buffer Pool Heap is of size 425984 bytes
   Buffer Pool Heap is of size 294912 bytes
   Buffer Pool Heap is of size 229376 bytes
   Lock Manager Heap is of size 89145344 bytes
   Database Heap is of size 9666560 bytes
   Other Memory is of size 0 bytes
   Total: 209076224 bytes
```

Currently the "-d" option is only supported on UNIX and Linux platforms, but not on Windows. There are architectural differences between DB2 memory management on Windows and on UNIX and Linux platforms. These differences make it difficult for DB2 UDB to distinguish between the instance and database memory on Windows systems. For that reason, on Windows, DB2 UDB reports the memory for the instance and the databases all together with the command **db2mtrk -i**.

The reason that there are so many buffer pool heaps shown in the example is that each DB2 UDB database maintains four hidden buffer pools, one with each page size: 4K, 8K, 16K, and 32K. The size of each hidden buffer pool is 16 pages. As a result, a database with one user defined buffer pool (IBMDEFAULTBP) will have a total of five buffer pools and accordingly will have five buffer pool heaps.

Example 8-13 shows using db2mtrk with option -p to see agent private memory.

It is a common practice to use the "-p" option along with the "-r <interval> <count>" option so that you can get private memory allocation information for all DB2 agents every several (<interval>) seconds for several (<count>) iterations.

Example 8-13 db2mtrk - agent private memory

```
% db2mtrk -p -r 2 3
Tracking Memory on: 2005/05/25 at 14:33:13

Memory for agent 25228

   Application Heap is of size 131072 bytes
   Application Control Heap is of size 16384 bytes
   Total: 147456 bytes

Memory for agent 5070

   Application Heap is of size 212992 bytes
   Application Control Heap is of size 16384 bytes
   Total: 229376 bytes

Memory for agent 13859

   Application Heap is of size 622592 bytes
   Application Control Heap is of size 16384 bytes
   Total: 638976 bytes

Memory for agent 13432

   Application Heap is of size 622592 bytes
   Application Control Heap is of size 16384 bytes
   Total: 638976 bytes
...
```

db2pd - monitor and troubleshooting tool

The *db2pd* tool is a comprehensive problem determination tool available at DB2 UDB V8.2. It is the equivalent of the well known `onstat` utility for Informix Dynamic Server.

`db2pd` can provide a wide range of information useful for troubleshooting and problem determination, performance improvements, and application development design, including:

- Locks
- Buffer pools
- Table spaces
- Containers
- Dynamic SQL statements
- Agents
- Applications
- Memory pools and sets

- Transactions
- Logs
- And others

db2pd collects this information without acquiring any latches or using any engine resources. It is therefore possible (and expected) to retrieve information that is changing while db2pd is collecting information; hence data may not be completely accurate. However, it has proven reliable even on the busiest systems with thousands of users or SAP connections. The benefits to collecting information without latching include faster retrieval and no competition for engine resources.

There are three ways to invoke db2pd functionality:

- Command line (with or without interactive mode):

  ```
  % db2pd -db sample -app
  ```

- Environment variable:

  ```
  % export DB2PDOPT="-db sample -app"
  % db2pd
  ```

- Command file:

  Edit a file containing the commands such as command.txt, then run

  ```
  % db2pd -command command.tx
  ```

As we mentioned earlier, db2pd can retrieve extensive instance and database related information from the live system. Below are two simple examples of the db2pd tool. Hope that you can take it as a start point and practise more about db2pd usage by following command details available in the DB2 UDB documentation, *Command Reference V8*, SC09-4828-01.

The db2pd -memsets and -mempools options report statistics about DB2 UDB memory sets and memory pools which can be very useful when trying to understand memory usage. Example 8-14 shows the usage of db2pd for monitoring memory usage.

Example 8-14 db2pd - monitoring memory usage

```
% db2pd -memsets -mempools

Database Partition 0 -- Active -- Up 0 days 00:07:34

Memory Sets:
Name   Address             Id      Size      Key         DBP Type   Ov OvSize
DBMS   0xC000000004900000  1515    43319296  0xF3409F61  0   0      Y  8224768
FMP    0xC000000008300000  916     23052288  0x0         0   2      N  0
Trace  0x0000000000000000  18014   39995248  0xF3409F74  0   -1     N  0

Database Partition 0 -- Active -- Up 0 days 00:07:34
```

```
Memory Pools:
Address         MemSet  PoolName  Id   Overhead  LogSz   Log UpBnd  LogHWM
0xC000000004900B10 DBMS   monh     11   48256    305584  524 288   306656
0xC000000004900A08 DBMS   resynch  62   7296     107344  835 5840  107344
0xC000000004900900 DBMS   apmh     70   0        97568   694 6816  103432
0xC00000004900 7F8 DBMS   kerh     52   0        50448   688 128   50720
0xC00000004900 6F0 DBMS   bsuh     71   0        76528   174 16192 89210
0xC00000004900 5E8 DBMS   sqlch    50   0        960530  983 040   960530
0xC00000004900 4E0 DBMS   pmth     80   0        1344    984 000   1600
0xC00000004900 3D8 DBMS   krcbh    69   0        21720   327 68    21720
0xC00000004900 2D0 DBMS   eduah    72   16192    81952   819 84    81952
0xC00000000830 02D0 FMP   undefh   59   145728   1179828 231 35360 1179828

PhySz       PhyUpBnd     PhyHWM       Bnd BlkCnt CfgParm
360448      524288       360448       Ovf 0      MON_HEAP_SZ
131072      8355840      131072       Ovf 0      n/a
114688      6946816      114688       Ovf 0      n/a
81920       688128       81920        Ovf 0      n/a
98304       17416192     98304        Ovf 0      n/a
983040      983040       983040       Ovf 0      n/a
16384       999424       16384        Ovf 0      n/a
32768       32768        32768        Ovf 0      n/a
98304       98304        98304        Ovf 0      n/a
1327104     23150592     1327104      Phy 0      n/a
```

db2pd -transactions provides the number of locks, first lsn, last lsn, logspace used, and space reserved. This can be useful for understanding the behavior of any transaction. Example 8-15 shows the command and output.

Example 8-15 db2pd - monitoring resource used by a transaction

```
% db2pd -transactions -db sample

Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:01:44

Transactions:
Address    AppHandl [nod-index] TranHdl  Locks    State   Tflag      Tflag2
0x14261540 95       [000-00095] 2        0        READ    0x00000000 0x00000000
0x14261FC0 96       [000-00096] 3        0        READ    0x00000000 0x00000000

Firstlsn       Lastlsn        LogSpace  SpaceReserved   TID
0x000000000000 0x000000000000 0         0               0x00000000000C6
0x000000000000 0x000000000000 0         0               0x00000000000C9

AxRegCnt   GXID
1          0
1          0
```

8.2.2 SAP tools for diagnostic purposes

SAP systems provide a collection of tools for diagnostic purposes for different areas: ST04: Performance allow you to monitor the performance of the whole system; ST05: SQL trace and ST04: SQL Cache helps you to identify and analyze expensive SQL statements; for general error logging between the DBSL and the CLI layer, we have the DBSL trace, developer trace, SM21: system logs and ST22: ABAP dump analysis; there is special tools like DBSL deadlock trace and `db6util trace` for tracking down deadlock problems; and script like `sapdb6st.ksh` collects diagnostic information in a file for you to send to SAP support for further analysis. In the following section, we explain what each of these tools are used for and how to use them.

sapdb6st.ksh - SAP diagnostic data collection script

sapdb6st.ksh is similar to the `db2support` script. It extracts important diagnostic data, such as operating system, hardware, network data, DB2 database environment, SAP application software environment, from your SAP and DB2 UDB environment (single partition or DPF environment) and gather the data together into a single compressed file. SAP support sometimes requires customers to run this script to collect diagnostic information for further investigation. Refer to SAP Note *83819* to download the latest version of the script from the note's attachment and the latest instruction on how to run the script.

Currently the script is only available on UNIX.

Here is an example to show the steps to use `sapdb6st.ksh` to collect diagnostic data from your system:

1. Download the `sapdb6st-<version>.SAR file from` SAP Note *83819*
2. As `<sid>adm`, unpack the `sapdb6st-<version>.SAR` file using the `SAPCAR` utility to get the `sapdb6st.ksh` file.

   ```
   heidelberg:hp1adm 22> SAPCAR -xvf sapdb6st-2-00-02.SAR
   SAPCAR: processing archive sapdb6st-2-00-02.SAR
   x sapdb6st.ksh
   SAPCAR: 1 file(s) extracted
   ```

3. Grant proper permission to the script for executing:

   ```
   heidelberg:hp1adm 23> chmod 755 sapdb6st.ksh
   ```

4. The script has the following syntax:

   ```
   ./sapdb6st.ksh sid=<sid> inqno=<problem_number>
   ```

   If our *<SID>* is HP1 and the SAP Message number is 17356, we would run:

   ```
   heidelberg:hp1adm 24> ./sapdb6st.ksh sid=HP1 inqno=17356
   ```

5. After the script finishes, you see the following files:

