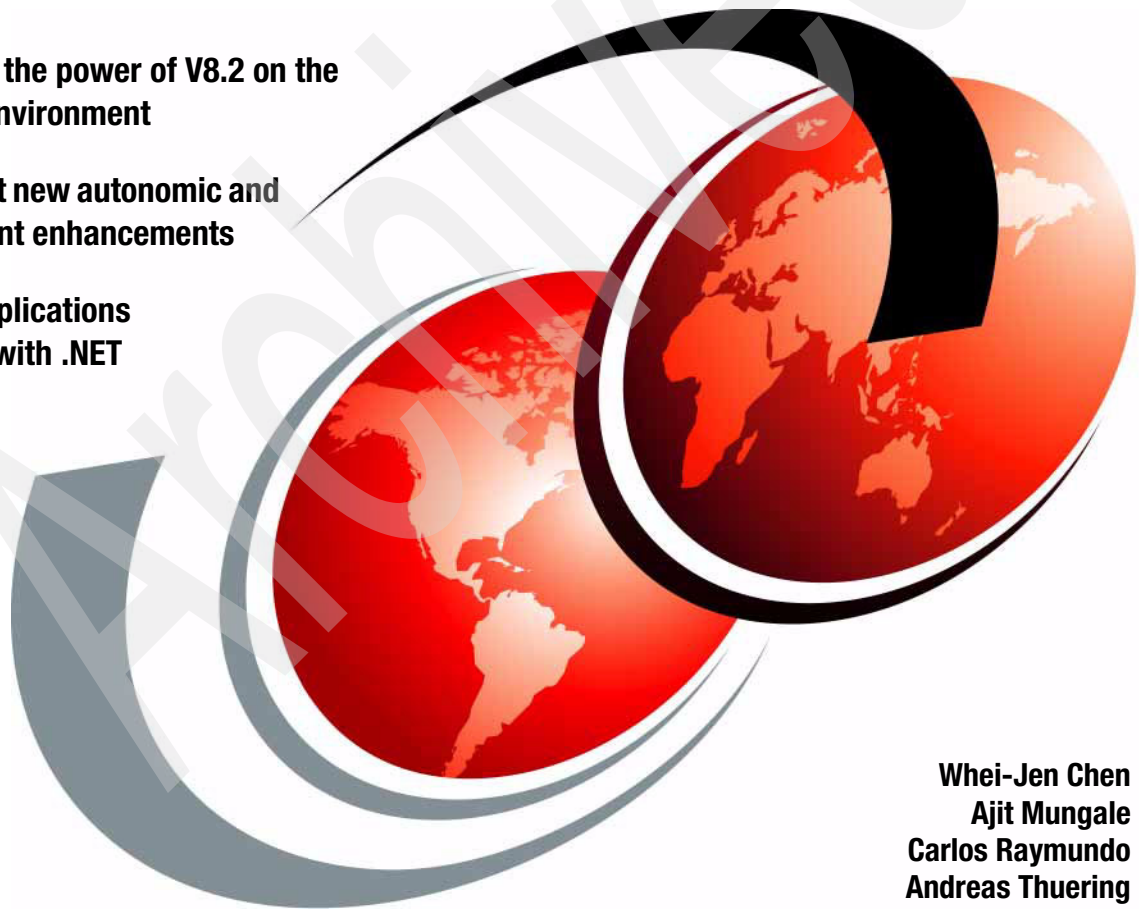


DB2 UDB V8.2 on the Windows Environment

Experience the power of V8.2 on the Windows environment

Learn about new autonomic and management enhancements

Develop applications using DB2 with .NET and Java



Whei-Jen Chen
Ajit Mungale
Carlos Raymundo
Andreas Thuerling



International Technical Support Organization

DB2 UDB V8.2 on the Windows Environment

October 2004

Archived

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

First Edition (October 2004)

This edition applies to DB2 Universal Database Version 8.2 for use with Microsoft Windows 2000 Server and Window Server 2003.

© Copyright International Business Machines Corporation 2004. All rights reserved.

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

Contents

Notices	vii
Trademarks	viii
Preface	ix
The team that wrote this redbook	x
Acknowledgements	xi
Become a published author	xi
Comments welcome	xii
Chapter 1. Introduction	1
1.1 DB2 UDB overview	2
1.1.1 DB2 technology strategy priorities	2
1.1.2 DB2 family	3
1.1.3 DB2 UDB products for Windows	9
1.2 DB2 UDB V8.2 technology highlights	11
1.2.1 SQL enhancements	12
1.2.2 Security enhancements	14
1.2.3 Business intelligence enhancements	15
1.2.4 Autonomic computing enhancements	16
1.2.5 Usability enhancements	20
1.2.6 High availability enhancements	21
1.3 DB2 UDB V8.2 integration with Windows	23
1.3.1 System security	23
1.3.2 Application development	24
Chapter 2. Installation and migration	27
2.1 DB2 UDB V8.2 installation	28
2.1.1 Installation overview for DB2 on Windows	28
2.1.2 DB2 UDB V8.2 server installation	31
2.1.3 DB2 UDB V8.2 client installation	36
2.2 Migration	40
2.2.1 Migrating DB2 between versions of Windows	41
2.2.2 Migrating between versions of DB2 UDB	47
2.2.3 Migrating from 32-bit to 64-bit	50
2.2.4 Migrating everything at once	50
Chapter 3. Administration and management	53
3.1 Backup, recovery, and logging	54
3.1.1 Automatic backup	54

3.1.2 Self-tuning backup and restore	64
3.1.3 Backup compression	66
3.1.4 Logs in backup images	67
3.2 Automated log file management	68
3.3 RECOVER command	69
3.4 Automated table maintenance	70
3.4.1 Automatic statistics collection	70
3.4.2 Automatic statistics profiling	76
3.4.3 Automatic reorganization	80
3.5 Integrated Design Advisor	86
Chapter 4. Security	117
4.1 Windows Domain handling and user IDs	118
4.1.1 Improved Windows Domain and Active Directory support	118
4.1.2 User ID and group name enhancements	127
4.2 Running DB2 under the local system account	132
4.3 Protecting DB2 UDB system files	136
4.4 Using DB2 with data encryption	138
4.5 The new DB2 security exit	141
4.5.1 DB2 authentication before Version 8.2	141
4.5.2 The new DB2 authentication model in Version 8.2	142
4.5.3 DB2 security plug-ins	143
4.5.4 Developing security plug-ins	152
4.5.5 Deploying a userid/password plug-in	157
4.5.6 Deploying a group retrieval plug-in	158
4.5.7 Deploying a GSS-API plug-in	158
4.5.8 Error handling	160
Chapter 5. Performance and monitoring	161
5.1 SQL query optimization enhancements	162
5.1.1 Native SQL procedures	162
5.1.2 SQL statement size limit increased to 2 MB	162
5.1.3 Data sampling in SQL queries	162
5.1.4 Dynamic SQL re-optimization	165
5.1.5 Specifying a lock wait mode strategy	166
5.1.6 Improved query execution plans	167
5.1.7 Multipage file allocation on SMS table spaces	167
5.1.8 Automatic setting of table space prefetch size	167
5.2 Multidimension clustering tables	167
5.2.1 MDC performance recommendations	170
5.3 Improvement of the RUNSTATS utility	171
5.3.1 Improved RUNSTATS performance through sampling	171
5.3.2 Throttling of the RUNSTATS utility	172

5.4 DB2 performance elements in Windows	173
5.5 DB2 Activity Monitor	175
Chapter 6. High availability.	189
6.1 High Available Disaster Recovery	190
6.2 Automatic client reroute	215
6.3 Index logging	218
Chapter 7. Using DB2 with Java.	221
7.1 Java Database Connectivity (JDBC)	222
7.1.1 Types of JDBC driver:	222
7.1.2 Writing a JDBC application	223
7.2 SQL Java (SQLJ)	231
7.2.1 Writing SQLJ applications	232
Chapter 8. Using DB2 with .NET framework.	237
8.1 An overview of ADO.NET	238
8.2 ADO.NET architecture	239
8.2.1 Connection	240
8.2.2 Command	241
8.2.3 DataReader	242
8.2.4 DataAdapter	243
8.2.5 DataSet	244
8.3 Connecting to the database	245
8.4 Data providers for DB2	246
8.4.1 OLE DB .NET Data Provider	246
8.4.2 ODBC .NET Data Provider	249
8.4.3 DB2 .NET Data Provider	254
8.4.4 Comparison between providers	256
8.5 Performing operations on a DB2 database	258
8.5.1 Using DataAdapter and DataSet (Disconnected model)	258
8.5.2 Calling stored procedures	261
8.5.3 Controlling transaction	263
8.5.4 Using large objects (LOBs)	265
8.5.5 Binding Data Controls with ADO.NET	268
8.5.6 Accessing DB2 with Web Forms	278
8.6 Add-ins and stored procedures in CLR	278
8.6.1 IBM DB2 Development Add-In overview	279
8.6.2 DB2 Toolbar	279
8.6.3 DB2 Database Project type	280
8.6.4 Data Connections folder in the IBM Explorer	281
8.6.5 DB2 SQL Editor	282
8.7 Developing DB2 stored procedures in .NET	282
8.7.1 Save and build the solution	287

Chapter 9. Consuming DB2 Web services in .NET.	295
9.1 Web services and DB2	296
9.2 Web Services Object Runtime Framework (WORF)	296
9.2.1 Document access definition extension (DADX)	297
9.2.2 DB2 XML Extender	300
9.2.3 How WORF processes a Web service request.	300
9.2.4 Setting up a WORF environment for Windows	301
9.2.5 Creating and deploying DB2 WORF service.	303
9.3 Consuming the DADX Web service using .NET client	313
Chapter 10. Problem resolution	317
10.1 RAS	318
10.2 PD/PSI	318
10.3 Information to collect	319
10.4 db2support utility	320
10.5 Understanding DB2DIAG.LOG	320
10.6 DB2DIAG tool	324
10.7 Prevention versus resolution	326
Related publications	333
IBM Redbooks	333
Other publications	333
Online resources	334
How to get IBM Redbooks	335
Help from IBM	335
Index	337

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®	Informix®	OS/390®
AS/400®	Intelligent Miner™	Redbooks™
DB2®	IBM®	Redbooks (logo)  ™
DB2 Connect™	IMS™	SQL/400®
DB2 Extenders™	iSeries™	WebSphere®
DB2 Universal Database™	MVS™	xSeries®
DRDA®	Net.Data®	z/OS®
@server®	Netfinity®	
Everyplace®	NetVista™	

The following terms are trademarks of other companies:

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

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

Intel and Intel Inside (logos) are trademarks of Intel Corporation in the United States, other countries, or both.

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

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

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

Preface

This redbook provides updated information about DB2 Universal Database™ (UDB) Version 8.2 from IBM® for Windows® environment. This book is organized as follows.

► **Chapter 1 Introduction**

In this chapter we introduce the DB2® technology strategy for the DB2 family, especially DB2 UDB V8.2. We provide a high-level description of the new features and functions introduced in DB2 UDB V8.2 and how the new capabilities and features of DB2 UDB V8.2 integrate with Windows Server 2003 and Visual Studio 2003.

► **Chapter 2 Installation and migration**

In this chapter we describe the basic procedures for installing 32-bit DB2 UDB V8.2 on Windows Server 2003. We also provide steps for various migration scenarios.

► **Chapter 3 Administration and management**

In this chapter we discuss the DB2 UDB V8.2 advancements in administration and management. The autonomic and management enhancements make the DBA's job easier than ever.

► **Chapter 4 Security**

In this chapter we discuss the new security enhancements of DB2 UDB V8.2 in the Windows environment such as Active Directory domains support, data encryption, and security plug-ins.