```
heidelberg:hp1adm 34> ls -lrt
total 17744
-rwxr-xr-x   1 hp1adm     sapsys      142449 May 25 17:32 sapdb6st.ksh
-rw-rw-rw-   1 hp1adm     sapsys      323991 May 25 17:43
README_HP1_17356.txt
-rw-rw-rw-   1 hp1adm     sapsys     8268317 May 25 17:43
17356_HP1_20050525.tar.Z
-rw-rw-rw-   1 hp1adm     sapsys      333504 May 25 17:44
17356_HP1_20050525.html
```

6. Send the <problem_number>_<SID>_<date>.tar.Z file to the SAP support. Tell SAP support the location of the file and the exact filename.

ST04: key performance indicators

The ST04 Performance screen in DBA Cockpit provides useful indicators for the performance of your system: buffer pool quality, package cache and catalog cache quality, sort heaps quality, etc. By examining these indicators and understand where the database performance is suboptimal, you can do the corresponding corrective measures. In the following sections, we discuss some key areas you should pay attention to in the ST04: Performance screen.

Buffer pool quality

A buffer pool is an area of memory used to cache table and index data pages as they are being read from disk, or being modified. Buffer pools allow the database to minimize disk accesses (physical reads, which is expensive) and to maximize buffer pool access (logical reads). Configuring buffer pools is one of the most important tuning areas. It has direct impact on the overall database performance.

You can monitor the buffer pool of your whole database by choosing **Performance** → **Database** in transaction ST04 (one of the entry point to *DBA Cockpit*). See Figure 8-5.

.

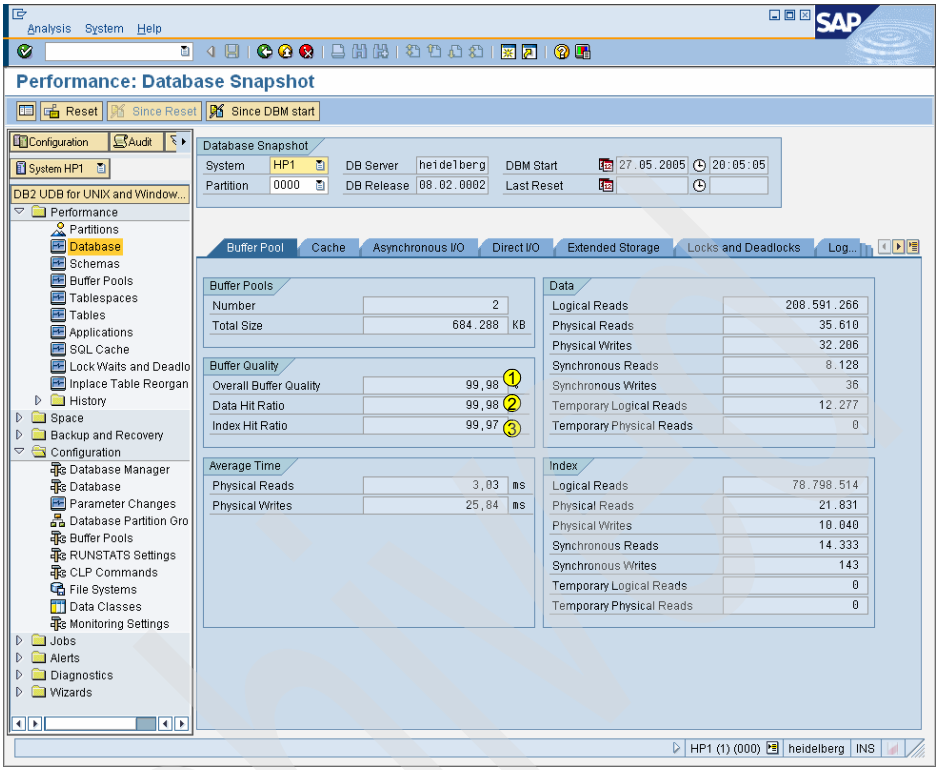


Figure 8-5 Buffer pool: overall buffer pool quality for database

1. *Overall buffer pool quality*: This represents the ratio of physical reads to logical reads of all buffer pools.

   ```
   ((data logical reads + index logical reads) - (data physical reads + index
   physical reads)) / (data logical reads + index logical reads) * 100%
   ```

   As a rule of thumb, the overall buffer pool quality for your database should be higher than 96%.

2. *Data hit ratio*: This represents the ratio of data physical reads to data logical reads of all buffer pools.

   ```
   (data logical reads - data physical reads) / (data logical reads) * 100%
   ```

   As a rule of thumb, data hit ratio should be higher than 95%.

3. *Index hit ratio*: This represents the ratio of index physical reads to index logical reads of all buffer pools.

   ```
   (index logical reads - index physical reads) / (index logical reads) * 100%
   ```

   As a rule of thumb, index hit ratio should be higher than 98%.

You can also monitor the quality of a particular buffer pool by choosing **Performance** → **Buffer Pools** (see Figure 8-6). As a rule of thumb, the buffer pool quality of each buffer pool should be higher than 96%.

L

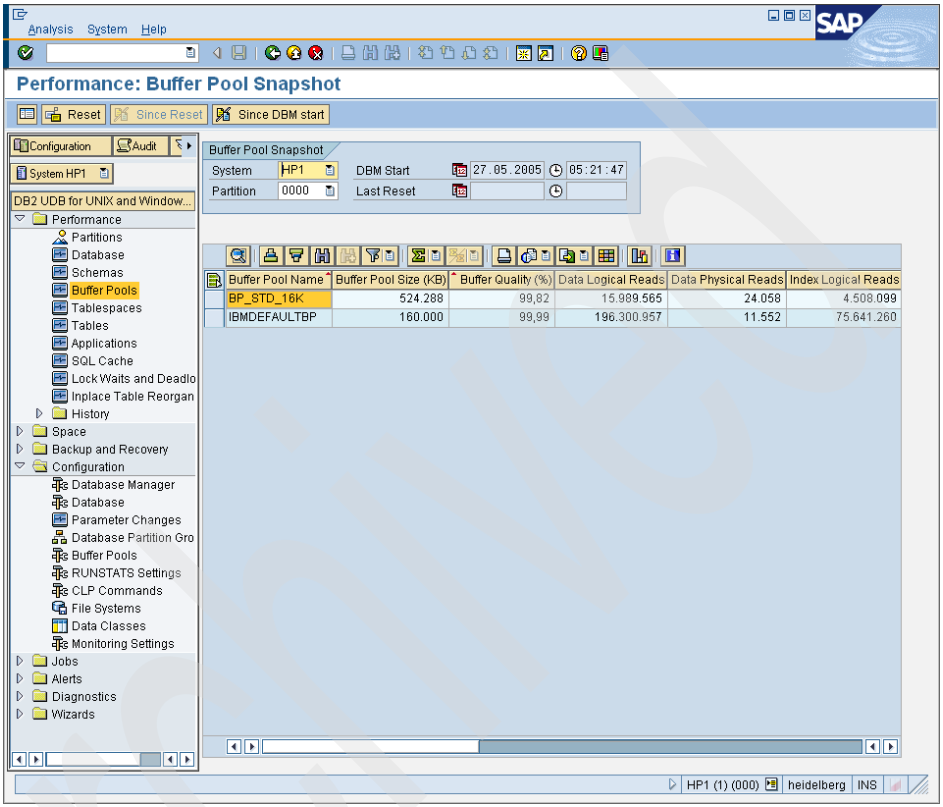


Figure 8-6 ST04: Buffer pool: buffer pool quality for each buffer pool

Buffer pool quality can be monitored on the table space level by choosing **Performance** →**Tablespaces** (Figure 8-7). You should pay attention to the table space that have large logical reads but low buffer quality.

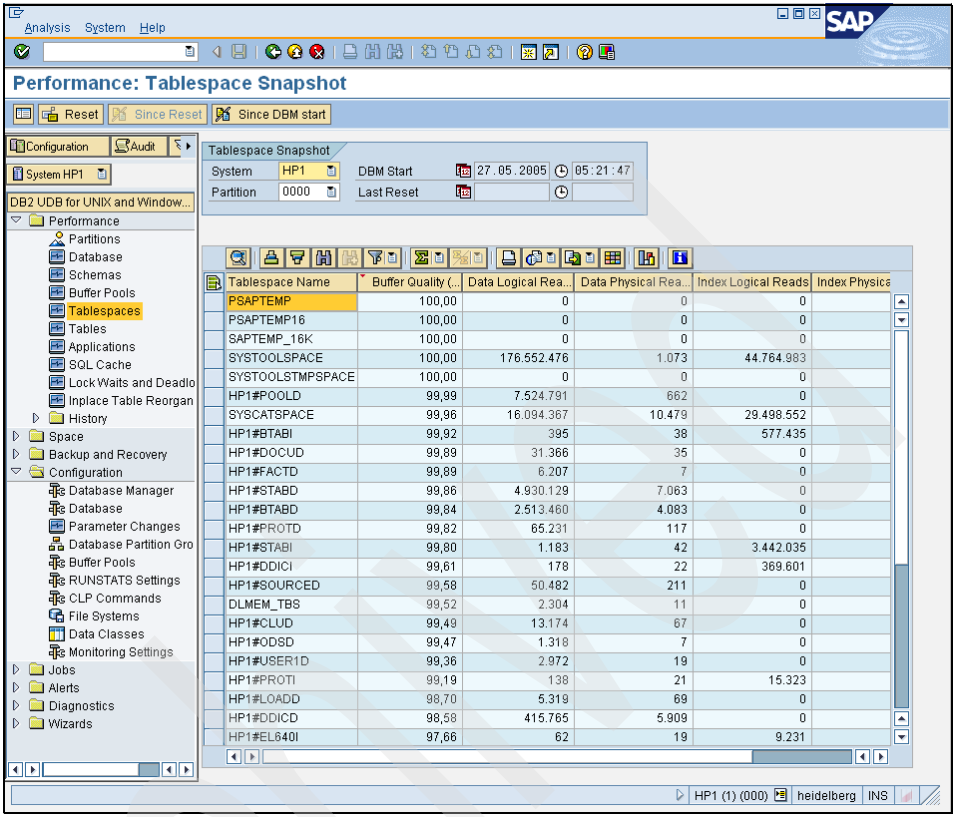


Figure 8-7 ST04: Buffer pool: buffer pool quality for each table space

Tip: To find out the which buffer pool is assigned to which table space, you can go to transaction DBA Cockpit: **Configuration → CLP Commands.** In the function box, choose **Buffer Pool to Table Space Assignment**.

If you find that any of the above values are too low, consider increasing the size of the buffer pools. To change the buffer pool size, you can use the DB2 command **`ALTER BUFFERPOOL`** statement; or in DBA Cockpit, choose **Configuration → Buffer Pools**; choose the buffer pool in the list and click the **Edit** button to change the buffer pool size. Choose the **immediate** check box if you want the new buffer pool size to take effect. The immediate option is useful if you need a bigger buffer pool to enhance the performance without restarting a system (especially a running production system).

Package cache quality and catalog cache quality

The *package cache* is used for caching sections for SQL statements. Caching packages allows the database manager to reduce its internal overhead by eliminating the need to access the system catalogs when reloading a package; or, in the case of dynamic SQL, eliminating the need for recompilation. This caching of the section can improve performance especially when the same statement is used multiple times by applications connected to a database. This is particularly important in an online transaction processing (OLTP) application.

The *catalog cache* stores binary and compressed descriptors for tables, views, and aliases. Caching catalog information allows the database manager to reduce its internal overhead by eliminating the need to access the system catalogs to obtain information that has previously been retrieved.

You can monitor the quality of package cache and catalog cache by choosing **Performance** → **Database** and choose the **Cache** tab (see Figure 8-8).

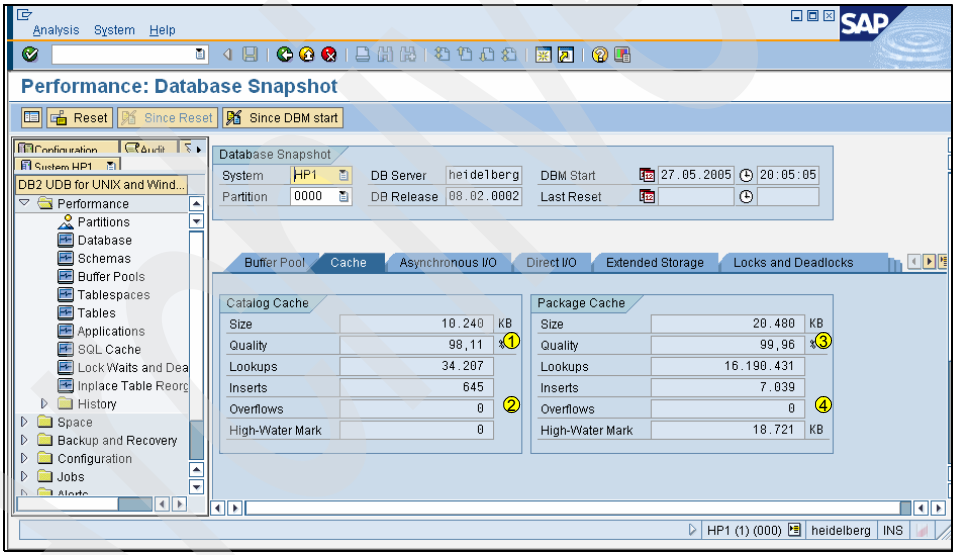


Figure 8-8 ST04: Package Cache and Catalog Cache

1. *Catalog cache quality*:

   The catalog cache hit ratio measures how well the catalog cache helps to avoid actual accesses to the catalog on disk:

   ```
   (1 - (catalog cache inserts / catalog cache lookups)) * 100%
   ```

   For online transaction processing (OLTP) system, this should be higher than 95%.

2. *Catalog cache overflow*:

   Number of times that an insert in the catalog cache failed because the catalog cache was full. Ideally this indicator should be 0. If you see catalog cache overflows, consider increasing the database configuration parameter `CATALOGCACHE_SZ` (catalog cache size). Since `catalog cache` is allocated from the database heap, you may need to increase the database configuration parameter `DBDEAP` as well.

3. *Package cache quality*:

   The package cache hit ratio is a percentage indicating how well the package cache helps to avoid reloading packages and sections for static SQL from the system catalogs as well as helping to avoid recompiling dynamic SQL statements:

   ```
   (1 - (package cache inserts / package cache lookups)) * 100%
   ```

   As a rule of thumb, the package cache hit ratio should be higher than 98%. If the package cache hit ratio is too low, consider increasing the database configuration parameter `PCKCACHESZ`.

4. *Package cache overflow*:

   Number of times that an insert in the package cache failed because the package cache was full. If you see package cache overflows, consider increasing the database configuration parameter `PCKCACHESZ`.

Sort overflows and non-piped sorts

Sorting is required when there is no index existing to satisfy a requested ordering, or an index exists but sorting would be more efficient then using the index. Sorting is an expensive operating, which requires both large CPU time and memory.

The database configuration parameter `SORTHEAP` controls how much memory can be used for each sort. The total amount of memory for sorting available in the DB2 instance is configured in the database manager configuration parameter `SHEAPTHRES` (sort heap threshold). This parameter is a soft limit. If the value of this parameter is reached, new sort requests will receive only small amount of memory.

Two kinds of problems that can happens to the sorting process:

- *Sort overflows:*

  Sort overflow occurs when the information being sorted is larger than the size of the sort heap. A temporary table is then created to store the temporary results. With small buffer pools, this temporary table might even spill over to disk storage of the `PSAPTEMP` (or `PSAPTEMP16`) table space. Sorts that do not overflow usually perform better than those that do.

- *Non-piped sorts* (when returning the results of the sorting)

  If sorted information can return directly without requiring a temporary table to store a final, sorted list of data, it is a *piped sort*. If the sorted information requires a temporary table to be returned, it is a non-piped sort. A piped sort usually performs better than a non-piped sort.

To monitor if there are any sort overflows, go to transaction ST04, choose **Performance** → **Database** and choose the **Sorts** tab. See Figure 8-9.

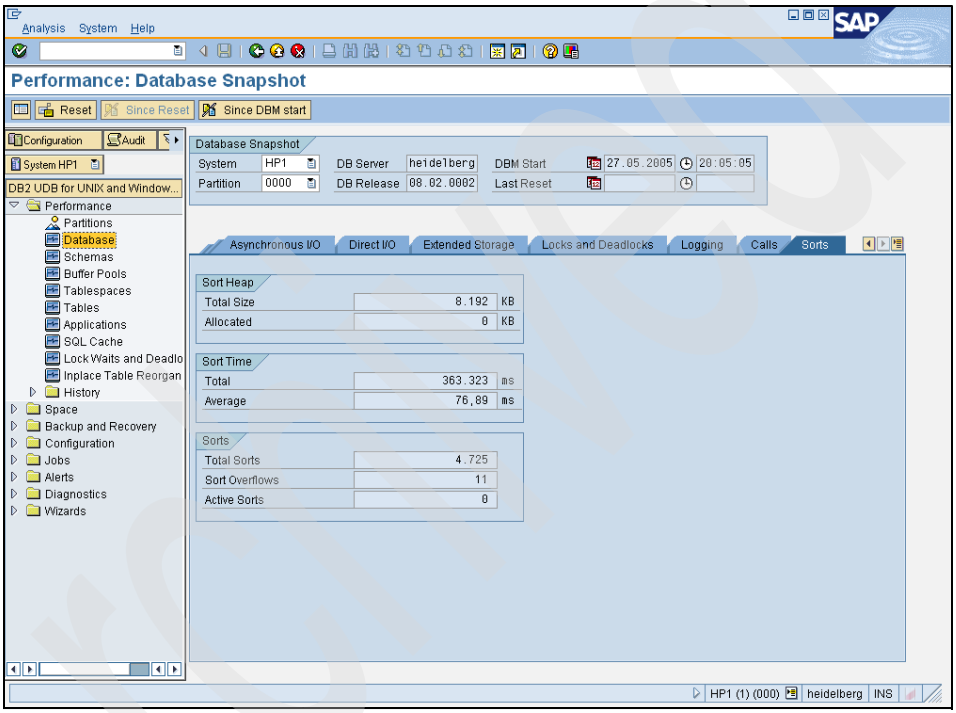


Figure 8-9 ST04: sort overflows

Normally, sort overflows should not exceed 1% of the total sorts. If the number of sort overflows is too high compare to the total sorts, consider increasing the database configuration parameter `SORTHEAP`. In this case, you will have to increase `SHEAPTHRES` as well.

Use database manager snapshot to monitor the piped sorts and *post threshold sorts*. Go to transaction `ST04`, choose **Configuration** → **CLP Commands.** In the function box, choose **002:GET SNAPSHOT FOR DATABASE MANAGER**. See Figure 8-10.

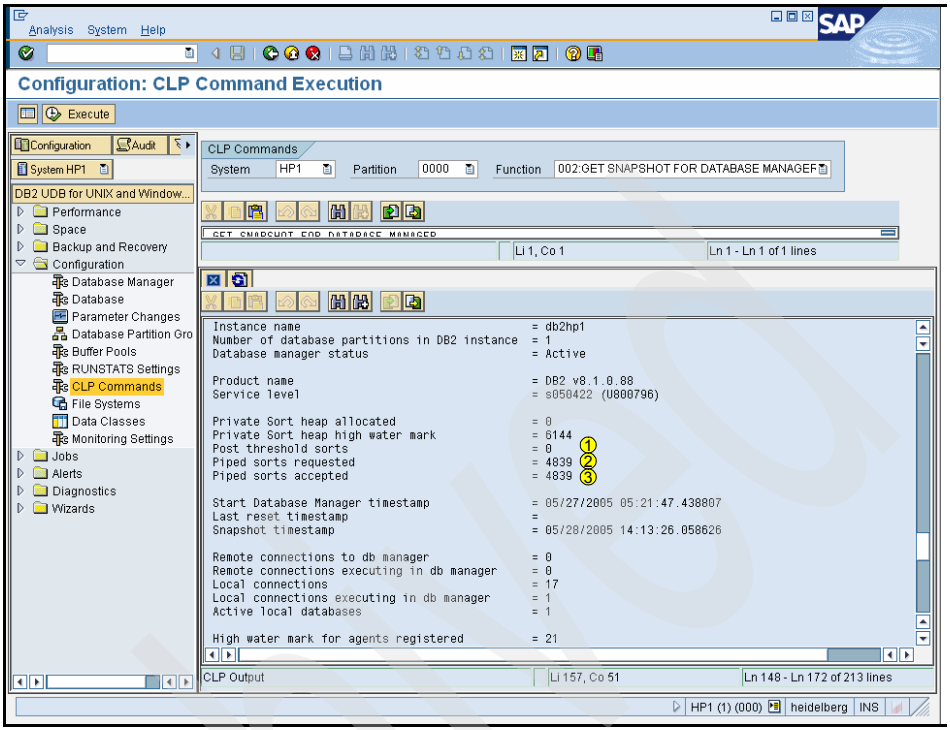


Figure 8-10 ST04: piped sorts and post threshold sorts

1. *Post threshold sorts*: Number of sorts that have required heaps after the sort heap threshold has been reached.
2. *Piped sorts requested*: Number of piped sorts that have been requested.
3. *Piped sorts accepted*: Number of piped sorts that have been accepted.

Deadlocks and lock escalations

Lock escalation occurs when the total number of locks held by an application reaches the maximum amount of lock list space available to the application, or the lock list space consumed by all applications is approaching the total lock list space. Lock escalations can lead to deadlocks or lock time outs. Refer to “Deadlock” on page 434 for more information.

To monitor lock escalations, go to transaction ST04, choose **Performance** → **Database** and choose the **Locks and Deadlocks** tab. See Figure 8-11.

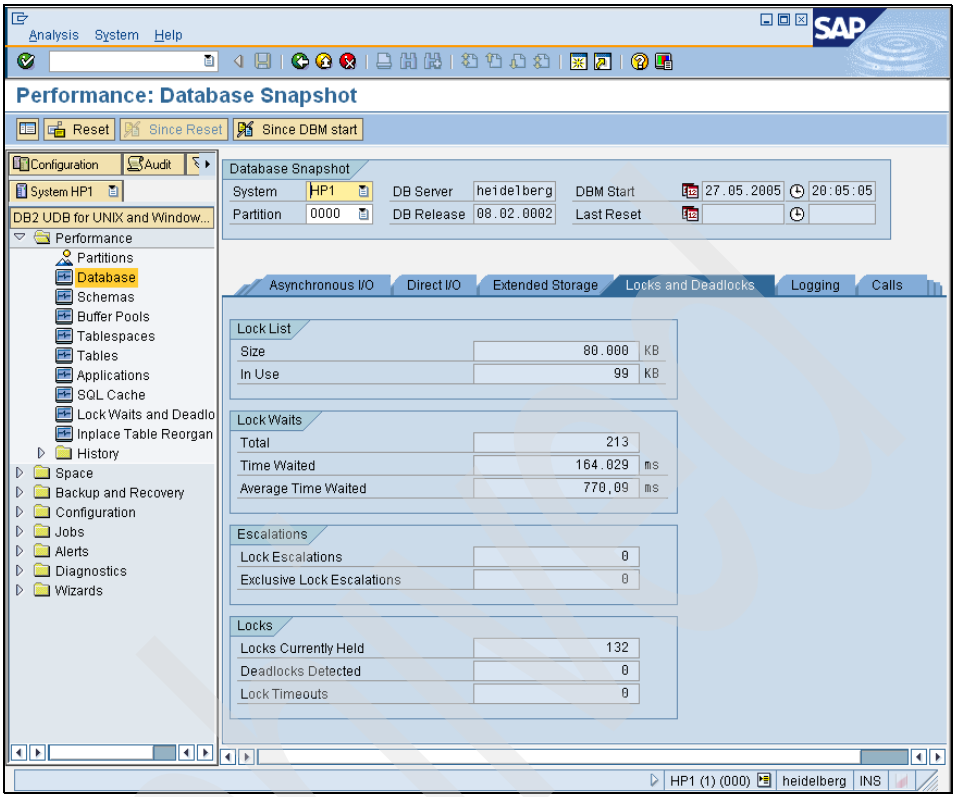


Figure 8-11 ST04: Deadlocks and lock escalations

Normally, lock escalation should be equal to 0. If you see high number of lock escalations, you may want to consider to increase the `LOCKLIST` configuration parameter value.

For more information on how to diagnose and analyze a deadlock situation, refer to “Deadlock” on page 434, “DBSL deadlock trace and db6util trace” on page 517.

ST04: SQL Cache - finding expensive statements

SQL Cache in `ST04` is a useful tool for finding expensive statements in your running system. Unlike the ST05: SQL Trace (discussed in “ST05: Performance Trace - SQL Trace” on page 490), which focuses on a particular transaction or report; ST04: SQL Cache tells you information about the runtime of the dynamic SQL statements in the SQL cache of the database that has been running in your system from all applications. By using the ST05: SQL trace, you can find out the most expensive statements in your whole system that require more analysis.

What is an expensive SQL statement?

When we talk about an “expensive” SQL statement, there are actually several levels of meaning. For the database manager to process a SQL statement, the database requires different kinds of resources:

- CPU resource: All processes (or threads, especially DB2 agents) involved in the execution of the SQL statement consume CPU cycles.
- Physical memory: During the process of running a SQL statement, memory has to be allocated from DB2 agent private memory like statement heap, query heap, sort heap and pages from buffer pool are needed.
- Physical disk storage: All data that is not readily available from the buffer pool has to be retrieved from the storage.

And the amount of resources needed to satisfy the processing of a SQL statement depends on:

- The size and the number of the objects (tables, indexes) involved.
- The complexity of the statement.
- The access method chosen by the optimizer (for example, table scan/index scan).

Statements that consume a lot of resources usually take a long time to run as well. We use execution time to measure the resource consumption.

To perform analysis using ST04: SQL Cache

To perform analysis using the SQL Cache, go to transaction ST04, choose **Performance → SQL Cache**. If you choose the default in the Selection Criteria popup box (the role of Selection Criteria in the analysis is described in the next section), you see the following screen (Figure 8-12).

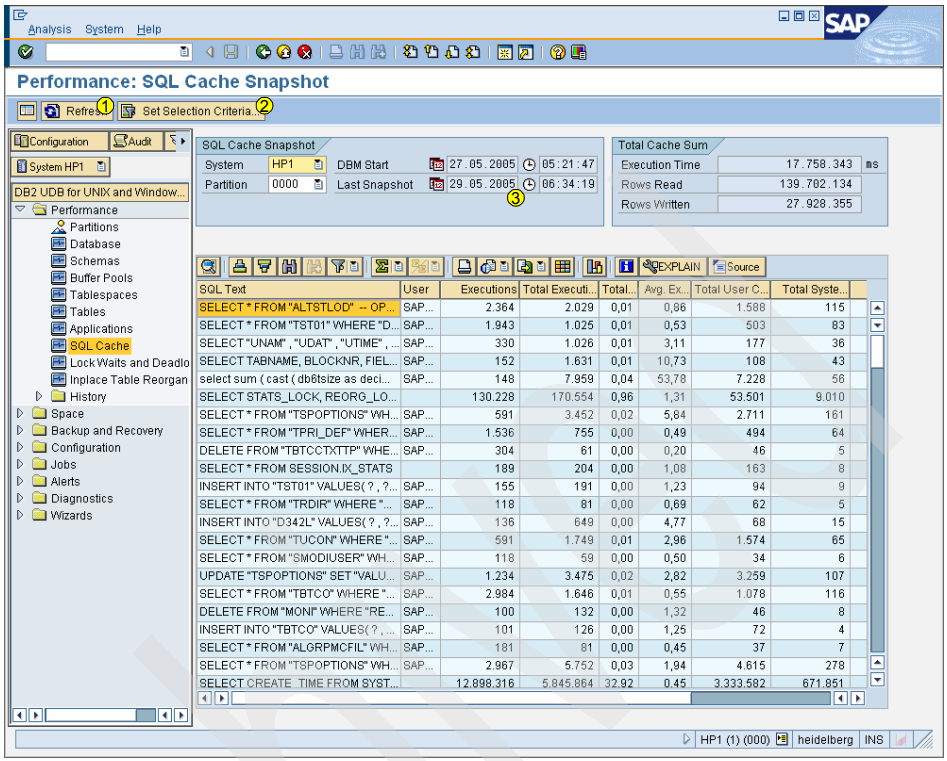


Figure 8-12 ST04: SQL Cache - main screen

Here is a description of the components of the ST04: SQL Cache main screen:

1. *Refresh* button: Retrieve the latest SQL cache from the database and stores this in an SAP table.
2. *Set Selection Criteria* button: Display the output of the SQL cache which is stored in the SAP system. This selection does not get the latest SQL cache from the database.
3. *Last Snapshot* field: The timestamp of the SQL cache output that is stored in the SAP system.
4. *Explain* button: Display the chosen execution or access plan and all associated information
5. *Source* button: Display the ABAP source code where the SQL statement comes from.

Important: Make sure that an SQL cache analysis is always performed in a "warm" system. You should ensure that there are already a significant number of statements that have been executed in order to see the representative content of the SQL cache. As a rule of thumb, at least 1,000,000 SELECT statements should have been executed in the system before starting an analysis. You can find out how many SELECT statements have been executed in your SAP system by going to ST04: **Performance** → **Database** under the tab *Calls*.

Approaches in performing SQL Cache analysis

There are different approaches in performing the SQL Cache analysis depending on which options in the popup screen *Set Select Criteria* you choose:

- Specifying `Executions >= 1` without other criteria (see Figure 8-13), is the most general approach. It displays the top 300 statements with regard to the `Total execution time (ms)` (this is an indicator of how expensive a statement is).

  We modify from the default of `Executions >= 100` to `Executions >= 1` because we also want to see those statements that are not executed very often (< 100 executions) yet consume a lot of system resources (those ones that have a high average execution time) and are therefore still considered expensive.

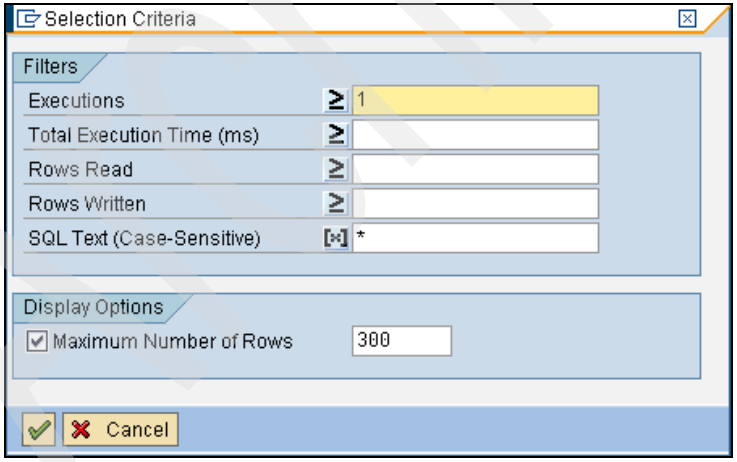


Figure 8-13 st04: SQL cache analysis - General approach

After confirming the select, you see a list of the SQL statements in the SQL cache. See Figure 8-14.

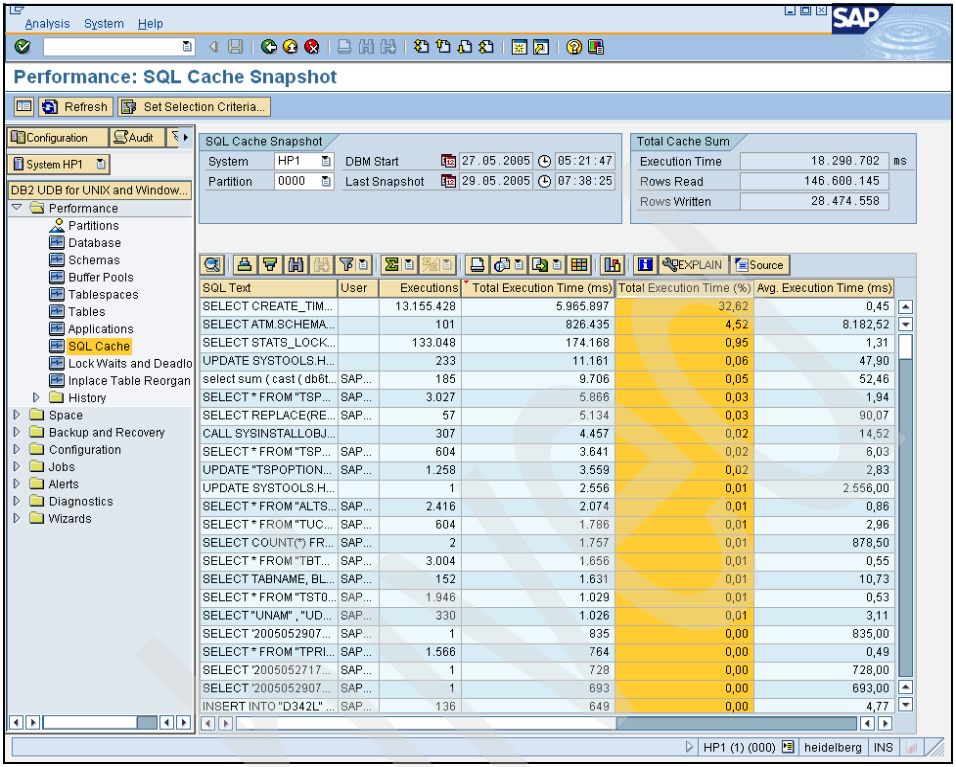


Figure 8-14 st04: SQL cache analysis - General approach - analysis

When using the general approach to perform a SQL Cache analysis, the SQL Cache content should be sorted in descending order by the column *Total Execution Time (ms)* (the column *Total Execution Time (%)* will then also be sorted in descending order). This sorting order emphasize those statements that have a greater share on the total response time. You should analyze the statements that have *Total Execution Time (%)* greater than 5%.

Use the values for *Avg. Execution Time (ms)* together with the number of *Executions* to determine if the statement has a large share of the *Total Execution Time* due to the following conditions:

- condition #1: The average execution time of a single statement is long. With a sufficient number of executions, the share on Total Execution Time is high.
- condition #2: Even though the average execution time of a single statement is not too long; because of the large number of executions, the share on Total Execution Time is high.

Both conditions need to be further analyzed to tune up the statements. In general, you should pay extra attention to the statements under condition #2 since these frequently-run statements impact your overall system performance.

The other area you should focus on when you are performing an SQL Cache analysis using the general approach analysis is sorting. You should check for statements that have a high number of sorts. Sort the SQL Sorts column in descending order. See Figure 8-15.

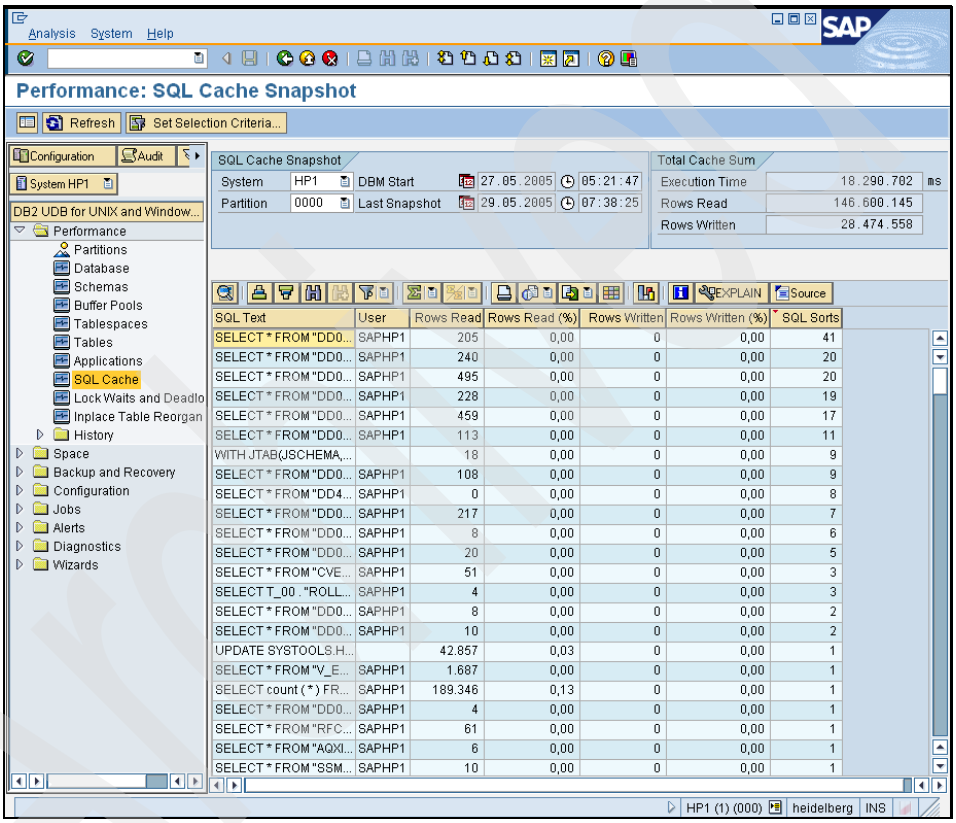


Figure 8-15 st04: SQL cache analysis - General approach - SQL sorts

A high number of sorts can be an indication that an appropriate index is not available or not used. For `SELECT` statements, the value in column *Rows Written* shows the number of rows inserted, updated and deleted in *temporary tables*. A high number here points to large sorts that should be analyzed in detail.

- On top of specifying `Executions >= 1`, if you additionally specify `Rows Read >= 1000000` (see Figure 8-16), the focus shifts more to those statements that consume a large amount of buffer pool resources and probably some other resources such as memory, CPU, and storage. This approach of analyzing the SQL trace is especially helpful when the performance of your buffer pool is suboptimal. See Figure 8-16.

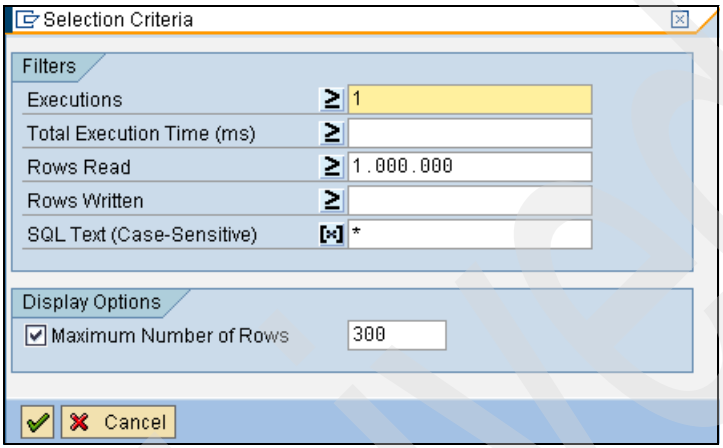


Figure 8-16 st04: SQL cache analysis - focus on large number of rows read

After confirming the select, you see a list of the SQL statements in the SQL cache. Sort the *Rows Read* column in descending order. See Figure 8-17.

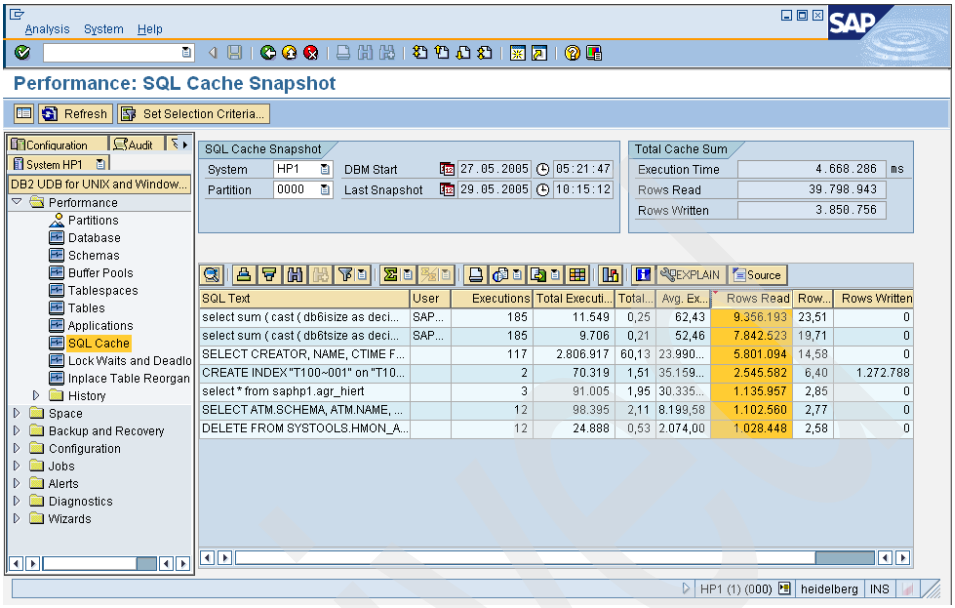


Figure 8-17 st04: SQL cache analysis - focus on large number of rows read - analysis

Then calculate the *average number of rows read per execution* for the top statements (that is, `Rows Read / Executions`). You should further analyze those statements with high average number of rows read. Often times, a high value of average number of rows read per execution indicates that there are table scans or index scans with little or no selectivity.

The identified top statements with regard to “Rows Read” will often be also the top statements regarding the “Execution Time”. But even if that is not the case for some statements, they should be analyzed nevertheless, because they consume a lot of database resources which impacts the performance of other SQL statements.

Attention: A common misconception is that the Rows Read column represents the number of rows in the result set of the query. Actually, the value in the Rows Read column describes the number of *touched* rows during the process of executing a statement.

Note: You can also use the *SQL Text* (Case-Sensitive) field to limit the display to those statements that access a certain table or view. Note that the selection criteria is case sensitive. Usually, if a statement comes from the ABAP coding, the object name is in upper case.

The most common reasons for expensive statements are:

- There is no suitable index for the “where” condition available.
- The suitable index is available but not used.

To find out whether a suitable index is used, use the *Explain* button to examine access plan of the statement of interest. Refer to 6.11.5, “Explain” on page 291 regarding how to use the Explain tool.

ST05: Performance Trace - SQL Trace

Under ST05: Performance Analysis, there is an option to perform SQL Trace, which is useful for performance investigation for a particular long running transaction or report. The SQL Trace option of the Performance Analysis tool allows you to see how the `OPEN SQL statements` that you use in ABAP programs are converted to standard SQL statements and the parameters with which the native SQL statements are passed to the database system.

Tip: If you start the ST05: SQL trace for a background job which is already running for a while, you may not be able to find the needed information any more. In this case, you may want to consider using transaction ST04: SQL Cache or ST04: Applications to gather the information.

Between the time you turn on the trace function and the time you turn it off later, all database activity occurring either for a specific user or for an entire system is recorded. The SAP system take `OPEN SQL statements` and converts them into native SQL statements that it passes to the database and makes the results available. The native SQL statement and its parameters are recorded in the SQL Trace file. The results of the SQL statement, like return code, number of entries retrieved, inserted, or deleted by the database are recorded in the SQL Trace file as well. The log file also contains the runtime of the statement and the places in the application program, and the corresponding transaction, from which it was called which enables additional analyses.

From the recorded SQL trace, you can get the following information:

- Which SQL statements your application carries out.
- Which run time values the system uses for specific database accesses and changes.
- How the system translates ABAP *OPEN SQ*L commands (such as `SELECT`) into standard SQL commands.
- Where your application positions `COMMIT` statements.
- Where your application makes repeated database accesses.
- What database accesses or changes occur in the update section of your application.

Concept of database operations in the SAP environment

SQL Trace records the database operations, some key functions, and measurements used in SQL Trace:

- Logical sequence of database operations:

  A database request is sent to the database with the following sequences. The names of these functions are used in the SQL Trace.

  a. The *DECLARE* function defines and numbers the cursor.

  b. The *PREPARE* function prepares a specific SQL statement (for example, `select * from address where CuNo eq 5`) and defines the access method before the system can transfer the request to the database. During this preparation, the system is concerned only with the structure of the SQL statement and not with the values it contains.

  c. The *OPEN* function takes the prepared `SELECT statement` and completes it with the run time values. In the above example, `OPEN` would issue the field `CuNo` with the value 5; Or, If the SQL statement makes changes in the database (`INSERT`, `UPDATE`, `DELETE`), `PREPARE` is followed by *EXEC*, which executes the statement. It also executes the first fetch operation automatically, transferring the first few rows of the result set into the application server side DB2 client code.

  d. *FETCH* passes the entries from the database to the database interface of the SAP System.

  If the system can refer back to an SQL statement that has already been prepared, there is no `PREPARE` operation, and the statement is executed using `REOPEN` or `REEXEC` as appropriate

- Measured database operations:

  Each SQL statement is broken down into database operations by the SAP System. The SQL Trace allows you to measure the runtime of each of the operations listed in Table 8-1.

Table 8-1 Operations measured in SQL Trace

Name of functions	Meaning
DECLARE	Defines a new cursor within an SAP work process and assigns the SQL statement to it. The short form of the statement is displayed in the list of trace records under statement. The cursor has a unique cursor ID. The Cursor ID is used in communication between the SAP System and the database system.
PREPARE	Converts the SQL statement and determines the access plan.
OPEN	Opens a cursor for a prepared (converted) `SELECT statement`. `OPEN` passes the parameters for the database access. `OPEN` is used only for `SELECT` statements.
FETCH	Passes one or more records selected in the `SELECT` statement to the database interface of the SAP System The data records are identified by the cursor ID.
REOPEN	Reopens a cursor that the system prepared for a `SELECT` statement and transfers the new parameter to the database.
EXEC	Passes the parameters for the database statement, and executes the statements that change data in the database (such as `UPDATE`, `DELETE`, or `INSERT`).
REEXEC	Reopens a statement the system already prepared for a previous `EXEC` statement.

Activating/Deactivating the SQL Trace

To activate or deactivate the SQL Trace, following this procedure:

- Activating the `ST05` Performance Trace/SQL Trace

  Figure 8-18 shows the initial screen of the transaction ST05 Performance.

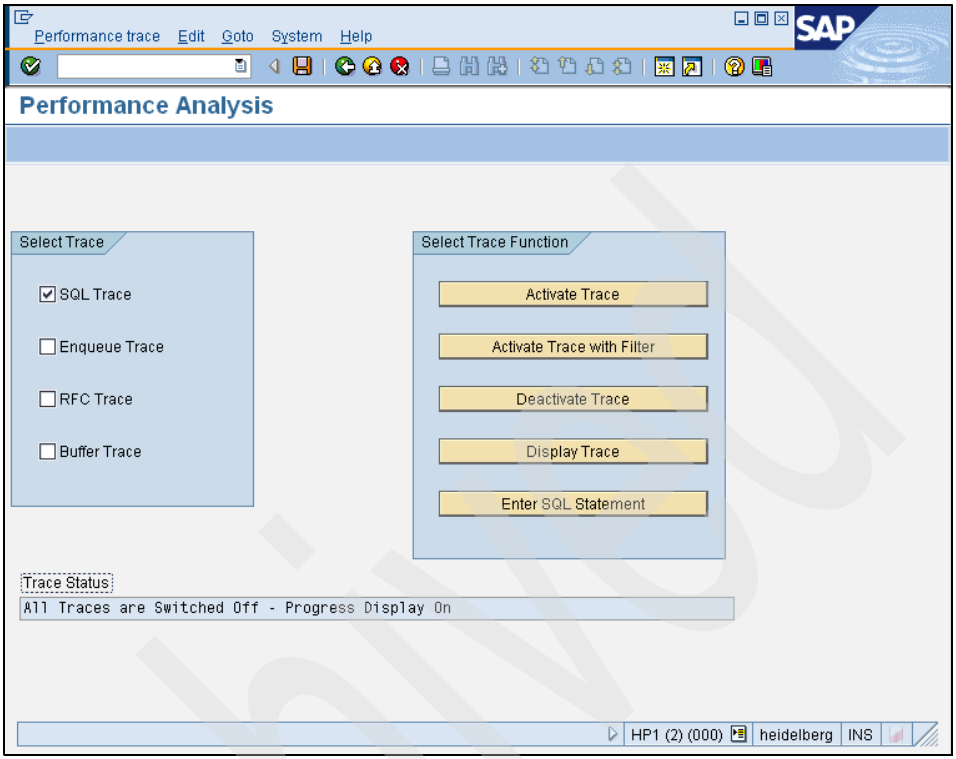


Figure 8-18 ST05 Performance Analysis: initial screen

In the lower part of the screen, the status of the Performance Trace is displayed. This tells you whether any of the Performance Traces are switched on, the users for which they are enabled, and the user that switched them on.

To switch on the SQL Trace:

a. Mark the **SQL Trace** check box on the left panel.

b. To switch on the trace under your user name, choose **Active Trace**.

or

You can choose **Activate Trace with filter** to specify a different user name, the specific program name or transaction name, or a list of table(s) to be included or excluded in the trace.

c. Run the program you want to analyze as normal.

- Deactivating the ST05 Performance Trace/SQL Trace:

Turn off the trace by clicking *Deactive Trace* in the ST05: Performance Analysis initial screen. For performance reasons, you should switch off the trace as soon as you have finished recording.

Displaying the SQL Trace

To display the SQL Trace after running the trace, you click the **Display Trace** button in the initial screen (Figure 8-18 on page 493). Then you have to specify your criteria for viewing the trace in the *Display Filter* screen. Afterwards, you see the screen, *Trace List* (or *Extended Trace List*) that display lists of trace records.

- Display filter:

  In this screen (Figure 8-19), you can choose the duration of the trace, the type of the trace (Trace List, Extended List, Trace List Sorted by Time) and other criteria. Choose **Extended Trace List** if you want to see the time at which the recorded was executed or the name of the program that executed the logged statement.

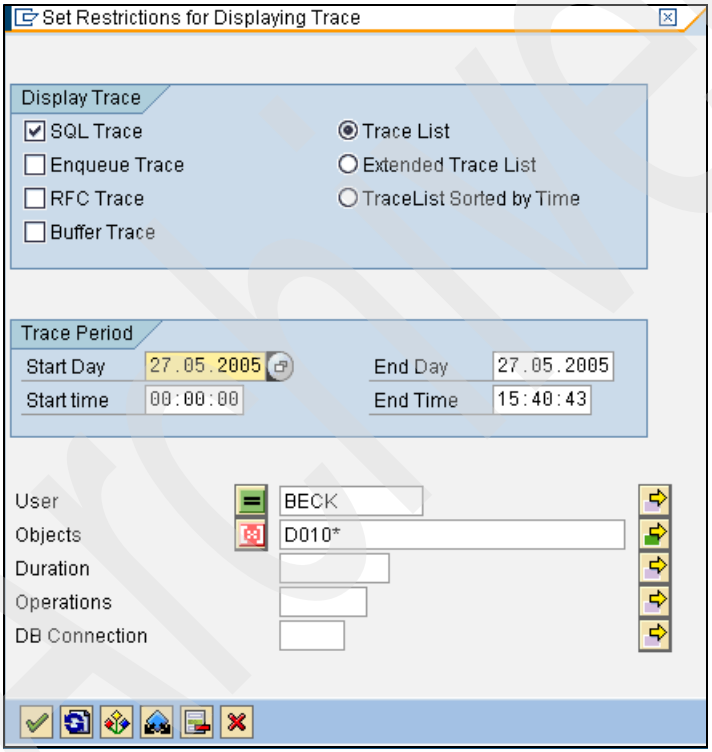


Figure 8-19 ST05 Performance Analysis: display filter

- Displaying Lists of Trace Records:

  This is where you see a list of traces. The kind of list can be the basic *Trace List* or *Extended Trace List* or *Trace List Sorted by Time* depending on what you have chosen in the previous display filter screen. See Figure 8-20 for an example of the Trace List (basic).

The first line of the list contains a subheader, which remains unchanged for all trace records of a program to be analyzed. It contains the following information:

- Name of the transaction
- Process identification number
- Process type
- Client
- User name

The next line contains the following headers:

- *Duration*: Runtime for the statement in the format milliseconds.microseconds. The runtime (duration) is highlighted in the list if the duration is too high. It is a good indication of a slow running query.
- *Object*: Name of the database table.
- *Op.*: Name of the operation to be performed on the database (for example, `PREPARE`, `OPEN`, `FETCH`, as described in “Concept of database operations in the SAP environment” on page 491).
- *Recs*: Number of records retrieved or processed and passed between the SAP System and the database.
- *RC*: Return code of the logged statement. It can tell you if there is any database error.
- *Statement* - Short form of the logged statement.

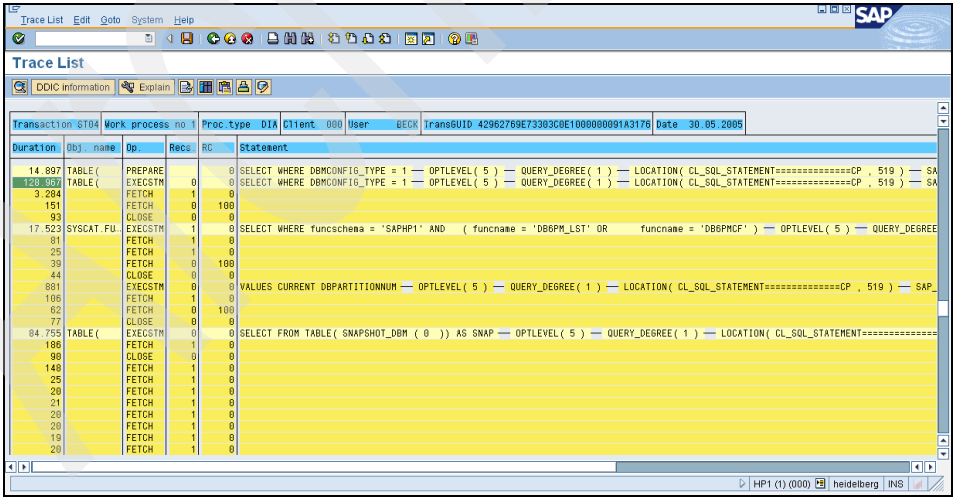


Figure 8-20 ST05 Performance Analysis: Trace List (Basic)

In the Trace List, look for the line that has high duration. If the duration is sufficiently high, the color of the field “duration” changes to red. You may want to pay extra attention to these statements. Use the **Explain** button to analyze why the duration is high. Refer to 6.11.5, “Explain” on page 291 on how to use the Explain tool.

Example: analysis of a slow running program using SQL Trace

In this example, we have a report that runs for a long time and we would like to pinpoint the cause of the problem. We use ST05: SQL Trace for our analysis. You may want to review 6.11.5, “Explain” on page 291 to get yourself familiar to the functionality provided by the Explain tool before proceeding.

1. Since we are going to activate, deactivate, and analyze the trace on one session and run our program on another session. Open two SAP GUI sessions.
2. On one of the SAP GUI session (we call this session #1 from now on), activate the ST05: SQL Trace by going to transaction ST05. Make sure “SQL Trace” is the only one checked for the “Select Trace” panel. Click the **Activate Trace** button (see Figure 8-21).

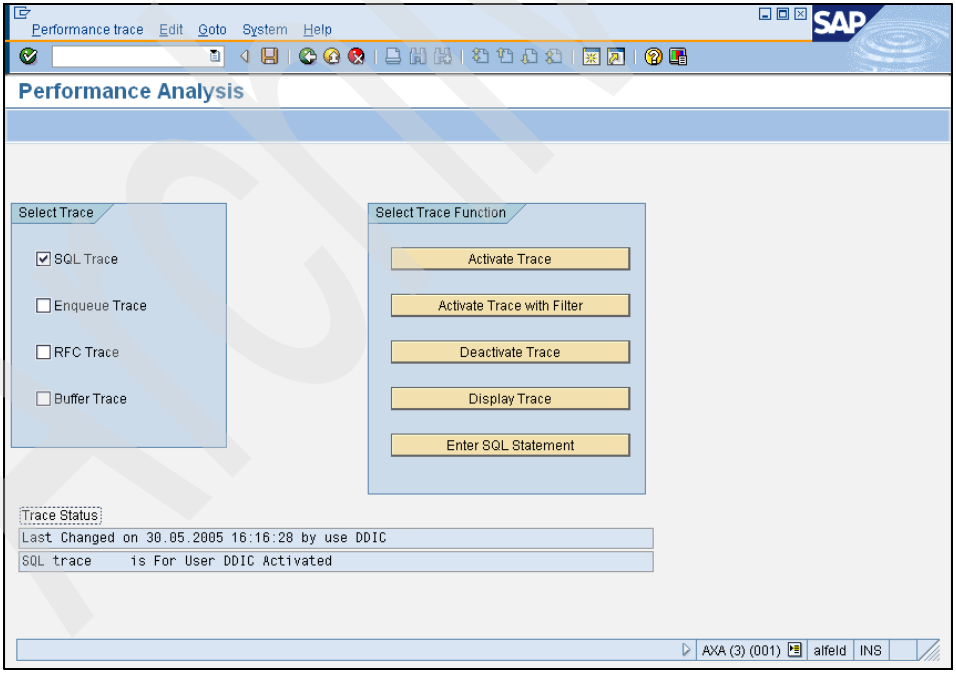


Figure 8-21 ST05 SQL Trace example: ST05 - activate trace

3. In another SAP GUI session (from now on, we call it session #2), go to transaction SE38 to run the slow report. Type in name of the report (see Figure 8-22) in the Program field and press F8 to execute the report. Wait for the report to finish.

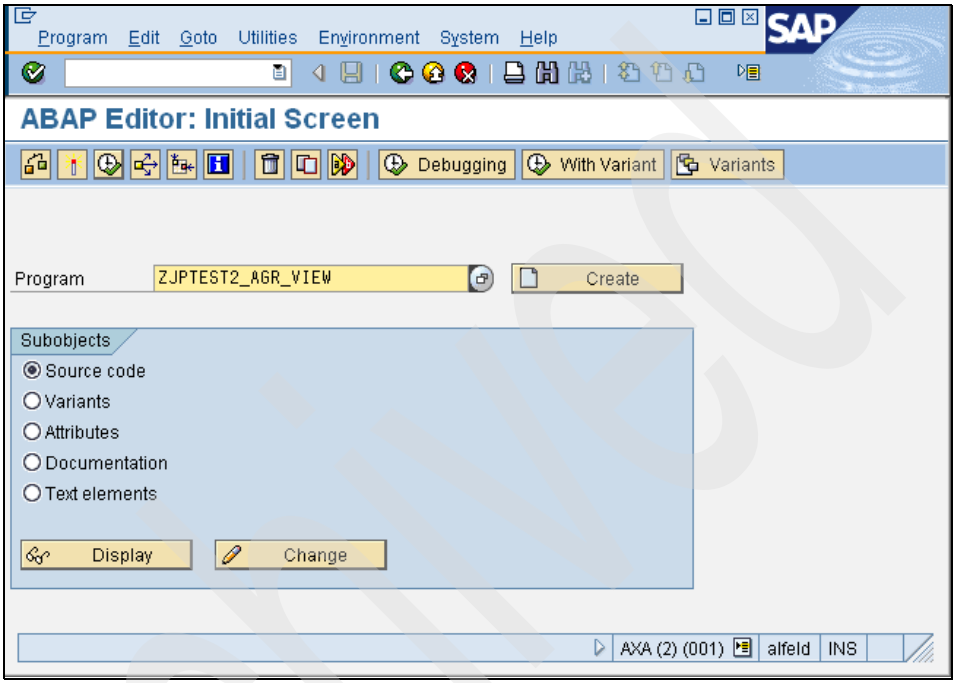


Figure 8-22 ST05 SQL Trace example: SE38 - run the slow program

4. After the report finishes running, go back to session #1 to switch off the trace by clicking the **Deactivate Trace** button. See Figure 8-23.

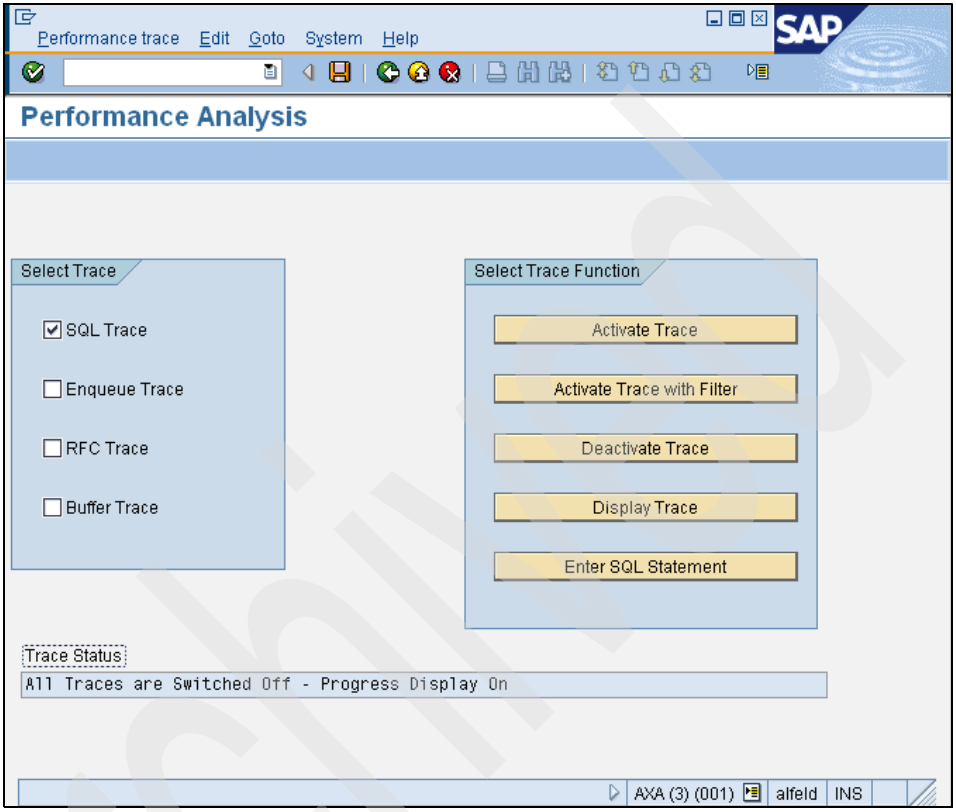


Figure 8-23 ST05 SQL Trace example: ST05 - display trace

5. In session #1, click the **DISPLAY Trace** button to launch the *Set Restrictions for Displaying Trace* (display filter) dialog box (Figure 8-24). Accept the default setting and continue.

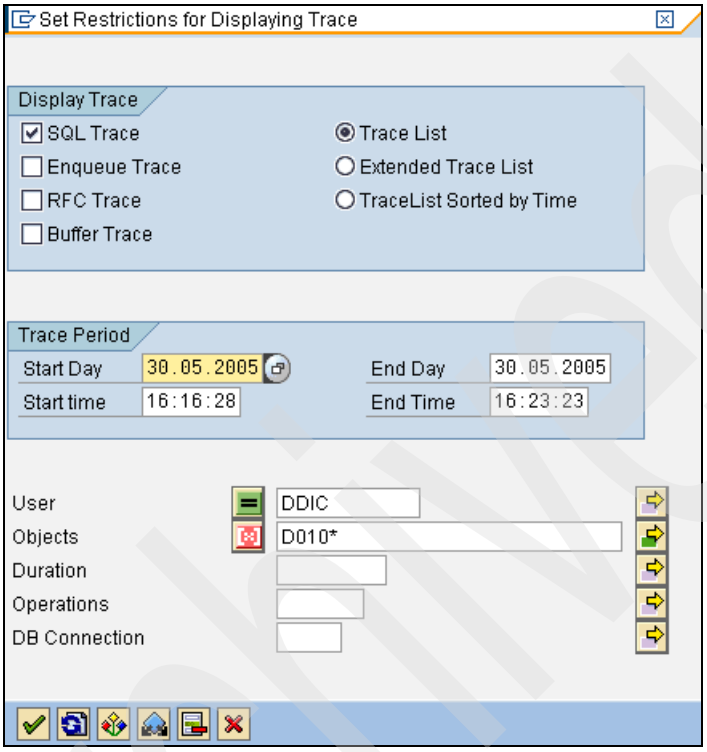


Figure 8-24 ST05 SQL Trace example: ST05 - display filer

6. Now, in session #1, you see the *Trace List* screen (Figure 8-25). Searching down the list, we found a statement that take a long time to OPEN: `SELECT WHERE "MANDT" = '001' AND "TEXT" = 'Systemnachrichten'` with a “red” duration of 4535654 microseconds (or ~ 4.5 seconds!). The slow query could be caused by a suboptimal access plan.

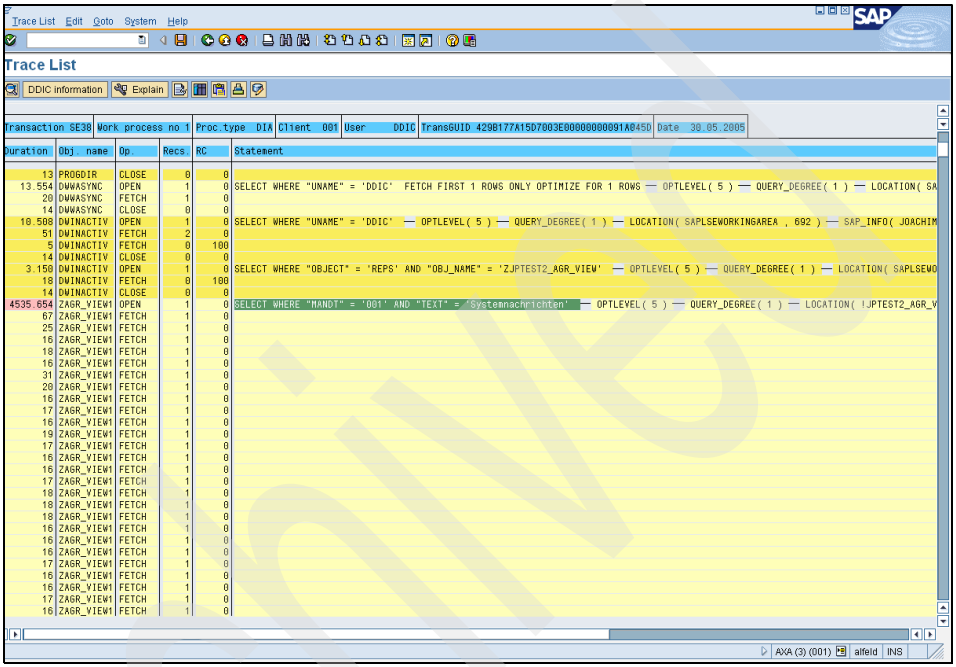


Figure 8-25 ST05 SQL Trace example: ST05 - trace list

7. Examine the access plan by choosing the line of the statement and click the **Explain** button (Figure 8-26). From the access plan, we see that this is a query against a view that is based on two tables (`AGR_HIERT` and `AGR_HIER`). DB2 UDB optimizer proposes a plan that does an index scan (IXSCAN) on each table (the index, `AGR_HIERT~0` for the table `AGR_HIERT`; and the index, `AGR_HIER~0` for the table `AGR_HIER`). Then a nested loop join (NLJOIN) is used to join the tables together.

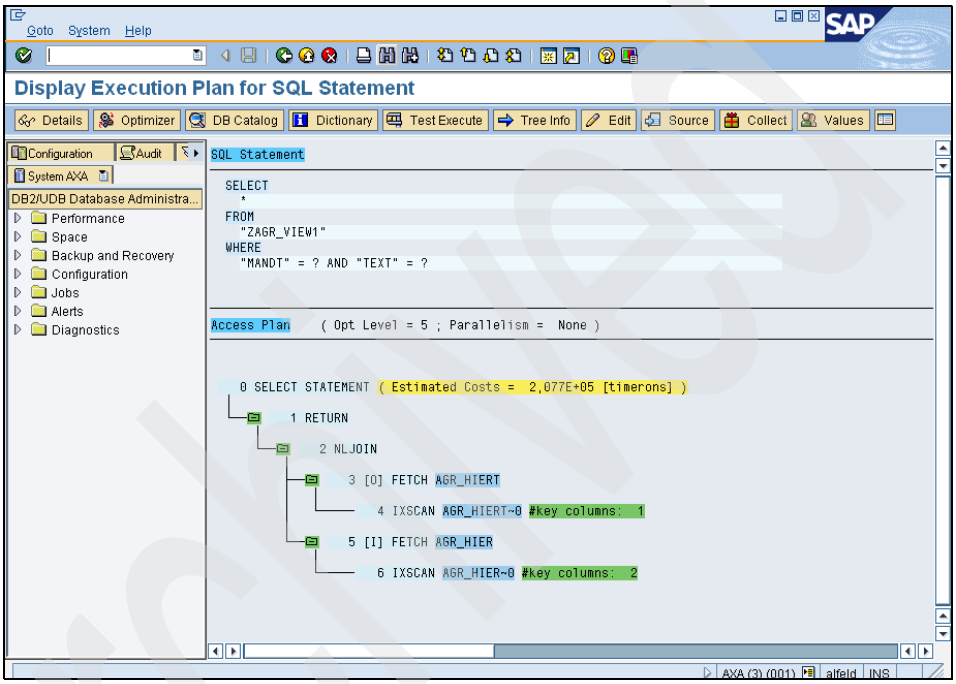


Figure 8-26 ST05 SQL Trace example: ST05 - Explain plan

8. Let's inspect the indexes that are used in this plan. Click the AGR_HIERT~0 index and click the **DB Catalog** button. You see the definition of the index AGR_HIERT~0 (see Figure 8-27). It consists of four columns: MANDT, AGR_NAME, SPRAS and OBJECT_ID. The index AGR_HIERT~0 only satisfies one of the predicates in the WHERE clause ("MANDT" = ?).

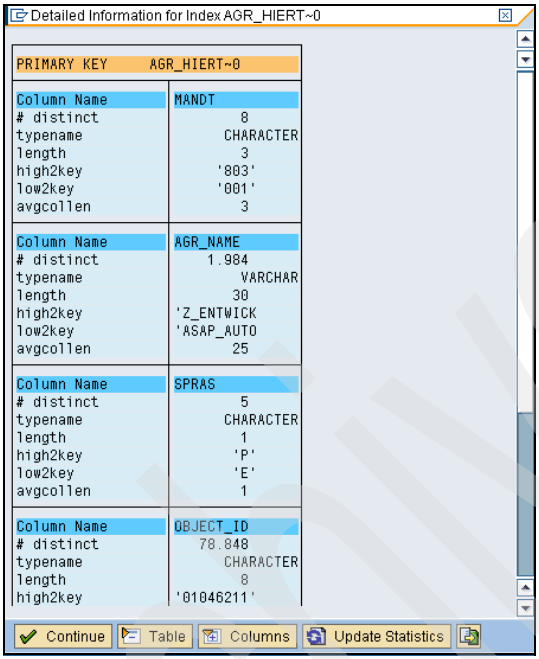


Figure 8-27 ST05 SQL Trace example: ST05 - DB Catalog for the index AGR_HIERT~0

9. Now inspect the AGR_HIER~0 index for the AGR_HIER table. Click the AGR_HIER~0 index and click the **DB Catalog** button. You see the definition of the index AGR_HIER~0 (see Figure 8-27). It consists of three columns: MANDT, AGR_NAME, and OBJECT_ID. Index AGR_HIER~0 only satisfies one of the predicate in the WHERE clause ("MANDT" = ?). As we can see from step 7 and step 8, neither of these two indexes AGR_HIER~0 nor AGR_HIERT~0 satisfy the predicate "TEXT" = ?. Therefore, the database cannot find an index to satisfy the predicate "TEXT" = ?

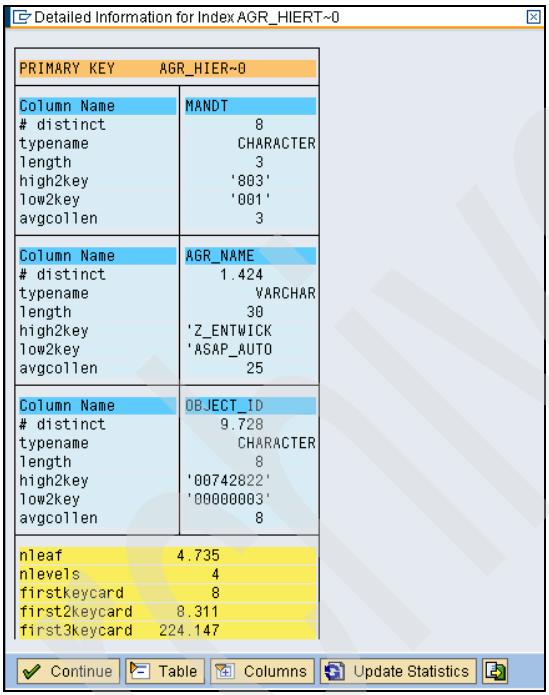


Figure 8-28 ST05 SQL Trace example: ST05 - DB Catalog for the index AGR_HIER~0

10. Now, let's look at in the data dictionary of the tables. Click the table AGR_HIERT and click the **Dictionary** button. This brings you to the definition of the table AGR_HIERT in the data dictionary. See Figure 8-29.

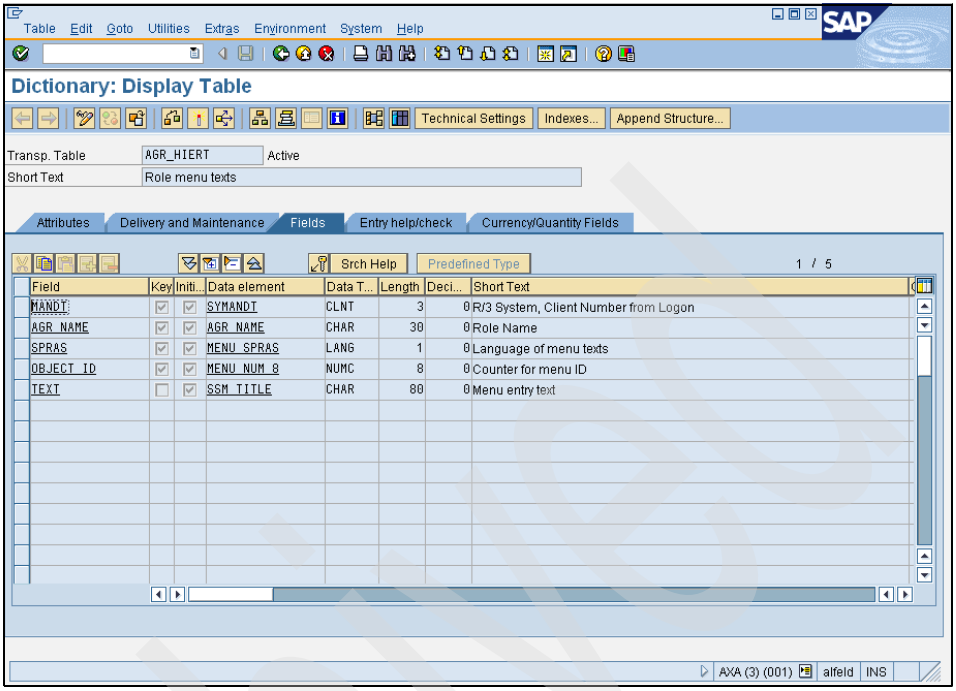


Figure 8-29 ST05 SQL Trace example: ST05 - dictionary for the table AGR_HIERT

11. In the *Dictionary: Display Table* screen for the table AGR_HIERT, click the **Indexes** button to check what indexes are created for this table (note that the primary index AGR_HIERT~0 is not shown here). See Figure 8-30. There is one index called ZT defined. Let's take a look at the definition of this index by double-clicking the line of the index.

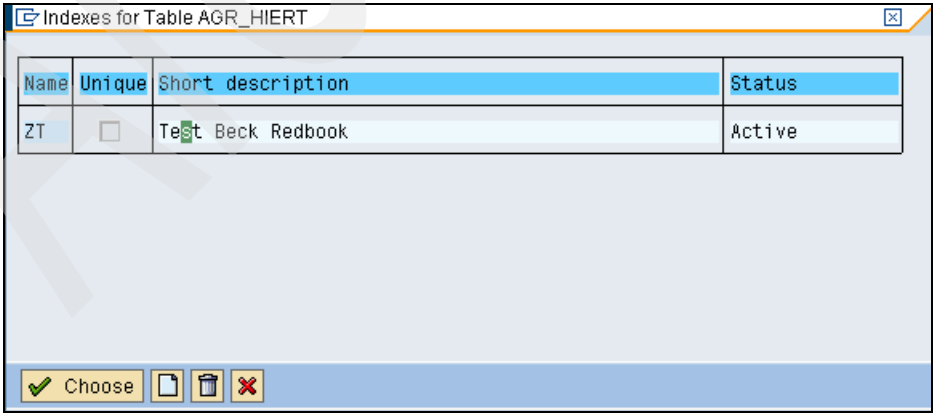


Figure 8-30 ST05 SQL Trace example: ST05 - index dialog for the index ZT

12. The *Dictionary: Display Index* screen shows up (Figure 8-31). From the Index field list at the bottom of the screen, we see that this index consists of the fields: `TEXT` and `MANDT`. This index contains the fields that would satisfy both predicates in the `WHERE` list (`"MANDT" = ?` and `"TEXT" = ?`). The problem is that the index does not exist in the database (Note the warning with `Index does not exist in database system DB6`). This index might have been dropped on the database accidentally before (but the definition still exists in the SAP data dictionary).

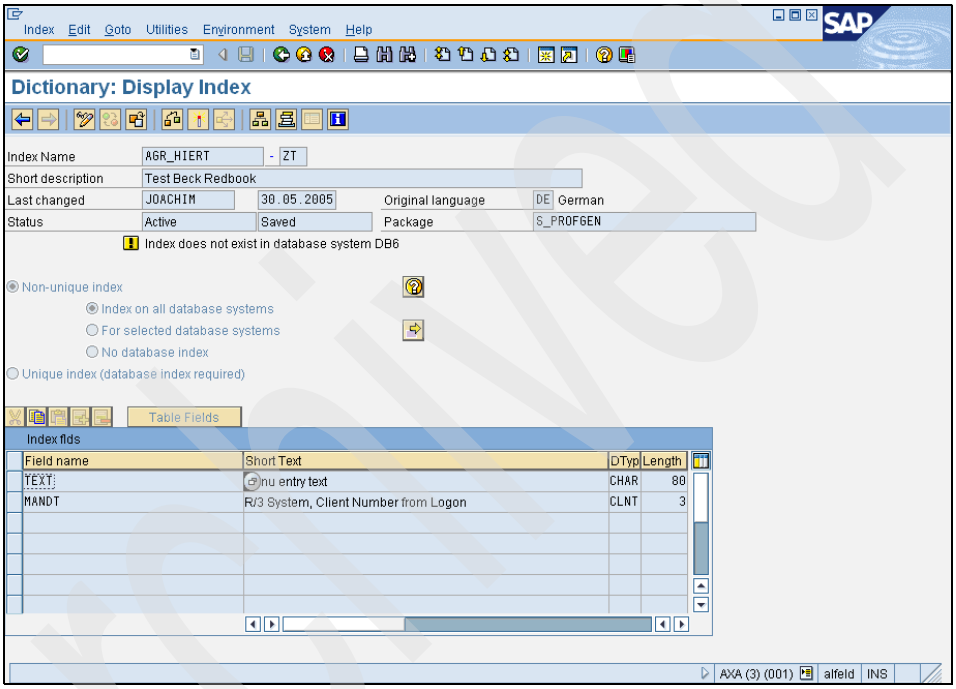


Figure 8-31 ST05 SQL Trace example: ST05 - display index definition for index ZT

13.). We are going to create the index in the database level. In session #2, go to transaction `SE14` (ABAP Dictionary: Database Utility), type in the table name `ARG_HIERT` in the *Obj. name* field. See Figure 8-32. Then click the **Edit** button.

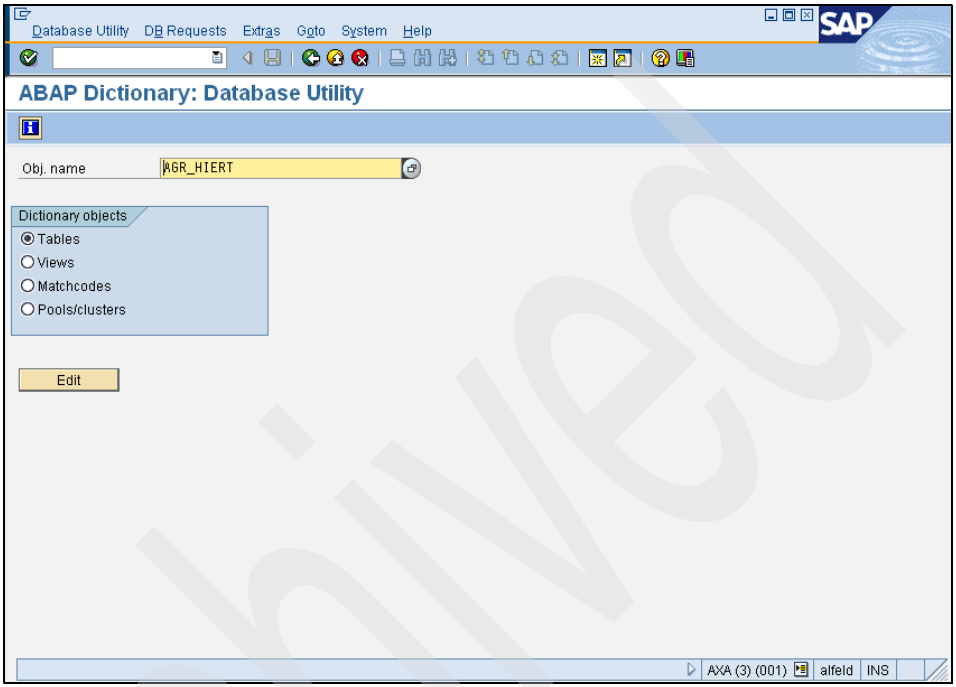


Figure 8-32 ST05 SQL Trace example: SE14 - Edit the table AGR_HIERT

14. You see the *ABAP Dictionary: Utility for Database Table* screen (Figure 8-33). Click the **Indexes** button to modify the index. In the popup screen (see Figure 8-34), choose the **ZT** index by double-clicking the row.

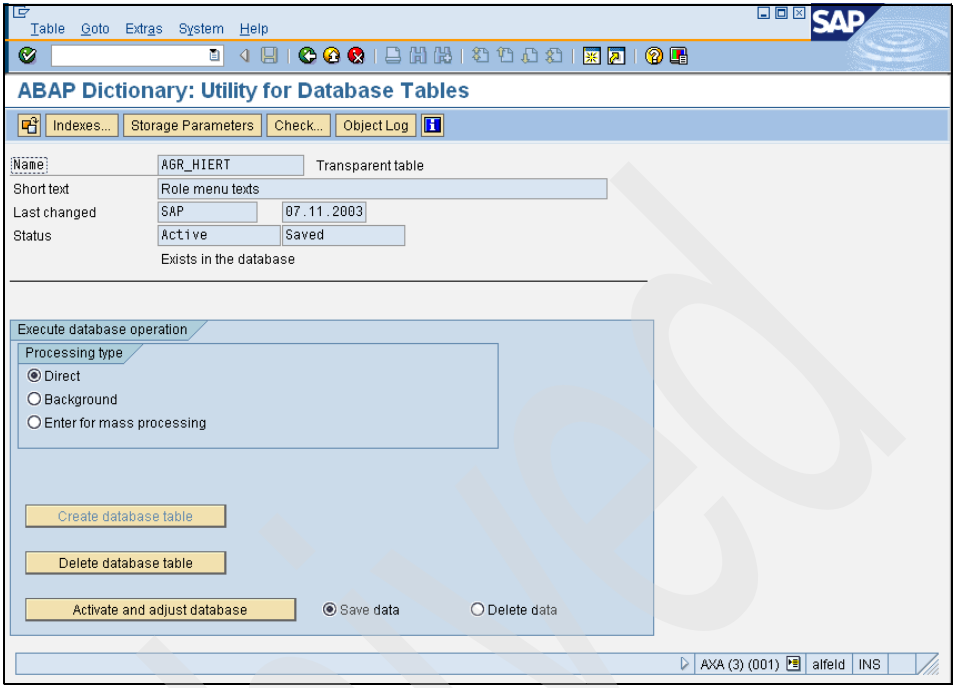


Figure 8-33 ST05 SQL Trace example: SE14 - inspecting the index

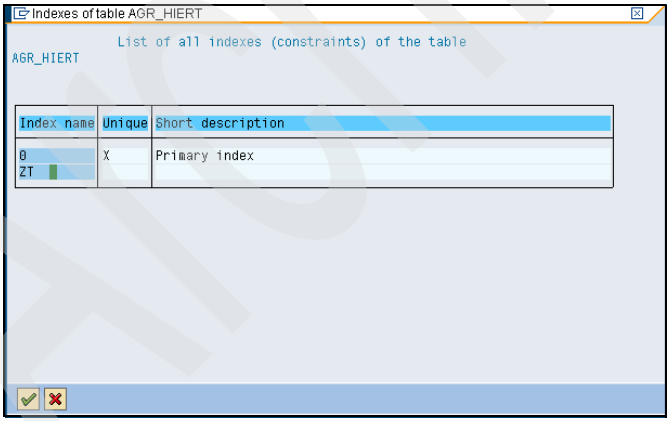


Figure 8-34 ST05 SQL Trace example: SE14 - choose the index ZT

15. The screen *ABAP Dictionary: Utility for Database Indexes* shows (Figure 8-35). (Note the comment below the Status field: "`Does not exist in the database`". It means that the index does not exist in the database level.) The creating index process takes a few minutes.

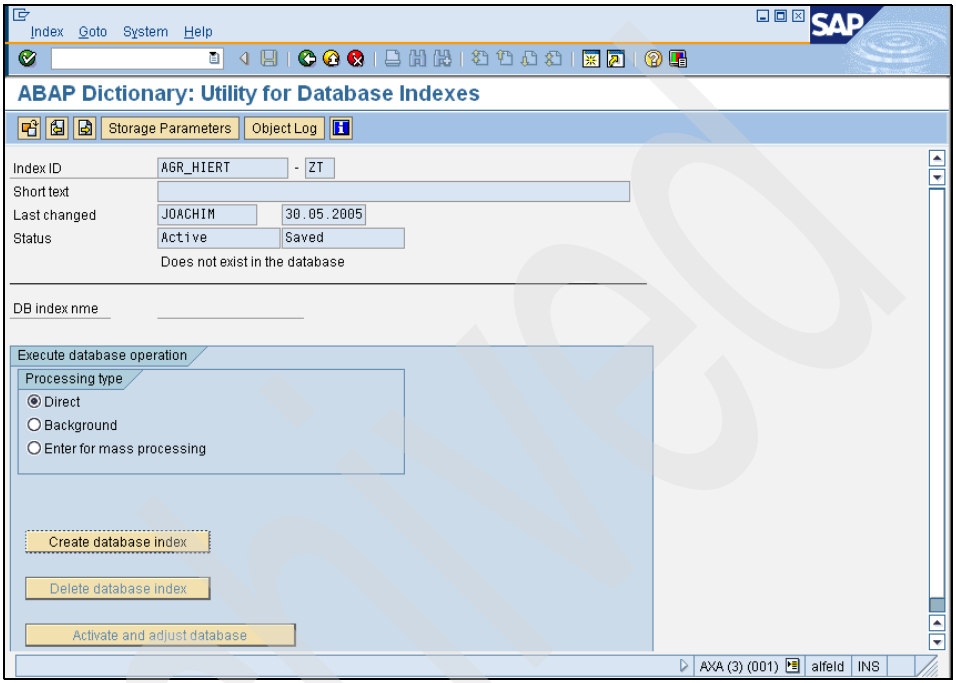


Figure 8-35 ST05 SQL Trace example: SE14 - create the index ZT in the database level

16. Now we have to update the statistics of the index AGR_HIERT~ZT (this is the full name of the index ZT). In session #2, go to the transaction DB20, enter the table name AGR_HIERT in the *Name* field. Then click the **Indexes** tab. Click the right arrow to choose the AGR_HIERT~ZT index (see Figure 8-36).

    Then click the **RUNSTATS in dialog** button to perform the update for the statistics on this index. After the RUNSTATS job finishes, you see that the statistics are updated (see Figure 8-37).

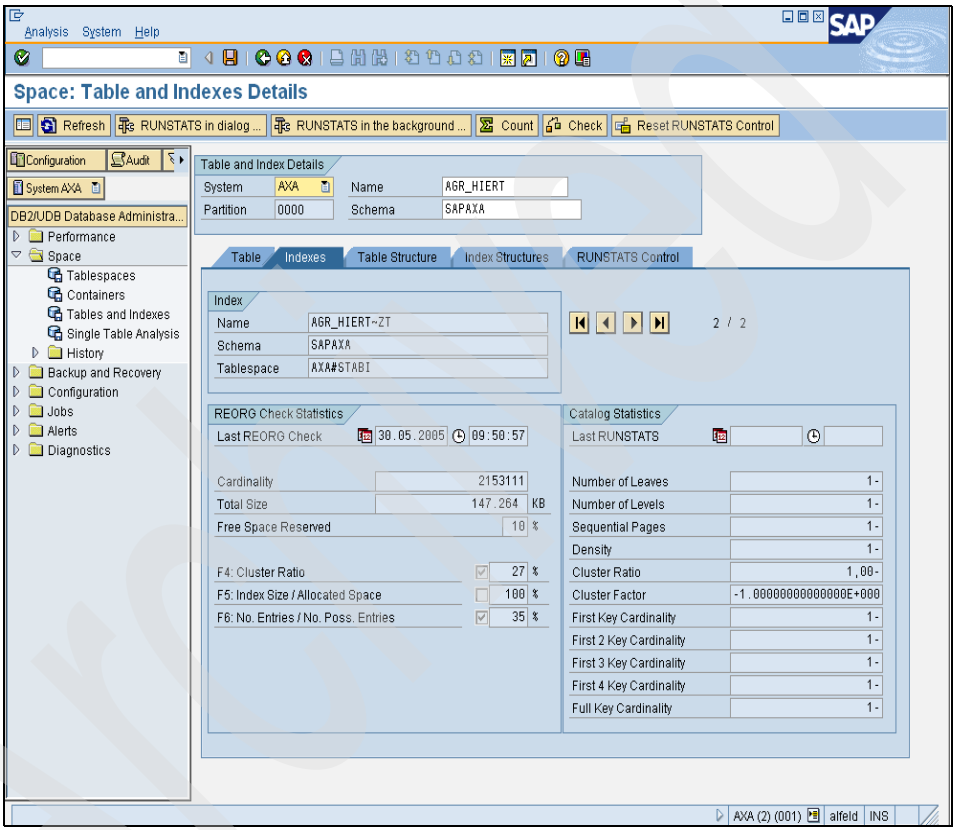


Figure 8-36 ST05 SQL Trace example: DB20 - before the statistics for index AGR_HIERT~ZT is updated

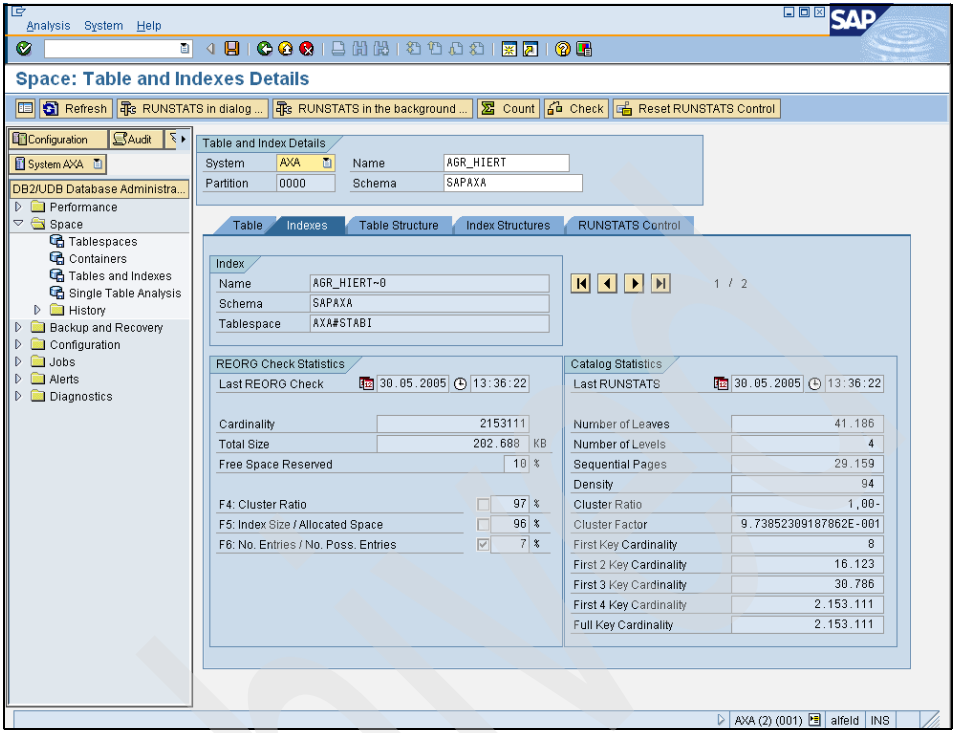


Figure 8-37 ST05 SQL Trace example: DB20 - after statistics for the index AGR_HIERT~ZT is updated

17. Now the DB2 optimizer should be able to take advantage of the `AGR_HIERT~ZT` index. Repeat Step 2 to Step 6 to check if we see any improvements. See Figure 8-38 for the trace list of the tracing of the previously slow running program. The duration dropped to 9266 microseconds (from 4535654 microseconds!).

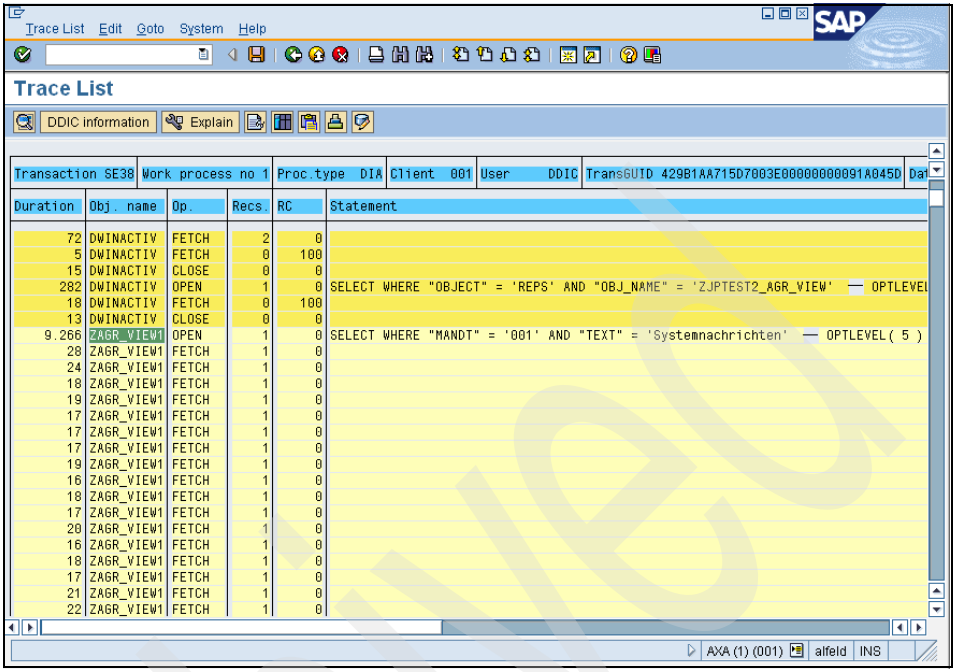


Figure 8-38 ST05 SQL Trace example: ST05: Trace List after creating the index AGR_HIERT~ZT

18. Let's click the **Explain** button to check if the index `AGR_HIERT~ZT` is indeed used. See Figure 8-39. As seen in the access plan, we see that the index `AGR_HIERT~ZT is` used for the two predicates in the `WHERE` list (`"MANDT" = ?` and `"TEXT" = ?`). That is why we see the dramatic improvement in the performance.

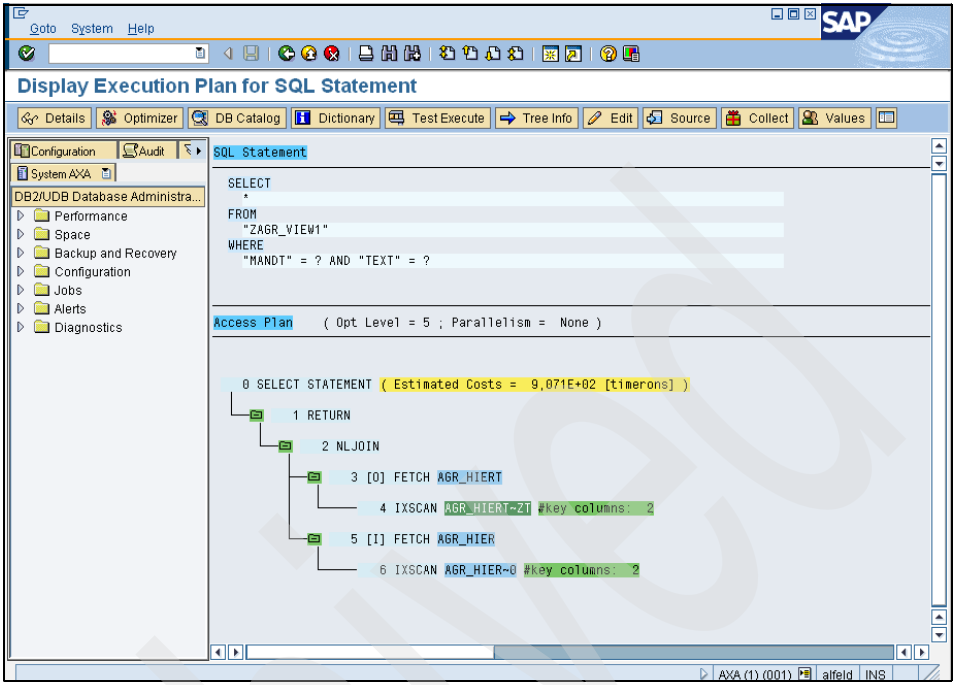


Figure 8-39 ST05 SQL Trace example: ST05 - Explain after creating the index AGR_HIERT~ZT

As you see in the preceding example, ST05: SQL Trace is a very useful tool in identifying expensive statements for a transaction or report; and with the integrated link to other features like inspecting the access plan in Explain and accessing to the data dictionary for checking the table/index definition, we can gather a lot of hints to find out the cause of slow statements.

Developer trace

Each work process writes a developer trace file (`dev_w*`) into the following directory:

- UNIX: `/usr/sap/<SID>/<instance>/work`
- Windows: `<DRIVE>:\usr\sap\<SID>\<instance>\work`

This developer trace does not have to be turned on explicitly. DBSL trace is always running in the background with *trace level* set to 1.

You can access the trace file by going to *transaction SM50: Process Overview*. Highlight the work process that ran the transaction/report that generates the error and choose the menu option: **Process** → **Trace** → **Display File.** See Figure 8-40.

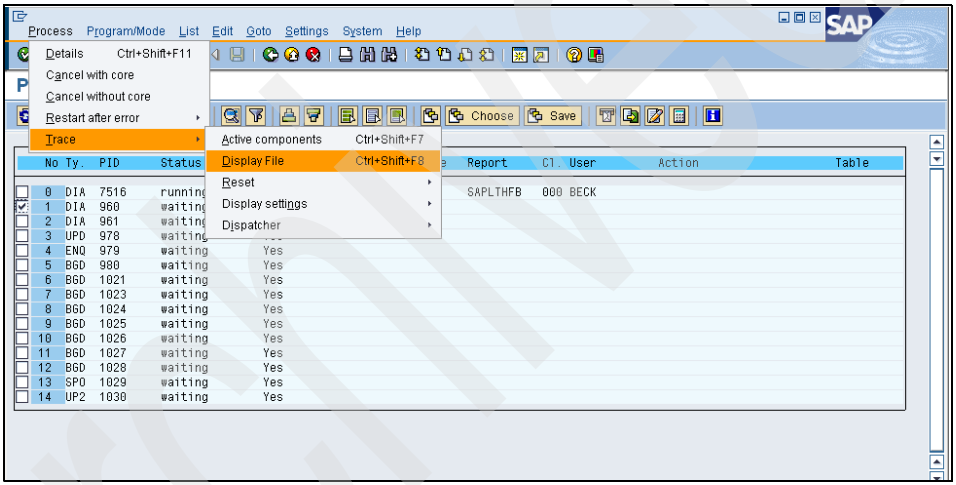


Figure 8-40 displaying developer trace files from SM50: Process Overview

With the default trace level of 1, the trace only dumps information when errors occur.

Now, we illustrate how to read the develop trace with an example. In the example, we have invoked a CLI error. We show how to use the developer trace to help us determine the cause of the error.

developer trace: header

Figure 8-41 shows the header of the developer trace.

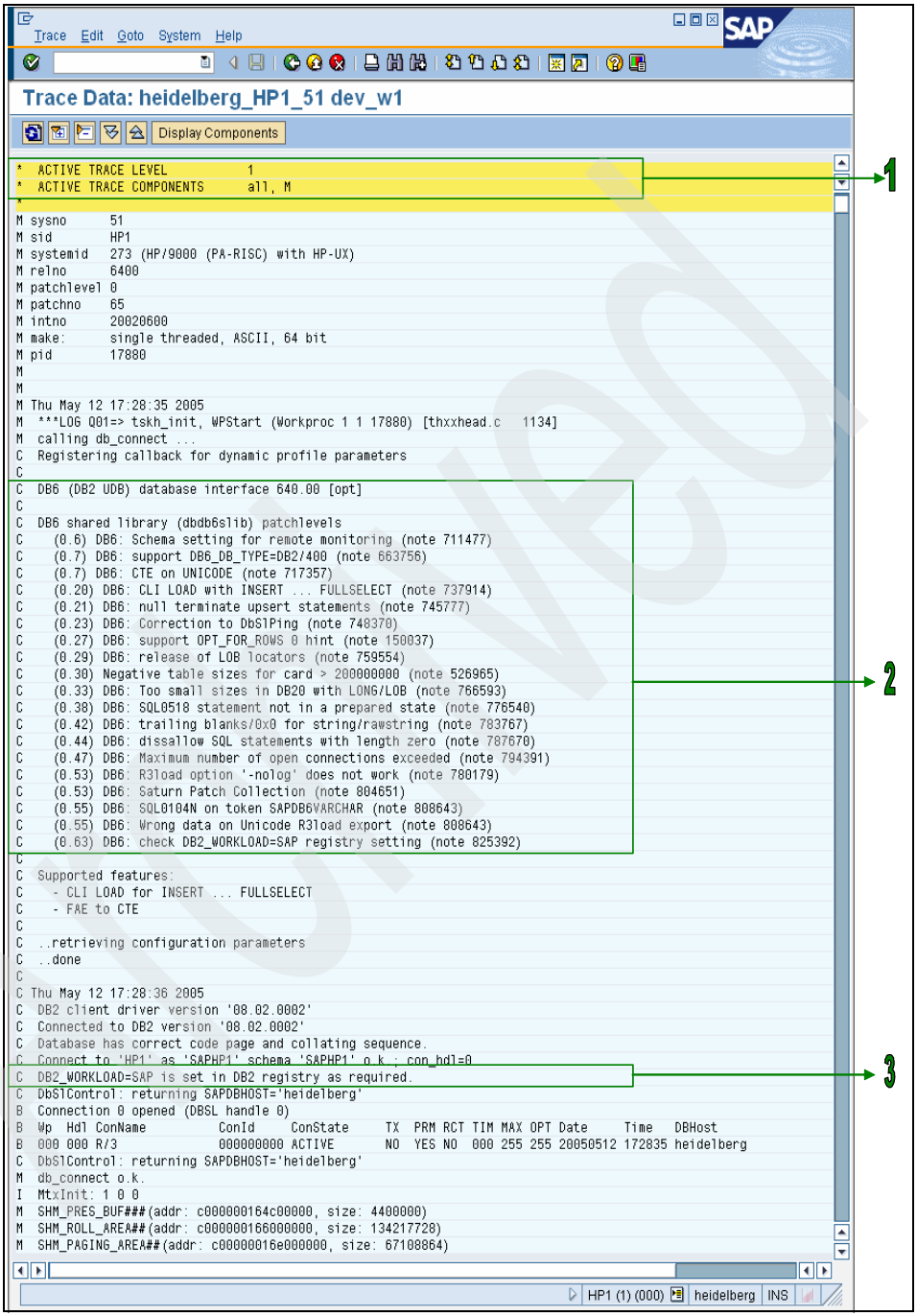


Figure 8-41 developer trace: header

Here are the explanations of each section each section in the header:

1. Trace file name and trace level (applies to all developer trace components).
2. Note the letter *C* on the left hand side of the trace. These are the traces written by the DBSL component.

   In this section, the DBSL version and patch level are shown. In this example, we are using DBSL version 640 (from the line "`DB6 (DB2 UDB) database interface 640.00 [opt]`"), patch level 63 (from the last line of the patch information, "`(0.63) DB6: check DB2_WORKLOAD=SAP registry setting (note 825392)`").
3. It is indicating that the DB2 UDB profile parameter `DB2_WORKLOAD` is set to `SAP` (that is, V8.2.2 feature `DB2 workload optimization for SAP environment`) as required by an SAP system running on DB2 UDB V8.2.2. This message only appears for an SAP system running on DB2 UDB V8.2.2.

developer trace: sample error entry

Figure 8-42 is a sample error entry in developer trace.

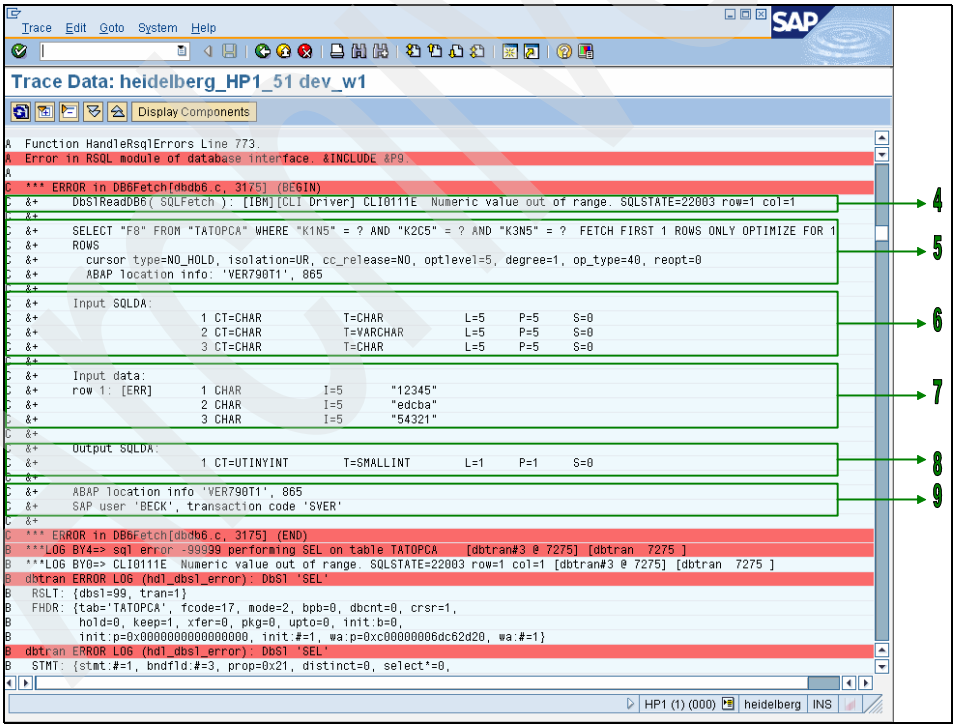


Figure 8-42 developer trace: sample error entry

Here are the explanations of each section in a sample error entry:

4. The error occurred in the DBSL function **DbslREADDB6**, CLI function **SQLFetch** with error text `CLI0111E Numeric value out of range` in row 1 column 1.
5. SQL statement texts and statement attributes (isolation level, optlevel, etc.) are shown.
6. Data type information for `INPUT` parameters (“?” or parameter markers in `WHERE` Clauses).
7. Bound values for `INPUT` parameters (“?” or parameter markers in the `WHERE` clauses).
8. Types of `OUTPUT` columns (columns from the `SELECT` list).
9. ABAP source location at Execution Time when error occurred (evaluate with report `RSDB6GETSOURCE` in transaction `SE38`. In our case, we put in `VER790TA` for `PROG` and `965` for `CONT` and we see the line of ABAP code that caused the error. See Figure 8-43). In this section, we also see the SAP user who run this report and the SAP transaction code name.

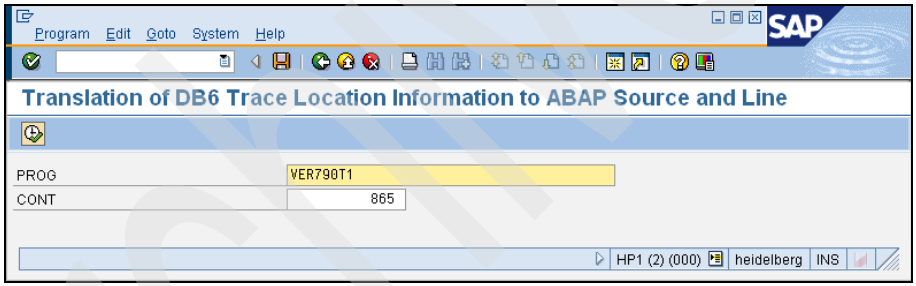


Figure 8-43 locating the error line in the report

In this example, the report tries to fetch a value defined as `SMALLINT` type in the database (which is 2 bytes long) into a `UTINYINIT` type defined in the ABAP program (which is 1 byte long). This is fine if the fetched value is smaller than a byte can hold (`2^8 - 1 = 255`). But if the fetched value is bigger than this limit, we see the `CLI0111E Numeric value out of range` error.

DBSL deadlock trace and db6util trace

If you have confirmed that the deadlock is not caused by lock escalations or inappropriate DB2 profile parameter settings, you may need to go on with more in depth deadlock analysis.

There are two SAP tools available for analyzing deadlocks: *DBSL deadlock trace* and the *db6util* tool. The db6util tool records a graph (in ASCII) of all the applications that participate in deadlocks or lock waits. The graph shows which application is waiting on which other application(s); along with the last SQL statement that is issued by each participating applications (without the runtime value). The *DBSL deadlock trace* gives you a list of SQL statements since the last commit (with runtime value) for candidate deadlock participants. Together, the two tools give the whole picture of the deadlock situation.

In DB2 UDB V8.2.2, there is a new deadlock event monitor which can provides the run time values of all the deadlock participants and the lock wait relationship among the deadlock participants. In SAP NetWeaver 2004s (Support Package 2), there is a new graphical user interface in the DBA Cockpit that use this new deadlock event monitor. Refer to , “New deadlock monitoring GUI in DBA Cockpit and the enhanced DB2 UDB deadlock event monitor” on page 528 for more information.

Important: When you need to use the DBSL deadlock trace and `db6util trace` for analyzing deadlock situation, it is important to have both traces switched on when reproducing the deadlock scenario. Missing one of the trace may not give you the full picture of the deadlock situation.

db6util

The *db6util* tool records a graph (in ASCII) of all the applications that participate in deadlocks or lock waits. The graph shows which application is waiting on which other application and the last SQL statement that was issued by each participating applications. Refer to SAP Note *327595* for more information.

The `db6util` takes snapshots at intervals to check whether there is a deadlock at that moment. It is crucial that the snapshot falls between the time that deadlock starts to occur and that deadlock get resolved. Otherwise, the deadlock is not recorded in the trace.

The `db6util` has two main options:

- `-sl`: for monitoring lock wait and deadlocks and
- `-sd`: for monitoring deadlock only.

It is mainly for analyzing two problems: deadlock (resulting in SQL0911 reason code *2*), and lock time out (when a lock wait exceeds the value set by the database configuration parameter LOCKTIMEOUT, resulting in SQL0911 reason code *68*) In either case, you can set sleep time and number of snapshots to have db6util check the system periodically. This is particularly useful when you do not know when the lock time out or deadlock occurs when reproducing the problem. You can optionally leave out the number of snapshots option and just put the sleep time option, which tells db6util to take the snapshots every <sleep time> seconds until you stop it with Ctrl-C (when the deadlock gets resolved). The -w option allows you to specify the name of the result file. The -o option allows you to specify a log file for the db6util trace. Here is the syntax map of db6util:

```
db6util -sl [sleep time] [number of snapshots]
            [ -o <logfile> ] [ -w <resultfile> ]

OR

db6util -sd [sleep time] [number of snapshots]
            [ -o <logfile> ] [ -w <resultfile> ]
```

For example, you would like to analyze a parallel job that sometimes results in deadlocks. The job run for 1 hour. You can set sleep time = 20 seconds and number of snapshots = 3600 sec / 20 sec = 180. And you chose log.txt as the name of the log and snapshot.txt as the name of the result file:

```
db6util -sd 20 180 -o log.txt -w snapshot.txt
```

In parallel, you should run DBSL deadlock trace at the same time to get the full picture of the deadlock.

Tip: Choosing a proper value for the sleep time is important for capturing a deadlock situation. If the value is too big compare to the database configuration parameter DLCHKTIME (Time interval for checking deadlock, default value is 30000 milliseconds or 5 minutes), deadlock could happen and got resolved in between two snapshots of db6util in which case no deadlock will be recorded in the trace. As a starting point, you can use 20 seconds for the sleep time parameter of db6tuil.

DBSL deadlock trace

The *DBSL deadlock trace* gives you a list of SQL statements since the last commit (with runtime value) for candidate deadlock participants. Refer to SAP Note *175036* for more information.

DBSL deadlock trace uses the following criteria to determine which applications for dumping out the SQL statements:

- If the application encounters `SQL911N` error (the deadlock victim).
- If the runtime of the SQL statements that the application is running exceeds the `trace time` (a parameter of the DBSL deadlock trace). Long runtime for an SQL statement could be caused by a lock wait or a slow running SQL statements. In the former case, the application is one of the deadlock participants, whereas in the latter case, the application just happens to be running a slow SQL statement. DBSL deadlock trace does not make the distinction between the two cases and dump out the SQL statements for both kind of applications. To find out which applications are actually involved in the deadlock situation, cross check with the db6util trace. In the following section, “Example: using DBSLdeadlock trace and db6util trace together” on page 520, there is an example on how to use the DBSL deadlock trace and the `db6tuil` trace together.

The `trace time` should be long enough to filter out applications that does not indeed involves in the deadlock; but not too long compares to the database configuration parameter `DLCHKTIME` to ensure that the DBSL deadlock trace captures the information from all deadlock candidates. For a `DLCHKTIME` interval value of 5 minutes, a `trace time` of about 20 seconds is useful.

There are two ways to activate DBSL deadlock trace:

- Dynamic activation of the deadlock trace via profile parameter:

  You can *dynamically* activate DBSL deadlock trace for all work processes using transaction `RZ11` by changing the SAP profile parameter:

  ```
  dbs/db6/dbsl_trace_deadlock_time = <seconds>
  ```

  You can use a value between 20 to 26 seconds.

  The default path for output is:

  - UNIX: `/tmp/TraceFiles`
  - Windows: `\\sapmnt\TraceFiles`

  You can change the trace path by setting:

  ```
  dbs/db6/dbsl_trace_dir = <tracepath>
  ```

- Activating the deadlock trace for all processes of a logon session:

  This method not only traces the work processes but *all R/3 processes* (`disp+work`, `tp`, `saplicense`, `R3trans`). This is helpful if the deadlock situation involves not only work processes but other processes like `R3trans`. This method is not dynamic. You need to restart the system for the trace to take effect. Refer to SAP Note *175036* for detail information on how to turn on the deadlock trace with this method.

Example: using DBSLdeadlock trace and db6util trace together

In this example, we illustrate how to use the *DBSL deadlock trace* and *db6util trace* together to analyze a deadlock scenario.

Before trying to reproduce the deadlock scenario, switch on both DBSL deadlock trace and `db6util` trace.

1. As **`<sid>adm`**, go to the default DBSL deadlock trace path and make sure it is clean:

```
# cd /tmp/TraceFiles/HP1
# rm *
# ls
```

2. Activating the DBSL deadlock trace:

   a. Logon to transaction `RZ11`. In maintain Profile parameters screen, enter `dbs/db6/dbsl_trace_deadlock_time` in *Para.Name* field (Figure 8-44).

   b. Click **Display** to display the current value of the parameter `dbs/db6/dbsl_trace_deadlock_time`.

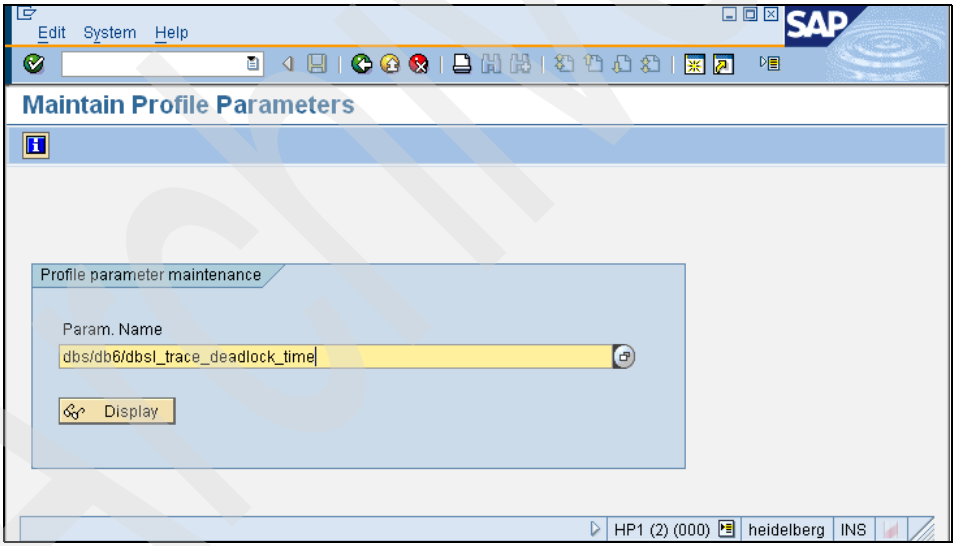


Figure 8-44 Activating DBSL deadlock trace: transaction RZ11

c. In *Display Profile Parameter Attributes* screen (Figure 8-45), to change the parameter value, click **Change Value**.

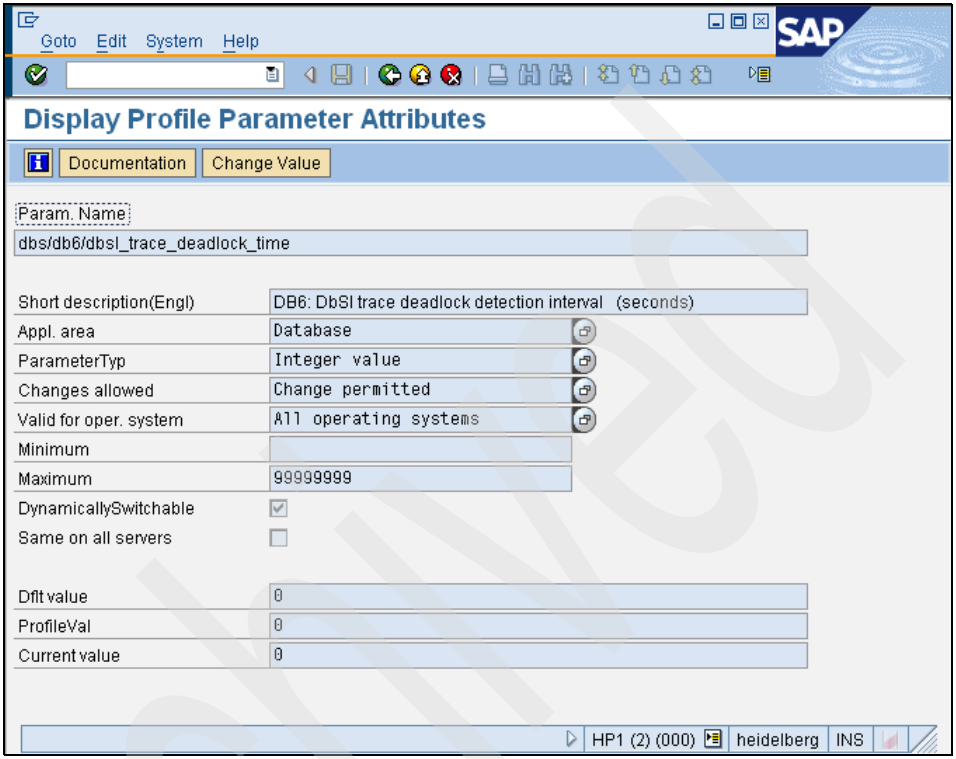


Figure 8-45 Activating DBSL deadlock trace: Display

d. In the *Change Parameter Value* screen (Figure 8-46), change the value of `dbs/db6/dbsl_trace_deadlock_time` to 20. Check **Switch on all servers**. Click **Save** to activate the DBSL deadlock trace.

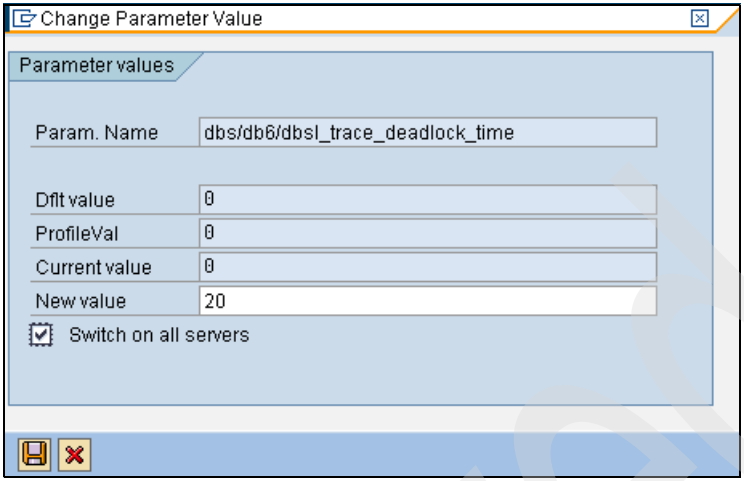


Figure 8-46 Activating DBSL deadlock trace: updating the value

3. Activate the db6util trace:

   a. As **<sid>adm**, switch on the `db6util` trace. Since our report normally takes 10 minutes to finish without deadlock, we are going to use 20 sec for the `sleep time` and 10 min X 60 sec/min / 20 sec = 30 for the `number of snapshots`:

      ```
      db6util -sd 20 30 -o log.txt -w snapshot.txt
      ```

4. Reproducing the deadlock scenario:

   a. Reproduce the deadlock scenario. Wait and see if one of the session gets cancelled.

5. Analyzing the deadlock scenario:

   a. If `db6util` has not finished collecting the data, use **`Ctrl-C`** to stop it.

   b. Use a text editor to view the trace file of `db6uti`, `snapshot.txt`. Example 8-16 is an excerpt of a db2util trace file.

Example 8-16 db6util trace file

```
...
SNAPSHOT TAKEN AT: 20050521143334
---------------------------------

No Applications are involved in a lock wait or deadlock situation


SNAPSHOT TAKEN AT: 20050521143354
---------------------------------

DEADLOCKS:
---------
```

```
Deadlock 1 :
                15                  12                  15
  <--    (PID:17879)  <--    (PID:17880)  <--    (PID:17879)  <--
          disp+work           disp+work           disp+work


DETAILED INFORMATION ABOUT LOCKED PROCESSES:
--------------------------------------------

  ID      PID       APPL-NAME        HOSTNAME  MODE RMODE OBJECT
      TABLE

  15    17879       disp+work       DB-server     X     X    ROW         SAPHP1.
/ISIS/DL_T1

  Last SQL STMT(s): UPDATE "/ISIS/DL_T1" SET "J" = ? WHERE "I" = ?  -- OPTLEVEL(
                     5 ) -- QUERY_DEGREE( 1 ) -- LOCATION( /ISIS/DL_OPENSQL_2 ,
                    112 ) -- SAP_INFO( BECK , SE38 ) -- SYSTEM( HP1 , SAPHP1 )

  12    17880       disp+work       DB-server     X     X    ROW         SAPHP1.
/ISIS/DL_T2

  Last SQL STMT(s): UPDATE "/ISIS/DL_T2" SET "L" = ? WHERE "K" = ?  -- OPTLEVEL(
                     5 ) -- QUERY_DEGREE( 1 ) -- LOCATION( /ISIS/DL_OPENSQL_1 ,
                    320 ) -- SAP_INFO( BECK , SE38 ) -- SYSTEM( HP1 , SAPHP1 )


SNAPSHOT TAKEN AT: 20050521143414
---------------------------------

DEADLOCKS:
---------

Deadlock 1 :
...
```

In each of the deadlock entries logged by the db6util trace, we first see the heading “DEADLOCK:”. In this section, we see which application is waiting for which other application represented by the “<--” symbol. For example, as shown in the db6util trace file:

```
<--    (PID:17879)  <--    (PID:17880)  <--    (PID:17879)  <--
```

The work process with PID (process ID) 17880 is waiting for work process with PID 17879; which in turn is waiting for work process with PID 17880. This loop which caused the deadlock can be seen in the ASCII graph.

In the section labeled “DETAILED INFORMATION ABOUT LOCKED PROCESSES:”, we see the last statement from each of the deadlock participant along with the kind of lock the application holds (under the column, MODE), the kind of lock the application requests (under the column, RMODE), the report/transaction name, and the user who run the report/transaction. From now on, we use the term work process to represent the application that runs on the work process:

- Work process with PID 17879:
  - Last statement: `UPDATE "/ISIS/DL_T1" SET "J" = ? WHERE "I" = ?`
  - Transaction name: `SE38`
  - Lock type that held by the other work process: `X`
  - Lock type that requested by this work process: `X`
  - Report name: `/ISIS/DL_OPENSQL_2`
  - User who ran the report: `BECK`
- Work process with PID 17880:
  - Last statement: `UPDATE "/ISIS/DL_T2" SET "L" = ? WHERE "K" = ?`
  - Transaction name: `SE38`
  - Lock type that held by the other work process: `X`
  - Lock type that requested by this work process: `X`
  - Report name: `/ISIS/DL_OPENSQL_1`
  - User who ran the report: `BECK`

However, we do not have the runtime value of the parameter markers (?) to determine which row the two `UPDATE` statements are going after. We can get this information from the DBSL deadlock trace.

Important: Under the column `OBJECT`, if we see `TABLE` as the lock level, it indicates a possibility of *Lock Escalations* (from row locks to table locks). You can confirm by checking the db2diag.log file to see if there is any lock escalation entries. Lock escalation occurs when the total number of locks held by an application reaches the maximum amount of lock list space available to the application, or the lock list space consumed by all applications is approaching the total lock list space. The amount of lock list space available is determined by the `maxlocks` and `locklist` configuration parameters. To resolve the situation, you can consider to increase the `locklist` or `maxlock` configuration parameter value.

c. As <*sid*>`adm`, go to the DBSL deadlock trace directory (that is, `/tmp/TraceFiles/HP1` in our example). List all the files there:

```
heidelberg:hp1adm 37> cd /tmp/TraceFiles/HP1
heidelberg:hp1adm 38> ls -l
total 70
-rw-rw----   1 hp1adm     sapsys        5966 May 21 14:36
DeadlockTrc17879.txt
-rw-rw----   1 hp1adm     sapsys        3224 May 21 14:36
DeadlockTrc17880.txt
```

Notice that the deadlock files has the naming convention: `DeadlockTrc<pid of the work process>.txt`. Make sure that all the work process involves in the deadlock shown by the **`db6util trace`** are there. In this case, both DBSL deadlock trace files for work process with PID 17879

(that is, `DeadlockTrc17879.txt`) and work process with PID 17880 (that is, `DeadlockTrc17880.txt`) are here.

d. View the content of `DeadlockTrc17879.txt` and `DeadlockTrc17880.txt` with a text editor:

Example 8-17 `DeadlockTrc17879.txt`

```
&
& DB2 UDB DbSl DEADLOCK trace
& --------------------------
&
& PID 17879 on heidelberg, system HP1 in database HP1
&
& DBSL_TRACE_DEADLOCK_TIME = 20
&
&
& LEGEND:
& -------
&
& CON   : connection handle
&
& hstmt : CLI statement handle
&
& c_id  : cache_id
&
& rows  : rows affected ( or 0 if n/a)
&
& timestamp: time of execution in format DD.MM hh:mm:ss.usec
&
& elapsed time: duration of execution in format mmm:ss.usec
&
& CL Classification of the elapsed time:
&       1!   50,000usec <= elapsed time <    125,000usec
&       2!  125,000usec <= elapsed time <    250,000usec
&       3!  250,000usec <= elapsed time <    500,000usec
&       4!  500,000usec <= elapsed time <  1,000,000usec
&       5!                elapsed time >= 1,000,000usec
&

&
& CON| hstmt |c_id|toplevel caller     |CLI function       |sql_rc|rows   |additional info        |CL| timestamp
|elapsed time
&
---+-------+----+--------------------+--------------------+------+-------+-----------------------+--+--------------------
+-------------
&+  0|       |    |    COMMIT[FORCED][DONT_KEEP_WITH_HOLD]
&   0| 1:126 | 125|DbSlModifyDB6       |SQLExecute          |     0|      1|                       |  |21.05
13:31:52.064271|000:00.001234
&+  0| 1:126 | 125|     row 1:                1 LONG            I=4        4
&+  0| 1:126 | 125|                           2 LONG            I=4        40
&+  0| 1:126 | 125|    INSERT INTO "/ISIS/DL_REF" VALUES( ? , ? )
&+ 0| 1:126 | 125|    cursor type=NO_HOLD, isolation=UR, cc_release=NO, optlevel=5, degree=1, op_type=15, reopt=0
&+  0| 1:126 | 125|      ABAP location info: '/ISIS/DL_OPENSQL_1', 113
&   0| 1:239 | 238|DbSlModifyDB6       |SQLExecute          |     0|      1|                       |  |21.05
13:31:52.066907|000:00.008043
&+  0| 1:239 | 238|     row 1:                1 LONG            I=4        50
&+  0| 1:239 | 238|    UPDATE "/ISIS/DL_T1" SET "J" = ?
&+ 0| 1:239 | 238|    cursor type=NO_HOLD, isolation=UR, cc_release=NO, optlevel=5, degree=1, op_type=19, reopt=0
&+  0| 1:239 | 238|      ABAP location info: '/ISIS/DL_OPENSQL_1', 280
&   0| 1:240 | 239|DbSlModifyDB6       |SQLExecute          |    -1|      1|                       |5|21.05
13:32:03.252394|004:39.659034
&+  0| 1:240 | 239|     row 1: [ERR]          1 LONG            I=4        100
&+  0| 1:240 | 239|                           2 LONG            I=4        1
&+  0| 1:240 | 239|    UPDATE "/ISIS/DL_T2" SET "L" = ? WHERE "K" = ?
```

```
&+ 0| 1:240 | 239|    cursor type=NO_HOLD, isolation=UR, cc_release=NO, optlevel=5, degree=1, op_type=19, reopt=0
&+ 0| 1:240 | 239|      ABAP location info: '/ISIS/DL_OPENSQL_1', 320
*** ERROR in DB6Execute[dbdb6.c, 4223] (BEGIN)
&+ 0| 1:240 | 239|     DbSlModifyDB6( SQLExecute ): [IBM][CLI Driver][DB2/HP64] SQL0911N  The current transaction has
been rolled back be
&+ 0| 1:240 | 239|    cause of a deadlock or timeout.  Reason code "2".  SQLSTATE=40001 row=1
&+ 0| 1:240 | 239|
&+ 0| 1:240 | 239|    UPDATE "/ISIS/DL_T2" SET "L" = ? WHERE "K" = ?
&+ 0| 1:240 | 239|    cursor type=NO_HOLD, isolation=UR, cc_release=NO, optlevel=5, degree=1, op_type=19, reopt=0
&+ 0| 1:240 | 239|      ABAP location info: '/ISIS/DL_OPENSQL_1', 320
&+ 0| 1:240 | 239|
&+ 0| 1:240 | 239|    Input SQLDA:
&+ 0| 1:240 | 239|                        1 CT=LONG            T=INTEGER         L=4     P=4     S=0
&+ 0| 1:240 | 239|                        2 CT=LONG            T=INTEGER         L=4     P=4     S=0
&+ 0| 1:240 | 239|
&+ 0| 1:240 | 239|    Input data:
&+ 0| 1:240 | 239|
&+ 0| 1:240 | 239|    ABAP location info '/ISIS/DL_OPENSQL_1', 320
&+ 0| 1:240 | 239|    SAP user 'BECK', transaction code 'SE38'
&+ 0| 1:240 | 239|
*** ERROR in DB6Execute[dbdb6.c, 4223] (END)
```

From Example 8-17, we see the content of the DBSL deadlock trace file DeadlockTrc17879.txt. We find out the run time values of the parameter of the last statement by examining the lines right before the SQL0911N deadlock entry (that is, [IBM][CLI Driver][DB2/HP64] SQL0911N The current transaction has been rolled back..). The presence of this entry indicates that work process with PID 17879 is the deadlock victim:

```
row 1: [ERR]       1 LONG            I=4       100
                   2 LONG            I=4       1
UPDATE "/ISIS/DL_T1" SET "J" = ? WHERE "I" = ?
```

The value of the first parameter marker is “100” and the value of the second parameter marker is “1”. Replacing the parameter markers with the run time value (Example 8-18), we get:

```
UPDATE "/ISIS/DL_T1" SET "J" = 100 WHERE "I" = 1
```

Example 8-18 DeadLockTrc17880.txt

```
&
&
& DB2 UDB DbSl DEADLOCK trace
& --------------------------
&
& PID 17880 on heidelberg, system HP1 in database HP1
&
& DBSL_TRACE_DEADLOCK_TIME = 20
&
&
& LEGEND:
& -------
&
& CON   : connection handle
&
& hstmt : CLI statement handle
&
& c_id  : cache_id
&
& rows  : rows affected ( or 0 if n/a)
```

```
&
& timestamp: time of execution in format DD.MM hh:mm:ss.usec
&
& elapsed time: duration of execution in format mmm:ss.usec
&
& CL Classification of the elapsed time:
&       1!   50,000usec <= elapsed time <    125,000usec
&       2!  125,000usec <= elapsed time <    250,000usec
&       3!  250,000usec <= elapsed time <    500,000usec
&       4!  500,000usec <= elapsed time <  1,000,000usec
&       5!                 elapsed time >= 1,000,000usec
&