► **Chapter 5 Performance and monitoring**

In this chapter we provide information about the new performance enhancements in DB2 UDB V8.2. The monitoring tool Activity Monitor is also discussed.

► **Chapter 6 High availability**

This chapter covers the new high availability features in the new release of DB2 UDB including High Availability Disaster Recovery (HADR) feature, Automatic Client Reroute, and Index logging.

► **Chapter 7 Using DB2 with Java™**

In this chapter we discuss various ways to use Java Database Connectivity (JDBC) to perform operations on the database and Structured Query Language for Java (SQLJ).

► **Chapter 8 Using DB2 with .NET framework**

In this chapter we discuss various features that are available for .NET

application developers. Topics covered are ADO.NET, various Providers, and application development using DB2 .NET Provider.

► **Chapter 9 Consuming DB2 Web Services in .NET**

In this chapter we introduce Web Services Object Runtime Framework (WORF) and show you how to use DB2 as a Web service provider and Web service consumer.

► **Chapter 10 Problem resolution**

In this chapter we introduce the enhanced DB2 diagnostic log and new tools for examining it.

The team that wrote this redbook

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

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.

Ajit Mungale has been working with IBM GSI as a Senior Software Engineer. He has extensive experience with Microsoft® technologies and has worked with almost all languages and technologies. He also has experience with IBM products, including WebSphere® and MQ. He is author of several other books and has published articles about Microsoft .NET.

Carlos Raymundo is a Independent Senior Consulting IT Specialist in Trujillo, Peru. He has 14 years of experience providing technical support across a variety of products and technologies in database management systems. He has more than 11 years of experience as a professor at the Computing and System Department at the Antenor Orrego University. He holds a Masters degree in Management Information Systems from the Instituto Tecnológico de Monterrey and the Antenor Orrego University, and is candidate for a Doctoral degree in System Engineering with a major in Computing from the Federico Villareal National University. He holds several industrial certifications in IT products such as IBM Certified Database Administrator, Oracle Database Administrator Master, and Microsoft Certified System Engineering. He can be reached at <mailto:craymundoi@upao.edu.pe>.

Andreas Thuerling is a Senior IT Specialist at IBM Global Services in Basel, Switzerland. He has 18 years of IT experience in the areas of application development, middleware implementation, database modeling, design, and

administration on the Linux®, AIX®, and Windows platforms. He has knowledge of several database systems and programming environments with different languages. Andreas has worked with DB2 on distributed platforms since 1995. He has project experience in the insurance, pharmaceutical, healthcare, telecommunications, and energy industries.

Acknowledgements

The authors express their deep gratitude for the help they received from the following people, who contributed advice, support, and written content.

Il-sung Lee is an IBM DB2 UDB developer. One of his areas of specialization is DB2 UDB security.

Volker Markl is a designer from IBM Almaden Research Center. One of his specialties is query optimization and self-managing databases.

Dale McInnis is an IBM DB2 UDB developer. Dale's area of specialty is DB2 UDB backup, recovery, and high availability.

Jan Nelken works in IBM DB2 UDB and DM support. One of his specialties is problem determination in DB2.

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

Mike Logan
IBM DB2 Product Management and Marketing

Andrew Hilden, Jackson Hui, Matt Huras, Leon Katsnelson, Desmond Lam, Ivan Lew, Garfield Lewis, Aslam Nomani, Jay Pederson, Hiep Phuong, Sam Qita, Scott Walkty, Shili Yang
IBM Toronto Laboratory

Michael Schenker, Dirk Wollscheid
IBM Silicon Valley Laboratory

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 will team with IBM technical professionals, Business Partners, and/or customers.

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us because 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

Introduction

In this chapter we introduce the DB2 technology strategy for the IBM DB2 family, especially the DB2 Universal Database (UDB) Version 8.2. We provide a high-level description of the new features and functions introduced in DB2 UDB V8.2 and how its new capabilities and features integrate with Windows Server 2003 and Visual Studio 2003.

This chapter contains the following sections:

- ▶ DB2 UDB overview
 - DB2 technology strategy priorities
 - DB2 family
 - DB2 UDB products for Windows
- ▶ DB2 UDB V8.2 technology highlights
 - SQL enhancements
 - Security enhancements
 - Business intelligence enhancements
 - Automatic computing enhancements
 - High availability enhancements
- ▶ DB2 UDB V8.2 integration with Windows
 - New features in Windows 2003
 - New features in Visual Studio 2003

1.1 DB2 UDB overview

In this section, we provide an introduction to the DB2 technology strategy and a description of the IBM DB2 family, especially DB2 UDB for Windows.

1.1.1 DB2 technology strategy priorities

The DB2 UDB database is the most advanced self-managing, self-configuring, self-optimizing database in the world. The IBM DB2 technology strategy for the next few years will be:

- ▶ Optimize for multiple workload environments.

From Palmtop to teraflop, DB2 supports various hardware platforms. DB2 UDB provides different editions that are suited to different business needs and workload environment. The highly scalable features enable the information system to grow as the business grows. The configuration, tuning, and monitoring tools enable system support personnel and DBAs to optimize the database systems for multiple workload environments.

- ▶ Deliver optimized transparent access to all forms of digitized information.

IBM is the only company that supports a customer's complete enterprise content management requirements, including support for all forms of information, document and records management, collaboration, Web content, information integration, digital rights management, and storage. IBM DB2 continuously delivers enterprise content management software that enables clients to link their business processes and gain a seamless flow of information across and beyond the enterprise.

- ▶ Ease application development.

In each release of DB2 UDB, IBM has brought application development functionality to a new level. DB2 UDB V8.2 accelerates the application development life cycle by simplifying and minimizing the development cycles and costs that are associated with building and deploying on demand e-business database applications. It provides significant new application development functionality such as new SQL features, seamless integration with Java and Microsoft .NET development environment, and the new DB2 Run-Time Client Lite, which makes building and redistributing DB2 applications easier.

- ▶ Minimize DBA skill requirements.

DB2 has been incorporating capabilities that have come from the ongoing IBM autonomic computing initiative for making computers more able to self-manage. New features such as Configuration Advisor and Design Advisor, which are provided by each release of DB2 database server

software, automate common administrative tasks and minimize DBA skill requirements, thereby reducing DB2 total cost of ownership.

- ▶ Deliver high-quality database services.

DB2 is the core of a wide variety of data management products and solutions. With the strong commitment to research and development, IBM Information Management has delivered high-quality database services in the area of business intelligence, enterprise content and records management, federation, and information integration.

- ▶ Be the best ISV partner.

IBM provides ISVs with comprehensive technical, marketing, and sales support to help meet the specific needs of medium-sized businesses. Aligned with this IBM strategy, DB2 UDB has continuously delivered functions and features that facilitate ISVs to provide customers with flexible and secure information management solutions that span multiple computing platforms, including Windows.

1.1.2 DB2 family

The IBM DB2 database software family, the worldwide leader in the industry, marks the next stage in the evolution of the relational database. DB2 is the industry's first multimedia, Web-ready relational database management system delivering leading capabilities in reliability, performance, and scalability. DB2 is the database of choice for customers and partners who develop and deploy critical solutions. The DB2 family is a consistent set of relational database management systems (RDBMS) that utilize shared technologies and a common application programming interface.

These are the specific RDBMS products that make up the DB2 family:

- ▶ DB2 Universal Database for z/OS® and OS/390®

The premier IBM enterprise RDBMS for use on the mainframe to run powerful enterprise applications and make large scale e-commerce a reality.

- ▶ DB2 Universal Database for iSeries™

An advanced, 64-bit relational database system that provides leading-edge performance in e-business and data warehousing environments. (The iSeries and DB2 UDB for iSeries in combination provide the flexibility and adaptability to support any type of workload, small or large.)

- ▶ DB2 Server for VSE and VM

A full-function RDBMS supporting production and interactive IBM VM and VSE environment for your company.

► DB2 Everyplace®

Mobile relational database and enterprise synchronization architecture for mobile and embedded devices.

► DB2 Universal Database V8.2

The IBM object-relational database solution for the UNIX®, Linux, and Windows operating environments. It is built on a solid foundation, bringing together a client/server database product with the IBM leadership in mission-critical relational database technology. The result is a highly scalable, highly extensible database that is very easy to use and manage and can be trusted with your most critical database applications. DB2 UDB is available on the following platforms:

- Windows NT/2000/XP/Server 2003, Windows 95/98/ME
- Linux (on Intel)
- Linux (on 390)
- AIX
- HP-UX
- Sun Solaris

DB2 UDB capabilities and benefits

The main capabilities and benefits of DB2 Universal Database are:

► Superior scalability

One of the best benefits of DB2 UDB is the scalability. It can run on a laptop or mobile device supporting a mobile user, or it can run on a massively parallel machine such as symmetric multiprocessors (SMPs) and clusters of SMPs to support multiple terabytes of data and thousands of users.

DB2 UDB scalability features can be grouped into three major capabilities:

- Advanced parallel processing
- High-performance computing
- Efficient large database operations

► Multimedia extensibility

Another major benefit of DB2 UDB is extensibility, and the key to this capability is *object-relational technology*. This is the ability to store and manage not only traditional relational tables with characters and numbers, but also multimedia and complex objects such as documents, images, audio, video, spatial information, and time-series data. This may also include industry-specific objects such as x-rays, fingerprints, engineering drawings, and insurance claim forms.

DB2 UDB object-relational capabilities enable you to add your own data types and business functions to the database, effectively tailoring the database to fit your specific business or application requirements.

DB2 UDB extensibility features can be grouped into the following categories:

- Universal data
- Business rules
- Advanced SQL
- DB2 Extenders™

► Web enablement

Customers must be able to make the data that is stored in their DB2 database systems accessible to employees of the company, and selectively to their suppliers and customers through private network (intranet) and public network (Internet) applications. DB2 UDB is fully integrated with Web technology so that data can be accessed easily from the Internet or from your intranet with complete security. The following facilities are included with UDB so you can Web-enable your database applications right out of the box:

- DB2 Java support
- Net.Data® — A Web server database gateway

► Partner solutions

In order to implement a data management solution that meets the changing needs of your business, you need a robust relational database and carefully chosen applications to run against it. DB2 UDB provides the rock-solid foundation that is required to successfully deploy enterprise solutions, and thousands of independent software vendors (ISVs) offer a diverse range of sophisticated applications that support DB2.

In close partnership with IBM, leading software developers such as Siebel, I2, SAP, Baan, Dassault, and PeopleSoft develop enterprise resource planning (ERP) solutions and CRM solutions as well as B2B solutions and other applications that deliver the functionality your user base needs. These premier ISVs recognize that DB2 UDB has the performance, reliability, and flexibility to meet the demands of today's operational and business intelligence applications.

For the most up-to-date information on solutions and to look through the online IBM Solutions Catalog, go to these two sites:

<http://www.software.ibm.com/data/partners/>
<http://www.software.ibm.com/solutions/isv>

► Business intelligence powerhouse

By business intelligence (BI), we mean such applications as data warehousing, data mining, online analytical process (OLAP), and decision support. Many customers are looking for ways to mine and analyze their operational data for competitive advantage. These are among the most important uses of data management technology today because they provide customers with excellent returns on their investment.