&
& CON| hstmt |c_id|toplevel caller     |CLI function        |sql_rc|rows   |additional info
|CL| timestamp           |elapsed time
&
---+-------+----+--------------------+--------------------+------+-------+-----------------------+--+
---------------------+-------------
&   0| 1:91  | 219|DbSlModifyDB6       |SQLExecute          |     0|      1|                       |
|21.05 13:31:55.326678|000:00.012327
&+  0| 1:91  | 219|     row 1:             1 LONG           I=4        222
&+  0| 1:91  | 219|                        2 LONG           I=4        2
&+ 0| 1:91 | 219|   UPDATE "/ISIS/DL_REF" SET "N" = ? WHERE "M" = ?
&+  0| 1:91  | 219|       cursor type=NO_HOLD, isolation=UR, cc_release=NO, optlevel=5, degree=1,
op_type=19, reopt=0
&+ 0| 1:91 | 219|    ABAP location info: '/ISIS/DL_OPENSQL_2', 68
&   0| 1:207 | 205|DbSlModifyDB6       |SQLExecute          |     0|      1|                       |
|21.05 13:31:55.339591|000:00.010225
&+  0| 1:207 | 205|     row 1:             1 LONG           I=4        3
&+ 0| 1:207 | 205|   DELETE FROM "/ISIS/DL_REF" WHERE "M" = ?
&+  0| 1:207 | 205|       cursor type=NO_HOLD, isolation=UR, cc_release=NO, optlevel=5, degree=1,
op_type=8, reopt=0
&+ 0| 1:207 | 205|    ABAP location info: '/ISIS/DL_OPENSQL_2', 79
&   0| 1:209 | 207|DbSlModifyDB6       |SQLExecute          |     0|      1|                       |
|21.05 13:31:55.351032|000:00.007467
&+  0| 1:209 | 207|     row 1:             1 LONG           I=4        50
&+ 0| 1:209 | 207|   UPDATE "/ISIS/DL_T2" SET "L" = ?
&+  0| 1:209 | 207|       cursor type=NO_HOLD, isolation=UR, cc_release=NO, optlevel=5, degree=1,
op_type=19, reopt=0
&+ 0| 1:209 | 207|    ABAP location info: '/ISIS/DL_OPENSQL_2', 98
&   0| 1:212 | 210|DbSlModifyDB6       |SQLExecute          |     0|      1|
|5!|21.05 13:31:55.358878|004:47.547086
&+  0| 1:212 | 210|     row 1:             1 LONG           I=4        100
&+  0| 1:212 | 210|                        2 LONG           I=4        1
&+ 0| 1:212 | 210|   UPDATE "/ISIS/DL_T1" SET "J" = ? WHERE "I" = ?
&+  0| 1:212 | 210|       cursor type=NO_HOLD, isolation=UR, cc_release=NO, optlevel=5, degree=1,
op_type=19, reopt=0
&+  0| 1:212 | 210|       ABAP location info: '/ISIS/DL_OPENSQL_2', 112
```

In Example 8-18, we see the content of the DBSL deadlock trace file `DeadlockTrc17880.txt`. We find out the run time values of the parameter of the last statement by examining the lines in the end:

```
row 1:              1 LONG            I=4         100
                    2 LONG            I=4         1
UPDATE "/ISIS/DL_T2" SET "L" = ? WHERE "K" = ?
```

The value of the first parameter marker is “100” and the value of the second parameter marker is “1”. Replace the parameter markers with the run time value, we get:

```
UPDATE "/ISIS/DL_T2" SET "L" = 100 WHERE "K" = 1
```

Now we know that work process with PID 17879 is trying to update the row I=1 from table /ISIS/DL_T1 and the work process with PID 17880 is trying to update the row K=1 from table /ISIS/DL_T2. But which statements were holding the locks for these rows?

e. Upon further investigation on the earlier part of the two DBSL deadlock trace files, we find that:

- DeadlockTrc17879.txt: work process with PID 17879 is trying to update all rows on table /ISIS/DL_T2

```
row 1:             1 LONG            I=4         50
UPDATE "/ISIS/DL_T2" SET "L" = ?
```

- DeadlockTrc17880.txt: work process with PID 17880 is trying to update all rows on table /ISIS/DL_T1

```
row 1:             1 LONG            I=4         50
UPDATE "/ISIS/DL_T1" SET "J" = ?
```

New deadlock monitoring GUI in DBA Cockpit and the enhanced DB2 UDB deadlock event monitor

In SAP NetWeaver 2004s Support Package 3, there is a new screen in the DBA Cockpit called *Deadlock Monitor* (**Diagnostics** → **Deadlock Monitor**). It is the graphical interface for using the DB2 UDB V8.2.2 new feature: *enhanced deadlock event monitor*. The Deadlock Monitor screen works together with new DB2 UDB enhanced deadlock event monitor to provide an unified tool to gather important diagnostic data for solving deadlock problem.

DB2 UDB V8.2.2 enhanced deadlock event monitor

In the new implementation of the deadlock event monitor, the **CREATE EVENT MONITOR** statement has two additional new arguments; namely HISTORY and VALUE.

- HISTORY:

  When specified, the event monitor data also includes the history of all statements in the current unit of work at the participating node (including WITH HOLD cursors opened in previous units of work) and the statement compilation environment for each SQL statement in binary format. SELECT statements issued at the uncommitted read (UR) isolation level are not included in the statement history (since these statements do not hold lock which causes deadlocks).

  The statement history information is stored in buffer of the deadlock event monitor which is allocated from the monitor heap (controlled by the database manager configuration parameter MON_HEAP_SZ). If the database transaction completes without hitting the deadlock, the statement history records related to this transaction are thrown away.

If the deadlock occurs, the statement history of all open transactions are stored either in tables, pipes or files. The deadlock event monitor created by the DBA Cockpit: Deadlock Monitor screen uses *deadlock event monitor tables* (for example, tables that stores every occurrence of deadlock, applications involve in a deadlock, list of previous statements in the unit of work etc.) to store the event monitor data.

In some cases, if the monitor heap is too small and there is high number of SQL statements issued by applications, the monitor heap gets full and statement history records are *prematurely dumped* to the event monitor tables; even though these prematurely dumped records may not be involved in the deadlock. When this happens, you may see the messages in the db2diag.log about "`Flush of inactive statements attempted`".

This mechanism of prematurely dumping of statement history records ensures that we have a complete statement history record from the beginning of a transaction involving in a deadlock situation. However, these premature dumped statement history records can sometimes occupy a lot of spaces in the event monitor tables. Make sure your monitor heap is big enough to avoid too many instances of premature dump of statement history records.

- `Value`:

  The run time data values used as input variables for each SQL statement. These data values will not include LOB data, long data, structured type data. Run time value is especially important for diagnosing a deadlock situation in an SAP environment because most SQL statements in SAP environment are prepared dynamically with parameter markers ("?"). With the knowledge of the data values for the predicates in the WHERE clause, we can find out which table row involves in the deadlock.

DBA Cockpit: Deadlock Monitor

DBA Cockpit: Deadlock Monitor is a new screen in SAP NetWeaver 2004s (Support Package 3) that provides a graphical user interface to present the deadlock diagnostic data in a logical and way. In the new DBA Cockpit: Deadlock Monitor screen, you can:

- Create and drop the deadlock event monitor
- Start and stop the deadlock event monitor
- Reset the deadlock event monitor
- Refresh the deadlock event monitor
- Analyze deadlock event monitor records

Creating and dropping the deadlock event monitor

DBA Cockpit: Deadlock Monitor allows you to create the deadlock event monitor with the click of a button.

The default options used for the create deadlock monitor includes the `History` and `Value` options discussed in the previous section. The deadlock event monitor writes event monitor data to the deadlock event monitor tables. In the Deadlock Monitor screen, you can specify which table space to store these tables.

Example: create deadlock event monitor through DBA Cockpit: Deadlock Monitor screen

Here we downstater how to create deadlock event monitor using Deadlock Monitor in DBA Cockpit.

1. Go to transaction `DBACOCKPIT`. Choose **Diagnostics** → **Deadlock Monitor**. If your system has no event monitor defined, you see a **Create Deadlock Monitor** button. To create the deadlock event monitor, click the **Create Deadlock Monitor** button. See Figure 8-47

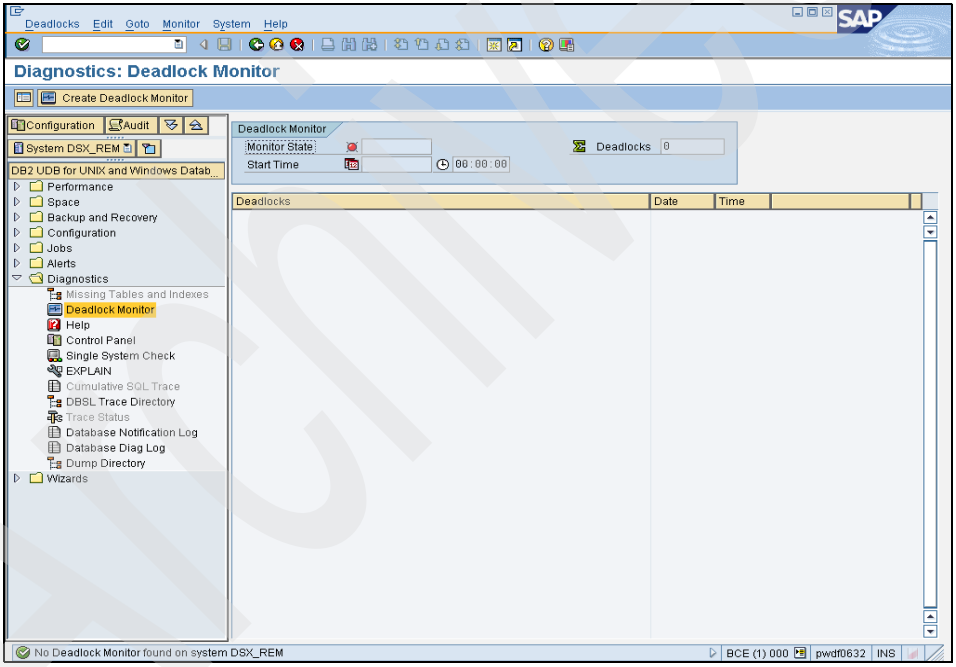


Figure 8-47 DBA Cockpit: Deadlock Monitor - create deadlock event monitor step 1

2. Since no deadlock event monitor with detailed statement history exists in the system, a wizard launches to guide you through the creation of the deadlock event monitor. Click **Continue**. See Figure 8-48.

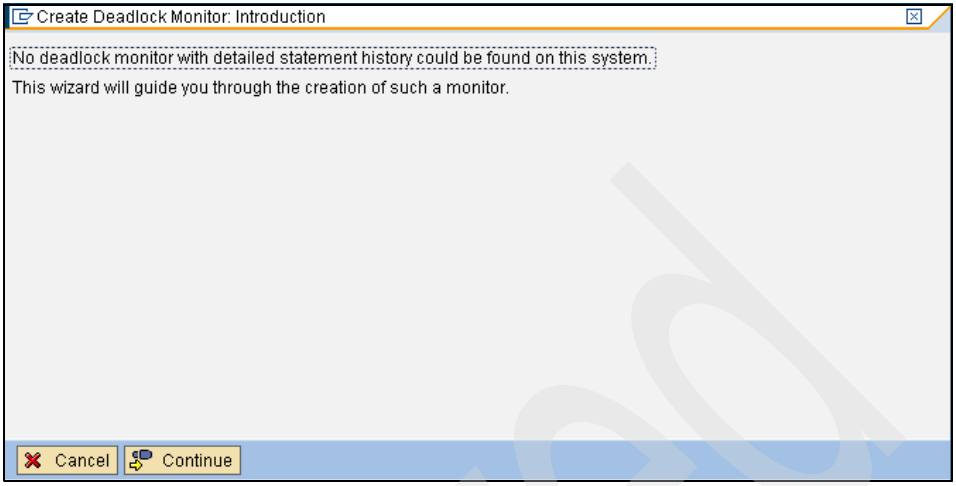


Figure 8-48 DBA Cockpit: Deadlock Monitor - create deadlock event monitor step 2

3. You choose which table space the deadlock event monitor tables resides in. If you are planning to monitor applications that has heavy workload, you may want to choose a dedicated table space for storing the big volume of statement history records and their corresponding data values.

   Click **Create Monitor** to create the deadlock event monitor. See Figure 8-49.

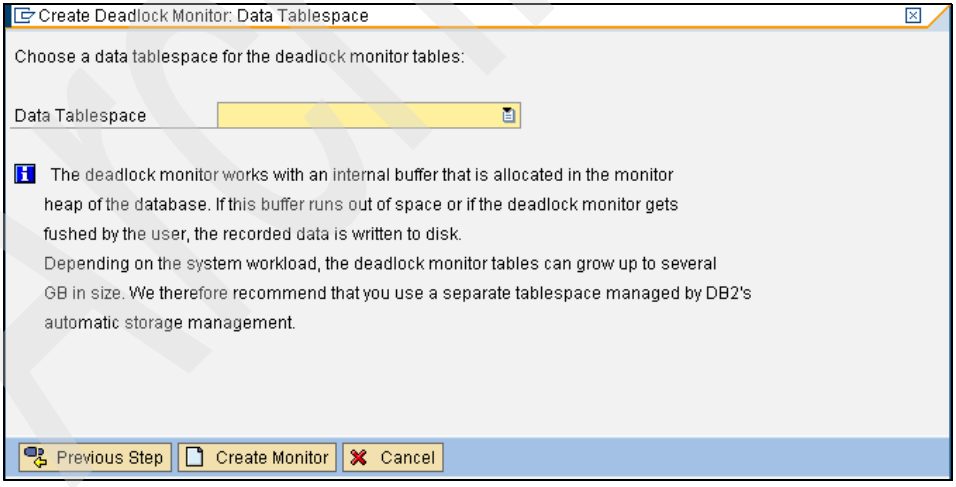


Figure 8-49 DBA Cockpit: Deadlock Monitor - create deadlock event monitor step 3

4. When the deadlock event monitor is created successfully, the DBA Cockpit: Deadlock Monitor screen changes. Now you see the **Start Monitor** button and the **Reset** button (Figure 8-50).

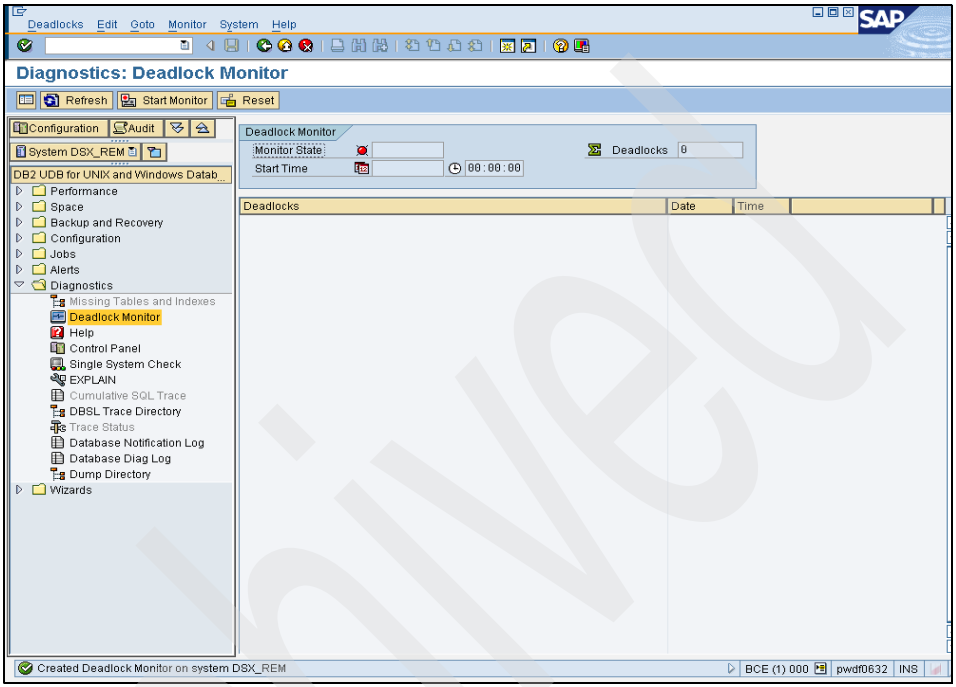


Figure 8-50 DBA Cockpit: Deadlock Monitor - create deadlock event monitor step 4

You may want to drop the Deadlock Event monitor and recreate it again (for example, if you want to put the deadlock event monitor tables in another table space). You can drop the deadlock event monitor by choosing the menu options: **Monitor** → **Drop Monitor**.

Starting and stopping the deadlock event monitor

Since the deadlock event monitor with detail statement history can have performance impact on the system, you only turn on the deadlock event monitor when you try to reproduce and analyze a deadlock situation. After the deadlock occurs, switch off the deadlock event monitor. To start the deadlock event monitor, click the **Start Monitor** button. To stop the deadlock event monitor, click the **Stop Monitor** button.

Make sure the database manager configuration MON_HEAP_SZ is big enough to hold the statement history to avoid too many instances of premature dumping. You may see the following error message if you try to start your deadlock event monitor but the MON_HEAP_SZ database manager configuration parameter is set too low (see Figure 8-51). Update the MON_HEAP_SZ to the value recommended in the popup dialog and restart the DB2 instance for the value to take effect.

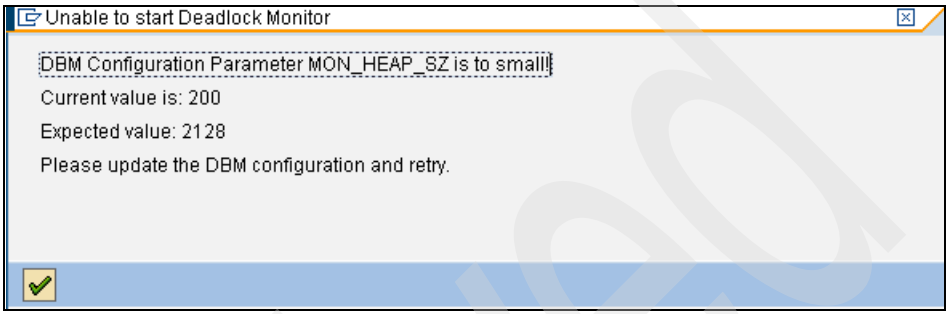


Figure 8-51 DBA Cockpit: Deadlock Monitor - error in starting the monitor - MON_HEAP_SZ too small

Resetting the deadlock event monitor

If you want to free up the space used by the deadlock event monitor, click the **Reset** button to clean up the contents in the event monitor tables.

Refreshing the deadlock event monitor

When you click the **Refresh** button, the new DBA Cockpit: Deadlock Monitor gathers and formats the deadlock information from the deadlock event monitor tables. Clicking the **Stop Monitor** automatically gather and format the deadlock information.

Analyzing deadlock event monitor records

After clicking the **Refresh** or **Stop Monitor** button, the deadlock data is flushed out to the deadlock event monitor tables. The deadlock data is organized in tree format in the *Main View*. See Figure 8-52.

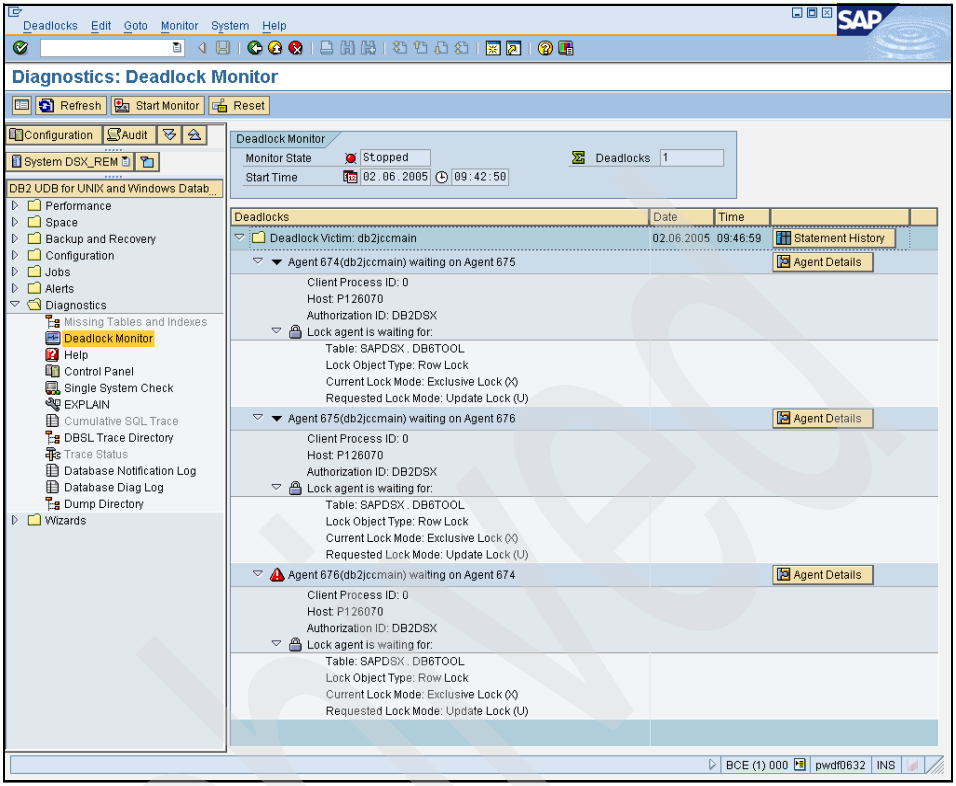


Figure 8-52 DBA Cockpit: Deadlock Monitor - Main View

In the Main View, the top level of the tree represents each incident of a deadlock. The root node is named in the format Deadlock *Victim: <rolled back application name>*. The data and time when the deadlock was detected is also shown.

When you open one of the root node, you see the 2nd-level subnodes. There are two possible kinds of 2nd-level subnodes:

- `Agent <Agent ID>` (`<Application Name>`) waiting on `Agent <Agent ID>`: These are the application(s) that have a lock wait on another application(s).
- *`Agent <rolled-back Agent ID>`* (`<Application Name of rolled-back Agent>`) waiting on `Agent <Agent ID>`: This is the victim of the deadlock that is chosen by the database deadlock detector to roll back.

With the above information, we can form the cyclic graph about which application is waiting on which other application. In the example shown in Figure 8-52, we can deduce from the 2nd-level nodes that `Agent 674`is waiting on `Agent 675` which is waiting on `Agent 676`. `Agent 676` in turn is waiting for `Agent 674`.

Right beside the root node (in this example, the folder icon which says “Deadlock Victim: db2jccmain”, see Figure 8-52), there is a **Statement History** button. By clicking the button, you see the “Diagnostics: Deadlock Monitor - Statement History” screen. Statement histories from *all agents* are listed here in chronological order (see Figure 8-53).

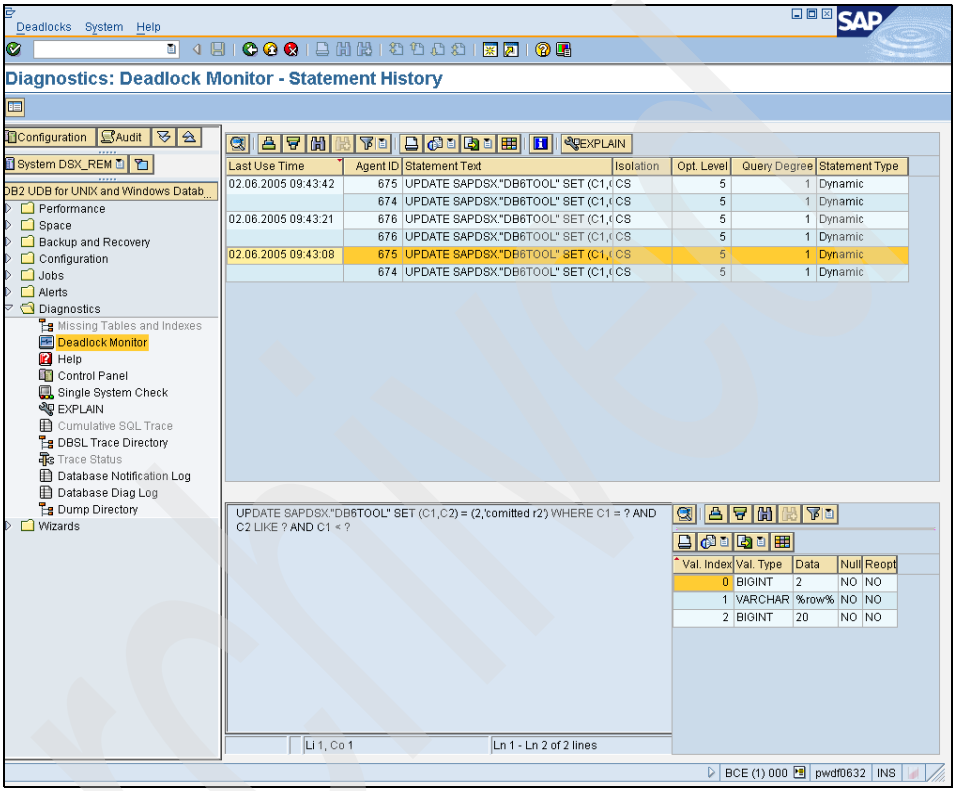


Figure 8-53 DBA Cockpit: Deadlock Monitor - Statement History (all agents)

To look at the run time values of a statement (for dynamic SQL statements that uses parameter markers “?”), double-click the statement. In the lower right hand corner of the screen, the run time values of dynamically prepared statements are listed. The column, “Val. Index” represents the position of the parameter marker in the statements (counting from 0). In this example, after substituting the parameter markers with the run time values, the WHERE clause becomes “`WHERE C1=2 AND C2 LIKE ‘%row%’ AND C2 < 20`”.

Now we go back to the Main View (see Figure 8-52). In the 2nd-level subnodes of the, there are the following information:

- Table: `<Schema>.<Table>`

- Lock Object Type: `<Lock Object Type>`
- Current Lock Mode: `<Lock Mode>`
- Requested Lock Mode: `<Lock Mode>`

In our example, under the subtree of the 2nd-level node of `Agent 17`, we see that `Agent 17` is trying to request a row lock (`Update Lock (U)`) on the table `SAPDSX.DB6TOOL`. But that row is currently locked by `Agent 18 (Exclusive Lock(X))`.

For each of the 2nd-level subnodes, you can click the **Agent Details** button to go to the *Agent Detail* screen to view the detail of certain agent. The Agent Detail screen has two tabs:

- *Locks Held* tab: In the Agent Detail: Lock Held tab (see Figure 8-54), you see the lock(s) that the agent is granted (holding) and the lock that it is waiting (requesting).

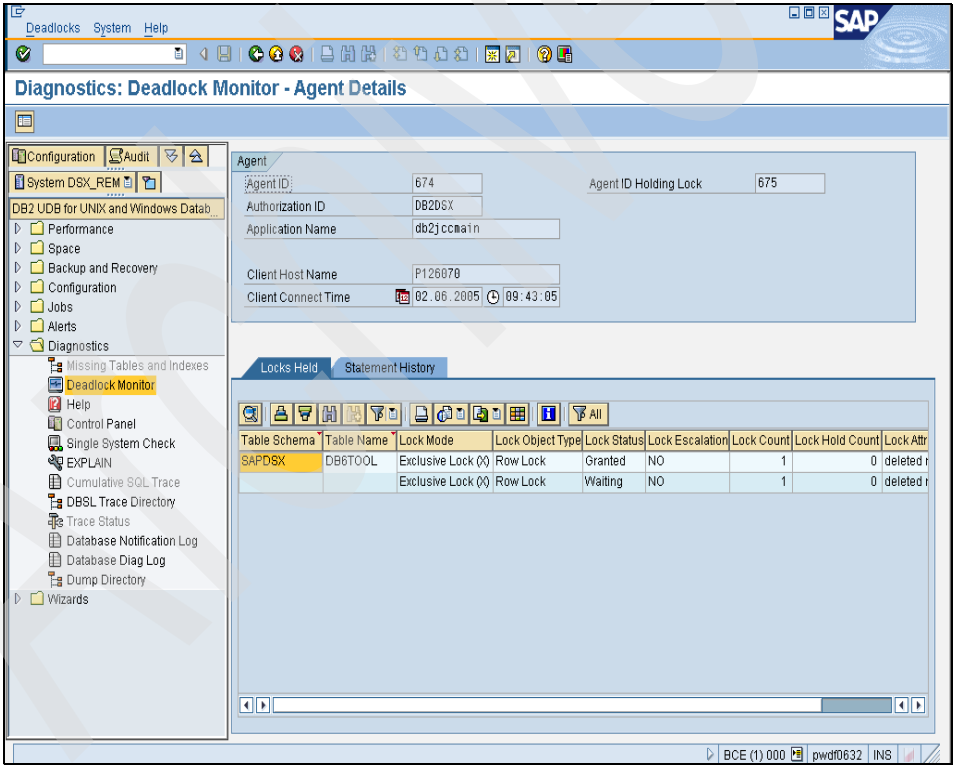


Figure 8-54 DBA Cockpit: Deadlock Monitor - Agent Details - Locks Held

- *Statement History* Tab (for particular agent): this tab (see Figure 8-55) shows a list of SQL statements issued by *this agent* chronologically since the last commit point. To look at the run time values of a statement (for dynamic SQL statements that uses parameter markers “?”), double-click the statement.

  In the lower right hand corner of the screen, the run time values of dynamically prepared statements are listed. The column, “Val. Index” represents the position of the parameter marker in the statements.

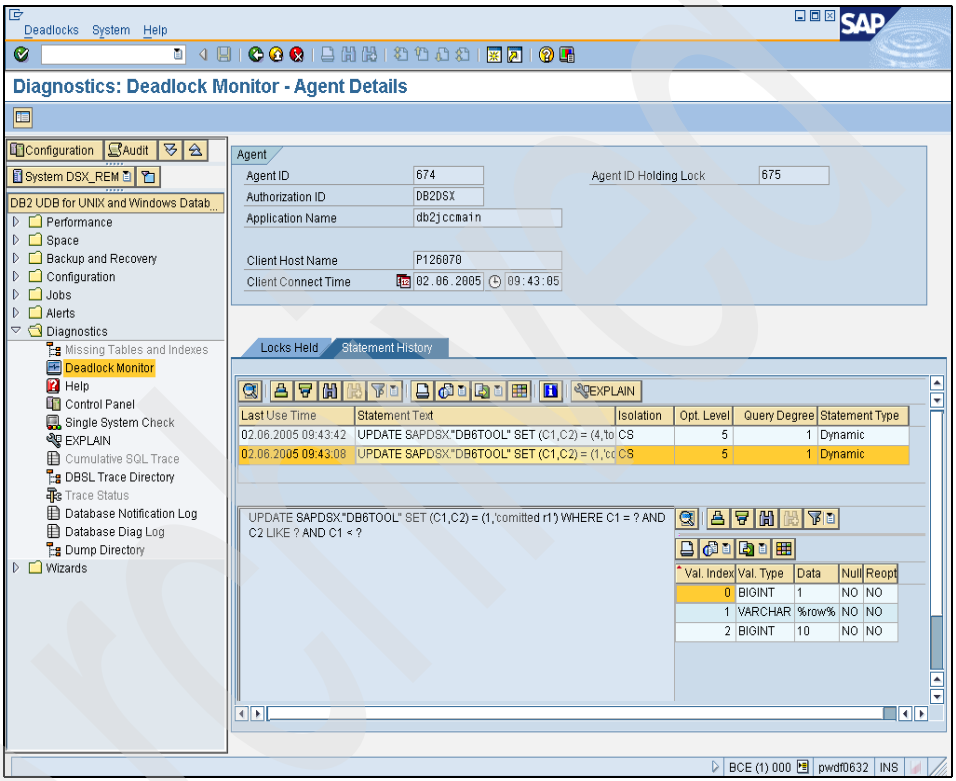


Figure 8-55 DBA Cockpit: Deadlock Monitor - Agent Details - Statement History (for one agent)

DBSL trace

DBSL trace is a useful tool for error analysis. It is especially useful for determining problems between the DBSL layer and the CLI layer. Details of DBSL Trace is described in SAP SAP Note *31707*. You may be asked by SAP support to provide DBSL trace to help problem analysis.

You can use DBSL trace to track all the function calls made from the database interface (DBSL) from SAP system to the DB2 UDB CLI layer. The logs for DBSL trace are saved as operating system files.

There are several ways to activate a DBSL trace:

- *Activate the DBSL trace via Transaction SM50*:

  This method tracks only single work process (that is, `disp+work` process). You do not need to restart the process to start the trace. The trace files are stored in:

  - UNIX: `/usr/sap/<SID>/<instance>/work`
  - Windows: `<DRIVE>:\usr\sap\<SID>\<instance>\work`

  Note that for this method, only the default DBSL Trace options are used.

- *Activating the DBSL trace via the SAP profile parameters*:

  This method tracks all the work processes (that is, `disp+work` processes). You can activate DBSL trace by setting SAP profile parameters (for example, `dbs/db6/dbsl_trace`). As of SAP Kernel Release 4.6A, some of the parameters can be switched on in transaction `RZ11` and the trace can be dynamically activated/deactivated. There is also an interface in DBA Cockpit (that is, **Diagnostics** → **Trace Status**) to update these SAP profile parameter for activating or deactivating the trace. The trace files are stored in:

  - UNIX: `/tmp/TraceFiles/<SAPSID>`
  - Windows: `\\<localhost>\sapmnt\Tracefiles\<SAPSID>`

  The name of the trace files are in the format: `TraceFile<`**`work process ID`**`>.txt`. If the trace has been switched on more than once, a number which is appended to the filename is incremented by one every time the trace is switched on again (that is, `TraceFile<work process ID>_<number>.txt`).

- *Activating the DBSL trace for all processes of a logon session*:

  This method tracks the calls from any SAP system processes (`disp+work`, `tp`, `R3trans`, `saplicense`). The trace is activated/deactivated in the operating system level. You need to restart the work processes after setting the environment variables for the trace to take effect. Refer to SAP Note *31707* for more information.

SAP profile parameters and environment variables for DBSL trace

Table 8-2 lists the SAP profile parameters and environment variables for the DBSL trace:

Table 8-2 SAP profile parameters and environment variables for DBSL trace

Profile Parameter	Environment Variable	Label in DBA Cockpit: Diagnostics → Trace Status	default	Meaning
dbs/db6/ dbsl_trace	DB6_DBSL_TRACE	Trace Level	0	The amount of data to be traced: *2*: Only statements are traced. *3*: Statements and results are traced.
dbs/db6/ dbsl_trace_flush	DB6_DBSL_TRACE_FLUSH	N/A	0	A synchronization of the file is done every <value> number of trace operation(s). There will be performance impact if the value is set to a non zero value. This is used in the case that process core dumps.
dbs/db6/ dbsl_trace_dir	DB6_DBSL_TRACE_DIR	N/A	as described above	Alternative trace directory
dbs/db6/ dbsl_trace_string	DB6_DBSL_TRACE_STRING	DBSL Trace Search String		Search string (statement text). If you are interested in a statement against a particular table, you can put the table name as the search string to filter the trace.

Profile Parameter	Environment Variable	Label in DBA Cockpit: Diagnostics → Trace Status	default	Meaning
dbs/db6/dbsl_trace_iocount	DB6_DBSL_TRACE_IOCOUNT	Number of I/O Records to Be T	5	Number of traced rows in array operation. A higher value may be needed in situation where the error only occurs after a big number of array operations.
dbs/db6/dbsl_trace_time	DB6_DBSL_TRACE_TIME	DBSL Trace Minimum Time Limit	0	Minimal time for operations to be traced (in micro seconds). Inexpensive query can be filtered out especially for performance analysis.
dbs/db6/dbsl_trace_str_len	DB6_DBSL_TRACE_STR_LEN	Display Length for String/Raw	64	The output length of the contents of long data fields

Example 1: Activating DBSL trace via transaction SM50

You would like to do a DBSL trace for a single work process, here is the process to turn on the trace.

1. Go to transaction `SM50`. Identify the work process you would like to run DBSL trace and highlight it as shown in Figure 8-56:

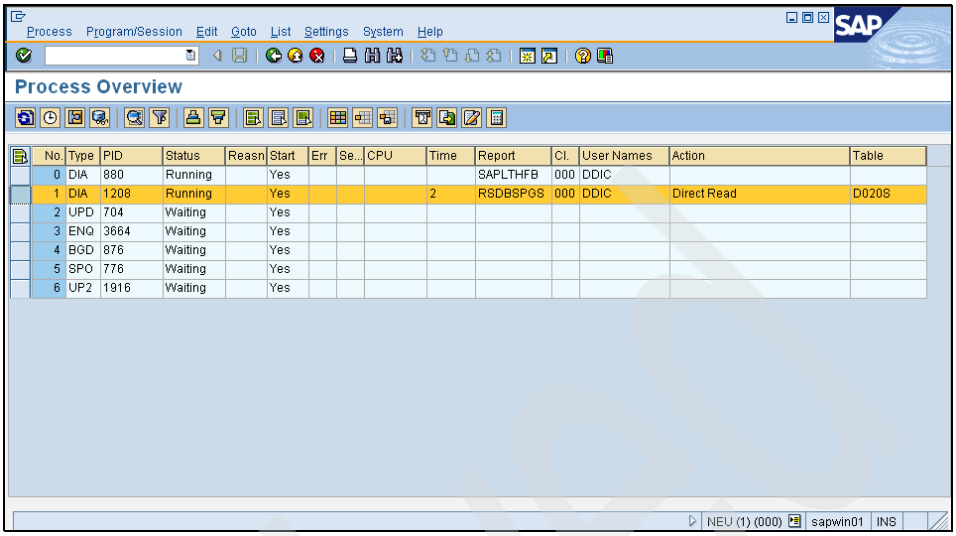


Figure 8-56 Activating DBSL Trace via Transaction SM50: highlighting the process

2. Choose **Process** → **Trace** → **Active Components** (Figure 8-57).

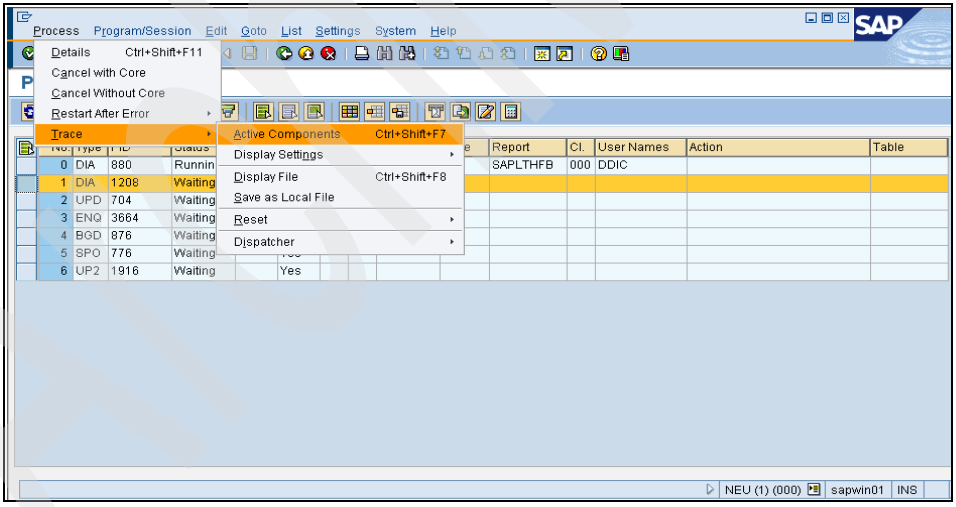


Figure 8-57 Activating DBSL Trace via Transaction SM50: Activate Components

3. In the *Change Trace Components* screen (Figure 8-58), choose **3** as the Trace level and mark the check box `Database (DBSL)`. You can leave the two other check box for the other components checked. Click the **Save** button and the trace starts.

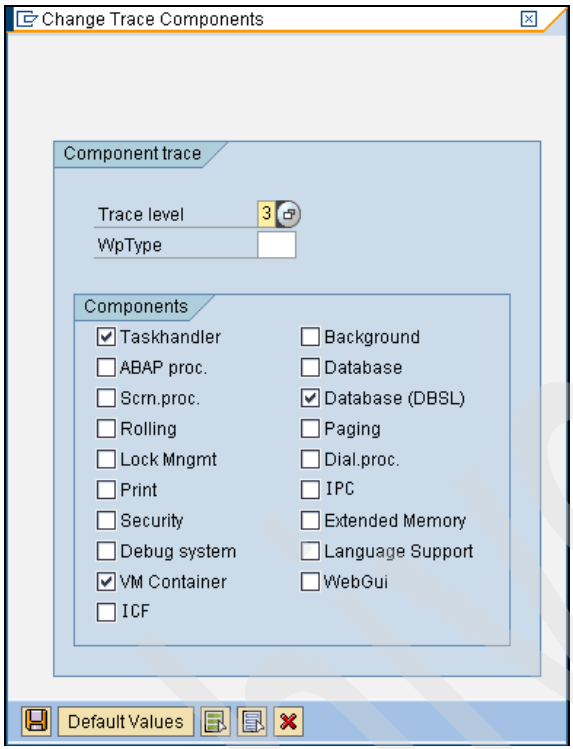


Figure 8-58 Activating DBSL Trace via Transaction SM50: Change Trace Components

4. As you can see in Figure 8-59, the color of the row is changed for the work process that have the trace turned on.

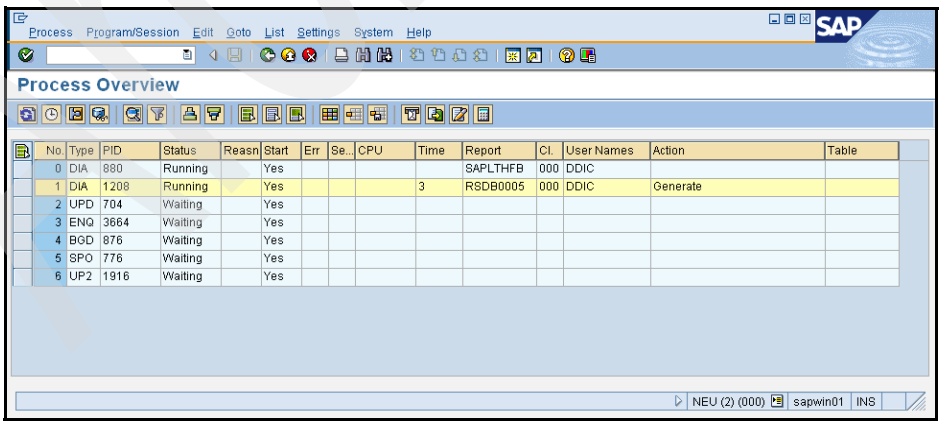


Figure 8-59 Activating DBSL Trace via Transaction SM50: the row that has the trace turned on has a different color

5. When you are done with the trace, switch off the trace by going to transaction SM50, **Process** →**Trace** → **Active Components,** and reset to the default setting.
6. You can use transaction AL11 or directly go to the /usr/sap/<SID>/<instance>/work (for UNIX) or <DRIVE>:\usr\sap\<SID>\<instance>\work to inspect the developer trace file where the trace is written to.

Example 2: Activating DBSL Trace via the SAP Profile Parameter

You would like to turn on DBSL trace for all work processes and you do not need to change the DBSL trace options that cannot be modified in the DBA Cockpit (for example, dbs/db6/dbsl_trace_flush). In this case, you can turn on the trace with the DBA Cockpit (**Diagnostics** →**Trace Status**).

1. Go to transaction DBACOCKPIT. Select **Diagnostics** → **Trace Status** and choose the **dbsl trace** options. In this example (Figure 8-60). We choose **3** for Trace Level to get a detail trace; **10** for Number of I/O Records; **128** for Display Length for String/Raw Data; and **5** microseconds for DBSL Trace Minimum Time Limit. Make sure you click the **UPDATE** button beside the value of each DBSL Trace option for the new value to take effect.

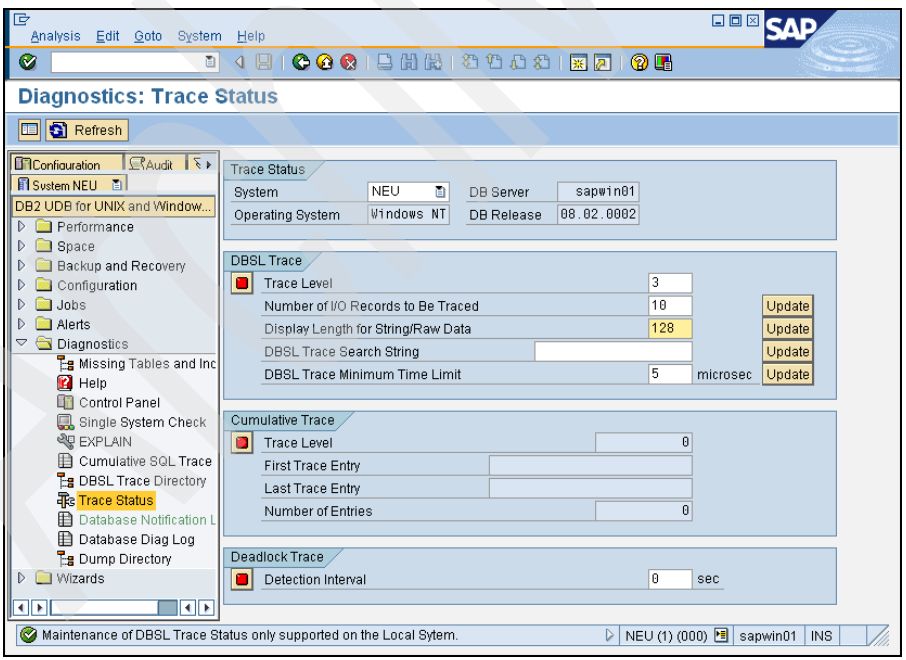


Figure 8-60 Activating DBSL Trace via the SAP Profile Parameter (DBA Cockpit): Diagnostics →Trace Status

2. Switch on the DBSL trace by clicking the red LED (Figure 8-61). When the LED is green, the trace is on.

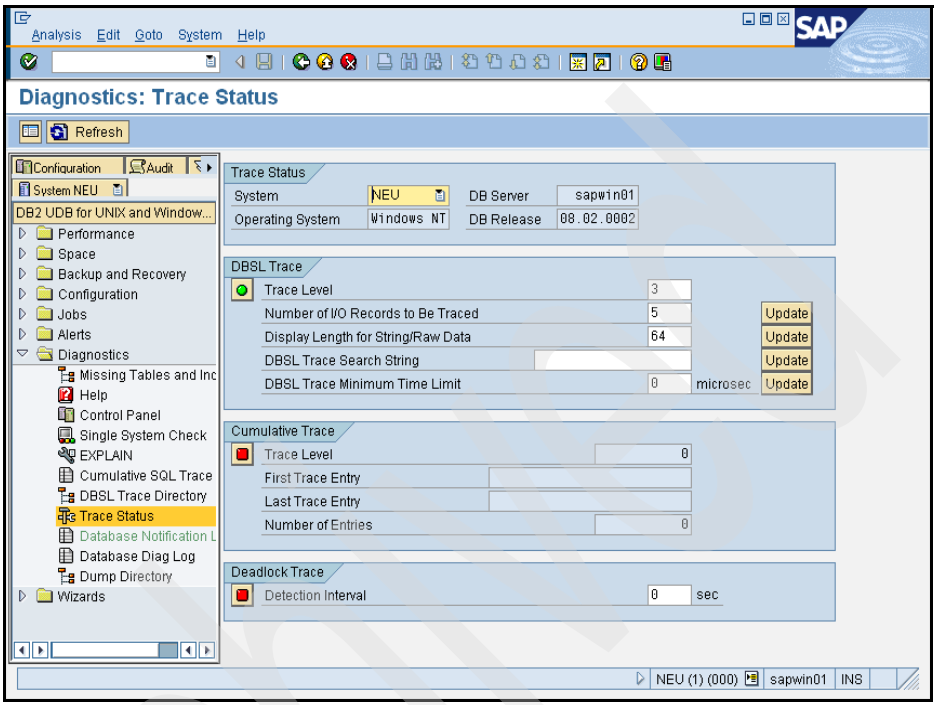


Figure 8-61 Activating DBSL Trace via the SAP Profile Parameter (DBA Cockpit): switch on the DBSL trace

3. Once you are done with the trace, switch off the trace by clicking the green LED . When the LED red, the trace is off.
4. Select **Diagnostics** → **DBSL Trace Directory** (Figure 8-62).

   Drill down the directory <SID> (that is, NEU in our example) and you see a list of DBSL trace files.

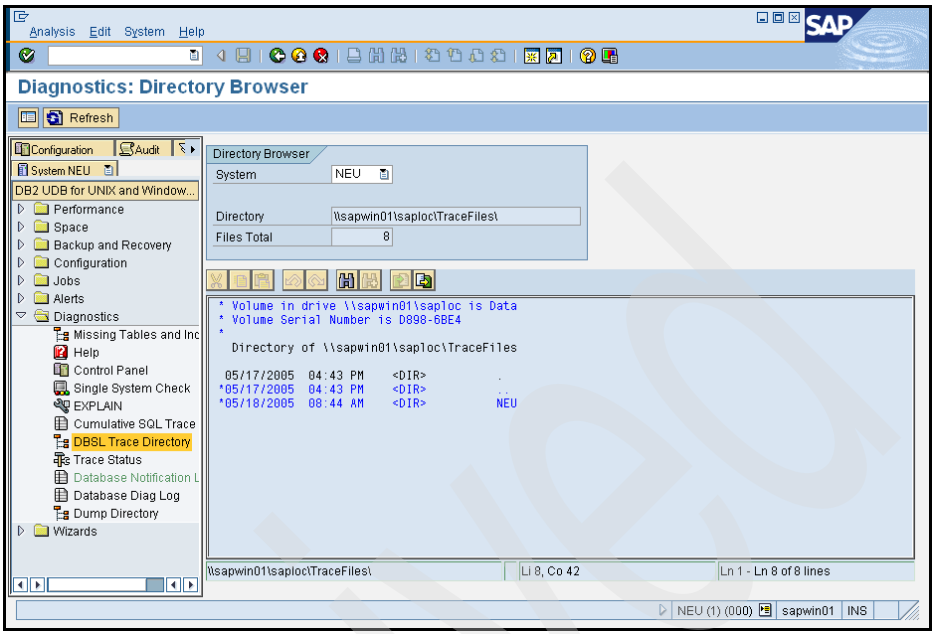


Figure 8-62 Activating DBSL Trace via the SAP Profile Parameter (DBA Cockpit): DBSL Trace Directory

ST22: ABAP dump analysis

When an ABAP program running in foreground or background hits a runtime error which causes immediate termination of the job, a ABAP dump (a.k.a. short dump) is usually generated. This dump contains a description of the cause of the error.

ABAP dump shows the line in the ABAP program which causes the run time error and the related run time environment. It also provides keyword for you to search in SAP Note to see whether this is a known problem. Sometimes, SAP support request the ABAP dump from the customer for further analysis.

You can access ABAP Runtime Error by going to transaction `ST22`.

In the initial screen, you specify whether you want to view today's dump or the dump from yesterday. If you want to be more specific, you can customize the criteria and click **Start**. See Figure 8-63.

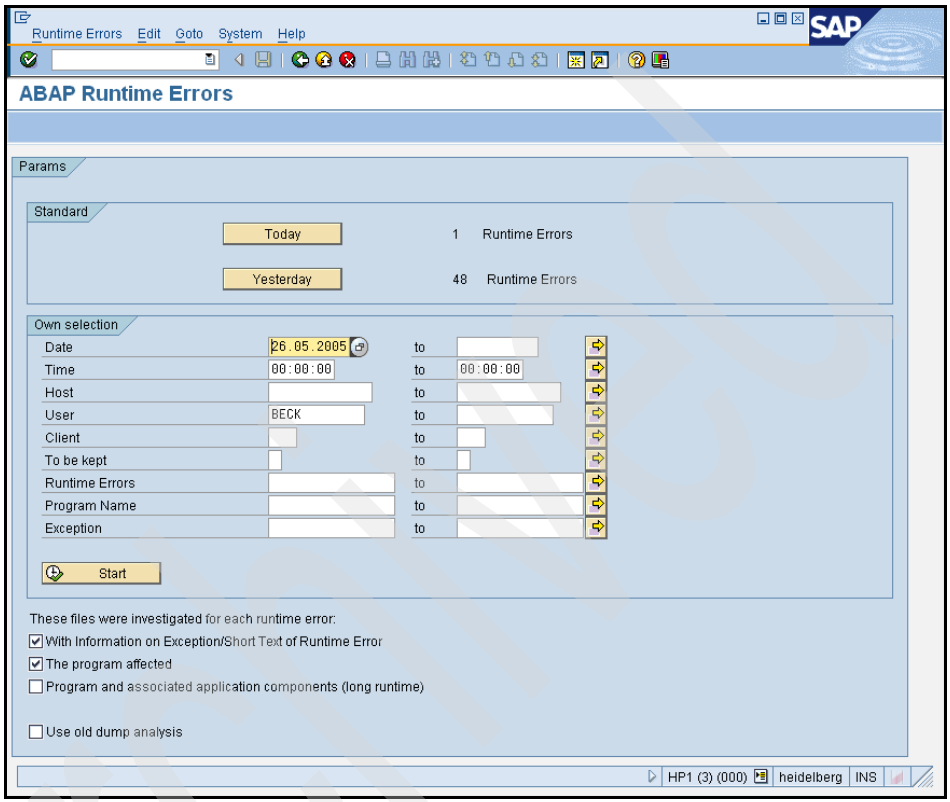


Figure 8-63 st22 ABAP Runtime Error: initial screen

The next screen (see Figure 8-64) shows the selected list of ABAP dumps with date, time, host, name of the user, name of runtime error, exception etc. Double-click the ABAP dump of interest to see the detail.

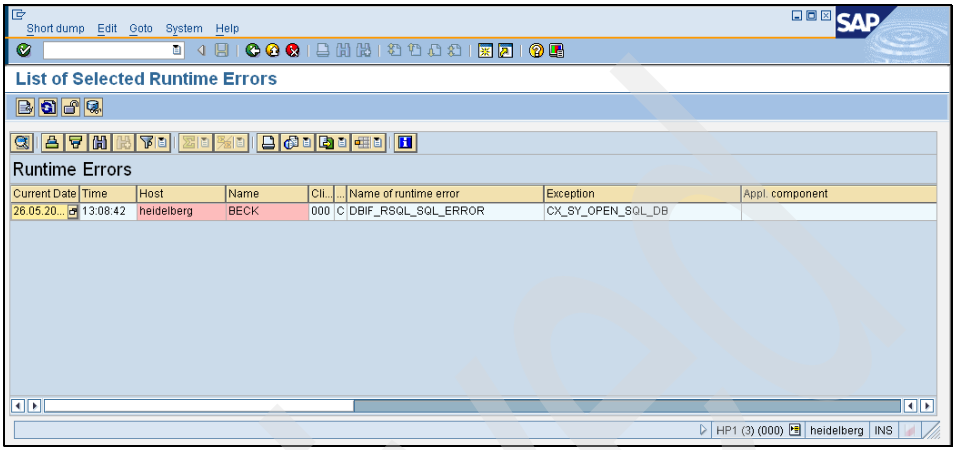


Figure 8-64 st22 ABAP Runtime Error: List of Selected Runtime Errors

In the `Runtime Error - Description of Exception` screen (see Figure 8-65), we see a short text about the error; the program name; and the section of program that causes the run time error. The error line is marked with the “>>>>>” sign. Click the **Long Text** button to see more information about the ABAP dump.

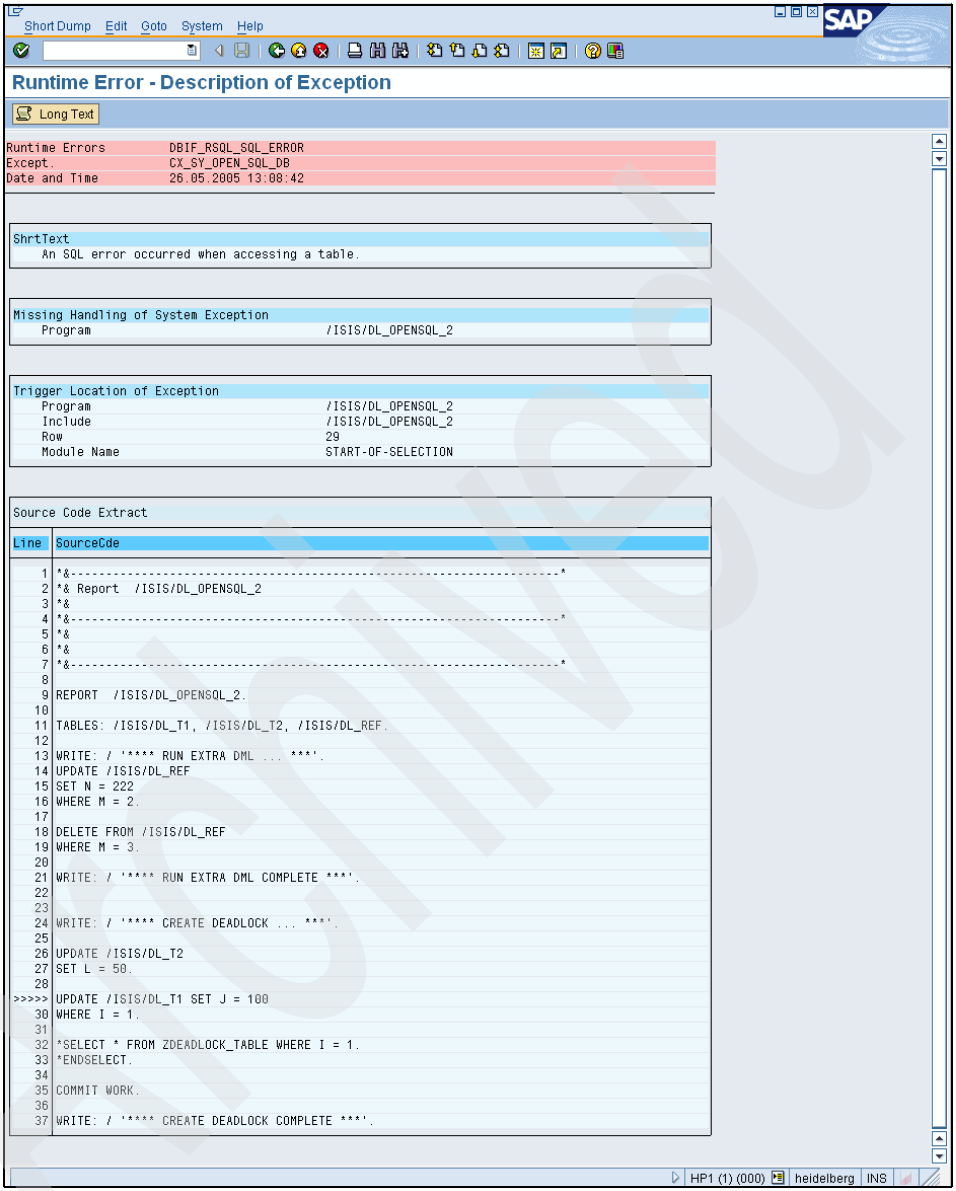


Figure 8-65 st22 ABAP Runtime Error: Runtime Error Long Text

In the `Runtime Error Long Text` screen (Figure 8-66), you can find extra information like the “How to correct the error” section. This section gives you a list of key words you can search in the SAP Note to check if this is a known problem.

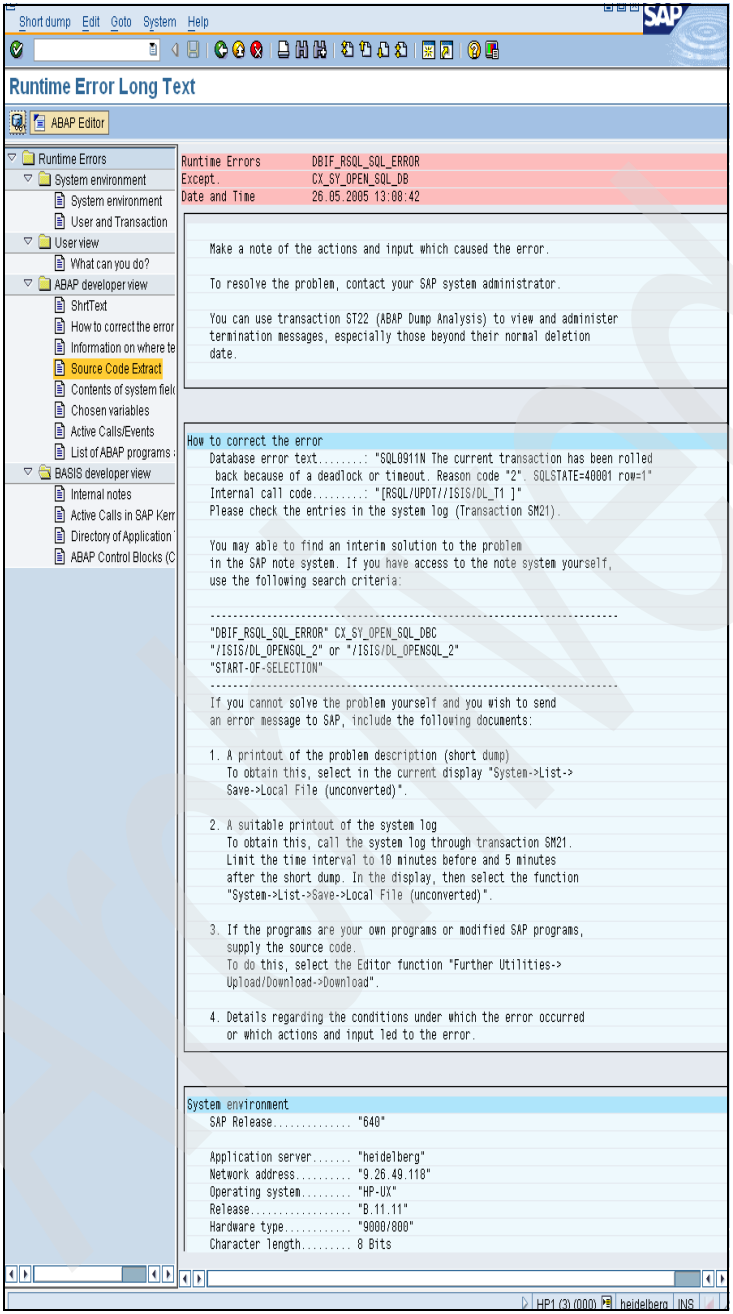


Figure 8-66 st22 ABAP Runtime Error: Description of Exception

SM21: SAP system logs

SAP system record events and problems in system logs. When you run an SAP transaction or report and hit a DB2 error, you can usually find the error entries logged in the system log as well.

To display the system logs, use transaction code `SM21`. You see the main screen of system log. See Figure 8-67

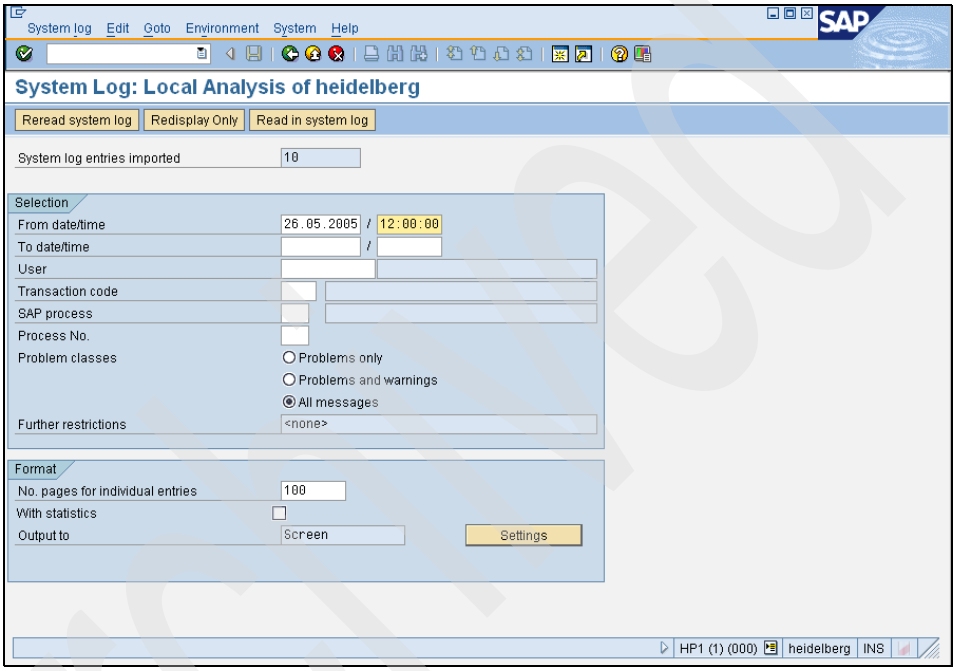


Figure 8-67 system logs: selection screen

The selection screen allows you to set criteria (for example, datatime, user, transaction code) to narrow down the error entries you are interested.

There are three button at the top:

- **Reread system log**: Completely reread the system log.
- **Redisplay only**: view the last system log you displayed.
- **Read in system log**: If you want to retain the data from the last read, but want to reread the system log with different selection criteria. Use the **Read in system log** button.

In Figure 8-68, you see a system log that shows a DB2 error:

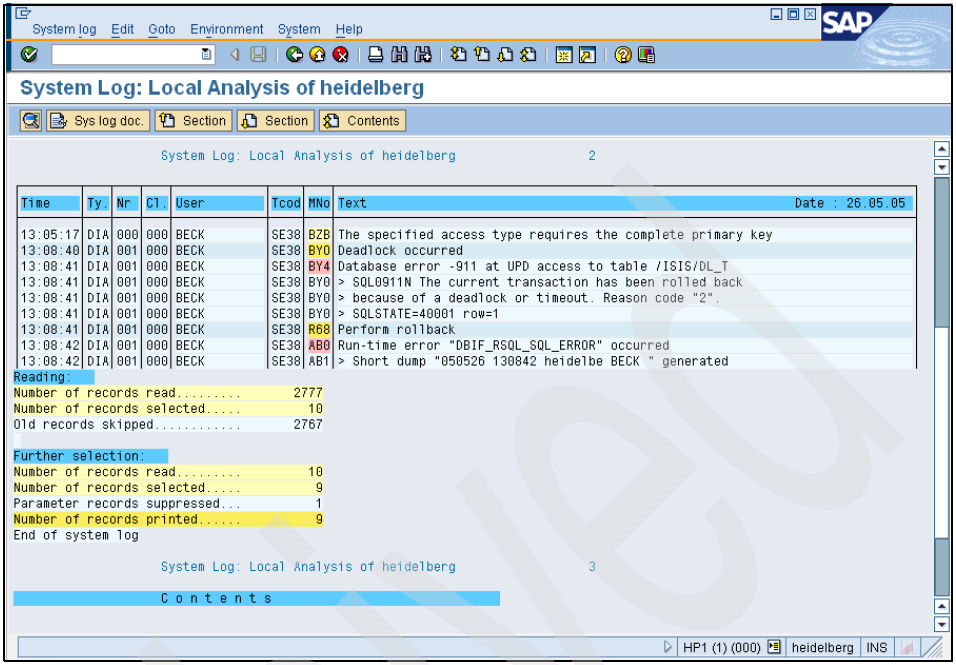


Figure 8-68 system logs: Sample Log report

As you can see in Figure 8-68, on `26.05.05` at `13.08.41`, user `BECK` ran a report in transaction code `SE38` as a foreground diaglog process (DIA in the second column) and hit the following error as shown in the text column:

```
Database error -911 at UPD access to table /ISIS/DL_T
> SQL0911N The current transaction has been rolled back
> because of a deadlock or timeout. Reason code "2".
> SQLSTATE=40001 row=1
```

From the error message, we know the report hits the database error `SQL0911N` which means we are either hitting a deadlock or timeout problem. Which one is it? You can use the command **db2 ?** to interpret the DB2 error code:

```
heidelberg:db2hp1 21> db2 \? sql0911n
SQL0911N The current transaction has been rolled back because of
         a deadlock or timeout.  Reason code "<reason-code>".
Explanation:
The current unit of work was involved in an unresolved contention
for use of an object and had to be rolled back.
The reason codes are as follows:
2 transaction rolled back due to deadlock.
68 transaction rolled back due to lock timeout.
...
```

In the error text, we see the error code `Reason Code` **“**2**”**. We can be sure that we are hitting a deadlock.

You can double-click the error line to see more details on of the error. See Figure 8-69.

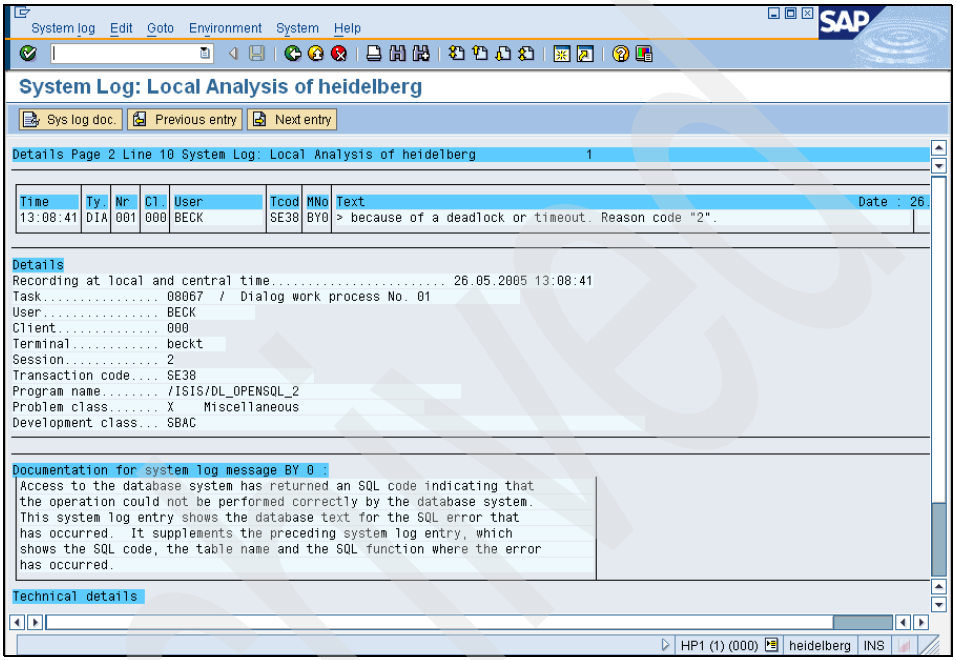


Figure 8-69 system logs: Display Detail

8.3 Support Integration of SAP and DB2

IBM and SAP works together tightly to maintain an SAP DB2 Support Structure to provide customer service in case of a customer problem on the SAP/DB2 UDB layer.

There are two types of DB2 UDB licenses which are based on your SAP Software licensing contract:

- *DB2 OEM License*:
  DB2 UDB is sold as integrated part of SAP software products. In this case, SAP sells DB2 UDB under OEM license from IBM.

- *DB2 Passport Advantage® License*:
  SAP software is sold without database license, in this case the customer must obtain entitlement for use DB2 UDB from IBM via IBM Passport Advantage.

In either case, the following global agreement between SAP AG and IBM Corporation exists:

- SAP Active Global Support is the Primary support contact for both SAP customers with DB2 OEM license and SAP customer with DB2 PPA license.
- Customers with IBM Passport Advantage support entitlement for DB2 UDB are strongly encouraged to report DB2 related problems of their SAP systems to SAP Active Global Support

This procedure is documented in SAP Note *686501* and IBM Tech Note 1051638.

The support structure consists of:

- SAP Active Global Support (SAP AG)
- SAP DB2 UDB Development Support (SAP AG)
- IBM DB2 Advance Support (IBM)

When you encounters a problem on your SAP/DB2 UDB system, you should first contact SAP Active Global Support to report the problem through the SAP Message system. SAP Active Global Support works 24h X 365d to provide initial problem analysis and resolution. For production system down situation that happens during non-business hours for SAP Development Support, SAP Active Global Support directly notifies IBM DB2 Advanced Support (without the need to go through IBM DB2 Level 1 support) via electronic interface.

In other situations that require more in-depth analysis, SAP Active Global Support notifies SAP Development Support to get engaged. SAP Development Support provides development-level problem resolution. The two centers for SAP Development Support located in Toronto, Canada and Walldorf, Germany work together to provide coverage for customer from different time zones in the world. If SAP Development Support determines that the problem requires more in-depth analysis in the DB2 UDB area, SAP Development Support opens a IBM DB2 PMR on behalf of the customer to engage IBM DB2 Advance Support.

Figure 8-70 illustrates the SAP IBM Support Process.

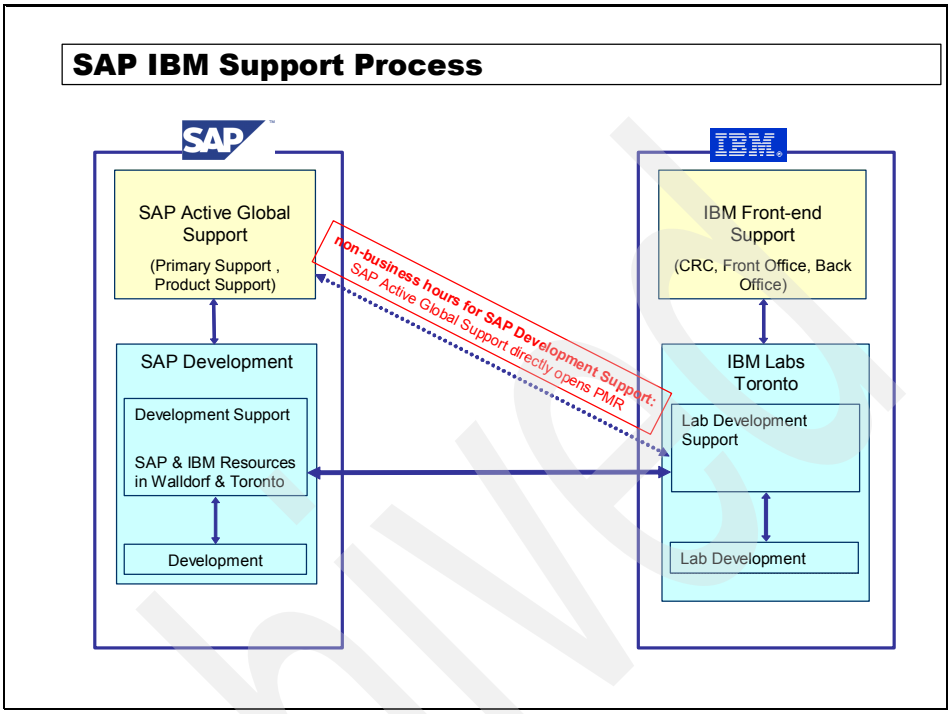


Figure 8-70 SAP IBM Support Process

8.4 Problem research resources

A wide range of knowledge base resources are available to assist you in using DB2 UDB and SAP products, and troubleshooting problems.

- **DB2 technical support Web site:**

  Refer to the DB2 technical support web site if you are experiencing DB2 UDB problems and want assistance on finding possible causes and solutions. The DB2 technical support site for Linux, UNIX, and Windows has the links to the latest DB2 UDB publications (product manuals, redbooks, white papers, and technotes, etc.), Authorized Program Analysis Reports (APARs, that is, defects), Fixpaks, and many other resources. You can search through this knowledge base to find possible solutions to your problems.

  Access the DB2 technical support web site at:

  `http://www.ibm.com/software/data/db2/udb/support`

- **DB2 Information Center:**

  DB2 Information Center is a search engine allowing the quick access to the entire DB2 UDB library made up of the complete set of DB2 UDB product manuals, release notes and other publications. You can refer to the “Support and troubleshooting” branch of the DB2 Information Center navigation tree in the left pane of the browser window to see a complete listing of the DB2 UDB troubleshooting documentation.

  Access the DB2 Information Center web site at:

  `http://publib.boulder.ibm.com/infocenter/db2help/index.jsp`

- **SAP Service Marketplace:**

  SAP Service Marketplace contains a variety of internet portals that enable the collaboration among SAP, its customers and partners, such as SAP support portal, SAP partner portal, SAP help portal, and SAP community etc.

  Access the SAP Service Marketplace Web site at:

  `http://service.sap.com`

- **The community for SAP on DB2 UDB for UNIX and Windows:**

  SAP-DB2.de provides you with

  – Up to date information and latest news for SAP on DB2 UDB for Unix and Windows

  – A forum where you can meet others, discuss topics and post questions

  – FAQs on technical issues as well as on this site

  – A link matrix which brings you directly at the nearest point in the SAP Service Marketplace for downloading support packages, binary patches, etc.

  – Most important SAP-Notes

  – A collection of SAP-Doku

  – Other useful Web Links

  SAP-DB2.de is a private, non-commercial initiative and not an official site of SAP AG or IBM. SAP-DB2.de Web site is

  `http://www.sap-db2.de`

- **Newsgroups:**

  There is a very active DB2 community online that interact via Newsgroups. By accessing Newsgroups, you have access to the knowledge, problems, and solutions of others on a massive number of topics.

  There are IBM hosted DB2 Newsgroups on the NNTP server `news.software.ibm.com`. Among all the IBM DB2 Newsgroup available there, `ibm.software.db2.udb` is the most heavily used.

- **Miscellaneous:**

  Other resources that can assist in problem determination are:

  - DB2 alerts E-mail:

    `http://www.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/db2alert.d2w/report`

  - DB2 support news:

    `http://www.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/newsletter.d2w/report`

  - DB2 support newsletter:

    `http://www-3.ibm.com/software/mailing-lists/`

  - DB2 online magazine:

    `http://www.db2mag.com/`

  - DB2 developer domain:

    `http://www.ibm.com/software/data/developer`

  - Community for SAP in a DB2 environment:

    `http://www.sap-db2.de`

  - SAP easy service marketplace:

    `http://www.easymarketplace.de`

9

Using features of DB2 UDB V8.2 and V8.2.2 with older SAP systems

The term, “older” SAP systems, refers to the SAP systems prior to SAP NetWeaver 2004s. In this chapter we discuss the following topics:

- How to upgrade DB2 UDB from V8.x to V8.2.2 on the older SAP systems
- Which DB2 UDB V8.2.2 features can be used with older SAP systems
- How to enable the supported DB2 UDB V8.2.2 features in older SAP systems

9.1 Upgrading DB2 UDB to V8.2.2

Upgrading DB2 UDB to V8.2.2 means to apply the FixPak 9 on a DB2 UDB running on V8.x. The following SAP Notes provide the procedures to upgrade DB2 UDB to version 8.2.2 on older SAP systems:

- UNIX: SAP Note *603972*
- Windows: SAP Note *603981*

Tip: To migrate an SAP system with DB2 UDB version below V8.x to V8, refer to the SAP documentation *Migration to Version 8 of the IBM DB2 Universal Database for UNIX and Windows* which can be downloaded from SAPNet. Accessing SAPNet requires an SAPNet user ID and password, on this site:

`http://service.sap.com/instguides:`

To download the document from this site:

- Click **Other Documentation**
- Click **Database Upgrades**
- Click **DB2 UDB**
- On the right, click to open the *Migration to Version 8 of the IBM DB2 Universal Database for UNIX and Windows* document

The following general steps are used to apply the FixPak 9 on a UNIX system:

- Download and extract the FixPak 9 from the SAPNet, for which you need an SAPNet user and password. On the SAPNet Web site:

  `http://service.sap.com/swcenter-3pmain`

  Download the Fixpak 9 as follows:

  - Click **DB2/UDB**
  - Click **DB2 Version 8 software download**
  - Click **Delta Upgrade**
  - From the list you can download the DB2 UDB FixPak 9 for your operating system by clicking it.
- Extract the downloaded ZIP file and place the files `SAPCAR`, `V8DB2FP9.SAR` and `db6_update_db.sh` in the temporary directory `<tmp>` in your database server.
- Use **`<tmp>/SAPCAR -xvf <tmp>/V8DB2FP9.SAR`** to extract the FixPak 9 to `<tmp>/FIXPAK`
- Check the prerequisites for your operating system, like specific APARs that are necessary on AIX. For this refer to SAP Note *603972*. For AIX, also refer

to the document *Known issues for DB2 Universal Database on AIX 4.3.3, 5.1, 5.2, and 5.3* which can be found at:

http://www-1.ibm.com/support/docview.wss?uid=swg21165448

- Stop your SAP systems, the database instances, the DB2 UDB administrator instance, and the DB2 UDB licence daemon on the server where you are going to apply the FixPak 9.
- Switch to the directory <tmp>/FIXPAK and start the installation of the FixPak 9 software with **./installFixPak -y**.

  Example 9-1 shows excerpts of the software installation on an AIX system. The output device of **installFixPak** is standard device. To save the output in file, redirect the output to file.

  The installation script will first check if the prerequisites are in place. However, it will continue the installation no matter if all the prerequisites are met. We recommend to check the output of the **./installFixPak** carefully to make sure all the required files are installed successfully.

Example 9-1 Installation of FixPak 9 on AIX (an excerpt)

```
> ./installFixPak -y

Checking level of DB2 V8 installation...
Running update all
geninstall -I "a -cgNqwX -J"  -Z   -d . -f File 2>&1

File:
    db2_08_01.ca                    8.1.1.88
    db2_08_01.cc                    8.1.1.88
    db2_08_01.ch.en_US.iso88591     8.1.1.88
    db2_08_01.cj                    8.1.1.88
    db2_08_01.client                8.1.1.88
    db2_08_01.cnvucs                8.1.1.88
    db2_08_01.conn                  8.1.1.88
    db2_08_01.conv                  8.1.1.88
    db2_08_01.cs.rte                8.1.1.88
    db2_08_01.das                   8.1.1.88
    db2_08_01.db2.engn              8.1.1.88
    db2_08_01.db2.rte               8.1.1.88
    db2_08_01.db2.samples           8.1.1.88
    db2_08_01.dj                    8.1.1.88
    db2_08_01.essg                  8.1.1.88
    db2_08_01.fs                    8.1.1.88
    db2_08_01.icuc                  8.1.1.88
    db2_08_01.icut                  8.1.1.88
    db2_08_01.inst                  8.1.1.88
    db2_08_01.jdbc                  8.1.1.88
    db2_08_01.jhlp.en_US.iso88591   8.1.1.88
    db2_08_01.ldap                  8.1.1.88
    db2_08_01.msg.en_US.iso88591    8.1.1.88
    db2_08_01.pext                  8.1.1.88
    db2_08_01.repl                  8.1.1.88
    db2_08_01.sqlproc               8.1.1.88
...
...
+-----------------------------------------------------------------------------+
             Installing Software...
+-----------------------------------------------------------------------------+
```

```
installp:  APPLYING software for:
    db2_08_01.ch.en_US.iso88591 8.1.1.88


. . . . . << Copyright notice for db2_08_01.ch.en_US.iso88591 >> . . . . . .

Licensed Materials - Property of IBM

 5648B9700

 (C) Copyright International Business Machines Corp. 1993, 2002.

All rights reserved.
US Government Users Restricted Rights - Use, duplication or disclosure
restricted by GSA ADP Schedule Contract with IBM Corp.
. . . . . << End of copyright notice for db2_08_01.ch.en_US.iso88591 >>. . . .

Restoring files, please wait.
1694 files restored.
3585 files restored.
5247 files restored.
Filesets processed:  1 of 26  (Total time:  3 mins 18 secs).

installp:  APPLYING software for:
    db2_08_01.cc 8.1.1.88
...
...
+-----------------------------------------------------------------------------+
                Summaries:
+-----------------------------------------------------------------------------+

Installation Summary
--------------------
Name                            Level           Part       Event       Result
-------------------------------------------------------------------------------
db2_08_01.ch.en_US.iso88591 8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.cc                8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.sqlproc           8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.repl              8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.pext              8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.msg.en_US.iso8859 8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.ldap              8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.jhlp.en_US.iso885 8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.jdbc              8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.inst              8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.icut              8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.icuc              8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.fs                8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.essg              8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.dj                8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.db2.samples       8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.db2.rte           8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.das               8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.cs.rte            8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.conv              8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.conn              8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.cnvucs            8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.client            8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.cj                8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.ca                8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.db2.engn          8.1.1.88        USR        APPLY       SUCCESS
db2_08_01.ca                8.1.1.88        USR        COMMIT      SUCCESS
db2_08_01.cc                8.1.1.88        USR        COMMIT      SUCCESS
db2_08_01.ch.en_US.iso88591 8.1.1.88        USR        COMMIT      SUCCESS
db2_08_01.db2.samples       8.1.1.88        USR        COMMIT      SUCCESS
db2_08_01.fs                8.1.1.88        USR        COMMIT      SUCCESS
```

```
db2_08_01.icuc              8.1.1.88        USR         COMMIT      SUCCESS
db2_08_01.icut              8.1.1.88        USR         COMMIT      SUCCESS
db2_08_01.jhlp.en_US.iso885 8.1.1.88        USR         COMMIT      SUCCESS
db2_08_01.msg.en_US.iso8859 8.1.1.88        USR         COMMIT      SUCCESS
db2_08_01.pext              8.1.1.88        USR         COMMIT      SUCCESS
db2_08_01.cnvucs            8.1.1.88        USR         COMMIT      SUCCESS
db2_08_01.client            8.1.1.88        USR         COMMIT      SUCCESS
db2_08_01.cj                8.1.1.88        USR         COMMIT      SUCCESS
db2_08_01.conv              8.1.1.88        USR         COMMIT      SUCCESS
db2_08_01.db2.rte           8.1.1.88        USR         COMMIT      SUCCESS
db2_08_01.jdbc              8.1.1.88        USR         COMMIT      SUCCESS
db2_08_01.ldap              8.1.1.88        USR         COMMIT      SUCCESS
db2_08_01.repl              8.1.1.88        USR         COMMIT      SUCCESS
db2_08_01.sqlproc           8.1.1.88        USR         COMMIT      SUCCESS
db2_08_01.conn              8.1.1.88        USR         COMMIT      SUCCESS
db2_08_01.cs.rte            8.1.1.88        USR         COMMIT      SUCCESS
db2_08_01.das               8.1.1.88        USR         COMMIT      SUCCESS
db2_08_01.db2.engn          8.1.1.88        USR         COMMIT      SUCCESS
db2_08_01.dj                8.1.1.88        USR         COMMIT      SUCCESS
db2_08_01.essg              8.1.1.88        USR         COMMIT      SUCCESS
db2_08_01.inst              8.1.1.88        USR         COMMIT      SUCCESS
```

- Update all instances on your system, starting with the administrator instance. For this you need the **./dasupdt *admininstance*** and **./db2iupdt db2*sapsid*** commands located in the <*dbsw*>/instance directory. The <*dbsw*> directory is /usr/opt/db2_08_01 on AIX and /opt/IBM/db2/V8.1 on Linux, HP and Solaris.

  Tip: To check the name of your admin instance, you can use the command **<*dbsw*>/instance/daslist**. The command **<*dbsw*>/instance/db2ilist** prints a complete list of all your DB2 UDB V8 instances.

  Example 9-2 shows the instance update on AIX from DB2 UDB V8.2 to V8.2.2. The example output script shown only covers the new DB2 V8.2.2 features.

Example 9-2 Updating the instances on AIX

```
> /usr/opt/db2_08_01/instance/daslist
dasusr1
> /usr/opt/db2_08_01/instance/dasupdt dasusr1
SQL4410W  The DB2 Administration Server is not active.
SQL4406W  The DB2 Administration Server was started successfully.
DBI1070I Program dasupdt completed successfully.

> /usr/opt/db2_08_01/instance/db2ilist
db2gr4
db2gr2
db2gr5
> /usr/opt/db2_08_01/instance/db2iupdt db2gr2
DBI1070I Program db2iupdt completed successfully.
```

```
> /usr/opt/db2_08_01/instance/db2iupdt db2gr4
DBI1070I Program db2iupdt completed successfully.

> /usr/opt/db2_08_01/instance/db2iupdt db2gr5
DBI1070I Program db2iupdt completed successfully.
```

- Use the script db6_update_db.sh from the downloaded ZIP file to create an output script for each database instance on your server with a list of recommendations what should be adapted on each database instance.

  It is possible to tell the script via options which DB2 UDB V8.2.2 features it should enable and which not. Per default the script enables the *auto-resize DMS table spaces* and DB2_WORKLOAD=SAP features; the *automatic statistic collection* feature must be explicitly enabled with an option. The name of the generated scripts for each instance are db6_update_db_out.sh. Before executing the generated scripts you should check the recommended commands.

  Example 9-3 shows the options for the script and also a sample script enabling auto-resize DMS table spaces and DB2_WORKLOAD_SAP. The db6_update_db.sh script also covers the new features of DB2 UDB V8.2, which is DB2 UDB V8.1 FixPak 7. In the example the upgrade has been done.

Example 9-3 Example of db6_update_db.sh usage

```
> ./db6_update_db.sh -h

usage: ./db6_update_db.sh [-d <DBSID>] [-r] [-k] [-a]

          -d <DBSID>    specify the databasename DBSID, if not, DB2DBDFT
is
                           used.
          -r            set autorunstats.
          -k            do not set DB2_WORKLOAD=SAP.
          -a            do not set autoresize on DMS tablespaces.


> db2start
05/25/2005 19:03:02     0   0   SQL1063N  DB2START processing was
successful.
SQL1063N  DB2START processing was successful.


> ./db6_update_db.sh -d GR2
db6_update_db_out.sh created. Please verify and run this script.