Several factors make DB2 UDB an outstanding data store for BI applications:

- These applications often involve large volumes of data. Typical small to medium-sized warehouses and datamarts might contain 50 GB to 300 GB. Large installations can contain several terabytes. DB2 UDB addresses these requirements with its outstanding scalability, including advanced parallel query and VLDB operations.
- Queries against the database tend to be very complex. DB2 UDB has the most advanced query optimizer in the industry, providing excellent query performance with a minimum of DBA time required for tuning.
- DB2 UDB has some features that are designed specifically to assist with OLAP support.
- DB2 also offers the Data Warehouse Manager feature, an integrated set of tools for building, managing, and maintaining data warehouses and datamarts. The Warehouse Manager tool works with the Warehouse Center, the integrated GUI tool that coordinates and automates the activities that are needed to extract, clean, and populate the data into your informational data store.

► Advanced optimizer

The DB2 SQL optimizer strengthens the DB2 leadership position in query optimization for traditional applications and also extends for object-oriented applications. This new optimization technology provides the function and performance needed by customers to analyze and exploit the vast amounts of valuable information stored in their databases, such as decision support applications.

The optimizer incorporates a very sophisticated query rewrite phase that automatically transforms a complex query into a simpler query that is easy to optimize and search for the best query execution plan, as well as employing more sophisticated techniques of modeling the cost of different ways of fetching the data from disk. As a result, the end user will realize the best possible performance regardless of the way a query is structured.

A related technology is the DB2 support of materialized query tables (MQT) or automated summary tables (AST). Using MQT queries that might take minutes or hours to complete can be shortened dramatically, often to seconds or even sub-second response times. This is done by precalculating summary information into a summary table, then using the power of the optimizer's query rewriting to change the submitted query so that it retrieves the information from the summary table, rather than recalculating it. Users do not have to change their queries to take advantage of this performance improvement, because it is handled automatically by the DB2 optimizer after the administrator defines the MQT.

► OLAP support

Data warehouse and OLAP applications are characterized by the use of a special design technique, which is called *star schema*, to model relational data for multidimensional analysis (MDA). Under this schema it is common for a large “fact” table to be joined to multiple “dimension” tables in a very complex SQL join query. Any optimization techniques that can be used to improve the performance of these joins can significantly improve the performance of the overall application. DB2 UDB has special optimization techniques to do this:

– Index ANDing using dynamic bit maps

UDB uses dynamic bit-map technology to efficiently combine multiple indexes. Performance is improved for queries that use columns that are key columns of different indexes over the same table. This includes the use of indexes for multiway joins.

– Star joins

The DB2 star join algorithm exploits dynamic bit maps to join a large fact table with a series of relatively small dimension tables, thus minimizing data I/O.

► Ease of use

DB2 UDB is one of the easiest databases in the industry to use and manage. It includes a complete suite of graphical tools to satisfy the needs of:

- Database administrators (DBAs)
- Application programmers

It also includes tools to assist with client setup and ad hoc query and reporting for end users.

– Administering databases with the DB2 Administration Tools

You can administer local or remote servers using the DB2 Administration Tools. Use the Control Center to perform administration tasks such as configuring DB2 instances and databases, backing up and recovering data, scheduling jobs, and managing media, all from a graphical interface.

– DB2 autonomic computing technology

IBM has put a lot of research and development effort into providing *autonomic computing* functionality within DB2 to help make database administration more manageable, increase efficiencies, and lower the costs associated with managing an IT infrastructure as data management systems continue to grow in size and complexity.

► Data access and replication

DB2 UDB is a distributed, fully networked RDBMS. It provides you with the flexibility of placing data anywhere in your network for optimum service and

productivity, and provides efficient means for clients to access that data over the network. DB2 provides the most efficient and seamless integration of data on mainframe and midrange data servers in the industry, enabling you to reduce costs and improve cycle times by leveraging your current investments in data, hardware, software, and skills.

DB2 UDB support for distributed data can be grouped into three categories:

- Client/server data access
- Data replication
- Host data integration through DRDA®

► Federated systems

A DB2 federated system is a special type of distributed database management system (DBMS). A federated system consists of a DB2 instance that operates as a server, a database that serves as the federated database, one or more data sources, and clients (users and applications) who access the database and data sources. With a federated system, you can send distributed requests to multiple data sources within a single SQL statement. The power of a DB2 federated system is in its ability to:

- Join data from local tables and remote data sources, as if all the data were local.
- Take advantage of data source processing strengths by sending distributed requests to the data sources for processing.
- Compensate for SQL limitations at the data source by processing parts of a distributed request at the federated server.
- Write capability to perform INSERT, UPDATE, and DELETE actions on data sources.
- Ability to create remote tables on relational data sources.

Federated database systems provide the middleware functionality for outstanding information integration. Built into DB2 Enterprise Server Edition is the ability to federate relational data across the IBM family of databases, including the DB2 family and Informix® IDS.

► Multi-platform support

DB2 UDB is one of the most open database platforms available. It runs on the most popular UNIX and Intel server platforms including AIX, HP-UX, Solaris, Linux, and Windows NT/2000/Server 2003/XP. It supports all major industry standards that are relevant to distributed data so that it can be accessed using thousands of existing tools and applications, and can be easily managed within an open network computing environment.

- ▶ Bulletproof reliability

DB2 UDB is setting the standard for quality and reliability in the client/server database industry. As more mission-critical applications are implemented on UNIX and Intel® platforms, the IBM ability to bring mainframe-level reliability to this environment has become a major factor in choosing DB2. Better reliability and availability can reduce your costs, while scalability both within and across platforms can reduce the risk of dead-end projects.

1.1.3 DB2 UDB products for Windows

DB2 UDB for Windows consist of several product offerings based on a single code base that provides you with a great deal of flexibility in licensing, depending on your requirements. In this section, we provide brief descriptions of the various DB2 UDB products for Windows.

The following are various DB2 UDB packages available for selection based on the business requirements:

- ▶ DB2 UDB Enterprise Server Edition

DB2 UDB Enterprise Server Edition (ESE) meets the database server needs of midsize to large businesses. It can be deployed in any environment on any size server. ESE is the ideal foundation for building data warehouses, transaction processing, and Web-based solutions as well as a back end for packaged solutions such as ERP, CRM, and SCM. Additionally, ESE offers connectivity and integration for other enterprise DB2 and Informix data sources.

- DB2 Database Partitioning Feature

The DB2 Database Partitioning Feature (DPF) enables DB2 UDB Enterprise Server Edition customers to partition a database within a single system or across a cluster of systems. The DPF capability provides the customer with multiple benefits, including scalability to support very large databases or complex workloads and increased parallelism for administration tasks.

- ▶ DB2 UDB Workgroup Server Edition

DB2 UDB Workgroup Server Edition (WSE) is the database server designed for deployment in a departmental or small business environment that involves a small number of internal users. WSE uses a licensing model that is designed to provide an attractive price point for smaller installations while still providing a full-function database server. WSE can be deployed in any environment on systems with up to four CPUs.

► DB2 UDB Workgroup Server Unlimited Edition

DB2 UDB Workgroup Server Unlimited Edition (WSUE) offers a simplified per-processor licensing model for deployment in a departmental or small business environment that has Internet users or a quantity of users that makes per-processor licensing more attractive than the licensing model for DB2 UDB Workgroup Server Edition. The WSUE can be deployed in Linux, UNIX, and Windows environments on systems with up to four CPUs.

► DB2 UDB Express Edition

The DB2 UDB Express is a specifically tailored database offering for small and medium businesses. The key features include simplified deployment, autonomic management capabilities, and application development support. It is designed for independent software vendors who need an easily installed database integrated into their application software solution. It is a multi-user version that supports local and remote applications in stand-alone and local area network environments.

► DB2 UDB Personal Edition

DB2 UDB Personal Edition (PE) provides a single user database engine ideal for deployment to PC-based users. PE can be remotely managed, making it the perfect choice for deployment in occasionally connected situations or remote office implementations that do not require multi-user capability.

► DB2 Universal Developer's Edition

DB2 UDB Universal Developer's Edition (UDE) offers a low-cost package for a single application developer to design, build, or prototype applications for deployment of any of the DB2 client or server platforms. It includes all client and server DB2 editions, DB2 Connect™, DB2 Extenders, DB2 Warehouse Manager, and Intelligent Miner™. The software in this package cannot be used for production systems.

► DB2 Personal Developer's Edition

DB2 UDB Personal Developer's Edition (PDE) enables a developer to design and build single user desktop applications.

In addition, IBM also offers following DB2 clients free of charge:

► Run-Time Client

Provides the functionality that is required for an application to access DB2 Universal Database servers and DB2 Connect servers. Its functionality includes communication protocol support and support for application interfaces such as JDBC, SQLJ, ODBC, CLI, OLE DB, and .NET.

- ▶ **Run-Time Client Lite**

This is a new addition in DB2 UDB V8.2. Run-time Client Lite is a smaller-footprint version of the DB2 Run-Time Client and is available only on Windows environments. It provides basic functions that enable your applications to access DB2 UDB server. The DB2 Run-Time Client Lite also contains support necessary for JDBC, SQLJ, ODBC, CLI, OLE DB, and .NET similar to the DB2 Run-Time client. With its reduced installation image size, DB2 Run-Time Client Lite is ideal for mass deployment or for bundling with your applications.

- ▶ **Administration Client**

Provides the ability for any client from a variety of platforms to access and administer DB2 databases. The DB2 Administration Client has all the features of the DB2 Run-Time Client and includes all of the DB2 administration tools and support for thin clients.

- ▶ **Application Development Client**

Used to develop DB2 database applications, including stored procedures, user-defined functions, and client applications, this contains all of the functionality available in the DB2 Run-Time Client. The Application Development Client is a collection of graphical and non-graphical tools and components for developing character-based, multimedia, and object-oriented applications. Special features include the Development Center and sample applications for all supported programming languages. The Application Development Client includes the tools and components that are provided as part of the DB2 Administration Client product.

1.2 DB2 UDB V8.2 technology highlights

IBM DB2 UDB V8.2 for Linux, UNIX, and Windows, codenamed Stinger, marks the next stage in the evolution of the relational database. Technology highlights for Version 8.2 are:

- ▶ SQL enhancements
- ▶ Security enhancements
- ▶ Business intelligences enmeshments
- ▶ Automatic computing enhancements
- ▶ High availability enhancements

We introduce the highlights of DB2 UDB V8.2. For more details about the specific enhancements, see the IBM DB2 Universal Database “Stinger” Web site:

<http://www.ibm.com/software/data/db2/udb/v82/>

1.2.1 SQL enhancements

The following enhancements have been added:

- ▶ Native PSM

The creation of SQL procedures does not require a C or C++ compiler on the server; therefore C or C++ compiler setup is not required. When you create a SQL procedure, its procedural statements are converted to a native representation called Native PSM (Persistent Storage Module). The new PSM generates a plan and bytecode that is stored in the catalog. When a SQL procedure is called, the native representation is loaded from the catalog and the DB2 engine executes the procedure at execution time.

- ▶ Support of CALL statement in a trigger body

You can now invoke procedures from triggers or any other dynamic compound statement, in single partition environments, by executing a CALL statement that references a procedure within a trigger action.

When a procedure is called by a trigger action, it enables you to encapsulate complex logic in the trigger, such as operations on other tables in the database, or operations external to the database (such as sending an e-mail or writing an audit record to a file in the file system of the database server).

- ▶ Support for REOPT bind option

The new REOPT option defers the compilation of the SQL statement until run time when the access plan can be optimized using the actual values for the input variables: host variables, special registers, and parameter markers. This new feature can improve the execution performance.

- ▶ Larger SQL statement

The previous 64 KB limit on statement size limited the total size of statements (for example, "CREATE PROCEDURE" or "CREATE TRIGGER"), which limited the size of the object. Now DB2 UDB V8.2 provides an increased SQL statement size up to 2 MB. The new 2 MB limit enables you to use large statements when your application logic is in stored procedures or triggers.