> cat db6_update_db_out.sh
```

```
#!/bin/sh

#*****************************************************************************
*
#
# This batch file was generated by db6_update_db.bat.
# Depends on your system configuration it enables db2 features:
# - run db2updv8
# - enable autoresize on DMS table spaces
# - create 16k bufferpool if necessary
# - create SYSTOOLSPACE table space if not exists
# - create SYSTOOLSTMPSPACE table spaces if not exists
# - set db2 registry DB2_WORKLOAD=SAP
# - enable AUTORUNSTATS if not set already
# - rebind packages
#
#*****************************************************************************
*

echo
echo "****************************************" | tee
db6_update_db_out.log
echo Connect to GR2 | tee -a db6_update_db_out.log
db2 connect to GR2 1> /dev/null
if [ $? != 0 ]; then
  echo Can not connect to GR2. | tee -a db6_update_db_out.log
  echo
  exit 1
fi

echo
echo "****************************************" | tee -a
     db6_update_db_out.log
echo run db2updv8. | tee -a db6_update_db_out.log
db2updv8 -d GR2 | tee -a db6_update_db_out.log

echo
echo "****************************************" | tee -a
      db6_update_db_out.log
echo Change DMS tablespace to AUTORESIZE | tee -a db6_update_db_out.log
db2 alter tablespace PSAPBTABD autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace PSAPBTABI autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace PSAPCLUD autoresize yes | tee -a db6_update_db_out.log
db2 alter tablespace PSAPCLUI autoresize yes | tee -a db6_update_db_out.log
db2 alter tablespace PSAPDDICD autoresize yes | tee -a
db6_update_db_out.log
```

```
db2 alter tablespace PSAPDDICI autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace PSAPDOCUD autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace PSAPDOCUI autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace PSAPEL46DD autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace PSAPEL46DI autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace PSAPES46DD autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace PSAPES46DI autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace PSAPLOADD autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace PSAPLOADI autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace PSAPPOOLD autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace PSAPPOOLI autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace PSAPPROTD autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace PSAPPROTI autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace PSAPSOURCED autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace PSAPSOURCEI autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace PSAPSTABD autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace PSAPSTABI autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace PSAPUSER1D autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace PSAPUSER1I autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace SYSCATSPACE autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace SYSTOOLSPACE autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace ZSAPAPQD32D autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace ZSAPAPQD32I autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace ZSAPBTAB2D autoresize yes | tee -a
db6_update_db_out.log
```

```
db2 alter tablespace ZSAPBTAB2I autoresize yes | tee -a
db6_update_db_out.log
db2 alter tablespace ZSAPBTAB3D autoresize yes | tee -a
db6_update_db_out.log
echo
echo "******************************************" | tee -a
db6_update_db_out.log
echo Create 16k bufferpool | tee -a db6_update_db_out.log
db2 create bufferpool BP_STD_16K database partition group IBMDEFAULTGROUP
size
      128 pagesize 16K | tee -a db6_update_db_out.log


echo "******************************************" | tee -a
      db6_update_db_out.log
echo Set DB2_WORKLOAD=SAP | tee -a db6_update_db_out.log
db2set DB2_INLIST_TO_NLJN=
db2set DB2_MINIMIZE_LISTPREFETCH=
db2set DB2_UPDATE_PART_KEY=
db2set DB2_REDUCED_OPTIMIZATION=
db2set DB2_EVALUNCOMMITTED=
db2set DB2_SKIPDELETED=
db2set DB2_RR_TO_RS=
db2set DB2_WORKLOAD=SAP | tee -a db6_update_db_out.log


echo "******************************************" | tee -a
      db6_update_db_out.log
echo Rebind packages. | tee -a db6_update_db_out.log
db2 bind ~/sqllib/bnd/@db2ubind.lst blocking all grant public | tee -a
      db6_update_db_out.log
db2 bind ~/sqllib/bnd/@db2cli.lst blocking all grant public | tee -a
      db6_update_db_out.log

echo
echo "******************************************" | tee -a
      db6_update_db_out.log
echo Terminate connection to GR2 | tee -a db6_update_db_out.log
db2 terminate 1> /dev/null

echo
echo "******************************************" | tee -a
      db6_update_db_out.log
echo Done. | tee -a db6_update_db_out.log
```

- Restart your databases and SAP systems.

Table 9-1 shows SAP Notes discussing DB2 UDB V8.2 and V8.2.2 features that can be used within older SAP systems.

Table 9-1 SAP Notes - DB2 UDB V8.2 and V8.2.2 features with SAP older systems

SAP Note number	SAP Note title
603972	DB6: Installing V8.1/ V8.2 FixPaks on UNIX
603981	DB6: Installing V8.1/V8.2 FixPaks on Win NT/Win 2000
780546	DB6: Use of new functions in DB2 UDB FixPak 9 (V8.2.2)
842898	DB6: dmdb6bkp Patch 11
825392	DB6: Checking the DB2 "DB2_WORKLOAD=SAP" registry
804651	DB6: SQL0437W reason code 13
584952	DB6: DB2 UDB ESE Version 8 Standard Parameter

10

Using SAP NetWeaver 2004s Business Intelligence with DB2 UDB

This chapter discusses new features of SAP NetWeaver 2004s Business Intelligence (SAP BI) with DB2 UDB V8.2.2. The topics include

- A short introduction to the architecture, framework, and components of SAP BI.
- Clustering support in DB2 UDB V8.2.2
- Clustering support within SAP BI:
  - Index clustering and MDC for InfoCube tables
  - MDC for DataStore Object tables
  - MDC for PSA tables
  - Re-clustering of existing InfoCubes and DataStore objects
- InfoCube Compression with SQL Merge statement
- Data delete with new SQL `DELETE` Statement
- Use of database sampled statistics within SAP BI
- Heterogeneous system copy of SAP BI systems to DB2 UDB

10.1 SAP BI technical overview

SAP BI supports both the strategic and the tactical (or operational) decision making process.

The SAP BI system is the only SAP system beside SCM that supports database partitions when installed with DB2 UDB. With the use of database partitions, data will be physically distributed between the partitions of the database and can be queried in parallel on all partitions.

This section provides a brief overview of the SAP BI information model and the underlying database tables. Furthermore, the most important SAP BI operations are briefly described. This provides you with a basic understanding of the architecture, framework, and components of SAP BI, and how they work together to enable an enterprise data warehousing solution. Please refer to the IBM Redbook *Building and Scaling SAP Business Information Warehouse on DB2 UDB ESE*, SG24-7094 for more detailed information.

10.1.1 SAP BI information model

The SAP BI information model supports the following conceptual layers of data warehousing:

- *Multi-dimensional models*, which enable views of data required for analysis.
- *Operational data store*, to hold current data updates from the operational transaction systems of the business.
- *Data warehouse*, to hold transformed and accurate data that has been integrated from the business processes across the enterprise to enable business decision-making.

The information model is based on a fundamental building block called an *InfoObject*. For example, 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, DataStore object, and InfoSource. These elements are described later in this section.

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.

Figure 10-1 shows the elements of the SAP BI information model, all of which store metadata. In addition, three of the four elements also store transactional or master data. These are the PSA, DataStore object, and InfoCube. Master data is data that remains unchanged over long periods of time. Examples of master data are items such as customer name and address, or an organizational structure.

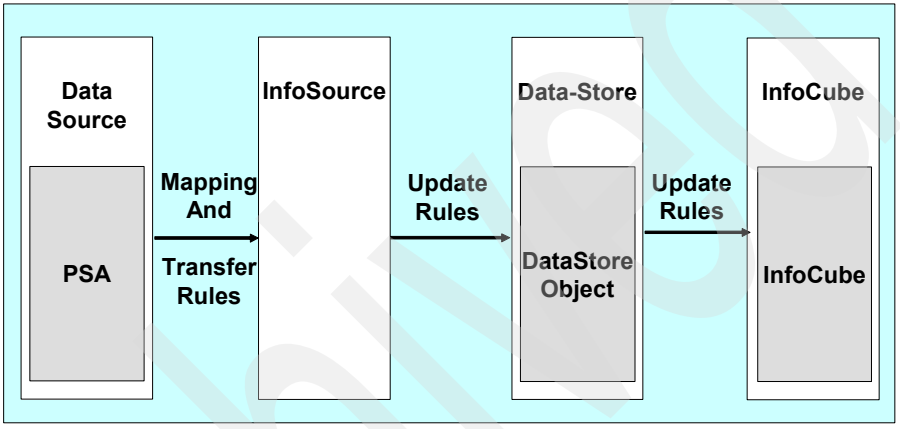


Figure 10-1 Elements of the SAP BI information model

In addition to InfoObjects, here is a list of other key elements in the information model:

- *DataSource*:

  Data is transferred into SAP BI in a flat structure. That is, it is a table rather than a multi-dimensional data structure. DataSources contain the definitions of the source data.

- *Persistent Staging Area (PSA)*:

  In the SAP BI information model, data is physically stored in PSA objects. These are collections of flat tables holding extracted data that has not yet been cleaned or transformed. The PSA is the initial storage area of data, where requested data is saved unchanged from the source system according to the structure defined in the DataSource.

- *InfoSource*:

  InfoObjects that belong together logically, from a business point of view, are grouped into InfoSources. InfoSources (and their underlying InfoObjects) can be filled with any data from within the enterprise or from external sources. They can hold both transactional data and master data.

  - Transactional data is generated from transactions in an Online Transaction Processing (OLTP) system, such as SAP R/3; it is quantifiable; and it can be granular.
  - Master data, such as a customer address or an organizational structure, typically remains unchanged over a long period of time. Master data in SAP BI includes attributes, texts, and hierarchies.

- *Operational data store (DataStore) object*:

  This describes a consolidated data set from one or several InfoSources. In contrast to the multi-dimensional data models of InfoCubes, data in DataStore objects is stored in flat, transparent, database tables. DataStore object data can be updated into InfoCubes or other DataStore objects using a delta update. Data in a DataStore object can be analyzed with the SAP BI Business Explorer (BEx) tool. It is typically used to integrate data that comes from different sources, for delta update into InfoCubes, and for day-to-day decision-making.

- *InfoCubes*:

  These are containers that organize data around its multi-dimensionality, in terms of business dimensions. They are used to answer complex business questions on topics such as revenues per region, revenues per office within each region, year-to-date revenues, and for comparisons with previous periods. InfoCubes can be accessed by the SAP BI Business Explorer for reporting and Online Analytical Processing (OLAP) analysis.

InfoObjects, InfoCubes, and DataStore objects belong to the so-called *InfoProviders*. These are objects in SAP BI that can be analyzed.

Figure 10-2 provides a more complete view of the possible flow of data in SAP BI. Data is normally loaded into a PSA first, and from there into DataStore objects and InfoCubes. It is also possible to directly load data into DataStore objects and InfoCubes, and from DataStore objects into InfoCubes.

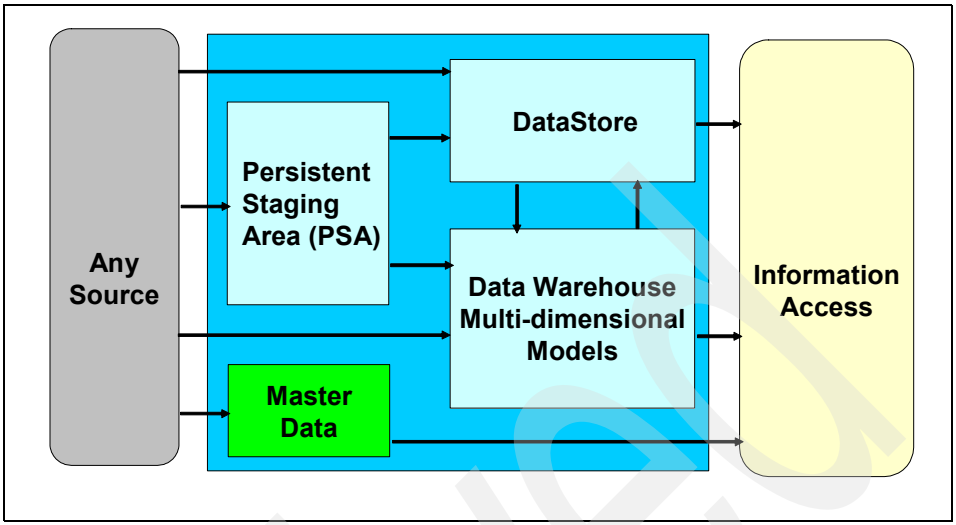


Figure 10-2 Data flow in SAP BI

Data is loaded into SAP BI PSA, DataStore, and InfoCubes through InfoPackages. An InfoPackage describes the data to be loaded from a source system, the data targets and how the data is to be processed. When an InfoPackage is scheduled to load data from a source system a new request ID is generated. The data to be loaded is structured into one or more data packages, each having a configurable maximum number of records. Requests are maintained in a request InfoObject named *0REQUEST* that contains the request IDs and 4-byte integer surrogate identifiers (SIDs) generated for them. Within the data packages of a request, the records are numbered.

PSA Tables

PSA tables consist of the columns defined in the DataSource plus the three additional columns request ID, data package and record number. The three additional columns form the primary key, used to address each record in the table. As each request corresponds to one data load operation, only one process writes into a single request at a time.

PSA tables can get very large because many customers keep data that has already been loaded into DataStore objects and InfoCubes in PSA. If data has to be reloaded or the contents of DataStore objects or InfoCubes have to be rebuilt, this avoids that data has to be extracted from the source systems again.

In DB2 UDB, PSA tables can reside in a partitioned table space. The partitioning key is always the record number column. This guarantees a balanced distribution of the data rows belonging to a single request over all the database partitions and thus enables full inter partition parallelism during request processing.

All PSA operations are based on requests: data load, propagation into data targets, and delete.

DataStore Objects

DataStore objects consist of key fields and data fields. Both key fields and data fields are InfoObjects. SAP BI distinguishes between two types of data used in analysis:

- *Key figures*: sales revenue, fixed costs, sales quantity, or number of employees, etc.
- *Characteristics*: customer type, fiscal year, period, or region are some 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.

The key fields of DataStore objects are usually characteristics InfoObjects (for example, order number). The data fields are usually key figure InfoObjects (for example, order quantity). As opposed to InfoCubes, fields in a DataStore object can be overwritten.

The DataStore 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 basic operations on DataStore objects are data load, data activation, delta update into data targets, and reporting. Except for reporting, these operations are based on requests.

SAP NetWeaver 2004s BI provides three types of DataStore objects that are described briefly in the following sections:

- Standard DataStore objects
- DataStore objects for direct update
- Write-optimized DataStore objects

Standard DataStore objects

At the database level, *standard DataStore objects* consist of three tables:

- Active table:

  The data in this table is used for reporting and delta update into data targets.

- Activation queue:

  This table contains data that is new or modified since the last activation. It is not yet available for reporting and delta update into data targets.

- Change log:

  The change log is used for the delta update from the DataStore object into other DataStore objects or InfoCubes. It records new data and changes to the existing active data of the DataStore object.

When loading data into a standard DataStore object it is first inserted into the activation queue. To make it available for reporting and delta update it has to be activated.

The *active table* has the key fields and the data fields of the DataStore object as columns. The key fields form the primary key. Users can define additional indexes on DataStore objects created on the active table to support queries.

The *activation queue table* has three additional columns for the SID of the request generated when loading the data, data package and record number. These form the primary key.

The *change log table* has the same columns as the active table and the additional three columns request ID (not request SID), data package and record number, forming the primary key of the table. During activation, several requests from the activation queue can be summarized into one new change log request.

In DB2 UDB, active table, activation queue, and change log can reside in a partitioned table space. The partitioning key of the active table consists of the logical key columns. This guarantees a balanced distribution of the data over all the database partitions. The partitioning key of the activation queue table and the change log is the record number column. This guarantees a balanced distribution of the data belonging to one request over all the database partitions.

DataStore objects for direct update

A *DataStore object for direct update* consists of only the active table. Data loaded into a DataStore object for direct update is directly available for reporting and delta update into data targets. Eliminating the activation queue and the change log reduces the processing overhead for data load, activation, and the tracking of changes. The implementation of the active table in DB2 UDB corresponds to the one for standard DataStore objects.

Write-optimized DataStore objects

A *write-optimized DataStore object* only consists of the active table. In this case, the active table has the three additional columns request ID, data package and record identifier. The user can decide whether to check uniqueness of the logical key of the DataStore object or not. The default is to check uniqueness. In this case, a unique index on the logical key columns is created. Write-optimized DataStore objects are specifically designed for fast data load, without the overhead of data activation and maintenance of a change log. Reporting only plays a minor role. The data in a write-optimized DataStore object is directly available for use when the InfoPackage has been loaded.

In DB2 UDB, the active table of write-optimized DataStore objects can reside in a partitioned table space. The partitioning key of the active table consists of the logical key fields. If the uniqueness of the logical key fields is to be checked, a unique index is created on them. However, the primary key index on request ID, data package and record number cannot be created because the columns of all unique indexes have to be part of the partitioning key. Therefore, in DB2 UDB, a non-unique index is created on request ID, data package and record number.

InfoCubes

SAP BI uses an extended star schema in defining and creating InfoCubes. InfoCubes contain key figures and characteristics.

An *InfoCube* is represented in the database as a set of relational tables, as shown in Figure 10-3. These are arranged according to the star schema, a technique that organizes data in terms of data facts and business dimensions.

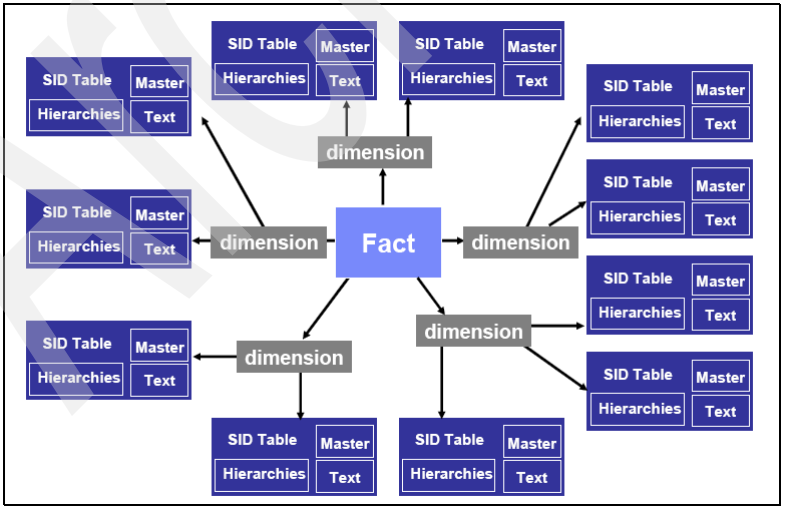


Figure 10-3 Extended star schema

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 BI can have up to 16 dimensions. By default, every InfoCube has the three standard dimensions Data Package, Time, and Unit.

SAP BI uses an extended star schema, which builds on the basic star schema by storing master data about attributes, hierarchies, and text, in separate tables that are shared between InfoCubes. This reduces data redundancies because master data is stored only once, and then used by various InfoCubes.

Characteristic values in the dimension table are replaced by surrogate identifiers (SIDs). These are numeric key values (4-byte integers) that 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.

The main operations on InfoCubes are data load, compression of requests, aggregate filling and roll-up, deletion of requests or selective deletion, and reporting. InfoCubes have two fact tables, the F fact table and the E fact table. Data is first loaded request-based into the F fact table. Requests can be deleted from the F fact table if the data is, for example, not consistent. If the quality status of a request is appropriate, the request can be compressed (see 10.5, “InfoCube compression” on page 608). During compression, data is transferred from the F fact table to the E fact table of the InfoCube or aggregate. Compressed requests cannot be deleted from the InfoCube anymore. Selective deletion can be used for deleting data from an InfoCube based on certain search criteria, for example data older than two years. It deletes data holding the search criteria from both fact tables, the F fact table and the E fact table.

In DB2 UDB, InfoCube fact tables can reside in a partitioned table space. The partitioning key consists of the dimension key columns without the package dimension. By default, single column indexes are created on the dimension key columns. On the E fact table, a combined index over all dimension key columns is created.

Aggregates

Aggregates represent another technique for improving query performance. They are materialized, summarized views of the data in an InfoCube, and store a subset of InfoCube data in a redundant form. When an appropriate aggregate for a query exists, the summarized data can be read directly from the database during query execution instead of having to perform this summarization during runtime. In SAP BI, the system generates suggestions for creating optimal aggregates. The system administrator can then decide whether to create those aggregates.

Aggregates reduce the volume of data to be read from the database, speed up query execution time, and reduce the overall workload 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 but are implemented using regular database tables.

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 InfoCube 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. 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.

If aggregates are defined, new data packets requests that are loaded into the InfoCube cannot be used for reporting until they are written to the aggregates by a so-called “roll-up”. In other words, data that has been recently loaded into an InfoCube is not visible for reporting, from the InfoCube or aggregates, until an aggregate roll-up takes place. During this process, you can continue reporting using the data that existed prior to the recent data load. The new data is only displayed by queries that are executed after a successful roll-up.

10.2 Clustering support in DB2 UDB V8.2.2

This section briefly describes the two clustering methods *index clustering* and *multi-dimensional clustering (MDC)* that DB2 UDB V8.2.2 supports. Range-clustering that is also available in DB2 UDB is not discussed here because it is not used in an SAP environment.

10.2.1 Index clustering

Regular tables have indexes that are record-based. Any clustering of the indexes is restricted to a single dimension. Prior to Version 8, DB2 UDB supported only single-dimensional clustering of data, through clustering indexes. Using a clustering index, DB2 UDB attempts to maintain the physical order of data on pages in the key order of the index when records are inserted and updated in the table. Clustering indexes greatly improve the performance of range queries that have predicates containing the key or keys of the clustering index. Performance is improved with a good clustering index because only a portion of the table needs to be accessed, and more efficient pre-fetching can be performed.

The table in Figure 10-4 has two record-based indexes defined on it:

- Clustering index on `Year`
- Another index on `Region`

The `Year` index is a clustering index, which means that as keys are scanned in the index, the corresponding records should be found for the most part on the same or neighboring pages in the table. In contrast, the `Region` index is un-clustered, which means that as keys are scanned in that index, the corresponding records will likely be found on random pages throughout the table. Scans on the clustering index will exhibit better I/O performance and will benefit more from sequential pre-fetching, the more clustered the data is to that index.

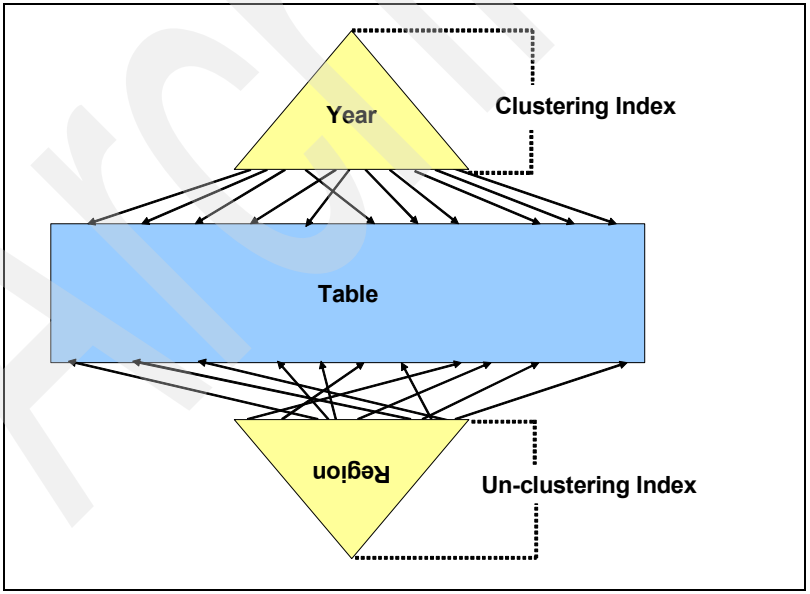


Figure 10-4 Table with a clustering index and an un-clustered index

Data clustering with a clustering index has some drawbacks. First, because space is filled up on data pages over time, clustering is not guaranteed. An insert operation will attempt to add a record to a page nearby to those having the same or similar clustering key values, but if no space can be found in the ideal location, it will be inserted elsewhere in the table. Therefore, periodic table reorganizations may be necessary to re-cluster the table and to setup pages with additional free space to accommodate future clustered insert requests. Second, only one index can be designated as the “clustering” index, and all other indexes are un-clustered, because the data can only be physically clustered along one dimension. Third, for all non-block indexes including clustering index, because record-based indexes contain a pointer for every single record in the table, they can be very large.

Creating tables with clustering indexes

To create a clustering index on a table you simply add the key word CLUSTER to the CREATE INDEX statement. In Example 10-1, the DDL statements create a table SALES_TABLE, a clustering index on column YEAR, and un-clustered indexes on columns REGION and PRODUCT.

Example 10-1 Creating table with clustering index

```
CREATE TABLE SALES_TABLE
  (REGION  VARCHAR(5) NOT NULL,
   YEAR    CHAR(4) NOT NULL,
   PRODUCT VARCHAR(30),
    ...
   QUANTITY INT          NOT NULL,
   PRICE    DECIMAL(17,2) NOT NULL,
   ...)
IN <TABLESPACE>;

CREATE INDEX REGION_INDEX
ON SALES_TABLE (REGION);

CREATE INDEX YEAR_INDEX
ON SALES_TABLE (YEAR) CLUSTER;

CREATE INDEX PRODUCT_INDEX
ON SALES_TABLE (PRODUCT);
```

10.2.2 Multi-dimensional clustering (MDC)

Multi-dimensional clustering (MDC) is used in data warehousing and large database environments. It provides an elegant method for clustering data along multiple dimensions in a flexible, continuous, and automatic way. MDC can significantly improve query performance while reducing the overhead of data

maintenance operations, such as reorganization, and index maintenance operations during insert, update, and delete operations. MDC has been introduced in DB2 UDB V8.1. In DB2 UDB V8.2.2 enhancements have been made that accelerate both data insertion and deletion with MDC tables (roll-in and roll-out).

MDC introduces a new type of index. *Block indexes* point to blocks or groups of records instead of to individual records. By physically organizing data in an MDC table into blocks according to clustering values, and then accessing these blocks using block indexes, MDC addresses all of the drawbacks of clustering indexes and provides significant additional performance benefits:

- MDC enables a table to be physically clustered on more than one key, or dimension, simultaneously. With MDC, the benefits of single-dimensional clustering are therefore extended to multiple dimensions, or clustering keys. Performance of queries with predicates on the clustered dimensions of a table is improved because these queries access only those pages having records with the correct dimension values. The qualifying pages are grouped into blocks enabling most efficient sequential pre-fetching.
- Unlike a table with a clustering index, an MDC table automatically maintains its clustering. This eliminates the need to reorganize an MDC tables for clutering.
- MDC's block-based indexes are much smaller than regular record-based indexes, so that they take up much less disk space and are faster to scan.

In an MDC table, every unique combination of dimension values forms a logical cell. Each cell may be physically made up of one or more blocks of pages. Each data page belongs to exactly one block, and all blocks consist of an equal number of pages. The logical cell will only have enough blocks associated with it to store the records having the dimension values of that logical cell. If there are no records in the table having the dimension values of a particular logical cell, no blocks will be allocated for that logical cell. The set of blocks that contain data having a particular dimension key value is called a *slice*. The size of a block is equal to the extent size of the table space, so that block boundaries line up with extent boundaries.

Block indexes are structurally the same as regular indexes, except that they point to blocks instead of records. Block indexes are smaller than regular indexes by a factor of the block size multiplied by the average number of records on a page. The number of pages in a block is equal to the extent size of the table space, which can range from 2 to 256 pages. The page size can be 4 KB, 8 KB, 16 KB, or 32 KB.

The table in Figure 10-5 is an MDC table that is organized such that records having the same `Region` and `Year` values are grouped together into separate

blocks, or extents. This is an MDC table with two dimensions. In this case, two block indexes are created, one for the `Region` dimension, and another for the `Year` dimension. These block indexes contain pointers only to the blocks in the table. A scan of the `Region` block index for all records having `Region` equal to `East` will find two blocks that qualify. All records, and only those records, having `Region` equal to `East` will be found in these two blocks. At the same time, and completely independently, a scan of the `Year` index for records between 1998 and 1999 will find three blocks that qualify. A data scan of each of these three blocks will return all records and only those records that are between 1998 and 1999, and will find these records clustered on the sequential pages within each of the blocks.

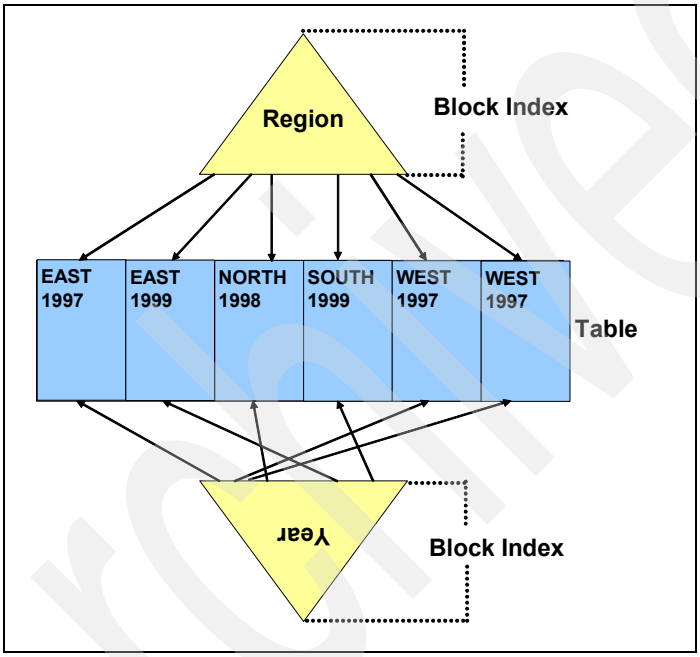


Figure 10-5 MDC table with two block indexes

Figure 10-6 shows the block indexes and the blocks that might have been allocated for the table in Figure 10-5 after a larger amount of data has been inserted. The blocks are represented by blue rectangles, and are numbered according to the logical order of allocated extents in the table. The grid represents the logical partitioning of these blocks, and each square represents a logical cell.

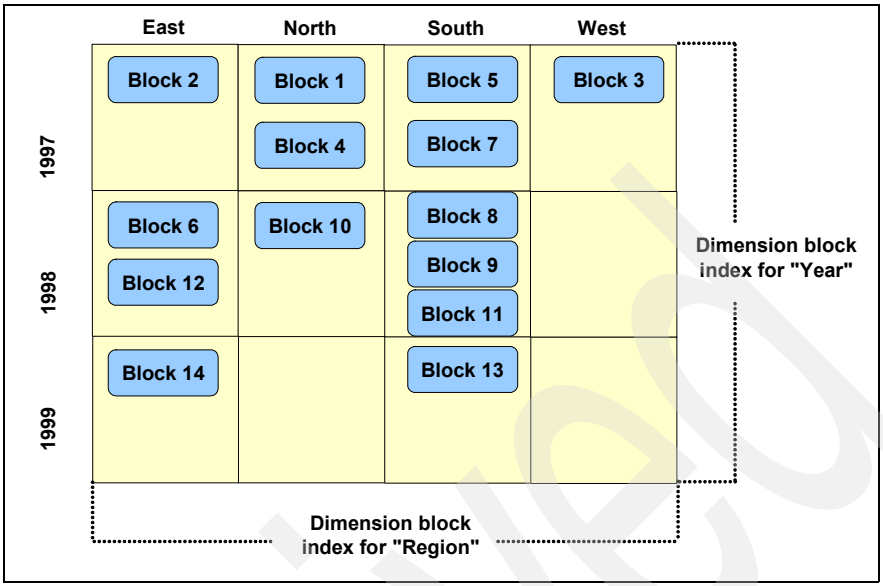


Figure 10-6 Blocks allocated and block indexes for an MDC table

Suppose a new row with `"Region" = "EAST"` and `"Year" = 1999` has to be inserted into the table. If there is still enough free space for the row in the blocks allocated for rows with `"Region" = "EAST"` and "Year" = 1999, it is inserted into one of the blocks. It is not necessary to update the block indexes because no new blocks have to be allocated. If there is not enough free space, either an existing empty block is re-used or a new block (extent) is allocated. The row is inserted into the block and the block indexes are updated with the new block ID. When the last row of a block is deleted the block ID is removed from the block indexes so that it can be reused in either the same or a different logical cell. When the dimension values of a row are updated such that the row belongs to another logical cell after the update, the row is deleted from the block where it is currently stored and inserted into a block that belongs to its new logical cell.

In addition to the clustering improvements described above, MDC tables provide the following benefits:

- Probes and scans of block indexes are much faster due to their small size in relation to record- based indexes.
- Block indexes and the corresponding organization of data allows for fine-grained “partition elimination”, or selective table access.
- Queries that utilize the block indexes benefit from the reduced index size, optimized pre-fetching of blocks, and guaranteed clustering of the corresponding data.

- Reduced locking and predicate evaluation is possible for some queries.
- Block indexes have much less overhead associated with them for logging and maintenance because they only need to be updated when adding the first record to a block, or removing the last record from a block.
- Data rolled in can reuse the contiguous space left by data previously rolled out.

Note: An MDC table defined with even just a single dimension can benefit from these MDC attributes, and can be a viable alternative to a regular table with a clustering index.

Creating MDC tables

To create an MDC table, you specify one or more keys as the dimensions. Each of these MDC dimensions can consist of one or more columns similar to regular index keys. A dimension block index will be automatically created for each of the dimensions specified, and it will be used by the optimizer to quickly and efficiently access data along each dimension. A composite block index will also automatically be created, containing all columns across all dimensions, and will be used to maintain the clustering of data over insert and update activity. A composite block index will only be created if a single dimension does not already contain all the dimension key columns. The composite block index may also be selected by the optimizer to efficiently access data that satisfies values from a subset, or from all, of the column dimensions.

Example 10-2 shows DDL statements which create an MDC table with two dimensions on the columns REGION and YEAR and an additional un-clustered index on PRODUCT.

Example 10-2 Creating MDC table with an additional un-clustered index

```
CREATE TABLE SALES_TABLE
  (REGION  VARCHAR(5) NOT NULL,
   YEAR    CHAR(4) NOT NULL,
   PRODUCT VARCHAR(30) NOT NULL,
   ...
   QUANTITY INT NOT NULL,
   PRICE DECIMAL(17,2) NOT NULL,
   ...)
ORGANIZE BY DIMENSIONS (REGION, YEAR)
IN <TABLESPACE>;

CREATE INDEX PRODUCT_INDEX
ON TABLE SALES_TABLE (PRODUCT);
```

Two block indexes for the two dimensions REGION and YEAR, respectively, and a composite block index on columns REGION and YEAR are created automatically. MDC can be combined with standard record-based indexes. Therefore, it is valid to create index PRODUCT_INDEX on the MDC table. However, it is no longer necessary to create the index REGION_INDEX on column REGION and YEAR_INDEX on column YEAR from Example 10-1 because block indexes are created on these columns. MDC tables cannot have a clustering index.

MDC fast insert and fast delete

MDC supports the typical data warehousing operations of loading data from external source systems (roll-in) and deleting old data no longer needed for reporting (roll-out).

If a large amount of data is inserted into separate cells by different processes, locking can be reduced by locking MDC blocks instead of single rows. Block inserting behavior can be enabled and disabled for a table with the ALTER TABLE statement:

```
ALTER TABLE <table> LOCKSIZE BLOCKINSERT
```

This will elimiate the need for a row locking for <*table*>.