The increase in the statement size also provides more flexibility to migrate a trigger or stored procedure statement from another RDBMS to DB2 UDB. The new statement limit enables you to record auditing context records that have statement text up to 2 MB.

- ▶ Support SET LOCK WAIT / NO WAIT

Traditional locking approaches can result in applications blocking each other. This happens when one application must wait for another application to release its lock. Strategies to deal with the impact of such blocking usually provide a mechanism to specify the maximum acceptable duration of the

block. That is the amount of time that an application will wait prior to returning without a lock.

An application or session can now specify a lock wait mode strategy, which is used when the session requires a lock that it cannot obtain immediately. The strategy indicates whether the application or session will:

- Return a `SQLCODE` and `SQLSTATE` when it cannot obtain a lock.
- Wait indefinitely for a lock.
- Wait a specified amount of time for a lock.
- Use the value of the *locktimeout* database configuration parameter when waiting for a lock.

The lock wait mode is specified through the new `SET CURRENT LOCK TIMEOUT` statement, which changes the value of the `CURRENT LOCK TIMEOUT` special register. The `CURRENT LOCK TIMEOUT` special register specifies the number of seconds to wait for a lock before returning an error indicating that a lock cannot be obtained.

► Changing generated column property

Now we can change generated column properties without having to re-create the table using the `ALTER COLUMN` clause in the `ALTER TABLE` statement to alter the various ways that a column value can be generated, such as:

- Add the generated expression attribute to an existing non-generated column.
- Drop the generated expression attribute from an existing generated expression column.
- Add the identity attribute to an existing non-identity column.
- Drop the identity attribute from an existing identity column.
- Alter a generated column from `GENERATED ALWAYS` to `GENERATED BY DEFAULT`, or from `GENERATED BY DEFAULT` to `GENERATED ALWAYS`.
- Drop the default attribute from a user-defined default column.

Until this enhancement, a generated expression or identity attributes could be assigned to a column only when created. Although the expression itself could be changed later, after a generated column was created it could not be changed to a non-generated column without dropping and then re-creating the table. Before dropping the table, you had to export the data in the table and then reload that data into the re-created table with a redefined column.

► Toggle generated column property

There are times when the default values of a column in a database table must be changed. The enhanced `ALTER TABLE SQL` statement enables you to set

the default value of a column to a different value. You have new options to change the generated values for a column. You also can alter identity columns and generated expression column.

► Nested savepoints

DB2 UDB now supports the nesting of savepoints. Multiple levels of savepoints can be active simultaneously inside an application and can roll back to any active savepoint as required. A rollback to a particular savepoint statement also releases any active nested savepoints within the savepoint being rolled back. Other new abilities of savepoints are:

- The ability to use savepoint-related statements within atomic compound SQL statements
- The ability to use atomic compound SQL statements within an active savepoint
- The ability to create a new savepoint level within a stored procedure invocation
- The automatic creation of a new savepoint level within each call to a user-defined function (UDF)

1.2.2 Security enhancements

In database security area, the following enhancements have been added:

► Data encryption

Two new authentication types are added to improve user data security during client/server communication over the network:

- `SQL_AUTHENTICATION_DATAENC` requires connections to use the data encryption.
- `SQL_AUTHENTICATION_DATAENC_CMP` allows for a compatibility mode with down-level products that do not support the new authentication type. So they will be allowed to connect with `SERVER_ENCRYPT` and not encrypt user data.

► DB2 security exit

Now you can create your own authentication and group management mechanisms using a DB2-provided plug-in framework to perform user authentication. This new feature presents customers with alternatives to the authentication methods currently provided by DB2. It enables customers to create their own authentication mechanisms to handle:

- Group membership
- Authentication on the client side
- Authentication on the server side

1.2.3 Business intelligence enhancements

In this release, several business intelligence enhancements have been added:

► Online import

The Import utility now supports both offline (ALLOW NO ACCESS) and online (ALLOW WRITE ACCESS) locking modes. The default is the offline mode. Prior to Version 8.1.4, only offline mode was available. The online mode provides better availability of the target table and potential increase in the performance of the import utility.

Online import is especially valuable when range-clustered tables (RCT) are used. Load into RCTs is not supported, so performing multiple concurrent imports into a single target table is the most efficient way to populate the database using supported DB2 utilities.

► Design Advisor

The DB2 Design Advisor wizard can help you significantly improve your workload performance. The task of selecting which indexes, MQTs, clustering dimensions, or partitions to create for a complex workload can be quite daunting. The enhanced DB2 Design Advisor includes the following features:

– Materialized query table advisor

Now you have an advisor to recommend creation (and removal) of materialized query tables (formerly known as ASTs) based on submitted SQL workload. The MQT advisor helps you improve your workload performance.

– Multidimensional clustering advisor

The MDC advisor feature recommends MDC clustering dimensions, including coarsifications on base columns in order to improve workload performance. This includes potentially recommending generated columns that define coarsification of dimensions. The MDC advisor has a goal to select MDC solutions that result in a moderate table expansion.

– Partitioning advisor

The partitioning advisor is an extension of the *db2adv* utility. The advisor takes a workload (consisting of SQL statements) as input and outputs the best partition (minimizing the cost of the workload) for each table in the workload. The tool can be used for the following applications:

- Choose partitioning for MQTs recommended by the MQT Advisor.
- Decide initial database partitioning before loading the data into the database.
- Migrating from a non-partitioned DB2 to partitioned DB2 ESE.

- Migrating to partitioned DB2 ESE applications from Informix or competing database systems.
- Find out the right database partitioning after the environment (workload, underlying data, number of partition groups) is changed.

You can use these features individually or in combination. The Design Advisor generates recommendations for

- New indexes
- New materialized query tables
- Indexes on the materialized query tables
- Conversion to multi-dimensional clustering tables
- Repartitioning of tables
- Deletion of objects unused by the specified workload

► RUNSTATS enhancements

DB2 query optimizer uses the table statistics to select the best access plan for any given query. It is important that statistics remain current to accurately reflect the state of a table at any given time. As the activity against a table increases, so should the frequency of statistics collection. With the increasing size of databases, it is becoming more important to find efficient ways to collect statistics.

Random sampling of table data on which to collect statistics can reduce the amount of time that it takes to collect statistics. For I/O-bound or CPU-bound systems, the performance benefits can be enormous. The smaller the sample, the faster statistics collection completes.

Now in DB2 UDB V8.2, the RUNSTATS command provides the option to collect statistics on a sample of the data in the table by using the TABLESAMPLE option. This feature can increase the efficiency of statistics collection because sampling uses only a subset of the data. At the same time, the sampling methods ensure a high level of accuracy.

1.2.4 Autonomic computing enhancements

The new Configure Automatic Maintenance wizard function can help DBAs automate database maintenance activities. The new enhancements include:

► Automated backup

This feature provides users with a solution to help ensure that their database is backed up both properly and regularly, without either having to worry about when to back up or having any knowledge of the backup command.

The new functions encapsulated in this feature include:

- Backup automation driven by the specified maintenance objectives.

- A new algorithm to identify the need for a backup on a particular database.
 - A new state-based Backup Health Indicator to surface the “backup required” state, when the backup cannot be accomplished as the specified maintenance objectives.
 - The addition of a new Database Configuration Parameter that will be a switch to turn the backup automation on/off (AUTO_DB_BACKUP).
- Self-tuning backup and restore
- DB2 backup and restore commands will automatically choose an optimal value for the number of buffers, the buffer size, and the parallelism settings for both backup and restore operations. The values chosen are based on the amount of memory available, the number of processors available, and the database configuration. The objective is to minimize the amount of time that is required for backup and restore operations to complete. The BACKUP DATABASE and RESTORE DATABASE commands automatically choose an optimal value for the following parameters whenever they are not specified:
- WITH num-buffers BUFFERS
 - PARALLELISM n
 - BUFFER buffer-size
 - For restore-database operations, a multiple of the buffer size used for the backup operation will always be used.
- Including of log files in online backup images
- The architecture of the backup and restore utilities prior to V8.2 required that the backup image and the required log files be shipped as separate objects. This introduced the possibility of losing the log files needed for recovery.
- When creating an online backup image, you can include the log files necessary for restoring and recovering a database in the image to a consistent point in time. This means that if you need to ship backup images to a disaster recovery site, you do not have to send the log files separately or package them together yourself. Further, you do not have to decide which log files are required to guarantee the consistency of an online backup, and you will not be able to delete the log files that pertain to a particular backup image.
- To use this feature, specify the INCLUDE LOGS option of the BACKUP DATABASE command. When you specify this option, the backup utility will truncate the currently active log file and copy the necessary set of log extents into the backup image.

► Integrated and automated log file management

DB2 UDB V8.2 introduces an autonomic log management function. The new log manager enables DB2 to manage the use, archive, retrieval, and eventual deletion of log files with minimal user interaction. The new history file entries:

- Indicate when and where log files are archived.
- Can determine which database backup and exactly which log files are required to restore to any given point in time.

► Simplified memory configuration

There are several disadvantages to configuring database heap sizes as maximum hard limits on memory usage:

- Determining the appropriate maximum size for a particular database heap requires extensive knowledge about how the heap will be used by DB2.
- Setting a limit on the amount of memory that can be allocated to a heap does not guarantee that memory will be available to that heap when it is required.
- The maximum size of a heap represents a hard limit that memory allocation for a heap cannot exceed, even for a short period of time.

To address these issues, V8.2 configures and interprets database shared memory heaps and instance shared memory heaps differently from the prior version. With the new simplified configuration of these heaps, you will now be able to provide a single value for how much memory you want DB2 to use for each active database. The configuration size for a heap now represents the guaranteed minimum, or reserved, size for that heap.

► Automatic RUNSTATS

DB2 optimizer uses catalog statistics to determine the most efficient access plan for any given query. Out-of-date or incomplete statistics for a table or an index could lead the optimizer to select a plan that is not optimal, and as a result slow down query execution. However, deciding which statistics to collect is complex, and keeping these statistics up to date is time-consuming.

With automatic statistics collection, you can let DB2 determine which statistics are required and which should be updated. The new function includes:

- RUNSTATS automation driven by the specified maintenance objectives.
- New database configuration parameter `AUTO_RUNSTATS`, used to control the on and off function.
- A health indicator, used to expose the need for runstats on a table if auto-runstats is disabled or fails for some reason.

With automatic statistics collection enabled, DB2 automatically runs the RUNSTATS utility in the background to ensure that the correct statistics are collected and maintained. This new feature makes process of statistics collection and maintenance completely transparent to users.

► Automated table reorg

Data defragmentation is a table/index maintenance task to utilize the disk space more efficiently and improve performance. In this version of DB2, the automated table reorganization feature is used to manage table and index reorganization without requiring manual intervention.

The new function includes:

- A state-based collection health indicator in the Health Monitor to surface any reorganization actions that cannot be handled by the specified maintenance objectives.
- Notification of the identification of tables requiring manual reorganization.
- Function to control database maintenance automation.
- Knowledge in DB2 to automatically identify the need for reorganization on a table.
- Knowledge in DB2 to determine the type of reorganization required and when to run it.
- Scheduling of reorganization actions based on automation specifications.

► Schema manipulation – alter table

DB2 UDB V8.2 goes beyond the functionality of the ALTER TABLE statement by offering the following new features:

- Renaming columns
- Removing columns
- Altering column type and transforming existing data using SQL scalar functions
- Increasing or decreasing column size
- Changing column default value
- Changing column from NOT NULL to NULLABLE
- Changing precision and scale for decimal

► Health Center Recommendation Advisor

The Recommendation Advisor helps you to resolve health alerts on DB2 objects by guiding you to take necessary actions to correct the issue that is

causing the health alert. The Recommendation Advisor answers the following questions:

- What is the health indicator on which the alert was raised?
- What is the alert condition/severity?
- Why is this health condition important to me?
- What are the possible courses of actions (recommendations) to resolve this problem?
- What is actually going to be done based on the recommendation I selected? Are there any additional steps I must take?

1.2.5 Usability enhancements

New usability enhancements provided in Version 8.2 include:

► **Novice Control Center navigation**

Now DB2 UDB 8.2 Control Center is available in three different views:

- | | |
|-----------------|---|
| Basic | This view provides core DB2 UDB V8.2 functionality, which includes the essential objects such as databases, tables, and stored procedures. This feature is designed to benefit developers from the aspect of providing easier and more efficient Control Center navigation. |
| Advanced | This view displays all objects and actions available in the Control Center. This is the view that you should select if you are working in an enterprise environment and want to connect to DB2 for z/OS or IMS™. |
| Custom | This view gives you the ability to tailor the object tree and the object actions to your specific needs. |

► **Throttle utilities supported by Control Center**

The supported throttle utilities include backup, rebalance, reorg, load, and runstats. This new feature:

- Enables DBA to reduce impact of running resource-intensive utilities on operational workload, such as backup, rebalance, or runstats.
- Supports setting and display for execute priority for the supported utilities.
- Supports new option to set priority in utility dialogs.
- Supports new option to set priority in SHOW COMMAND.
- Displays priority of executing utilities, and the current priority set for the session.

1.2.6 High availability enhancements

The following enhancements have been added:

- High availability disaster recovery (HADR)

DB2 UDB High Availability Disaster Recovery (HADR) is a data replication feature that provides a high availability solution for both partial failure (hardware, network, DB2, or operating system failure) or complete site failure (fire, earthquake, flood, or any failure that causes the entire site to be destroyed). HADR protects DB2 against data loss by replicating data changes from a source database, called the primary, to a target database, called the standby. Without HADR, the length of time it takes to solve the failure is unpredictable. It can take several minutes or hours before the failure is solved and the DBMS is available.

HADR enables failover and fallback between the two systems. The standby database can take over as the primary database with full DB2 functionality. After the failed old primary is repaired, it can rejoin the HADR pair as a standby database if the two copies of the database can be made consistent. After the original primary database is reintegrated into the HADR pair as the standby database, a fallback operation can be performed so that the original primary database is once again the primary database.

HADR requires the same hardware, OS, and DB2 software on the two systems (except for some minor differences during rolling upgrade).

- Client reroute

The automatic client reroute feature enables client applications to recover from a loss of communication with the server in order to continue working with minimal interruption. After a loss of communication occurs, the client application attempts to reconnect to the server, but if this fails, the client is then rerouted to a different server. So when the connection is re-established, the application receives an error that informs it of the transaction failure, but the application can continue with the next transaction.

The main goal of this feature is to enable a DB2 UDB client application to recover from a loss of communications so that the application can continue its work with minimal interruption. As the name implies, rerouting is central to the support of continuous operations. But rerouting is only possible when there is an alternate location that is identified to the client connection.

Automatic client reroute works in conjunction with HADR to enable a client application to continue its work with minimal interruption after a failover of the database being accessed.

► Set up HADR wizard

This wizard facilitates the process to set up and configure primary and standby databases for high availability disaster recovery. It guides the user to perform the following tasks:

- Identify the HADR pair.
- Prepare the primary database for log shipping.
- Perform database backup.
- Copy backup image to secondary server.
- Perform restore on the selected standby database.
- Move any database objects not included in the backup image.
- Update service files.
- Update HDR-related configuration parameters on both databases.
- Provide an option to Start HDR.

► HADR action windows

Use this tool to configure and to check the status of your HADR system. This window will notify you if there are any problems with the HADR configuration. From this window, you can perform the following tasks:

- Refresh the status information.
- Start HADR on one or both of the databases.
- Stop HADR on one or both of the databases.
- Initialize a takeover operation.
- Check or alter database configuration parameters related to HADR.
- Catalog an HADR database (if required).

The GUI enhances the CLP by enabling the current role and state of a database to be used to determine which commands and options are valid before submitting the command to the engine.

► Q replication architecture

Q replication enables you to replicate committed transactional data from DB2 UDB sources to targets by using two programs: Q Capture and Q Apply. The Q Capture program runs on the source system; it reads DB2 recovery logs for changed source data and writes the changes to WebSphere MQ queues.

The Q Apply program runs on the target system. It retrieves captured changes from queues and writes the changes to targets. Both programs use a set of DB2 tables to track the information that they require to do their tasks and to store information that they generate, such as information that you can

use to find out how well they are performing. You create these tables before you tell Q replication what your replication sources and targets are.

The features in Q replication architecture are:

- Each message representing a transaction
- XML format option for publishing
- Highly parallel apply process
- Differentiated conflict detection and resolution
- Staged availability of heterogeneous support

1.3 DB2 UDB V8.2 integration with Windows

When providing DB2 on a platform, IBM seeks to offer more than just another port. Wherever possible, IBM customizes DB2 to exploit the capabilities of a platform. In this section, we discuss the new capabilities of DB2 UDB V8.2 specific to the Microsoft Windows environment.

IBM DB2 UDB has earned 17 Windows Server 2003 certifications across the Standard, Enterprise, and Datacenter editions—even more certifications than SQL Server. DB2 was also the first database to qualify for the Microsoft .NET Connected logo to certify that DB2-based Web Services can easily integrate with .NET connected solutions.

1.3.1 System security

DB2 UDB V8.2 introduces the following enhancements to system security on the Windows platform:

► Windows domain handling and user IDs

Improved Windows Domain handling support includes:

- Support using cached credentials for DB2 authentication and group lookup
- Support nested group semantics
- Support domain local group
- Support implicit trusts between domains

The user ID enhancements include:

- Accepting additional special characters, including &, -, and blank, in user IDs and security mechanism group names (and consequently in authorization names and authorization IDs).
- Accepting security mechanism group names longer than eight characters

- Allowing two-part names on CONNECT and ATTACH that contain a Windows domain name and the user ID to avoid the network traffic associated with looking up the user name in the trusted domain forest.
 - Enhanced support for Active Directory domains has been enhanced (for example, support for implicit trusts between domains, domain local groups, and nested global groups)
- Local System Accounts
- Support for Windows local system account (LSA) is provided for both the various DB2 services, as well as for applications that access them. The DB2 installation process enables DB2 services to run under SYSTEM (local system account) as an alternative to running under a dedicated user account. You can also perform the DB2 installation itself from a process that is running under the context of the LSA.
- This feature makes it easier to deploy DB2 by avoiding, where appropriate, the complexities associated with handling of user IDs and passwords. Another enhancement is the ability of DB2 applications that are running in the context of the LSA to access and use the local DB2 server.

1.3.2 Application development

By using DB2 UDB V8.2, the large community of Visual Studio .NET developers will benefit from the advancements all through the context of their natural development environment. DB2 UDB V8.2 provides tools that take advantage of the latest features of Microsoft .NET to DB2 users even before SQL Server customers. Following are enhancements added in DB2 UDB V8.2:

- DB2 development add-in for Visual Studio .NET 2003
- Version 8.2 of DB2 UDB has extended DB2 family support on:
- DB2 UDB for Linux, UNIX, and Windows
 - DB2 for OS/390 and z/OS V6, V7, and V8
 - DB2 for iSeries V5.2 and V5.3
- You can now develop a DB2 stored procedure using Common Language Runtime (CLR) from existing methods in a .NET-managed language, such as C# and Visual Basic. You also can perform source-level debugging of SQL procedures using the DB2 database project. Other new features include:
- Enhanced data connections support
 - Enhanced SQL text box control
 - Simple SQL parser
 - Improved managed provider tooling support

► DB2 administration add-in for Visual Studio .NET 2003

The DB2 administration add-in provides developers a convenient way to create DB2 object via Visual Studio .NET 2003. This feature extends the IBM Explorer in Visual Studio .NET 2003 to include:

- Adding DB2 remote data connections that are not cataloged on the local client
- Adding a “create new” feature for supported DB2 objects from the data connection folder directly through the use of existing designers or wizards as applicable
- Adding a “generate DDL” feature for supported DB2 objects from the data connection folder into a new script file or through drag and drop onto a script file opened by the DB2 SQL editor.

► Microsoft .NET CLR stored procedures

DB2 UDB V8.2 now enables you to create stored procedures and UDFs using any .NET CLR-compatible programming language, including C#, Visual Basic, managed C++, and all other CLR compatible languages.

You can create CLR routines as other external (non-SQL) routines are created: by executing a CREATE statement that associates a database routine signature with a .NET assembly residing on the database server. You can use the routines to encapsulate commonly used database operations and logic, to extend the functionality of SQL, and to improve the performance of client applications.

► DB2 .NET Data Provider additional support

The DB2 .NET Data Provider is an extension of the ADO.NET interface that enables .NET applications to access a DB2 database through a secure connection, execute commands, and retrieve results. Version 8.2 features the following improvements to the DB2 .NET Data Provider:

- Faster performance
- Support for the Microsoft .NET Framework V1.1 and Visual Studio 7.1
- A new ConnectionString keyword to specify the isolation level
- New DB2Connection properties that enable DBAs to attribute workload to a particular source
- Access for .NET applications to the following database management systems through the DB2 .NET Data Provider:
 - DB2 UDB Version 5, Release 1 (or later) for AS/400® and iSeries, through DB2 Connect
 - DB2 UDB Version 7.3 (or later) for VSE and VM, through DB2 Connect

► DB2 Web Services Support

DB2 UDB V8.2 is a release for application developers with significantly better support for Web services and seamless integration with Microsoft .NET development environment. The DB2 tools add-ins extend Microsoft Visual Studio .NET to support WOLF (Web Services Object Runtime Framework) Web services including:

- Add wizard to deploy an embedded WebSphere Application Server Web Service for DB2 SQL statements and stored procedures.
- Add wizard to generate C# Web methods using ADO.NET for DB2 SQL statements and stored procedures.

Installation and migration

In this chapter we describe the basic procedures of installing DB2 UDB V8.2 on Windows Server 2003. We also provide steps for various migration scenarios.

This chapter contains the following sections:

- ▶ DB2 UDB V8.2 installation
 - Installation overview for DB2 on Windows
 - DB2 UDB V8.2 server installation
 - DB2 UDB V8.2 client installation
- ▶ Migration
 - Migrating DB2 between versions of Windows
 - Migrating between versions of DB2 UDB
 - Migrating from 32-bit to 64-bit
 - Migrating everything at once

2.1 DB2 UDB V8.2 installation

In this section, we discuss what preparation steps are necessary before you begin the installation of 32-bit DB2 UDB V8.2 on Windows 2003. Before you install, prepare your computer for installation as follows:

- ▶ Verify that your computer meets the necessary installation requirements.
- ▶ Ensure that your system has enough memory.
- ▶ Ensure that your system has enough disk space.
- ▶ Ensure that you have the necessary user accounts and authority for installation and setup. You require one user account for installation and two user accounts for setup. The user accounts required for setup can be created before you install or you can let the DB2 Setup wizard create them for you. The user that you install DB2 UDB with must either be the administrator on the local system or domain, or have administrator security access.

2.1.1 Installation overview for DB2 on Windows

The installation methods that are available for DB2 UDB V8.2 on Windows are:

- ▶ DB2 Setup wizard

A GUI installer available on Windows operating systems, it provides an easy interface for installing DB2 UDB and performing initial setup and configuration tasks. It also can be used to create instances and response files.