```
ALTER TABLE <table> LOCKSIZE ROW
```

This will reset locking to row level.

LOCKSIZE BLOCKINSERT should be used carefully if concurrent processes are inserting data simultaneously into the same cell. In this scenario, it is possible that some transactions will allocate new blocks to avoid lock contention. This may result in sparely populated blocks if there are not enough records to fill up the blocks.

MDC also provides a mechanism to quickly delete (roll-out) data along dimension boundaries. Instead of writing individual log entries for every deleted record, DB2 writes a single log entry for each data page containing delete records. The block ID is removed from the block indexes. Additionally, for each data page one log record has to be written. If there are record-based indexes defined on the table, the index entries have to be processed in the standard way. Blocks can be re-used as soon as the rollout is committed. To make best use of space and to minimize logging it is recommended to COMMIT between rollout and a new roll-in of data.

Considerations for the selection of MDC dimensions

When deciding the dimensions of an MDC table, besides query performance optimization, the expected cells density plays an important role. Depending on query performance to be achived, you can choose a set of dimensions that cause

the potential number of cells in the table to be very large due to the number of possible values for each of the dimensions. The number of possible cells in the table is equal to the Cartesian product of the cardinalities of each of the dimensions. For example, if you cluster a table on dimensions `DAY`, `REGION` and `PRODUCT` and the data covers 2 years, you might have `730 days * 12 regions * 100 products = 876 000` different possible cells in the table. Every cell that only contains a few records will still have an entire block of pages allocated to it. If the block size is large, this table could end up being much larger than it really needs to be.

Several design factors can contribute to optimal cell density:

- Varying the number of dimensions:

  Choosing only `Day` and `Region` as dimensions in the example above would reduce the number of different possible cells to 8760. You might have to define an additional record-based index on `PRODUCT` to support your queries in this case.

- Varying the granularity of one or more dimensions:

  Choosing `MonthAandYear` instead of `Day` as a dimension would reduce the number of different possible cells to 28800. If most of your queries restrict on month rather than day this might be a good choice. Note that you will have to add an additional, possibly generated, column to your table if it only contains `Day` so far.

- Varying the block (extent) size and page size of the table space:

  If you create the table in a table space with extent size 2 instead of 16 only two pages will be allocated for each block. This will make your blocks four times smaller so that there might be enough records in each cell to fill them.

10.3 Clustering support in SAP NetWeaver 2004s BI

Since release SAP Business Information Warehouse 3.0A, index clustering is supported for the fact tables of InfoCubes and Aggregates. In release SAP NetWeaver 2004s Business Intelligence (BI), MDC support is added for the following objects:

- Persistent Staging Area (PSA)
- Standard DataStore
- DataStore for Direct Update
- Write-optimized DataStore
- InfoCube
- Aggregate

Table 10-1summarizes the default MDC settings for SAP BI object tables.

Table 10-1 Default MDC settings for SAP BI object tables

Table	MDC Dimension(s)	Fixed MDC-Dimensions	Clustering Change-able	Locksize
PSA	Request ID	Yes	No	BLUCKINSERT
Standard DataStore - Activation Queue	Request SID	Yes	No	BLUCKINSERT
Standard DataStore – Change Log	Request ID	Yes	No	Row
Standard DataStore – Active	User-defined, only key fields	No	Yes	Row
DataStore for Direct Update – Active	User-defined, only key fields	No	Yes	Row
Write-optimized DataStore	Request ID	Yes	No	BLUCKINSERT
F Fact Table - InfoCube	Fix: Request Dimension ID field, User-defined: only dimension ID fields and one time characteristic (`OCALMONTH` or `OFISCPER`), same as E Fact Table	No	Yes	BLUCKINSERT
E Fact Table - InfoCube	User-defined: only dimension ID fields and one time characteristic (`OCALMONTH` or `OFISCPER`),same as F Fact Table	No	Yes	BLUCKINSERT

Table	MDC Dimension(s)	Fixed MDC-Dimensions	Clustering Change-able	Locksize
F Fact Table - Aggregate	Fix: Request Dimension ID field, Inherit MDC-Dimensions from InfoCube	Yes (Inheritance)	No (Inheritance)	BLUCKINSERT
E Fact Table - Aggregate	Inherit MDC-Dimensions from InfoCube	Yes (Inheritance)	No (Inheritance)	BLUCKINSERT

In the next sections, we describe how MDC is applied to each object.

10.3.1 Persistent Staging Area (PSA)

We have seen in “PSA Tables” on page 571 that all operations on the PSA are performed request-wise: data load, read and delete. Database performance can be improved by clustering the table by request. Therefore, a PSA table is created with MDC having the request ID field as the only MDC-dimension. This enforces that only records belonging to the same request are stored in the same MDC cell.

The risk of high space consumption is low because MDC is done on a single column and it is likely that a request has more records than can be stored in a single block or extent.

The lock size of a PSA table is set to `BLOCKINSERT` to enable MDC fast insert. This reduces locking during insert and speeds up loading data into PSA.

The clustering settings of a PSA table cannot be changed in the SAP BI Data Warehousing Workbench. If you want to disable MDC for PSA tables, set the new RSADMIN parameter `DB6_MDC_FOR_PSA` to `NO`. The default value for this parameter is `YES`. The RSADMIN parameter is described in SAP Note *832621*.

10.3.2 DataStore objects

There are three types of DataStore objects:

- Standard DataStore
- DataStore for Direct Update
- Write Optimized DataStore

Standard DataStore

To improve data load and activation performance the activation queue table is created with MDC having the request SID as the only MDC dimension. The change log table is created with MDC having the request ID as the only MDC dimension. For both tables this enforces that only records belonging to the same request are stored in the same MDC cell.

As for PSA, the risk of high space consumption is low because requests usually have more records than can be stored in a single block or extent.

The lock size of activation queue tables is set to `BLOCKINSERT` for accelerating data load. This is feasible because only one process writes into a single request. Therefore, the risk of several parallel processes allocating blocks in the same MDC cell and only filling them to a certain percentage does not exist. The lock size of change log tables is set to `ROW` because during activation several parallel processes can process a single request.

The clustering settings cannot be changed for these tables in the SAP BI Data Warehousing Workbench. If you want to disable MDC for activation queue and change log tables, set the new RSADMIN parameter `DB6_MDC_FOR_PSA` to `NO`. The RSADMIN parameter is described in SAP Note *832621*. The default value for this parameter is `YES`.

To improve query performance on the active table the user can create secondary indexes or define MDC on InfoObject columns when creating the DataStore object. MDC-dimensions can only be InfoObject fields that were selected as key fields. How to define MDC is described in 10.3, “Clustering support in SAP NetWeaver 2004s BI” on page 584. The lock size is always set to `ROW`, even if MDC is selected since multiple activation processes may write concurrently into the active table.

By default, no secondary indexes are created and no clustering is used for the active table. While additional indexes can be created at any time, even when the DataStore object already contains data, the MDC settings can only be changed as long as the DataStore object is empty. For defining MDC for a filled DataStore object or changing the MDC settings, re-clustering is available. Re-clustering creates a new active table with the MDC settings selected by the user and copies the data from the old active table to the new one. Re-clustering is described in detail in 10.4.2, “Re-clustering for DataStore objects” on page 604.

DataStore for direct update

MDC for the active table is defined in exactly the same way as for the active table of standard DataStore objects. MDC dimensions on the active table are user defined when the DataStore object is created. The active table has lock size set to `ROW` because several parallel processes can process one request.

Write Optimized DataStore

Write-optimized DataStore objects are specifically designed for fast data load. Reporting only plays a minor role. They only consist of the active table that contains the additional three columns *request ID*, *data package,* and *record number*. For accelerating data load active tables of write-optimized DataStore objects are created with the request ID as the only MDC column.

The lock size is set to `BLOCKINSERT`.

10.3.3 InfoCube fact tables

The F fact table has dimension ID fields for all dimensions and fields for all key figures. A primary key is not created for this table. Since the release of SAP NetWeaver 2004s BI, a compound non-unique index on all dimension ID fields, the P-index, is not created anymore. Secondary indexes are created on each dimension ID field.

Line Item Dimensions have a field with SID-values for the Line Item characteristic in the F fact table instead of a dimension ID field.

The table can reside in a partitioned table space and the partitioning key contains all dimension ID fields except for the request dimension ID field.

Data is loaded and deleted request-wise into the F fact table. To improve query performance, *index clustering* is used by default for the F fact table. The clustering Index is the secondary index on the time dimension ID field. This is to improve performance of queries restricted on the time dimension, which is in general the case for all queries.

If the InfoCube does not contain any time characteristics, the clustering index is the secondary index on the request dimension ID field. In this rare case, Index clustering improves only the delete performance while it has no influence on the query performance.

The clustering can be manually changed to MDC for an InfoCube. For more details on how to define MDC-dimensions for an InfoCube, see 10.3.6, “Defining MDC for DataStore objects” on page 596. The clustering settings are set for an InfoCube and take affect on both fact tables in the same manner. MDC-dimensions can be the time dimension ID field and all dimension ID fields of the user-defined dimensions. The request dimension ID field cannot be selected and is by default always the first MDC-dimension on the F fact table. This ensures faster load and delete performance for this table.

In place of the time dimension ID field the time characteristic `Calendar Month` (`0CALMONTH`) or `Fiscal Period` (`0FISCPER`) can be selected as a MDC-dimension.

In this case, an additional field is added to the table that contains the SID values for this characteristic and the MDC-dimension is the SID-field.

If a dimension is selected as a MDC-dimension the secondary index on the dimension ID field is not created. Instead, a database-internal block index is created that does not appear in the ABAP dictionary.

If MDC is selected, the lock size of the table is `BLOCKINSERT`.

The E fact table has exactly the same fields as the F fact table. It also does not have a primary key but a unique, compound index is created that contains all dimension ID fields. the compound index is called p-index and is the clustering index in case of index clustering. The partitioning key is the same as for the F fact table.

Data is inserted into the E fact table only by compression. The relation to a request is lost and only selective deletion can be done.

MDC is applied to the E fact table in the same way as to the F fact table except for the request dimension ID field, which is not added to the MDC-dimensions.

10.3.4 Aggregates

An *aggregate* is a copy of its Base-InfoCube with the reduction of one or several characteristics, dimensions or key figures. The key figures are aggregated for the remaining dimensions.

If an aggregate contains an unchanged dimension of the Base-InfoCube with the same characteristics, the dimension table of the Base-InfoCube is reused by the aggregate rather than a new dimension table is created. The fact tables contain the same dimension IDs for this dimension as the fact tables of the Base-InfoCube.

If an aggregate contains a changed dimension of the Base-InfoCube that has fewer characteristics, a new dimension table is created.

The fact tables contain the dimension ID fields of the aggregate dimensions and the fields of the aggregate key figures. The layout of indexes and partitioning keys are the same as for the fact tables of an InfoCube.

The clustering of the aggregate's fact tables is inherited from the clustering of the Base-InfoCube. If Index Clustering is used the Clustering Index for the F fact table is the secondary index on the time dimension ID field or if not present the secondary index on the request dimension ID field. The Clustering Index for the E fact table is the P-index.

If MDC is defined on the Base-InfoCube, a dimension ID field in the aggregate's fact table will be a MDC-dimension if the dimension is unchanged and points to a dimension of the Base-InfoCube that has been defined as a MDC-dimension. If MDC is defined on a time characteristic, the aggregate's fact tables will only be clustered on the time characteristic if the aggregate contains the time characteristic. If the aggregate does not contain any unchanged MDC-dimensions Index Clustering is used instead.If you do not want aggregates to inherit the MDC settings of the InfoCube you set the RSADMIN parameter `DB6_MDC_FOR_AGGREGATES` to `NO`. SAP Note *832621* describes the parameter.

10.3.5 Defining clustering for InfoCubes

To define index clustering for an InfoCube and its aggregates, proceed as follows:

1. In the Data Warehousing Workbench, when creating or changing the InfoCube, select **DB Performance** → **Clustering** from the *Extras* menu (Figure 10-7).

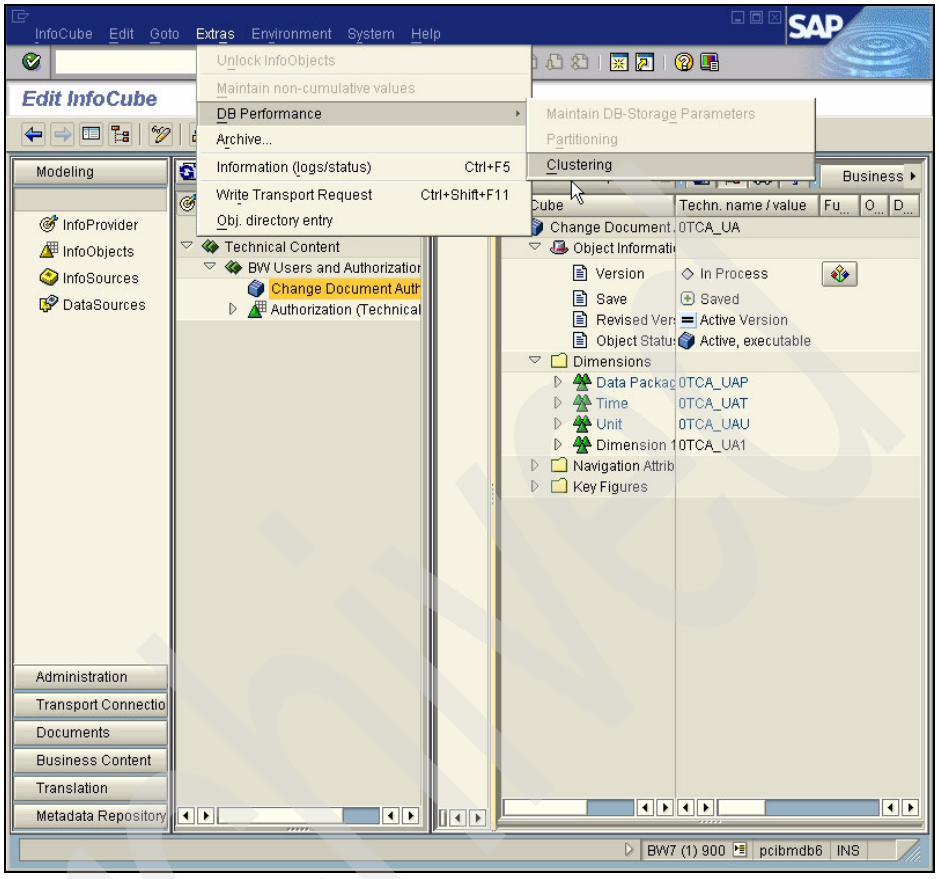


Figure 10-7 Defining clustering for InfoCubes

2. In *Selection of Clustering* dialog (Figure 10-8), select **Index Clustering** and save.

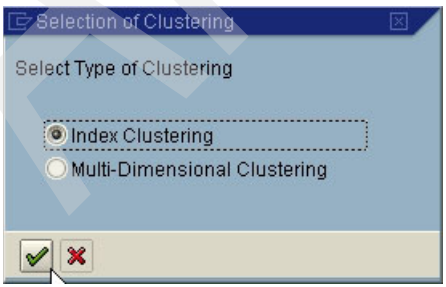


Figure 10-8 Clustering selection - index clustering

The fact tables of the InfoCube will be created with a clustering index. MDC will not be used. On the E fact table, the clustering index is the combined index on the dimension key columns. On the F fact table, it is the index on the time dimension key column if the InfoCube has a time dimension and the index on the package dimension key column otherwise.

To define MDC for an InfoCube and its aggregates, proceed as follows:

1. In the Data Warehousing Workbench, when creating or changing the InfoCube, **DB Performance** → **Clustering** from the *Extras* menu (Figure 10-7 on page 591).
2. Select **Multi-dimensional clustering** and save (Figure 10-9).

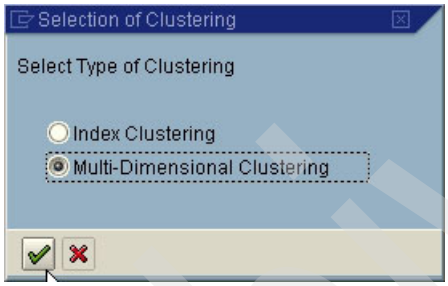


Figure 10-9 Clustering selection - multi-dimensional clustering

3. Selecting the MDC dimensions in the Multi-dimensional Clustering dialog:
   - In the *Time Dimension,* from the drop-down box of *Selected Column* field (select the MDC dimensions), you can select a field of the time dimension as MDC dimension. If available in the InfoCube, you can choose the key field of the time dimension, the additional SID field of the time characteristic calendar month (`0CALMONTH`), the additional SID field of the time characteristic financial year/period (`0FISCPER`), or no field, if you don't want to select the time dimension as MDC dimension.

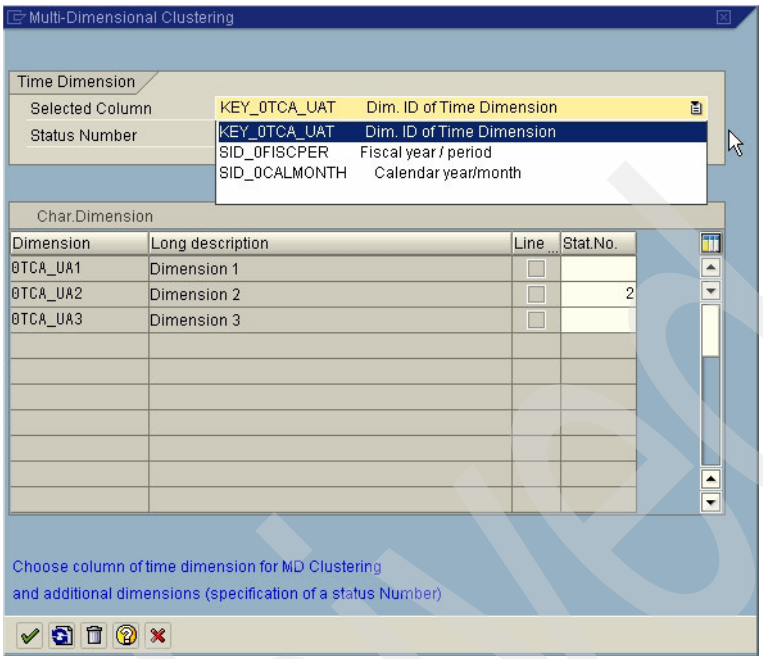


Figure 10-10 MDC InfoCubes - choosing the time dimension

– The *Status Number* field of the *Time Dimension* receives automatically the ordinal number *1* (Figure 10-11). The ordinal number indicates whether a field is selected as MDC dimension, and determines the order of the MDC dimensions in the compound block index.

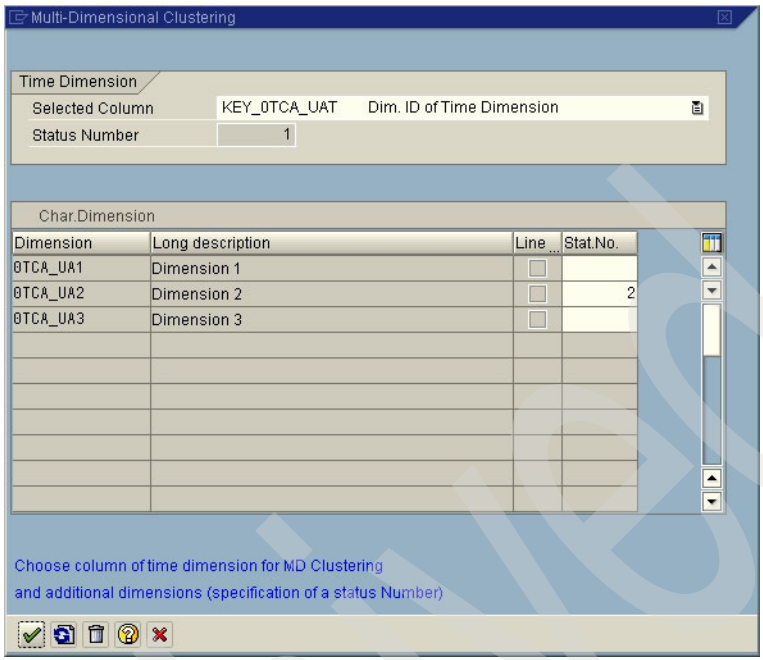


Figure 10-11 MDC for InfoCubes - Time Dimension selected

4. In the C*haracteristic Dimensions*, you can select further MDC dimensions by entering consecutive numbers in the *OrdNum* fields of the dimensions you want to select for MDC. As for the time dimension, the ordinal number determines the order of the MDC dimensions in the compound block index.

Figure 10-10 on page 593 shows the choices you have for the time dimension if it contains calendar month (`0CALMONTH`) and fiscal period (`0FISCPER`). The default for MDC is the dimension ID of the time dimension. If you select either calendar month or fiscal period, an additional column for the SID of the time characteristic will be inserted into the fact tables of the InfoCube.

Considerations when using MDC for InfoCubes

Consider these things when selecting MDC dimensions for DataStore objects:

- Select dimensions for which restrictions in your queries occur frequently.
- Select dimensions with a rather low cardinality. The columns available for MDC dimension are the dimension keys (DIMIDs) of the InfoCube. The cardinality is equal to the number of different combinations of the characteristics of the dimension. Therefore, you should select dimensions with only one or a small number of characteristics and with only a small number of characteristics values.

- If you want to create an MDC dimension for a specific characteristic with low cardinality, you can define this characteristic as a line item dimension in the InfoCube. This is opposed to the general rule that line item dimensions are characteristics with a very high cardinality. For MDC the advantage is that for line item dimensions the fact table contains the SID of the characteristic instead of the DIMID so that database queries can be directly restricted on the SID column.
- You cannot select more than three dimensions.
- You cannot select all dimensions of an InfoCube.
- When specifying the order numbers of the MDC dimensions sort the dimensions according to their selectivity and according to the frequency with which they occur in queries. That is, specify lower order numbers for dimensions that occur more often and select lower order numbers for dimensions with a larger cardinality.

You should always take into account the space consumption for the MDC dimensions you select. For each combination of values in the MDC dimensions, at least one block is reserved. If there are only a few rows for each value combinations, only a small portion of the block might actually be filled with data while the rest remains empty. The number of data page storage space occupied by the table would be much larger than required. This not only wastes disk space but It also has a negative impact on query performance because many partially filled data pages have to be read instead of a few filled ones.

The block size is determined by the page size and the extent size of the table space. For fact data the default page size is 16 KB. The default extent size in SAP NetWeaver 2004s BI is 2, in former releases it was 16. With a page size of 16 KB and an extent size of 2, 32 KB would be reserved for each block. The number of bytes for each record in a fact table depends on the number of dimensions and the number and data type of the key figures.

For example: If the InfoCube has the three standard dimensions, 5 customer defined dimensions, and 20 key figures of data type decimal, each record needs `3 * 4 + 5 * 4 + 20 * 9 bytes = 212` bytes. About 154 records would fit into one block. If you have selected the time characteristic `0CALMONTH`, the unit dimension, and one of the customer defined dimensions for MDC there should be at least 150 records per request, calendar month, unit and dimension id of the customer dimension to optimally use the allocated space in the F fact table. For the E fact table this is per calendar month, unit and dimension id of the customer dimension.

10.3.6 Defining MDC for DataStore objects

When creating the DataStore object, to define MDC dimensions for the active table of DataStore objects, select **DB Performance** → **Clustering** from the *Extras* menu. The dialog shown in Figure 10-12 is displayed. You can only change the MDC settings if the DataStore object does not yet contain any data. Otherwise, the current MDC settings are displayed in read-only mode in the dialog.

You can select MDC dimensions from the InfoObjects belonging to the logical key of the DataStore object. Enter consecutive order numbers starting with 1 for the InfoObjects you select as MDC dimensions. The order numbers indicate that the InfoObjects have been selected as MDC dimensions and they determine the order of the MDC dimensions in the composite block index.

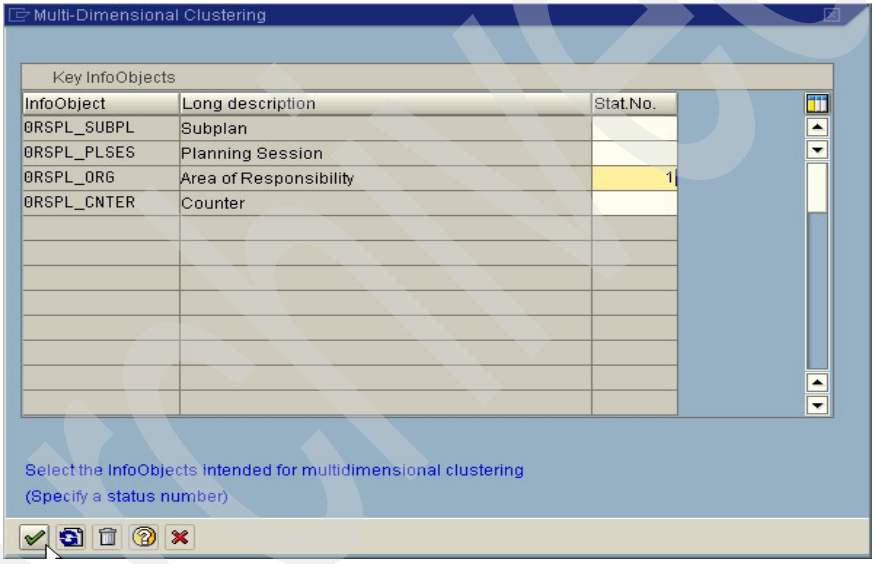


Figure 10-12 Multi-dimensional Clustering for DataStore objects

Considerations when using MDC for DataStore objects

You should consider the following when selecting MDC dimensions for InfoCubes:

- Select InfoObjects for which restrictions in your queries occur frequently, for example a time characteristic.
- Select InfoObjects with a rather low cardinality, for example select `0CALMONTH` instead of `0CALDAY`.

- You cannot select more than three InfoObjects as MDC dimensions, to reduce the risk of high space consumption
- You cannot select all InfoObjects as MDC dimensions
- When specifying the order numbers of the MDC dimensions sort the dimensions according to the frequency with which they occur in queries. Specify lower order numbers for dimensions that occur more often. You should sort dimensions according to their selectivity, that is, select lower order numbers for dimensions with a larger cardinality.

As for InfoCubes, you should always take into account the space consumption for the MDC dimensions you select.

10.4 Re-clustering InfoCubes and DataStore objects

MDC and index clustering settings of InfoCubes and DataStore objects can only be changed for new, empty objects. Re-clustering allows changing clustering settings for InfoCubes and DataStore objects that already contain data. If you upgrade from an older SAP Business Information Warehouse release to SAP NetWeaver 2004s BI and want to use MDC for some of your existing InfoCubes and DataStore objects, you execute re-clustering for these objects. Re-clustering involves the creation of shadow tables for the fact tables of InfoCubes and the active table of DataStore objects with the new clustering settings and copying the data from the old tables to the new ones.

You can call re-clustering for an InfoCube or a DataStore object from its context menu in the Data Warehousing Workbench (Figure 10-13) or by selecting **Reclustering** from the *Administration* screen of the Data Warehousing Workbench (Figure 10-14).

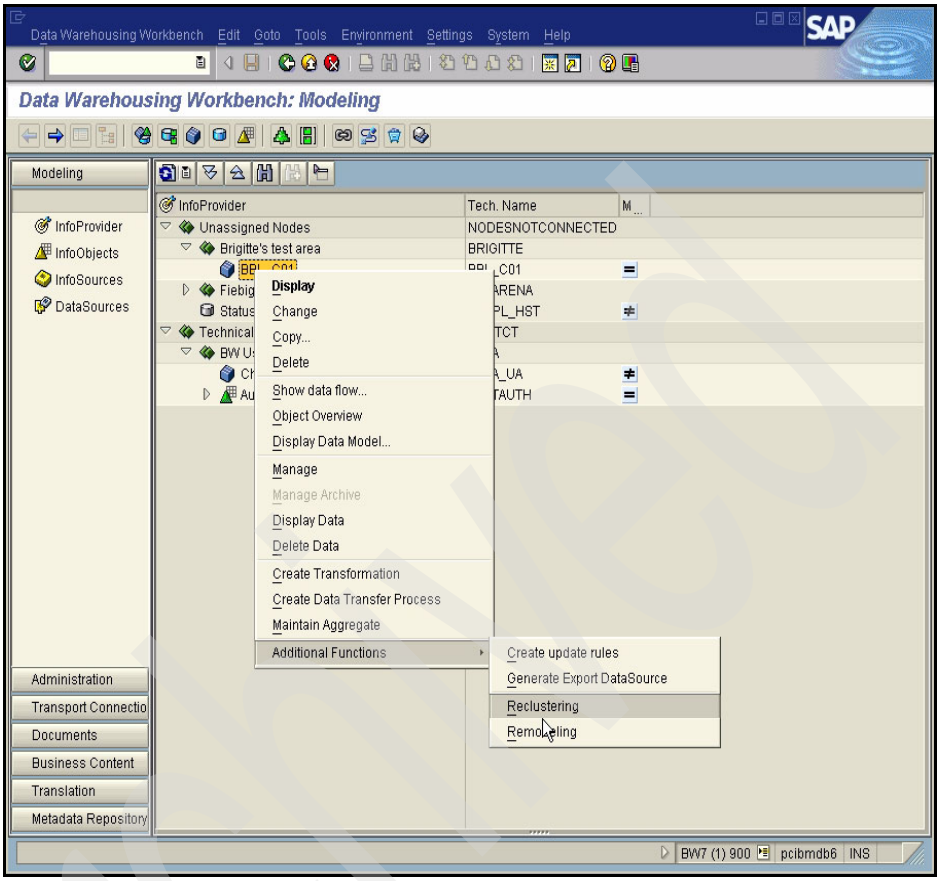


Figure 10-13 Invoking re-clustering from the context menu of an InfoCube

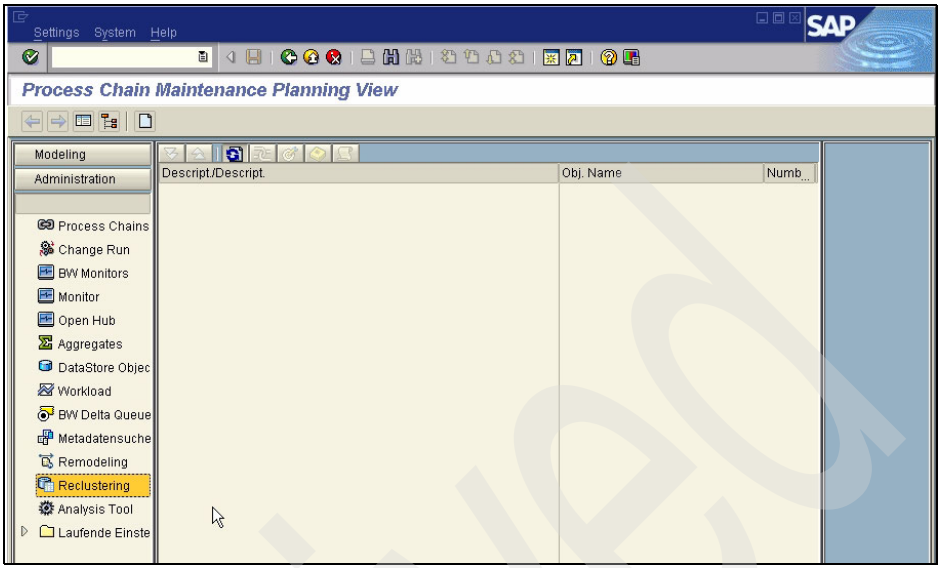


Figure 10-14 Invoking re-clustering from the Administration screen of the Data Warehousing Workbench

Figure 10-15 shows the re-clustering dialog, when invoked from the context menu of InfoCube `BBL_C01`.

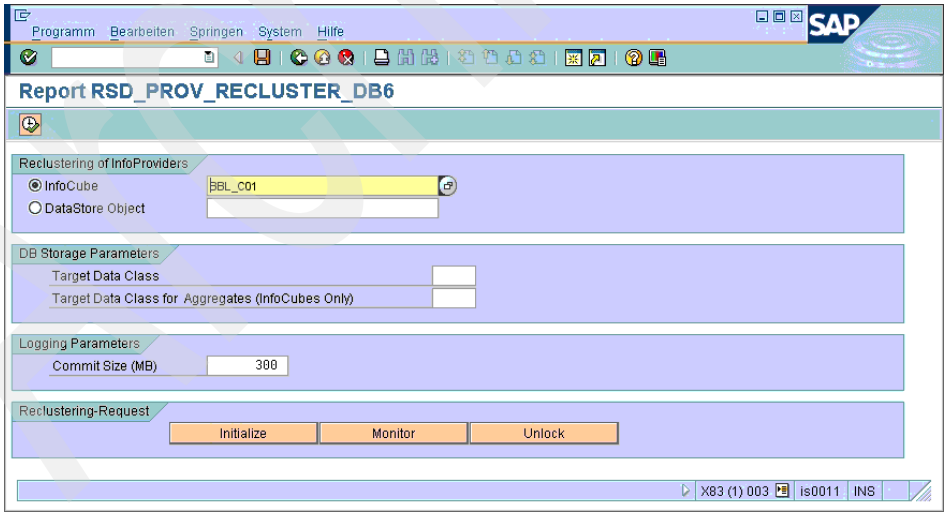


Figure 10-15 re-clustering Dialog

When you call re-clustering from the context menu of an InfoCube, the radio button *InfoCube* is selected and the object's name is displayed in the entry field for InfoCubes.

In the section *DB Storage Parameters,* you can specify a new data class for the InfoCube and a second one for its aggregates. For example, if the fact tables reside in a table space with extent size 16, you decide to use MDC for them and you want to control the MDC block size, you can create new table spaces with a smaller extent size, define new data classes for them and enter them here as target data classes for the InfoCube and aggregate fact tables.

In the section *Logging Parameters* you enter the maximum size of data packages to be copied in one transaction from the source fact tables to the shadow tables. This restricts the amount of log space that is needed for copying the data.

When you call re-clustering from the context menu of a DataStore object, the radio button *DataStore Object* is selected and the object's name is displayed in the entry field for DataStore objects.

In the section *DB Storage Parameters,* you can specify a new data class for the DataStore object. For example, if the active table resides in a table space with extent size 16, you decide to use MDC for them and you want to control the MDC block size, you can create a new table space with a smaller extent size, define new data classes for it and enter it here as target data class for the active table of the DataStore object.

In the section *Logging Parameters*, you enter the maximum size of data packages to be copied in one transaction from the source active table to the shadow table. This restricts the amount of log space that is needed for copying the data.

The action *Initialize* creates a new re-clustering request.

The action *Monitor* switches to the monitor that displays the progress of the re-clustering request. The action *Unlock* removes an existing lock from the InfoCube or DataStore object. You can use it when reclustering has failed and the lock on the InfoCube or DataStore object has not been removed.

Before re-clustering an InfoCube or DataStore object, you should make sure that you have a backup available. This is to make sure you can go back to the current state of the InfoCube or DataStore object if something goes wrong during re-clustering. When you select **Initialize**, the system will first prompt whether you have performed a database backup (Figure 10-16).

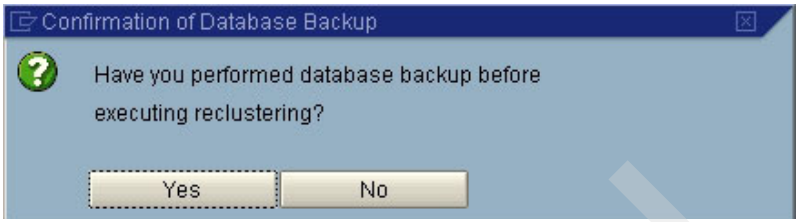


Figure 10-16 Confirmation of availability of a database backup before re-clustering

10.4.1 Re-clustering for InfoCubes

Re-clustering for InfoCubes creates shadow tables with the new clustering settings for the fact tables, copies the data from the original fact tables to the shadow tables, and exchanges the shadow tables with the original fact tables. The aggregates of the InfoCube are deactivated and recreated with their clustering settings adapted to the new ones of the InfoCube. If you specify new data classes for the InfoCube fact tables and/or the aggregate fact tables, the shadow tables are created in the table spaces associated with the data classes. The SAP BI metadata of the InfoCube and its aggregates is changed to reflect the new clustering settings and data classes.

When initializing a new re-clustering request for an InfoCube, the clustering selection screen is displayed (Figure 10-17).

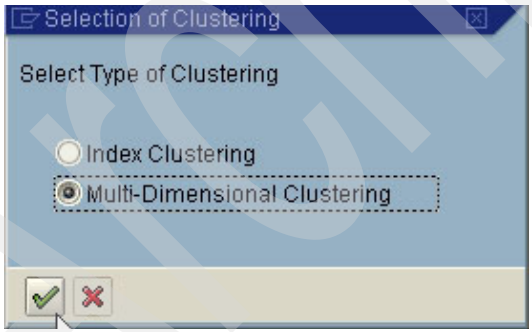


Figure 10-17 Clustering selection

If you select *Index Clustering,* the shadow fact tables will be created with a clustering index. If you select *Multi-dimensional Clustering*, the *Multi-Dimensional Clustering* dialog for InfoCubes is displayed (Figure 10-18) and you can select the MDC dimensions as described in 10.3.5, “Defining clustering for InfoCubes” on page 590.

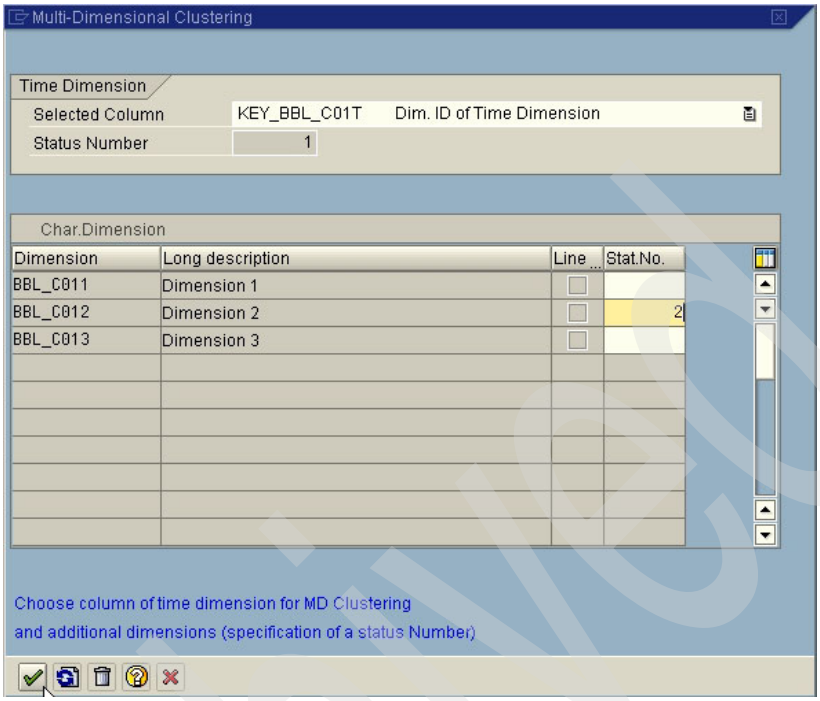


Figure 10-18 Multi-Dimensional Clustering dialog for re-clustering InfoCubes

After you have defined the new clustering settings for the InfoCube, the system informs you that a new re-clustering request has been created (Figure 10-19).

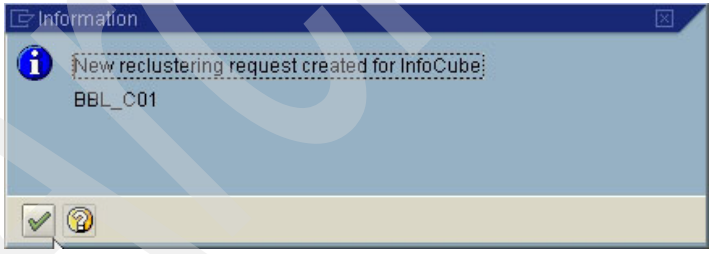


Figure 10-19 Confirmation of re-clustering request creation

It then displays the job scheduling screen on which you can schedule the re-clustering job as a batch job (Figure 10-20).

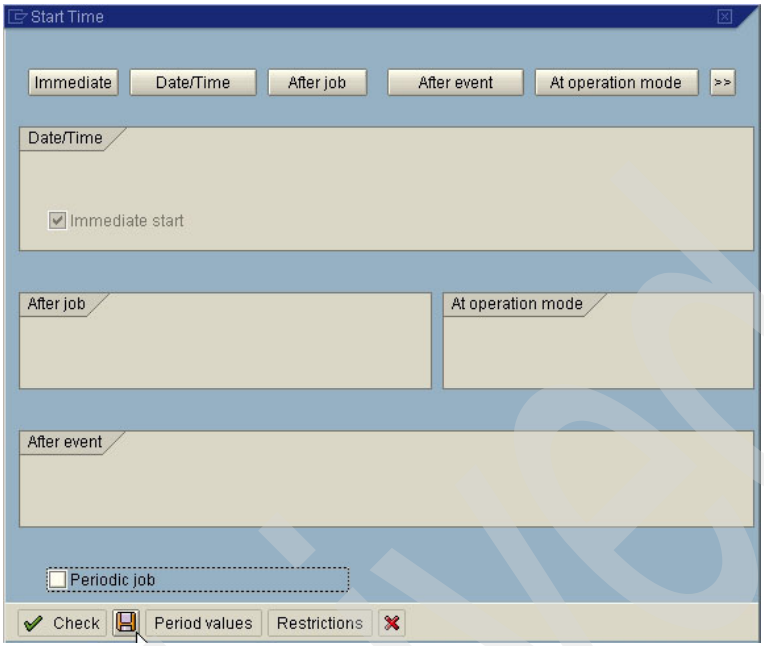


Figure 10-20 Scheduling Batch Job

The system then informs you that the new re-clustering request has been scheduled or already started as batch job (Figure 10-21).

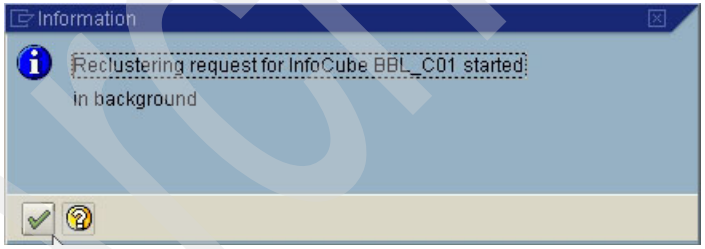


Figure 10-21 Message that re-clustering job has been started

The re-clustering job executes the following steps:

- Creation of shadow tables for the E and the F fact table of the InfoCube.
- Data copy from the original tables to the shadow tables.
- Creation of the required secondary indexes on the shadow tables.
- Setting of a read lock on the InfoCube to block read operations while old and new fact tables are being exchanged.

- Deactivation of the aggregates.
- Dropping of the UNION ALL view on the old E and F Fact table.
- Consistency check of the copied data.
- Exchange of the shadow tables and the old fact tables. The old fact tables are saved under the name of SAP BI temporary tables. The shadow tables are renamed to the names of the old fact tables.
- Recreation of the UNION ALL view on the new fact tables.
- Adaptation of the SAP BI metadata of the InfoCube and reactivation of the transfer structures.
- Statistics collection on the new fact tables.
- Removal of the read lock.
- Reactivation of the aggregates (this will adapt their layout to the new clustering settings of the InfoCube).
- Cleanup.

Re-clustering can be applied only to InfoCubes, not to aggregates. Clustering settings of aggregates are automatically adapted to the clustering settings of the InfoCube during re-clustering of the InfoCube.

10.4.2 Re-clustering for DataStore objects

Re-clustering for DataStore objects creates a shadow table with the new clustering settings for the active table, copies the data from the original active table to the shadow table, and exchanges the shadow table with the original active table. If you specify a new data class for the DataStore active table, the shadow table is created in the table space associated with the data class. The SAP BI metadata of the DataStore object is changed to reflect the new clustering settings and data class.

When initializing a new re-clustering request of a DataStore object the *Multi-dimensional Clustering* screen for DataStore objects is displayed (Figure 10-22).

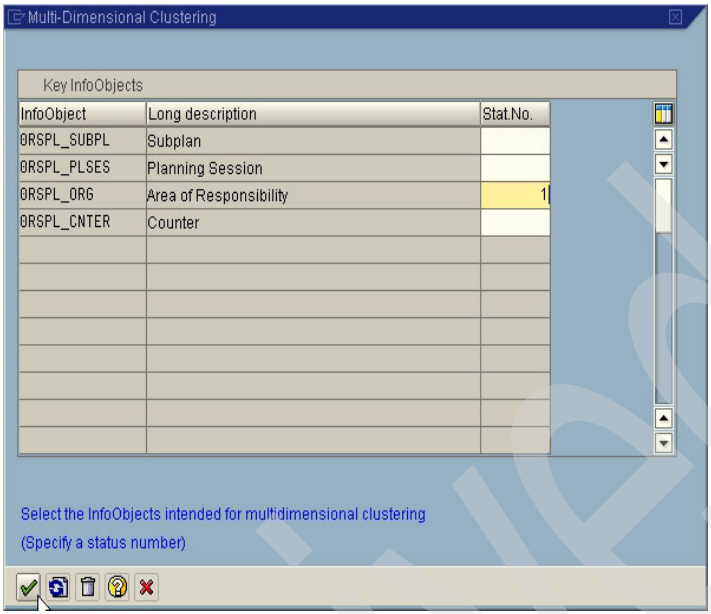


Figure 10-22 Multi-dimensional Clustering for re-cluster DataStore objects

After you have defined the new clustering settings for the DataStore object, you schedule the re-clustering job as batch job on the job schedule screen, in the same way as for InfoCubes.

The re-clustering job executes the following steps:

- Creation of the shadow table for the active table.
- Data copy from the original active table to the shadow table.
- If the active table has secondary indexes, creation of the secondary indexes on the shadow table.
- Consistency check of the copied data.
- Exchange of the shadow table and the original active table. The old active table is saved under the name of an SAP BI temporary table. The shadow table is renamed to the names of the old active table.
- Adaptation of the SAP BI metadata of the DataStore object.
- Statistics collection on the new active table.
- Cleanup

Re-clustering can be applied only to standard DataStore objects and DataStore objects for direct update. MDC settings for write-optimized DataStore objects cannot be changed.

10.4.3 Monitoring re-clustering requests

Re-clustering requests are monitored with the new SAP BI monitor. You call the monitor by selecting **Monitor** from the *Re-clustering* screen. Figure 10-18 shows the *Monitor Request* screen. The monitor displays the progress of SAP BI batch jobs, including re-clustering. Steps already completed are displayed with a green icon. The step that is currently processed is displayed with a yellow icon. When an error occurs, the step causing the error and the re-clustering request itself are displayed with a red icon.

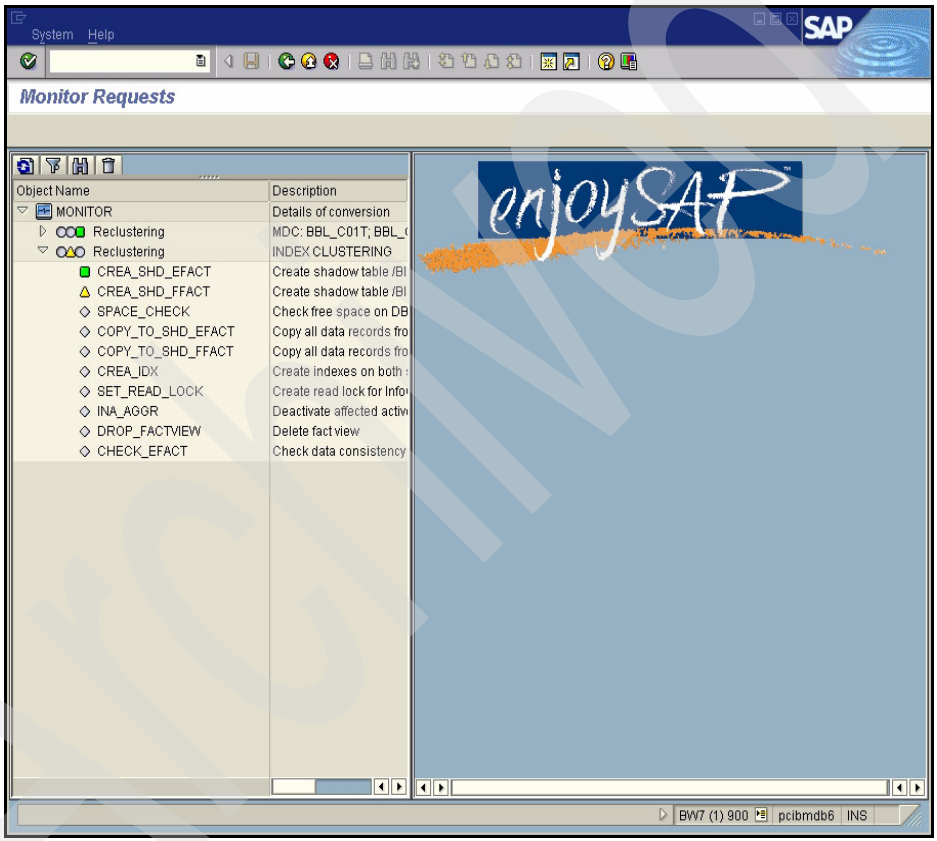


Figure 10-23 Monitoring re-clustering

You can double-click a step in the left frame to display its details in the right frame (Figure 10-24).

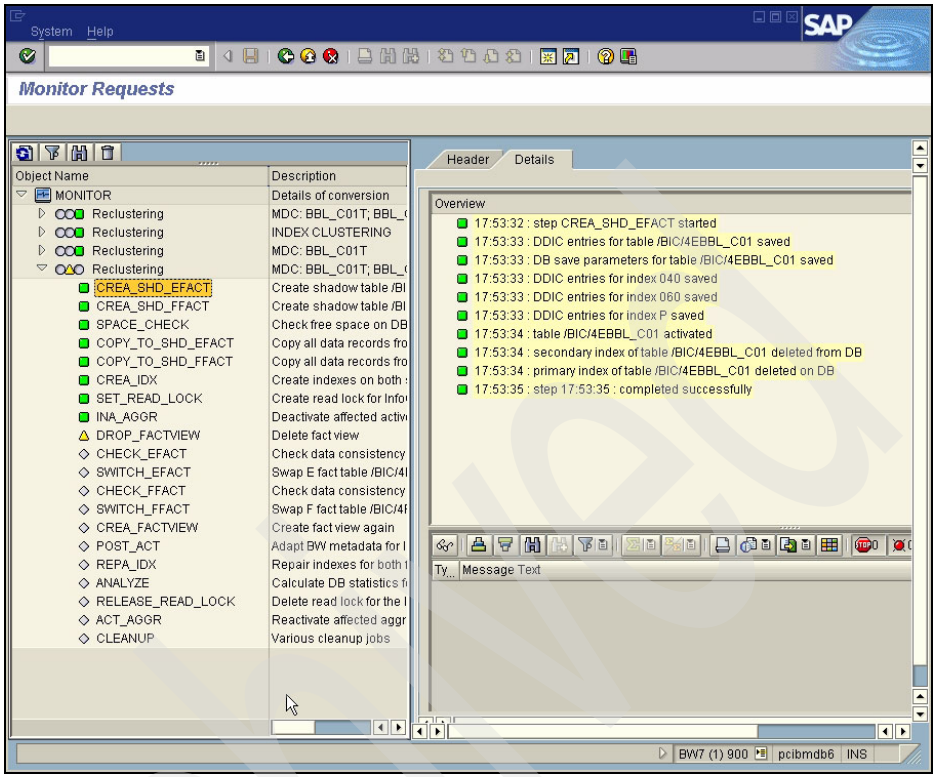


Figure 10-24 Displaying details of a re-clustering step

From the context menu of a re-clustering request (Figure 10-25), you can select the following actions:

- *Delete Request*
  Delete the re-clustering request. Deleted requests are removed from the monitor. All tables remain in the current state. This works for aborted and completed requests. If the request aborted, the InfoCube or DataStore object can be inconsistent.
- *Reset Request*
  Reset the re-clustering request. This works for aborted requests. Locks on the InfoCube or DataStore object are removed and the shadow tables are deleted.
- *Restart Request*
  Restart the re-clustering in batch. This works for aborted requests.

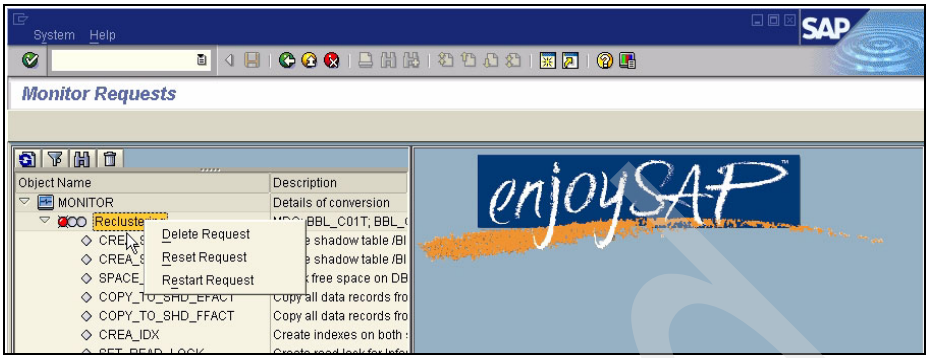


Figure 10-25 Context menu of re-clustering requests

When a processing step aborts, you can restart it from its context menu.

10.4.4 Data copy

By default, data is copied with maximally 6 parallel processes. The default value for the number of parallel processes can be reduced by setting RSADMIN parameter `RSDU_REPART_PARALLEL_DEGREE` to a lower value.

10.5 InfoCube compression

There are basically two application areas for InfoCube compression. First, the elimination of duplicated rows and second, the update of reference points in an InfoCube containing non-cumulative key figures.

A single row represented by a combination of dimension IDs in all dimension fields (except for the request dimension) may appear multiple times within a single request or across multiple requests with different values in the key figures. Note, that the F fact table has no primary index nor an unique index. During reporting, the key figures of all duplicated rows are accumulated so that a single row is returned for each dimension ID combination.

An example scenario is an InfoCube with multiple data sources delivering same or intersecting sets of data. In such a scenario, InfoCube compression can reduce the required storage space by transferring an accumulated, unique row to the E fact table. Note, that the E fact table has an unique, compound index on all dimension ID fields. During compression, a row is inserted if the E fact table does not yet contain a row with the same dimension ID combination. If the E fact table already contains a row with the same dimension ID combination, the row is updated by accumulating the key figures to a single result.

A non-cumulative key figure represents a stock value, for example a banking account balance, a warehouse stock, or an inventory. For a fast access to the current stock value, an InfoCube stores for each stock item (same dimension ID combination) a reference point in the E fact table containing the latest stock value. Compression is the only method to update reference points and should therefore be done regularly for InfoCubes with non-cumulative key figures. During compression, transaction data is transferred to the E fact table as described above and reference points are updated in the same manner by inserting and updating unique rows.

Compression jobs are scheduled on the tab *Collapse* (Figure 10-26) of the *Manage* screen of an InfoCube (see Figure 10-13 on page 598). Here you specify the highest request ID for a compression job. All uncompressed requests having a request ID lower or equal to the specified ID will be compressed in this job. If "*With Zero Elimination* is selected, compression avoids rows having all key figures equal to zero in the E fact table. An InfoCube is locked and cannot be read during compression.

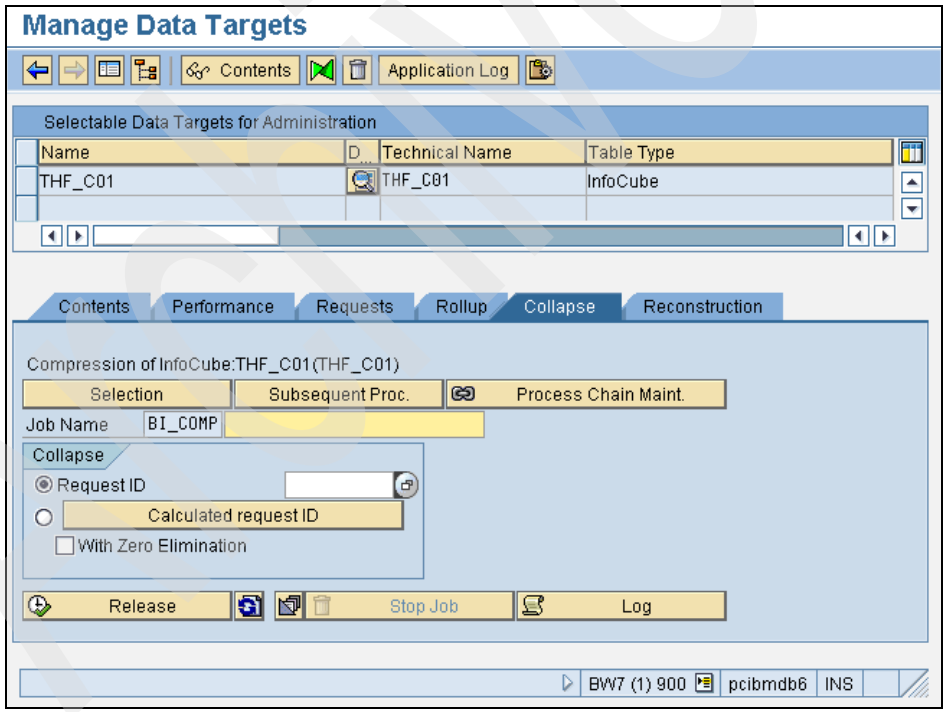


Figure 10-26 Scheduling compression jobs

The compression implementation has been completely revised in the release SAP NetWeaver 2004s BI. The new implementation takes advantage of DB2's SQL MERGE statement. DB2 UDB Version 8.2.2 contains improvements to the SQL MERGE statement that are dedicated to the InfoCube compression and that accelerate execution on partitioned tables and when using intra-parallelism.

The new implementation uses the SQL MERGE statement to insert and update all unique rows into the E fact table in a single statement execution. This has several advantages over the previous implementation with a searched SQL INSERT statement and a searched SQL UPDATE statement:

- No compound index on the F fact table is required

  The searched INSERT/UPDATE approach required a non-unique, compound index on all dimension fields on the F fact table for a faster execution. Therefore, prior release SAP NetWeaver 2004s BI, this index has always been created and is called the P-index. The SQL MERGE statement achieves best compression performance without using the P-index on the F fact table. To save index space and to reduce index maintenance overhead during data load and data deletion the P-index on the F fact table is not created anymore in release SAP NetWeaver 2004s SAP BI. SAP Note *832274* describes how to remove P-indexes created in earlier releases.

- Zero elimination is done during data transfer

  Without the SQL MERGE statement, zero elimination is done after transferring the data from the F fact table to the E fact table by executing a SQL DELETE statement restricted on all rows having all key figures equal to zero. This SQL DELETE statement always requires a full table scan on the E fact table. With the SQL MERGE statement, it is possible to only transfer those rows from the F fact table to the E fact table that are not resulting into a row having all key figures equal to zero. This makes the SQL DELETE statement with the full table scan unnecessary and depending on the size of the E fact table improves overall execution time of the compression job.

- Overall performance improves on average by 20%

  Internal tests show an average improvement by 20% in overall execution time of InfoCube compression using the SQL MERGE statement in comparison to the compression implementation of release SAP Business Information Warehouse 3.5 with the searched SQL INSERT/UPDATE statements.

The main features of the searched SQL INSERT/UPDATE statements -- all processing is done on the database server with no data transfer to the application server and use of collocated joins in partitioned fact tables -- are preserved by the SQL MERGE statement.

Example 10-3 shows the SQL MERGE statement which transfers a request from the F fact table /BIC/FCUBE to the E fact table /BIC/ECUBE. The InfoCube CUBE contains two user-defined dimensions (KEY_CUBE1, KEY_CUBE2) and two cumulative key figures (/BIC/KYFA,/BIC/KYFB). The USING-clause defines a view on the unique, accumulated rows of one request. Note that for all rows (and for all requests), the dimension ID in the request dimension KEY_CUBEP will be 0, that means the relation to the request is lost.

Example 10-3 Transferring request using SQL MERGE statement

```
MERGE INTO "/BIC/ECUBE" AS E
USING (
   SELECT 0 AS "KEY_CUBEP",
          F."KEY_CUBET" AS "KEY_CUBET",
          F."KEY_CUBEU" AS "KEY_CUBEU",
          F."KEY_CUBE1" AS "KEY_CUBE1",
          F."KEY_CUBE2" AS "KEY_CUBE2",
          SUM( F."/BIC/KYFA" ) AS "/BIC/KYFA",
          SUM( F."/BIC/KYFB" ) AS "/BIC/KYFB"
   FROM   "/BIC/FCUBE" AS F
   WHERE  F."KEY_CUBEP" = ?
   GROUP BY F."KEY_CUBET", F."KEY_CUBEU", F."KEY_CUBE1", F."KEY_CUBE2"
) AS AV
ON E."KEY_CUBEP" = AV."KEY_CUBEP" AND
   E."KEY_CUBET" = AV."KEY_CUBET" AND
   E."KEY_CUBEU" = AV."KEY_CUBEU" AND
   E."KEY_CUBE1" = AV."KEY_CUBE1" AND
   E."KEY_CUBE2" = AV."KEY_CUBE2"
WHEN NOT MATCHED THEN
   INSERT ( E."KEY_CUBEP", E."KEY_CUBET", E."KEY_CUBEU",
            E."KEY_CUBE1", E."KEY_CUBE2",
            E."/BIC/KYFA", E."/BIC/KYFB" )
   VALUES ( AV."KEY_CUBEP", AV."KEY_CUBET", AV."KEY_CUBEU",
            AV."KEY_CUBE1", AV."KEY_CUBE2",
            AV."/BIC/KYFA", AV."/BIC/KYFB" )
WHEN MATCHED THEN
   UPDATE SET ( E."/BIC/KYFA", E."/BIC/KYFB" ) =
   ( E."/BIC/KYFA" + AV."/BIC/KYFA", E."/BIC/KYFB" + AV."/BIC/KYFB" )
```

The SQL MERGE Statement in Example 10-4 is an example of zero elimination. The USING- and ON-clause is the same as in the Example 10-3. Note that a row is deleted from the E fact table if an update would result in a row having all key figures equal to zero.

Example 10-4 Zero elimination