DB2 Setup Wizard has the following parameters:

/?	Generates help message
/f	Force DB2 services to stop
/i xx	Install with preferred language, where xx means the language
/l	Full path and file name of log file
/u	Full path and file name of response file
/t	Creates a file with install trace information

- ▶ Response file installation

An ASCII file that contains setup and configuration values. The file is passed to the DB2 setup program, and the installation is performed according to the values that have been specified. Make sure that all DB2 processes have been shut down gracefully, using a DB2 command such as DB2STOP, to prevent data loss. Response files can contain the following information to do automatic installation and configuration:

- DB2 installation type

- DB2 products path
- DB2 accounts information
- Global DB2 registry variables
- Instance variables
- Instance database manager configuration settings

Here is a sample of a DB2 Enterprise Server Edition (ESE) response file provided with the DB2 ESE Install CD:

```
*DB2_USE_JDK12           = BLANK, YES or NO
*DB2_VI_ENABLE           = BLANK, ON or OFF
*DB2_VI_DEVICE           = BLANK or char()
*DB2_VI_VIPL             = BLANK or char()

* General information for instance to be created
* -----
INSTANCE                  = DB2
DEFAULT_INSTANCE          = DB2
DB2.NAME = DB2
```

Comments are made by placing either a * or a # at the start of a line, or by placing ** or ## after the start of a line to comment out the rest of the line.

Do not uncomment selected components (the COMP keywords) unless you change the INSTALL_TYPE to CUSTOM.

If you specify a default instance (DEFAULT_INSTANCE) in the response file, the USERNAME and PASSWORD keywords for the instance will be used for all other USERNAME and PASSWORD values unless they are explicitly specified. Some of the variables that would be applied are:

- DAS_USERNAME
- DAS_PASSWORD
- MD_DB.USERNAME
- MD_DB.PASSWORD

For descriptions of DB2 registry variables and configuration parameters, see *Administration Guide: Performance*, SC09-4821.

For more information about how to work with response files, refer to *DB2 Installation and Configuration Supplement*, GC09-4837.

Important: Response files should be handled confidentially, because they contain the user name and password in plain text.

There are three ways to create a response file:

- Using the response file generator

You can create a response file that replicates an existing installation. For example, you might install a DB2 client, fully configure the client, then generate a response file to replicate the installation and configuration of the client to other computers.

- Customizing sample response files

You can manually create response files as a plain text file. DB2 installation CDs provide a sample of each product installation on server and client machines that can be used as a base for more elaborate installation procedures. They are on `db2/platforms/samples` path, where *platform* refers to the platform that you are currently working with. In our case it is *Windows*. These response files with default entries are available on the respective CD:

db2ese.rsp	DB2 Enterprise Server Edition
db2wse.rsp	DB2 Workgroup Server Edition
db2pe.rsp	DB2 Personal Edition
db2conee.rsp	DB2 Connect Enterprise Edition
db2conpe.rsp	DB2 Connect Personal Edition
db2adcl.rsp	DB2 Application Development Client
db2admcl.rsp	DB2 Administration Client
db2rtcl.rsp	DB2 Run-Time Client

- Using the DB2 Setup wizard

You can create a response file for Windows installations. The selections that you make as you proceed through the DB2 Setup wizard are recorded in a response file that you can save to a location on your system.

Select the **Save your settings in a response file** check box as shown in Figure 2-1 on page 31.

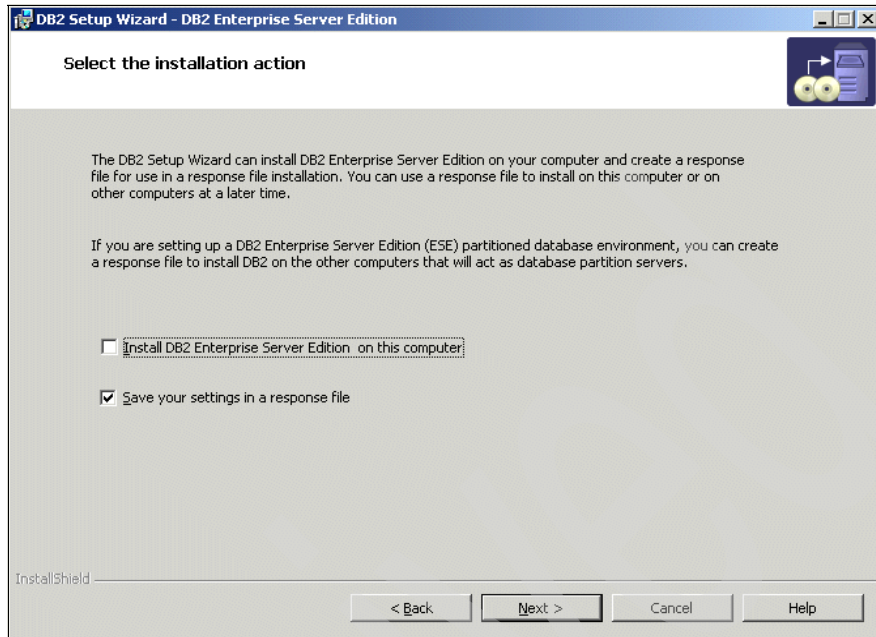


Figure 2-1 Generating a response file through DB2 Setup wizard

Tip: Disabling product installation at the generate response file dialog in the DB2 Setup wizard is an easy way to generate the script with all options selected.

- Reverse-engineer a current installation
Use the command **db2rspgn** to reverse-engineer a specific installation. See *DB2 Installation and Configuration Supplement Version 8*, GC009-4837, for more details.

2.1.2 DB2 UDB V8.2 server installation

The procedure to install DB2 Enterprise Server Edition or Workgroup Server Edition in a single-partition database environment on Windows using DB2 Setup wizard and response file is provided in this section.

We used the following hardware and software in our lab to install DB2 using the wizard:

- IBM xSeries 330 with 3.68 GB RAM memory and 16.9 GB HDD
- Windows 2003 Standard Edition

Using the DB2 setup wizard

In this section, we discuss the basic DB2 UDB installation procedure using the DB2 setup wizard. This is a one-step installation to install DB2 UDB V8.2 server, Administration Client, and Application Development client.

Note: On some dialogs on the DB2 Setup wizard, when user ID and password are required to be validated, DB2 will search locally for the user. To validate domain user accounts, add the domain name plus a slash before user name. Example: On Data Warehouse authentication dialogs, enter UDBRED\db2admin.

The step-by-step procedure to perform the installation for DB2 Server (ESE or WSE) is as follows:

1. Place the DB2 Server (ESE or WSE) Install CD in the drive and wait for the setup initial dialog. If the dialog is not shown, or the install files are on the network, search and run **setup.exe** at the directory of db2 installation. On the IBM DB2 Setup Launchpad dialog, click **Install Products**, as in Figure 2-2.

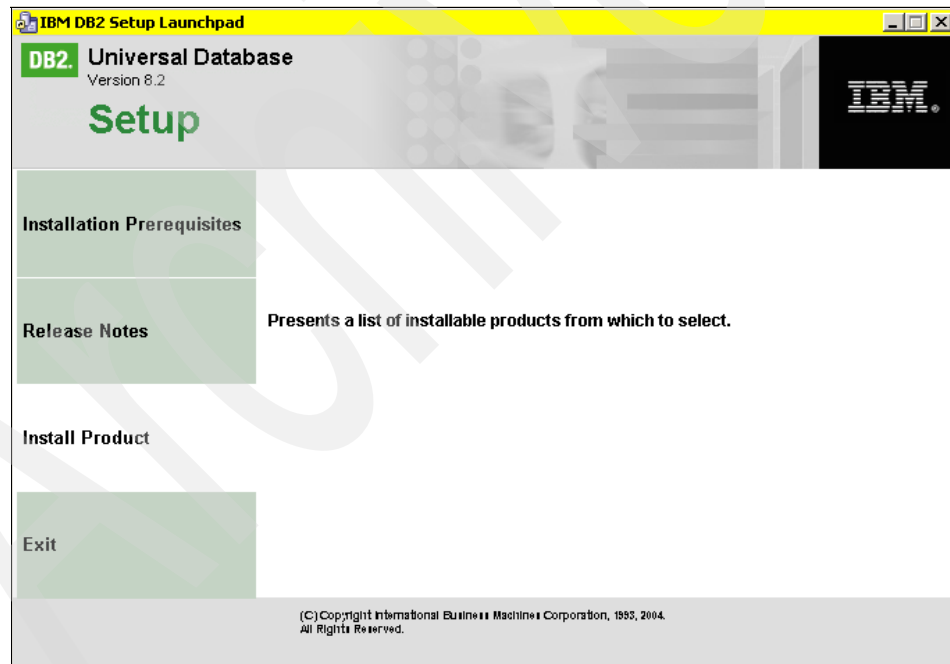


Figure 2-2 DB2 Setup Launchpad

2. Make sure that the **DB2 UDB Enterprise Server Edition** option is selected and click **Next**.

3. On the Welcome to the DB2 Setup wizard dialog, click **Next**.
4. After you read the license agreement on the License Agreement dialog, choose **I accept the terms of the license agreement** option and click **Next**.
5. The Select the installation type dialog opens, asking for the desired type of installation. We continue with the **Custom** selection in order to select the features we wanted to install and specify configuration options. We want to install DB2 Development Client at the same time the server is installed. If you do not plan to install the Development Client on the same machine as the server, select Typical.

Typical	This option installs the visual studio add-ins and the stored procedure builder, plus the administration tools to navigate through DB2 servers and objects, and the client features that are used most often, including all required features, ADO, OLEDB, ODBC support, Control Center and CCA. Also, it configures with default values. Select the Data warehousing option if you need data warehousing tools.
Compact	This option installs only the basic development features and client features to query and connect to DB2 servers.
Custom	This option lets you select the required client features.

Note: DB2 requires the English language to work. A Typical installation includes English plus the language defined locally. If you need support on another language, choose the Custom installation type.

6. The Select the installation action dialog opens. Make sure that **Install DB2 Enterprise Server edition on this computer** is selected. If you want to generate a script with the installation procedure in a response file, select **Save your settings in a response file** option. Click **Next**.
7. In the Select the features you want to install dialog, you can customize the options to be installed based on your type of environment (production or development) and your type of application (OLTP, OLAP, Web). The Client Support, Server Support, and Administration Tools options must be selected. If you need other options later, rerun the DB2 Setup wizard again. You can also select the installation folder (drive and directory) where you want DB2 to reside. By default, the C: drive and the c:\Program Files\IBM\SQLLIB directory will be used. Click **Next** to proceed to the next step.
8. The Select languages to install dialog opens. By default English is selected, but you can add one or more languages to support the user interface and product messages. (Note that multiple languages increase the disk space requirements.) If you need support in another language, click the **Available Languages** list to select the language and click > to add it to the Selected

Languages list. This list is sorted alphabetically. You also can select the installation folder (drive and directory). Proceed by clicking **Next**.