```
MERGE INTO "/BIC/ECUBE" AS E
USING (
  ...
) AS AV
ON ...
WHEN NOT MATCHED AND ( AV."/BIC/KYFA" <> 0 OR
                       AV."/BIC/KYFB" <> 0 ) THEN
   INSERT ( E."KEY_CUBEP", E."KEY_CUBET", E."KEY_CUBEU",
            E."KEY_CUBE1", E."KEY_CUBE2",
            E."/BIC/KYFA", E."/BIC/KYFB" )
   VALUES ( AV."KEY_CUBEP", AV."KEY_CUBET", AV."KEY_CUBEU",
            AV."KEY_CUBE1", AV."KEY_CUBE2",
            AV."/BIC/KYFA", AV."/BIC/KYFB" )
WHEN MATCHED AND ( E."/BIC/KYFA" + AV."/BIC/KYFA" = 0 AND
                   E."/BIC/KYFB" + AV."/BIC/KYFB" = 0 ) THEN
   DELETE
WHEN MATCHED AND ( E."/BIC/KYFA" + AV."/BIC/KYFA" <> 0 OR
                   E."/BIC/KYFB" + AV."/BIC/KYFB" <> 0 ) THEN
   UPDATE SET ( E."/BIC/KYFA", E."/BIC/KYFB" ) =
   ( E."/BIC/KYFA" + AV."/BIC/KYFA", E."/BIC/KYFB" + AV."/BIC/KYFB" )
ELSE IGNORE
```

You may refrain from compressing an InfoCube if none of the described areas of application applies. For example, an InfoCube in a retail scenario that stores unique receipt numbers usually has neither any duplicated rows nor non-cumulative key figures. Compressing such an InfoCube has no gain in required storage space nor in query performance. This is possible since DB2 allows similar indexing of the F fact table and of the E fact table and thereby the query performance on the two fact tables does not differ.

This is also true if using MDC because both fact tables have the same MDC dimensions (except for the request dimension that is added to the MDC dimensions on the F fact table only). In contrast to that range partitioning still requires compression in this scenario to gain in query performance since the F fact table is partitioned on the request dimension and the E fact table is partitioned on the time dimension which usually results in a better query performance on the E fact table.

10.6 Data deletion with new SQL `DELETE` statement

Within SAP BI, there are two functions available to delete data from SAP BI InfoProvider tables:

- Selective Delete:

  Data is deleted from BW InfoCube fact tables or BW DataStore Active Tables. You call `Selective Deletion` either via transaction `delete_facts bye` selecting **Direct deletion** (Figure 10-27), or by clicking **Selective Deletion** on the contents tab of the *Manage Data Target* screen of InfoProviders (Figure 10-28).

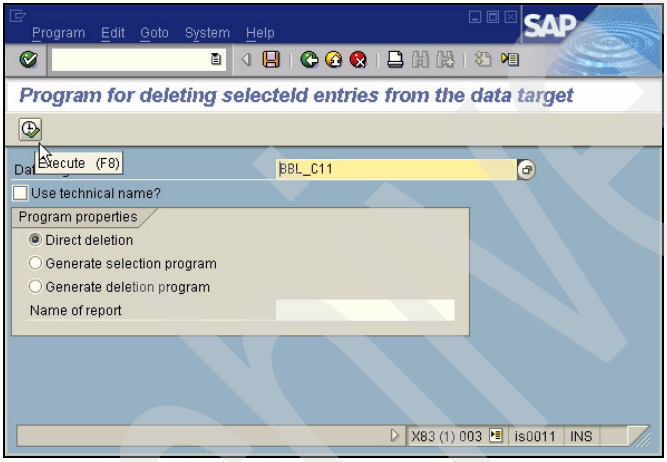


Figure 10-27 Selective Deletion via transaction delete_facts

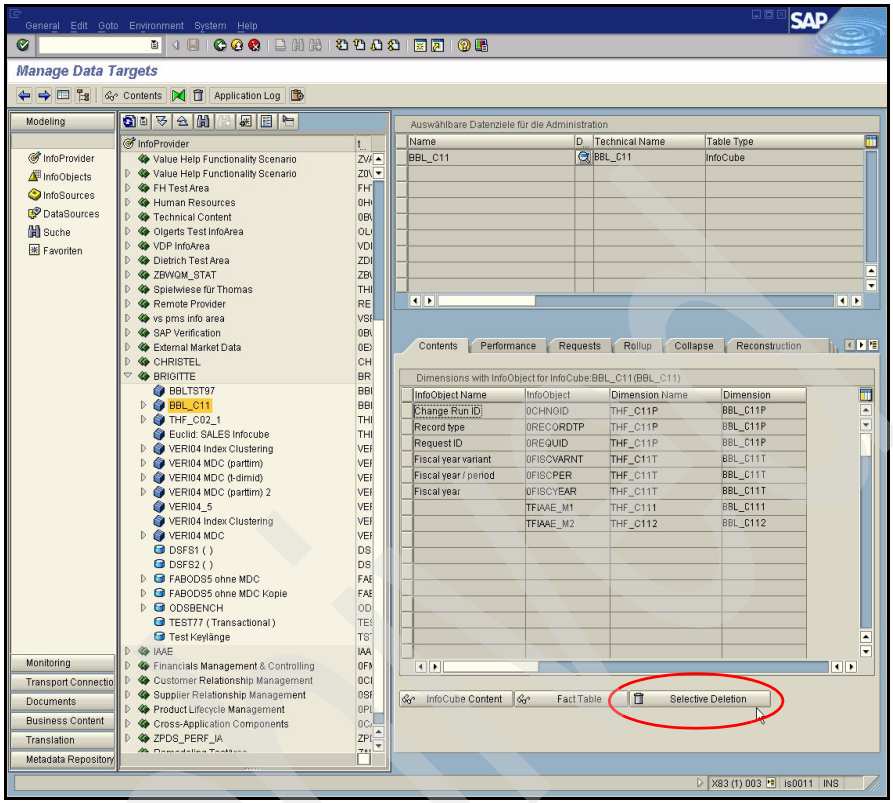


Figure 10-28 Selective Deletion on maintenance screen of InfoProvider

- Request Delete:

  Data requests are deleted from BW InfoCube F fact tables, PSA tables or from BW DataStore activation queue tables. You call request deletion via the Requests tab of the *Manage Data Targets* screen of InfoProviders (Figure 10-29).

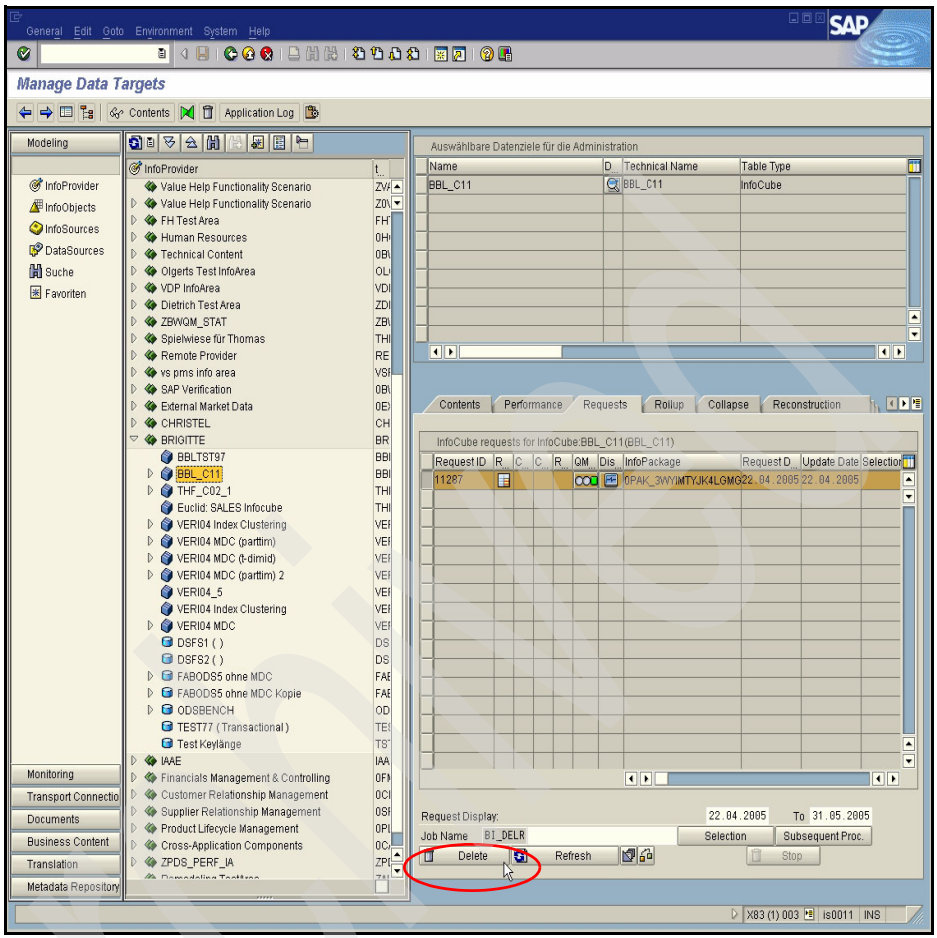


Figure 10-29 Deletion of requests via maintenance screen of InfoProvider

In both cases the new SQL `DELETE` Statement, which is available since DB2 UDB V8.1 FixPak 4, is used. The advantages of the new SQL `DELETE` statement are:

- Delete performance improvements.
- SQL access plans are simpler using the new SQL Delete statement and more robust, if database statistics are not up to date.

In general, two kinds of SQL `DELETE` statement are used:

- Simple delete: All data is deleted with one SQL `DELETE` statement.
- Delete in packages: data is deleted in packages to avoid log file full situations and avoid long database rollback time, if an error occurs.

With Request Deletion, a simple delete in one step is executed on MDC tables, which have no secondary RID indexes. In such cases the new DB2 UDB V8.2.2 feature "MDC Faster Delete" is applied, which improves delete performance as well as log space consumption, because delete processing and logging is based on block level instead of row level.

For non-MDC tables or MDC tables with secondary RID indexes, larger amounts of data have to be deleted in packages to avoid log overflows

Example 10-5 shows the old `DELETE` statement used prior to DB2 UDB V8.1 FixPak 4 for deleting data in packages. It uses the `rownumber()` function for determining the first 100000 rows of the F fact table that belong to the request with package dimension ID 8 and deletes them. This is repeated until all rows belonging to the request are deleted.

Example 10-5 DELETE statement prior to DB2 UDB V8.1 Fixpak 4

```
DELETE
FROM
  sapx83."/BIC/FBBL_C11" A
WHERE EXISTS
  ( SELECT *
     FROM TABLE
        ( SELECT "KEY_BBL_C11P", "KEY_BBL_C11T", "KEY_BBL_C11U",
                 "KEY_BBL_C111", "KEY_BBL_C112" ,
    ROWNUMBER() OVER() AS ROW
          FROM  sapx83."/BIC/FBBL_C11"
          WHERE "KEY_BBL_C11P" = 8
        ) AS B
     WHERE ROW <= 100000 AND
         A."KEY_BBL_C11P" = B."KEY_BBL_C11P" AND
         A."KEY_BBL_C11T" = B."KEY_BBL_C11T" AND
         A."KEY_BBL_C11U" = B."KEY_BBL_C11U" AND
         A."KEY_BBL_C111" = B."KEY_BBL_C111" AND
         A."KEY_BBL_C112" = B."KEY_BBL_C112"
  )
```

Example 10-6 shows the new `DELETE` statement (since DB2 UDB V8.1 FixPak 4) that uses the `FETCH FIRST <n> ROWS ONLY` clause for determining the first 100000 rows of the result set. It is much simpler because it doesn't require joining the F fact table with itself.

Example 10-6 New Delete statement

```
DELETE
FROM ( SELECT "KEY_BBL_C11P", "KEY_BBL_C11T", "KEY_BBL_C11U",
              "KEY_BBL_C111", "KEY_BBL_C112"
           FROM  sapx83."/BIC/FBBL_C11"
           WHERE "KEY_ BBL_C11P" = 8
           FETCH FIRST 100000 ROWS ONLY ) ;
```

In SAP BI systems on DB2 UDB V8.1 FixPak 4 or a higher FixPak level, the new Delete statement is used for both Selective Deletion and Request Deletion from PSA, DataStore, and InfoCube tables, resulting in better performance and stable access plans.

10.7 Use of sampled statistics

SAP NetWeaver 2004s supports the automatic statistics calculation of DB2 UDB. It is no longer necessary to schedule daily and weekly statistics collection jobs.

In SAP BI, during or after certain operations like data load, data deletion, and compression, statistics are collected for InfoCube tables. Until Before SAP NetWeaver '04 (SAP Business Information Warehouse 3.5), full statistics were collected on all table columns. SAP Business Information Warehouse 3.5 introduced the collection of statistics on the key columns of fact tables only. This reduces the time required for collecting statistics considerably, especially for fact tables with a large number of key figures. SAP NetWeaver 2004s BI additionally supports data sampling when collecting statistics for fact tables.

You can set RSADMIN parameters to enable or disable the following statistics options for InfoCube and aggregate fact tables:

- Running statistics only on the key columns of fact tables
- Running sampled statistics for the indexes of fact tables. Both DB2 UDB V8.1 and V8.2 support CPU sampling for indexes. A sampling rate cannot be specified.
- Running sampled statistics on fact table data. DB2 UDB V8.2 supports sampled statistics on table data based on a sampling rate.

These parameters do not apply to non-fact tables. For all non-FACT fact tables full statistics on all columns is always collected.

The RSADMIN parameters are as follows (see also SAP Note *801011*):

- `DB6_STATS_ON_KEYCOL_ONLY`

  Used to specify whether to collect statistics on fact tables on the key columns only. This speeds up statistics collection considerably. The default value for the parameter is `YES`, that is, statistics are collected on the key columns only. To collect statistics on all columns set the parameter value to `NO`.

- `DB6_SAMPLED_INDEX_STATS`

  Used to specify whether to collect sampled index statistics or not. The default value for the parameter is `NO`, that is, full index statistics are collected. To collect sampled index statistics set the parameter value to `YES`. For indexes, DB2 UDB uses CPU sampling. No sampling rate needs to be specified.

- `DB6_SAMPLED_TABLE_STATS`
  Used to specify the sampling rate for statistics on the table data. The default value for the parameter is `100`, that is, no data sampling will be used. To use data sampling set the parameter to a value between `1` and `99`. This parameter is used as the lower boundary for data sampling when statistics for fact tables is collected during BW SAP BI operations like data load, data deletion, and compression.

Execute function module `RSDU_FACTTB_ADJUST_DBSTATC_DB6` to change the statistics options stored for the fact tables of an existing InfoCube or for all existing InfoCubes.

You can change the data-sampling rate for InfoCubes on the *Performance* tab of the *Manage Data Targets* dialog of an InfoCube. The default value is a sampling rate of 10% (Figure 10-30).

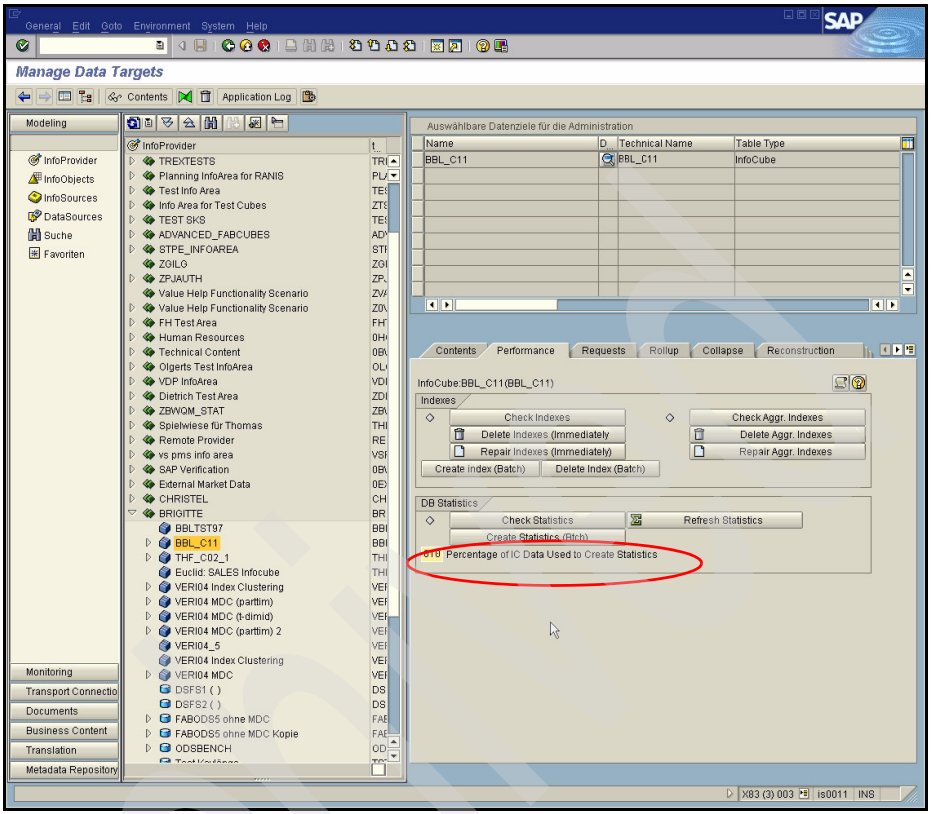


Figure 10-30 Figure caption

10.8 Migration of SAP BI systems to DB2 UDB

Migration of SAP systems to a new database platform or another operating system usually requires unloading the data in the source database and loading it into the new target database. The SAP term for migration is heterogeneous system copy. SAP provides a tool called R3load for migration. Migration is also necessary when converting an SAP system from non-Unicode to Unicode. In this case the database code page changes, so that database tools like restoring a database backup cannot be used.

SAP BI systems and systems based on it, like SEM and SCM, are difficult to migrate to other database platforms because, to get the maximum of performance, SAP BI makes use of specific features of the database platforms it supports. This includes hash partitioning for DB2 UDB and range partitioning for several other database platforms. Additionally the number, structure, and uniqueness properties of indexes on InfoCube fact tables may differ on the different database platforms. For range-partitioned InfoCubes, fact tables have an additional partitioning column.

SAP BI on DB2 UDB makes use of the following database specific features:

- Hash partitioning, requiring the specification of partitioning keys for tables to be created.
- Clustering indexes.
- MDC, requiring the specification of the MDC dimensions in an ORGANIZE BY clause for tables to be created.

With the following support packages and patches, SAP released a migration procedure for SAP Business Information Warehouse 3.0B and 3.1 Content and for SAP NetWeaver 2004:

- SAP Business Information Warehouse 3.0B / 3.1 Content:
  - R3load, Patch 61 of SAP Basis 6.20
  - SAP Basis 6.20 Support Package 46
  - SAP Business Information Warehouse 3.0B Support Package 25
  - SAP Business Information Warehouse 3.1 Content Support Package 19
- SAP NetWeaver '04:
  - R3load, Patch 17 of SAP Basis 6.40
  - SAP NetWeaver '04 Support Package Stack 10

The procedure can be applied to both migrations from other database platforms and to migrations from non-Unicode to Unicode. It is released for all source and target database platform combinations that have already been piloted. For new source and target data-base platform combinations, the first customer will become pilot customer and the procedure will be released after the first successful migration.

In the following sections, we briefly describe the migration procedure for SAP BI. You can find the details about the released source and target database platform combinations and the required patches on the SAP Service Marketplace:

`http://service.sap.com/bw-Services&Implementation-Migration`

Details can also be found in the SAP Notes *771209* and *777024*.

10.8.1 Migration procedure

The standard R3load based SAP migration procedure with consists of the shown in Figure 10-31.

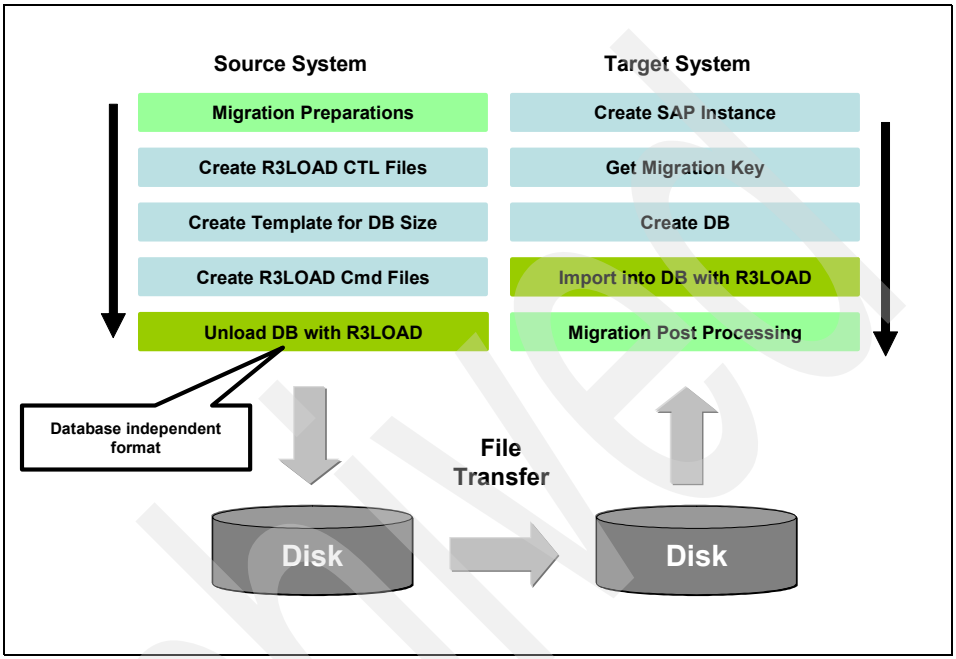


Figure 10-31 SAP migration procedure

In the source system, a number of reports have to be executed to prepare the migration. Then control files for the R3load tool, a template containing estimates for the target database size, and command files for the R3load tool are generated. The database content is exported in a database independent format with the R3load tool. The export consists of files containing the data and of files that describe the columns and data types of the tables and the columns and uniqueness properties of the indexes.

The export is transferred to the target server. On the target server the SAP instance is created. The customer gets a migration key and then creates the target database. The data is loaded with the R3load tool. In the new target system, a number of reports and transactions have to be executed to complete the migration.

SAP BI migrations contain two additional steps, Create DDL for Target DB and BW Migration Post Processing, marked in red in the Figure 10-32.

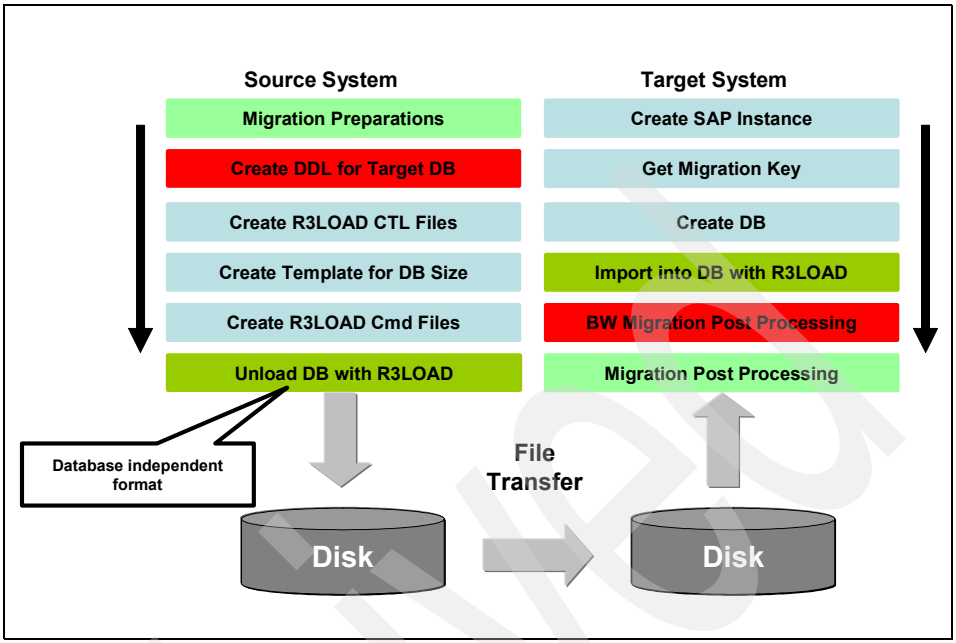


Figure 10-32 SAP BI system migration

On the source system, report `SMIGR_CREATE_DDL` has to be executed to create DDL statements for the tables and indexes that use database specific features or have different properties or a different layout on the target database platform. The DDL statements are stored in files that have to be copied to the export directory, and have to be transferred to the target server together with the rest of the export. The new, modified version of R3load uses the DDL to create the tables and indexes correctly in the target system.

On the target system, the BW SAP BI specific post migration report `RS_BW_POST_MIGRATION` has to be executed.

The SAP BI migration procedure is only available for releases greater or equal to SAP Business Information Warehouse 3.0B. For SAP Business Information Warehouse 2.0B and 2.1 Content systems there are two options:

- Upgrade the system to SAP Business Information Warehouse 3.0B / 3.1 Content and then migrate using the procedure described here.
- Register an SAP BI migration project at SAP and migrate the system with the help of SAP.

The SAP Service Marketplace and the SAP Notes *771209* and *777024* contain up to date information about the released source and target database platform combinations and about the required support packages and patches.

10.8.2 Generation of the DDL for the target database platform

To create the DDL for the SAP BI object tables and indexes for the target database system, report `SMIGR_CREATE_DDL` is run in the source system. Figure 10-33 shows the dialog that is displayed when running the report.

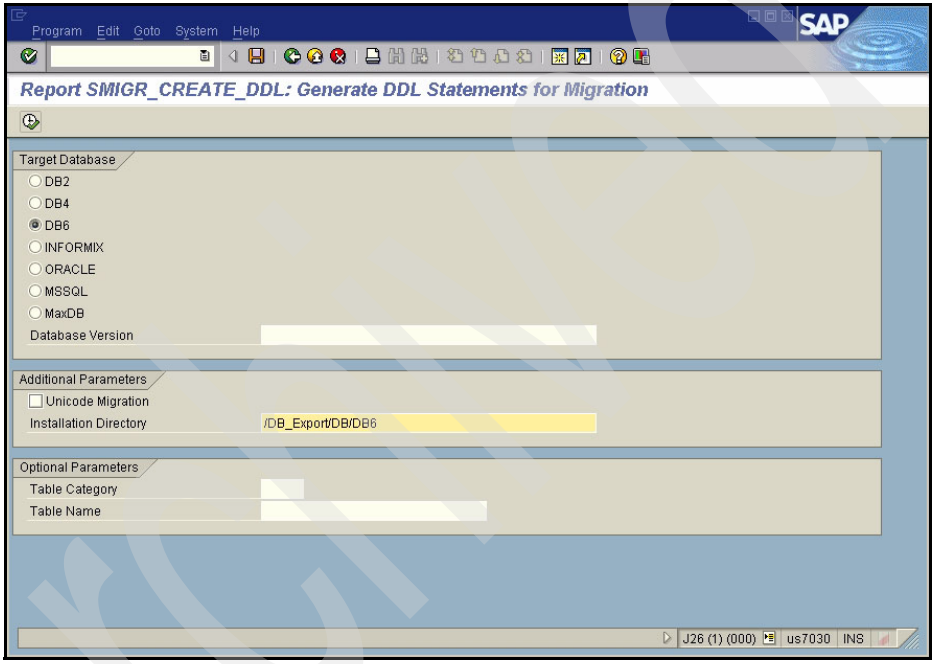


Figure 10-33 Report `SMIGR_CREATE_DDL`

The user has to

- Select the target database platform
- Specify whether a non-Unicode to Unicode conversion is involved
- Enter the output directory for the DDL

For test purposes, DDL generation can be restricted to a single table or a single SAP data class. The DDL is stored in files named *<dataclass>*`.SQL` for each SAP data class that contains SAP BI objects with database specific features or different implementations on the source and target database platforms.

The output files have to be placed in the DB/DB6 subdirectory of the source database export directory. R3load will use them as additional input files when creating the tables and indexes in the target database.

For migrations to DB2 UDB, SQL is created to handle the following database specific features and implementation differences:

- Partitioning keys for Fact, DataStore, and PSA tables
- Clustering indexes on Fact tables
- Indexes missing on the source database platform but required for DB2 UDB
- Indexes existing on the source platform but not required for DB2 UDB
- Differences in uniqueness and column order of indexes on the source platform

It is important that after the report has been executed and before the database export, no more changes to SAP BI database objects (for example, activations, field changes) should be made. The safest thing is to call the report directly before the export while the system is still locked for users.

Applying the DDL when creating the tables and indexes in the target database leads to a target system where the state in the SAP data dictionary is not consistent with the state in the database. SAP BI migration post processing will adapt the information about tables and indexes in the SAP data dictionary to the state in the DB2 UDB database when the target system has been created and loaded.

10.8.3 Exporting the source database

SAPinst is used to export the source database. For SAP NetWeaver 2004, simply select **ABAP Database Content Export** from either the Unicode or the non-Unicode branch of the SAPinst menu tree, as shown in Figure 10-34. For SAP NetWeaver 2004s, the export is contained in the *Lifecycle Management - System Copy* branch of the SAPinst menu tree.

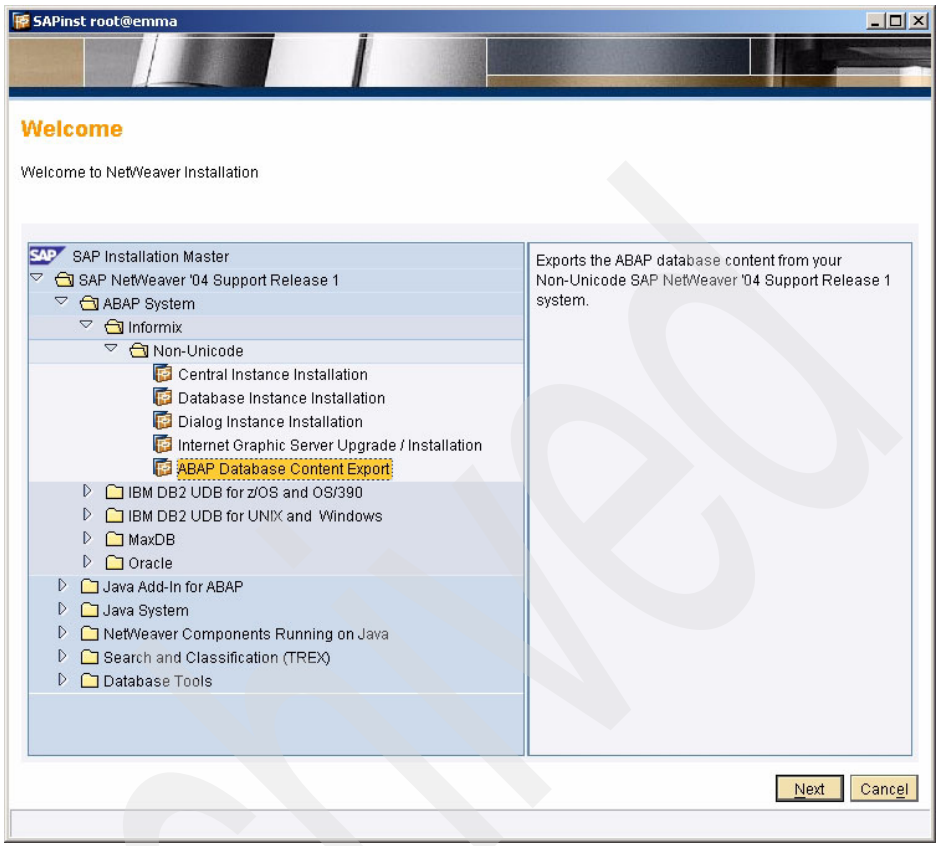


Figure 10-34 Source database export with SAP NetWeaver 2004 SAPinst

For SAP Business Information Warehouse 3.0B and SAP BI 3.1 Content, R3load based database export is not supported for all database platforms. It is recommended to export the source database with the SAPinst tool of R/3 Enterprise 4.7 Service Release 1 instead.

10.8.4 Creating and importing the target database

SAP NetWeaver 2004 and SAP NetWeaver 2004s fully support heterogeneous system copy with `SAPinst`. For SAP NetWeaver 2004, migration target system installation and import is contained in the standard database instance installation. For SAP NetWeaver 2004s, it is contained in the *Lifecycle Management - System Copy* branch of the `SAPinst` menu tree as a separate item.

The following screen shots are for SAP NetWeaver 2004. For SAP NetWeaver 2004s they are very similar.

- At some point in time during database instance installation `SAPinst` prompts whether to perform a standard installation or a system migration with R3load (Figure 10-35). Select **Standard System Copy / Migration (R3load-Base)**.

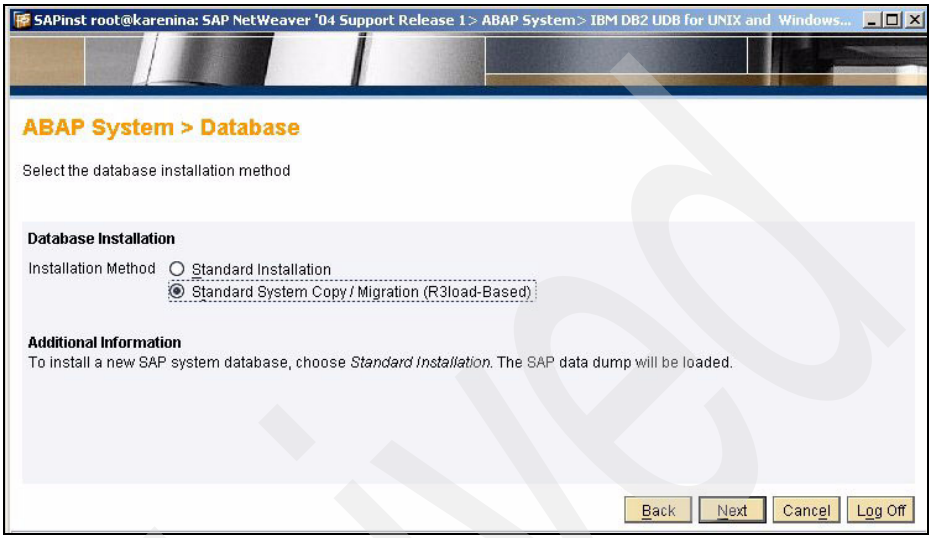


Figure 10-35 SAP NetWeaver 2004 - Target database installation 1

- It then prompts whether to use *R3load* or database backup and restore for the target database (Figure 10-36). Select `R3load`.

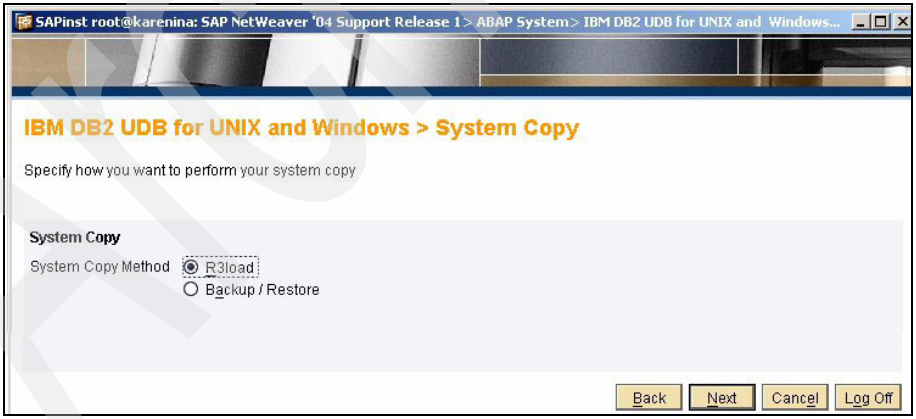


Figure 10-36 SAP NetWeaver 2004 - Target database installation 2

When prompted for the migration export location (Figure 10-37), enter the export directory of the source database as the package location for “Migration EXPORT”

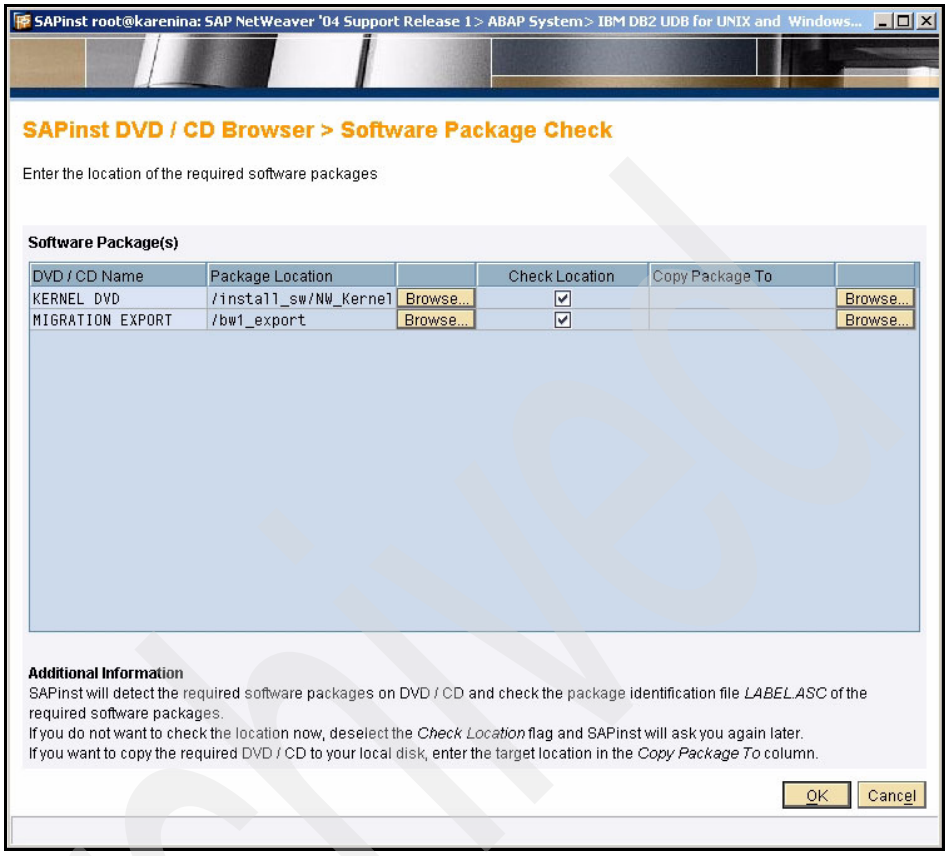


Figure 10-37 SAP NetWeaver 2004 - Target database installation 3

SAP Business Information Warehouse 3.0B and 3.1 content

SAPinst of SAP Business Information Warehouse 3.0B and 3.1 Content do not fully support heterogeneous system copy. However, it has to be used because only the SAPinst procedures for SAP Business Information Warehouse and SAP NetWeaver support DB2 UDB multi-partitioning. An SAP Business Information Warehouse 3.0B or 3.1 Content installation always requires SAPinst for the SAP Business Information Warehouse. The recommendation is to use SAPinst of SAP Business Information Warehouse 3.1 Content also for SAP Business Information Warehouse 3.0B installations.

SAPinst first prompts whether to perform a homogeneous system copy with the backup/restore method (Figure 10-38). Select **No** here and SAPinst will assume a standard installation.

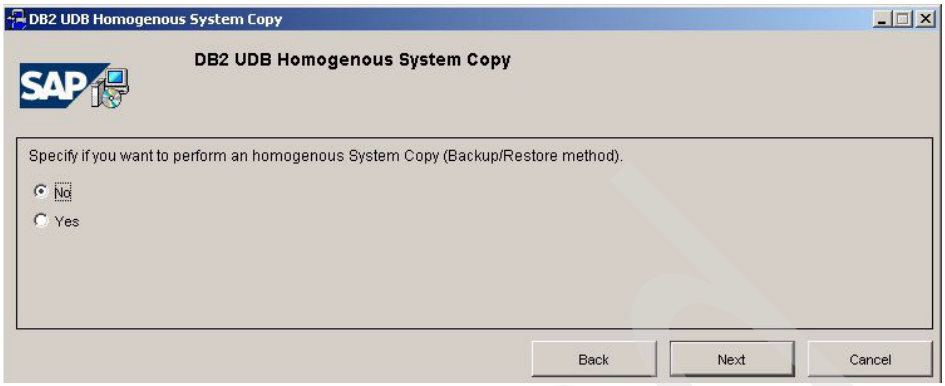


Figure 10-38 SAP BI Warehouse BW 3.0B and 3.1 content - target DB installation 1

When `SAPinst` prompts for the location of the first Export CD (Figure 10-39), you enter the directory where the export of the source database resides. Because `SAPinst` will check the CD label, you might have to replace the CD label of the export directory with the one from the first export CD of a BW SAP Business Information Warehouse 3.0B or 3.1 content standard installation.

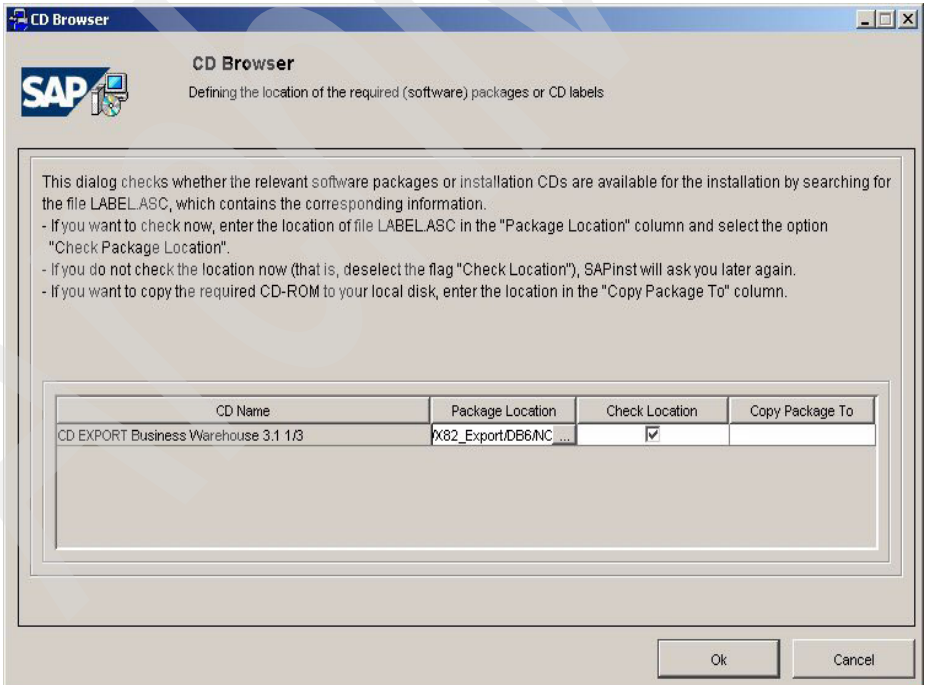


Figure 10-39 SAP BI Warehouse BW 3.0B and 3.1 Content - Target DB installation 2

When prompted for the database load parameters (Figure 10-40),select **load packages from the MIGRATION CD**, enter the migration key, and enter the directory where the export of the source database resides.

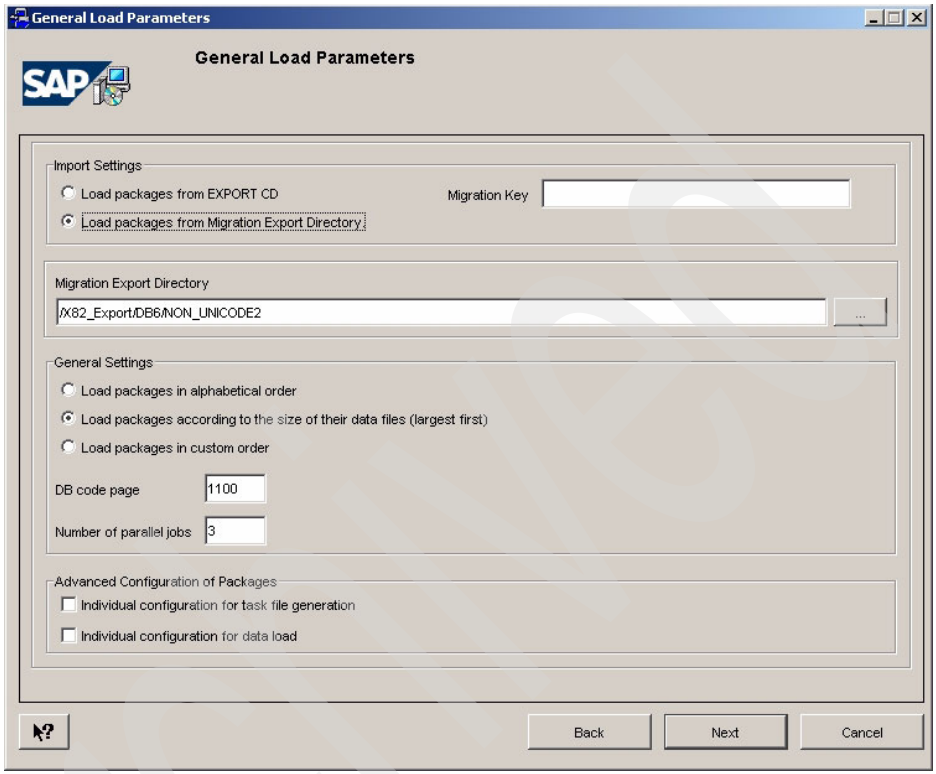


Figure 10-40 SAP BI Warehouse BW 3.0B and 3.1 Content - Target DB installation 3

10.8.5 SAP BI migration post processing

After database import, report `RS_BW_POST_MIGRATION` has to be executed in background on the target system. There are two variants defined for the report:

- Variant `SAP&POSTMGR` is used when the database platform has not changed, that is, in the case of a non-Unicode to Unicode migration.
- Variant `SAP&POSTMGRDB` is used when the database platform has changed. It includes a number of additional repair operations that are necessary because of the implementation differences on the different database platforms.

The two variants are defined for background processing only. It is recommended to always run the report in background because it might run for several hours.

It is very important that all migration post-processing steps are executed successfully. Otherwise, all kinds of errors can occur on the migrated system and it will not be usable. `RS_BW_POST_MIGRATION` writes a log that should be scanned carefully for any error messages.

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 633. Note that some of the documents referenced here may be available in softcopy only.

- *Oracle to DB2 UDB Conversion Guide,* SG24-7048
- *Database Transition: Informix Dynamic Server to DB2 Universal Database*, SG24-6367
- *MySQL to DB2 UDB Conversion Guide*, SG24-7093
- *Building and Scaling SAP Business Information Warehouse on DB2 UDB ESE*, SG24-7094

Other publications

These publications are also relevant as further information sources:

IBM publications:

- IBM DB2 UDB documentation, *What’s New V8,* SC09-4848-01
- IBM DB2 UDB documentation, *Administration Guide: Implementation V8*, SC09-4820-01
- IBM DB2 UDB documentation, *Administration Guide: Performance V8*, SC09-4821-01
- IBM DB2 UDB documentation, *Administration Guide: Planning V8*, SC09-4822-01
- IBM DB2 UDB documentation, *Application Development Guide: Building and Running Applications V8*, SC09-4825-01
- IBM DB2 UDB documentation, *Application Development Guide: Programming Client Applications V8*, SC09-4826-01

- IBM DB2 UDB documentation, *Application Development Guide: Programming Server Applications V8*, SC09-4827-01
- IBM DB2 UDB documentation, *Call Level Interface Guide and Reference, Volume 1*, *V8*, SC09-4849-01
- IBM DB2 UDB documentation, *Call Level Interface Guide and Reference, Volume 2, V8*, SC09-4850-01
- IBM DB2 UDB documentation, *Command Reference V8*, SC09-4828-01
- IBM DB2 UDB documentation, *Data Movement Utilities Guide and Reference V8*, SC09-4830-01
- IBM DB2 UDB documentation, *Data Recovery and High Availability Guide and Reference V8*, SC09-4831-01
- IBM DB2 UDB documentation, *Guide to GUI Tools for Administration and Development*, SC09-4851-01
- IBM DB2 UDB documentation, *Installation and Configuration Supplement V8*, GC09-4837-01
- IBM DB2 UDB documentation, *Quick Beginnings for DB2 Clients V8*, GC09-4832-01
- IBM DB2 UDB documentation, *Quick Beginnings for DB2 Servers V8*, GC09-4836-01
- IBM DB2 UDB documentation, *SQL Reference, Volume 1, V8,* SC09-4844-01
- IBM DB2 UDB documentation, *SQL Reference, Volume 2, V8,* SC09-4845-01
- IBM DB2 UDB documentation, *System Monitor Guide and Reference V8,* SC09-4847-01
- IBM DB2 UDB documentation, *Data Warehouse Center Application Integration Guide Version 8 Release 1*, SC27-1124-01-01
- IBM DB2 UDB documentation, *DB2 XML Extender Administration and Programming Guide Version 8 Release 1*, SC27-1234-01
- IBM DB2 UDB documentation, *Federated Systems PIC Guide Version 8 Release 1*, GC27-1224-01

SAP publications:

- *Database Administration Guide, IBM DB2 Universal Database for UNIX and Windows: New Log File Management*

Online resources

These Web sites and URLs are also relevant as further information sources:

DB2

- Database and Data Management

  http://www.ibm.com/software/data/
  http://www.ibm.com/software/data/highlights/db2tco.html

- DB2 Universal Database

  http://www.ibm.com/software/data/db2/udb/
  http://www.ibm.com/db2/v8

- DB2 developerWork

  http://www.ibm.com/developeworks/db2/

- DB2 Universal Database V8 Application Development

  http://www.ibm.com/software/data/db2/udb/ad

- DB2 Technical Support

  http://www.ibm.com/software/data/db2/udb/support/index.html

- DB2 Extenders

  http://www.ibm.com/software/data/db2/extenders/

- IBM Manuals for Data Management Products

  http://www.ibm.com/software/data/technical/BOOK/

- DB2 Migrate NOW!

  http://www.ibm.com/db2/migration

SAP

- SAP Service MarketplaceDescription2

  http://service.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

Index

D

E

F

G

H

I

P

R

U

V

W

SAP Solutions on IBM DB2 UDB V8.2.2 Handbook

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

SAP Solutions on IBM DB2 UDB V8.2.2 Handbook

Redbooks

Leverage highly integrated SAP and DB2 UDB for your enterprise system

Data protection strategies and monitoring using DBA Cockpit

Problem determination and diagnostics

IBM and SAP share a vision for responding to customers' specific requirements for tighter integration, platform flexibility, lower total cost of ownership, and improved performance. DB2 UDB V8.2.2 is an SAP-optimized version that uses autonomic computing technologies specifically tuned to help SAP users streamline installation, improve performance, and increase availability.

This IBM Redbook provides information for SAP consultants and SAP system administrators. It includes the latest DB2 UDB architecture details, SAP-related features and functions, practical advice on SAP NetWeaver 2004 installations, and DB2 configuration in SAP environments.

We describe DB2 UDB's logical and physical database objects, storage management, SAPs data classes, and space reclamation strategies, and provide the best practices for intelligent storage subsystems.

We also discuss how to use DBA Cockpit to monitor your system, DB2 UDB log file management, database backup and recovery, and high availability and clustered solutions. We introduce the Problem Description / Problem Source Identification (PD/PSI) methodology, and the troubleshooting approaches for common problems. Finally, we include a discussion of the new features of SAP NetWeaver 2004s Business Intelligence (SAP BI) with DB2 UDB V8.2.2.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6765-00 ISBN 0738493880