9. The Specify the location of DB2 Information Center dialog opens. Use the default **On the IBM Web site** unless you do not want to install Information Center or if the Information Center will be installed in different server.
10. The Set user information for DB2 Administration Server dialog is shown. DB2 Setup Wizard now asks for a user account that will start the DB2 administration service (DAS). Enter the domain, user name, and password, and confirm the previously defined password. The db2admin ID is recommended. Also, leave the **Use the same user and password for the remaining DB2 services** check box selected if you want to do so. Click **Next**.
11. In the Set up the administration contact list dialog, designate where the contact list should be placed. In our case, it will be created locally. Select **Local - Create a contact list on this system** and proceed by clicking **Next**.
12. A warning is displayed advising that there is no SMTP server specified to send notifications. Click **OK**. We will set it after the installation procedure.
13. The Create a DB2 instance dialog is shown. We chose the default to have the Setup wizard create the instance for us. You can create a DB2 instance later by using the **db2icrt** command.
14. In the next window, configure the DB2 instance and Control Server instance in order to set up the communications protocols and start-up options of this instance. By default only TCP/IP is configured, but you can add more communications protocols, depending on your environment. Click **Next**.
15. In the Select the metadata you want to prepare dialog, you can request that DB2 prepare a tools catalog and warehouse control database metadata. In order to use certain DB2 tools such as Task Center and Scheduler, you must create the DB2 tools catalog that contains task metadata. Select **Prepare the DB2 tools catalog** and proceed by clicking **Next**.
16. In the Specify the schemas for your metadata dialog, we use the default tools schema **SYSTOOLS**.
17. In the Specify a contact for health monitor notification dialog, we select **Defer the task until after installation is complete** and proceed by clicking **Next**.
18. The Enable operating system security for DB2 objects dialog is shown. Select **Enable operating system security** and proceed by clicking **Next**.
19. The Start copying files dialog opens. Select **Install** to start the file copy. At the end of the installation process, click **Finish**. Depending on the system configuration, it could be necessary to reboot the machine. If the wizard detects anything requiring a reboot, it will ask you to do so.

For information about errors encountered during installation, see the db2.log file, which stores general information and error messages resulting from the

install and uninstall activities. By default, the db2.log file is located in the `\MyDocuments\DB2LOG` directory. The location of the *MyDocuments* directory depends on the settings on your computer.

Using the DB2 setup response file

In this section, we provide a step-by-step procedure for response file installation. As mentioned before, for more information, refer to *Quick Beginnings for DB2 Servers Version 8*, GC09-4836.

Here are the steps you need to follow to generate a response file script:

1. Execute steps 1 to 6 of “Using the DB2 setup wizard” on page 32. In step 5 on page 33, choose carefully between the Typical and Custom options, because the install profile is generated differently between these options. In our example, we proceed with the **Custom** option.
2. Check **Save your settings in response file** and uncheck the **Install DB2 Enterprise Server Edition on this computer** check box. Proceed to the next step by clicking **Next**.
3. Continue executing the steps from 7 to 18. After step 18 on page 34, the DB2 Setup wizard displays a text box where you can enter the path and the file name of the script. We proceeded by entering `c:\db2_prod_ese.rsp` and then clicked **Finish**.

DB2 Setup wizard generates only the necessary parameters in the response file. If you need to add more parameters, we recommend using the response files provided on the DB2 Install CDs and modify them accordingly.

Follow these steps to perform the installation using the response file script:

1. Place the DB2 ESE Install CD in the drive if DB2 install files are not available on the network. Open a command prompt session and execute:

```
start /wait [path1]\setup.exe /u [path2]\db2_prod_ese.rsp /!
[path2]\db2_prod_ese.log /t [path2]\db2_prod_ese.trc.
```

PATH1 is the root directory of DB2 installation where setup.exe resides. PATH2 is the path where the response file is and where the log files will be placed. When done this way, the shell waits for the process to complete, enabling you to add this command to a batch file and handle the return code in an adequate fashion. The default successful return code for DB2 Setup Wizard is 0.

2. During the install process, DB2 Setup wizard generates a temporary log file, db2wi.log. You can verify that DB2 Setup wizard is working if this file grows. If something fails, open it and go to the last few lines to see what happened. After the return of the prompt, we recommend that you open it to check for message codes or warnings. Remove it if you wish.

2.1.3 DB2 UDB V8.2 client installation

In this section, we discuss the basic installation procedure for each DB2 UDB client. The client products for DB2 UDB V8.2 on Windows are:

- ▶ DB2 UDB Application Development Client
- ▶ DB2 UDB Administration Client
- ▶ DB2 UDB Run Time Client
- ▶ DB2 UDB Run Time Client Lite

The steps for installing these clients using the DB2 Setup wizard are mostly identical. DB2 UDB Run-Time Client Lite is the new addition to the DB2 UDB client product, which has a fewer installation steps. We provide the step-by-step installation procedure for both Run-Time Client Lite and Application Development Client. You can use the Application Development Client installation procedure to install the Administration Client and Run-Time Client.

In our lab, we used the following hardware and software (environment) to install DB2 UDB client using the Wizard:

- ▶ IBM NetVista™ with 1.00 GB RAM memory and 37.2 GB HDD
- ▶ Windows Server 2003

Installing Run-Time Lite Client

In this section we provide a step-by-step procedure for installing Run-Time Lite Client using the DB2 UDB Setup wizard. Follow these steps:

1. Place the DB2 Run-Time Client Install CD (RTCL) in the drive, search for and run **setup.exe** at the directory of DB2 installation. On the Choose Setup Language dialog (Figure 2-3), select the language you need. The default is English (United States). Click **OK**.

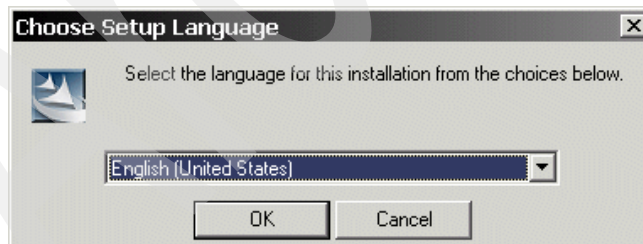


Figure 2-3 DB2 Run-Time Lite Client: Choose Setup Language

2. On the Welcome to the DB2 Setup wizard dialog (Figure 2-4 on page 37), click **Next**.

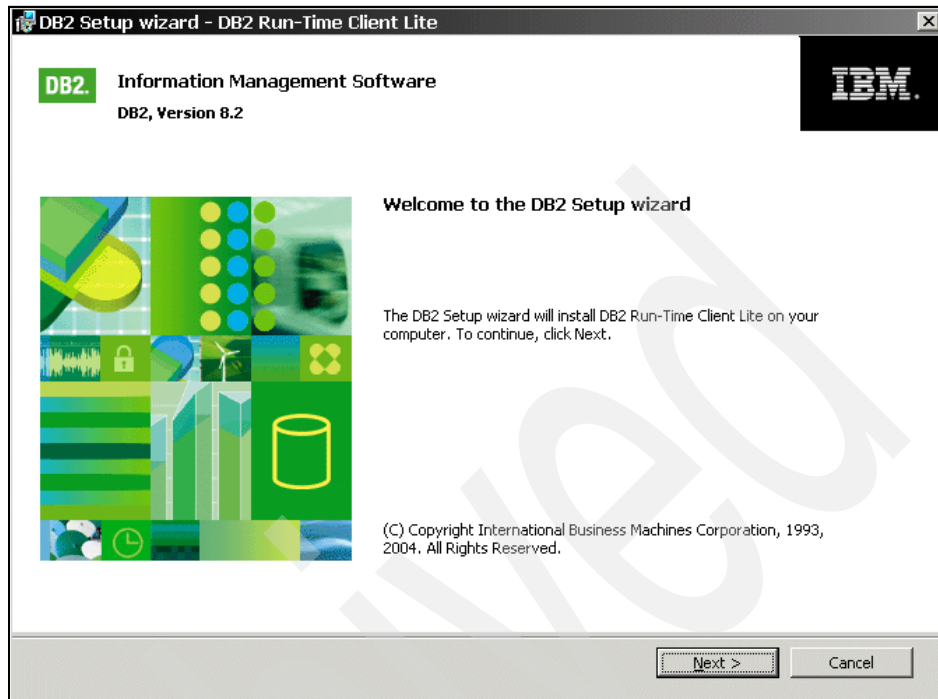


Figure 2-4 DB2 Run-Time Lite Client Setup Welcome

3. Read the license agreement on the **License Agreement** dialog, choose **I accept the terms of the license agreement** and click **Next**.

4. The Select the installation type dialog opens, requesting the type of the installation. There are three types of installations:

Typical This option installs the client features that are used most often, including all required features, ADO, OLEDB, and ODBC support. Also, it will configure with default values.

Compact This option installs only required client features and basic libraries to connect to DB2 servers.

Custom This option lets you select the required client features.

Now we continue the setup using the Custom option. Click **Custom** and click **Next** to proceed to the next step.

If you choose Typical installation, the **Start copying files** window opens as described in step 6.

5. The **Custom setup** dialog is shown. Customize the options to be installed based on your type of environment (Production or Development) and the type of application (OLTP, OLAP, Web). But the **Client Support** option must be

checked. If you need other options later, rerun the DB2 Setup Wizard. You can also select the installation folder (drive and directory). By default, the C: drive and the c:\Program Files\IBM\SQLLIB directory are used. Click **Next**.

6. The Start copying files window opens. Select **Install** to start the file copy. At the end of the installation process, click **Finish**. Depending on your system configuration, it could be necessary to reboot the machine. If DB2 Installation Wizard detects anything requiring a reboot, it will ask you to do so.

The db2.log file stores general information and error messages resulting from the install and uninstall activities. By default, this file is located in the `\MyDocuments\DB2LOG` directory. The location of the *Documents* directory depends on the settings on your computer.

Application Development Client installation

In this section, we provide a step-by-step procedure for installing Application Development Client using the DB2 Setup wizard. The Application Development Client is composed of the Administration Client and the add-ins for Visual Studio and the Development Center. This client enables you to explore the benefits of .NET. For more information, refer to *Quick Beginnings for DB2 Clients Version 8*, GC09-4832.

Here are the steps for installing this product:

1. Place the DB2 Application Development Client Install CD (ADCL) in the drive and wait for the setup initial dialog. If the dialog is not shown, or the install files are on the network, search for and run **setup.exe** at the directory of db2 installation. On the IBM DB2 Setup Launchpad dialog (Figure 2-5 on page 39), click the **Install Products** tab.

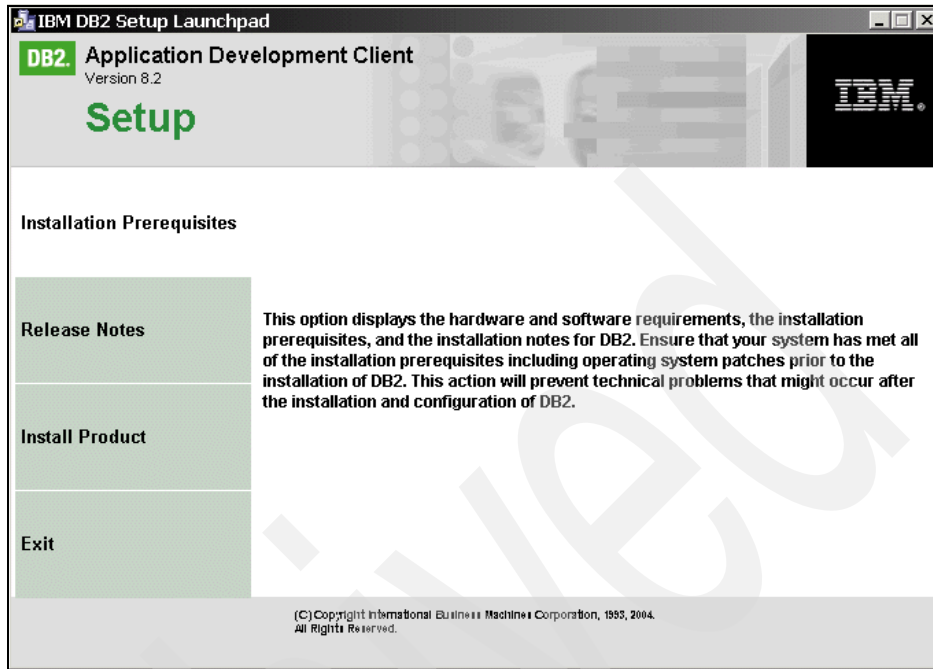


Figure 2-5 DB2 Development Client Setup Launchpad

2. In the Select the product you want to install dialog, make sure that the **DB2 UDB Application Development Client** option is selected and click **Next**.
3. Click **Next**.
4. After you read the license agreement, choose **I accept the terms of the license agreement** and click **Next**.
5. In the Select the installation type dialog, designate the type of installation:

Typical This option installs the Visual Studio add-ins and the stored procedure builder, plus the administration tools to navigate through DB2 servers and objects, and the client features that are used most often (including all required features, ADO, OLEDB, ODBC support, Control Center, and CCA). Also, it will configure with default values. Select **Data warehousing** if you need data warehousing tools.

Compact This option installs only the basic development features and client features to query and connect to DB2 servers.

Custom This option lets you select the required client features.

We continue with **Custom** selection in order to select the features we wanted to install and specify configuration options.

6. In the Select the installation action dialog, check **Save your settings in a response file** if you want to generate a script with the installation procedure in a response file. Click **Next**.
7. In the Select the features you want to install dialog, customize the options to be installed, If you need other options later, rerun the DB2 Setup wizard. You also can select the installation folder where you want to install the client. By default, the installation location is the C: drive and the directory is c:\Program Files\IBM\SQLLIB. Click **Next**.
8. The Select languages to install dialog opens. By default, English is selected, but you can add one or more languages to support the user interface and product messages. (Note that multiple languages increase the disk space requirements.) If you need support in another language, click the **Available Languages** list to select the language and click > to add it to the Selected Languages list. This list is sorted alphabetically. You also can select the installation folder (drive and directory). Proceed by clicking **Next**.
9. In the Specify the location of DB2 Information Center dialog, designate a location from which you will access the DB2 Information Center. We use the default: On the IBM Web site. Click **Next**.
10. In the Configure NetBIOS dialog, the wizard now asks if you want to connect to DB2 server using the NetBIOS protocol. To configure it, select **Configure NetBIOS for connections to DB2 servers** and enter the workstation name and adaptor number. Click **Next**.
11. The Enable operating system security for DB2 objects dialog opens. Select **Enable operating system security** and proceed by clicking **Next**.
12. In the Start copying files dialog, select **Install** to start the file copy. At the end of the installation process, click **Finish**. Depending on your system configuration, it could be necessary to reboot the machine. If the DB2 Installation Wizard detects anything requiring a reboot, it will ask you to do so.

The db2.log file, which stores general information and error messages resulting from install and uninstall activities, is located by default in the *MyDocuments\DB2LOG* directory. The location of this directory depends on the settings on your computer.

2.2 Migration

The migration to Windows Server 2003 can be a complex process because most likely you are not only moving from Windows 2000 Server to Windows Server 2003 and 32-bit to 64-bit, but also from DB2 UDB V7.2 to DB2 UDB V8.2. The migration process should be treated as two separate components: Upgrading

versions of Windows and migrating versions of DB2 UDB. In this section, we discuss different migration scenarios.

2.2.1 Migrating DB2 between versions of Windows

Windows Server 2003 offers a great deal of new functionality. One of the greatest selling points for migrating to Windows Server 2003 is its ability to make use of the 64-bit architecture for a Windows system. This section covers what is required to migrate DB2 UDB between different releases of Windows.

Migrating DB2 from Windows 2000 to Windows 2003

The migration between different releases of the operating system can be a complex task because it often involves moving to a new system or cleaning out the contents of the C: drive. Several options are available, depending on which version of DB2 UDB you are using. The following options apply only if the DB2 UDB version remains the same. The scenarios in which the version of DB2 UDB has changed, such as from V7.2 to V8.2, are covered in the next section.

Using DB2 Backup and Restore

This migration is an easy process because the DB2 backup utility enables you to take a backup on one Windows server and restore to another Windows server. The backup image is OS-version-independent, so restoring to Windows Server 2003 from Windows 2000 Server, or vice versa, will work without any problems.

To use the DB2 Backup and Restore function to migrate a DB2 database from Windows 2000 Server to Windows Server 2003:

1. Install the same release of DB2 UDB on the new target Windows system.
2. Take the database offline on the original system.
3. Take a full offline backup of the database. Only offline backups can be used to restore to a newer version of DB2 UDB, as replaying of downlevel log records is not supported.
4. Copy the database backup image to the target system.
5. Start DB2 on the target system.
6. Restore the database. If the file systems that will contain the DB2 database are not identical on both systems, then a redirected restore will be required.
7. Check that the database was restored properly by checking on any restore error messages and items in the db2diag.log file.
8. Test a connection to the database on the target system.
9. Recatalog the database on your client systems.

Figure 2-6 illustrates the restrictions on which images can be restored from and to combinations of Windows, 32-bit and 64-bit, and releases of DB2 UDB.

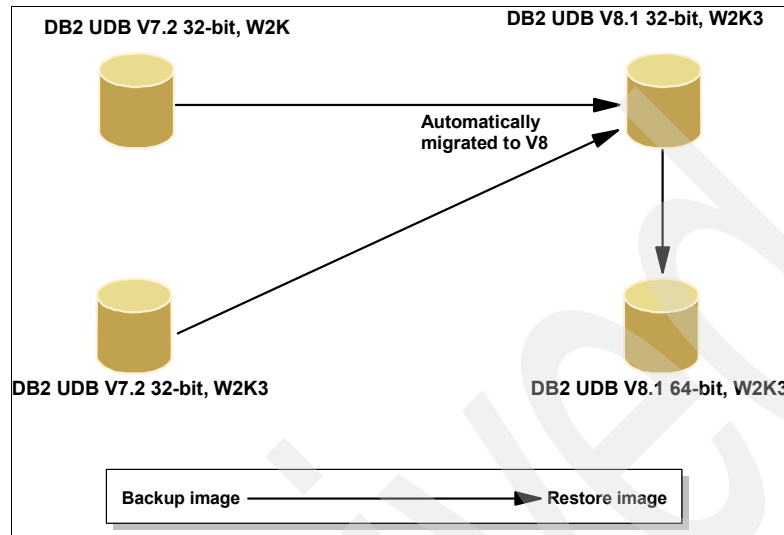


Figure 2-6 Backup and restore combinations with DB2 UDB V8.1 for Windows

Note: It is possible to take a backup image on DB2 UDB V7 and restore it on a different version of Windows on DB2 UDB V8.1 or Version 8.2. The new releases of Windows must be of the same bit level, so you cannot move from Windows 2000 32-bit to Windows Server 2003 64-bit.

If you try and restore from an unacceptable combination, you will receive the following message:

```
SQL2570N  A database cannot be restored on a platform that does not match
the platform on which the backup image was created.
```

Moving disks to a new system

Sometimes the same external set of disks, be it in an array or a disk system, will be switched over to the new system. There are two distinct sets of steps to take, depending on whether the disk tables have changed.

- **Scenario 1:** The disk assignments have not changed.

If the disks will be laid out and labeled the same way on the new system, no work is needed. After you install DB2 UDB you can catalog the new database and DB2 will be able to use your old database. This option is sometimes

referred to as “plug & play.” In order to use the plug & play approach, you must ensure the following:

- The physical disk arrangement, file paths, and drive letters are preserved when existing disks are plugged into the target system.
- If raw disks are used for containers or a database log, they are referenced either as logical raw partitions (for example, `\\.\D:` on Windows) or as physical drives.

Note: On Windows, this method is not supported for raw devices referenced by globally unique identifiers (GUID) that are unique to each server.

- Instance names are the same on both systems.
- The number of database partitions must be the same on both source and target systems.

► Plug & play steps

Follow these steps to use this method:

- a. Install DB2 UDB on the target server with the same version and fix pack as on the original server.
- b. Stop DB2 UDB on the original server.
- c. Unplug the disks (or disk subsystem) from the source machine in a manner approved by the operating system. On Windows, you should disable the disk using the Device Manager utility before unplugging it in order to prevent system crash or data loss.
- d. Enable the disks (or disk subsystem) on the target machine in an approved manner. On Windows, you should use the Disk Management tool to rescan disks.
- e. Verify that all disks are online and drive letters are identical on both systems.
- f. Make the database available to the database manager on the target server (for example, by using the `db2 catalog` command).
- g. Proceed through the normal performance tuning tasks to ensure that DB2 UDB meets the performance expectations on the target server.

Note: In order to use this approach you must be aware of the following:

- ▶ The physical disk arrangement, file paths, and drive letters are preserved when existing disks are plugged into the target system.
- ▶ If raw disks are used for containers or the database log, they are referenced either as logical raw partitions (for example, \\.\D: on Windows) or as physical drives. Note that on Windows this method is not supported for raw devices referenced by globally unique identifiers, which are unique to each server.
- ▶ Instance names are the same on both systems.
- ▶ The number of database partitions must be the same on both source and target systems.

- ▶ Scenario 2: The disks assignments have changed.

Sometimes the DBA may not want to back up and restore the database because all of the database information is stored on a separate disk system, which can be unplugged from the old OS release system and connected to the new Windows 2003 server machine. The DB2 command **db2relocatedb** enables a DBA to relocate a database based on information in a configuration file. This command can also be used to inform DB2 if the location of the database files has changed. Detailed information about the command can be found in *IBM DB2 Universal Database Command Reference*, SC09-4828-01.

The command syntax is:

```
db2relocatedb -f configFileName
```

The configFileName file contains all of the information describing the database. The command alters the information for the database concerning where its files are stored. If the data still resides on a drive but the drive has been relabeled on the new system, issue the command so that DB2 will recognize the database and be able to use it after DB2 UDB has been installed. The format of the configuration file is described below.

The **db2relocatedb** command configuration file has the following keywords:

- DB_NAME

The name of the database to be relocated.

- DB_PATH

The path where the database is stored. This is the root directory for the database.

- INSTANCE

The name of the instance that the database belongs to.

- NODENUM

The node number of the database. This is used only for partitioned databases.

- LOG_DIR

The directory where the log files are stored.

- CONT_PATH

This is a list of container paths that have to be changed. There must be a separate entry for each container that is being changed in the database. If you are moving a large number of database containers, then you should use a script or editor (such as 'vi') to generate a script to make the db2relocatedb configuration file.

- The parameters are set using this format:

```
parameter=oldValue,newValue
```

- Steps for using **db2relocatedb**

To use the **db2relocatedb** command to migrate a database:

- Install DB2 UDB on the target server with the same version and fix pack as on the original server.
- Stop DB2 on the original server.
- Unplug the disks (or disk subsystem) from the source machine in a manner approved by the operating system. On Windows, you should disable the disk using the Device Manager utility before unplugging it, in order to prevent a system crash or data loss.
- Enable the disks (or disk subsystem) on the target machine in an approved manner. On Windows, you should use the Disk Management tool to rescan disks.
- Verify that all disks are online and drive letters are properly assigned.
- Run the **db2relocatedb** utility to modify the desired configuration on the target system. (**db2relocatedb** will automatically catalog the database indicated in the configuration file.) Proceed through the normal performance tuning tasks to ensure that DB2 meets the performance expectations on the target server.

Example of relocation

A database is currently laid out on a Windows 2000 server. The DBA would like to move the database over to a new Windows 2003 server. The new server, however, has an additional drive on it, which means that all of the drive letters on the disk array that was used for the database will have to be changed. The old server is laid out as in Example 2-1 on page 46.

Example 2-1 Relocate database: old system configuration

```
database name = redDB
instance name = redInst
database path = e:
dms container=e:\redbook\data
database log path = f:\db2\redDB\logs
```

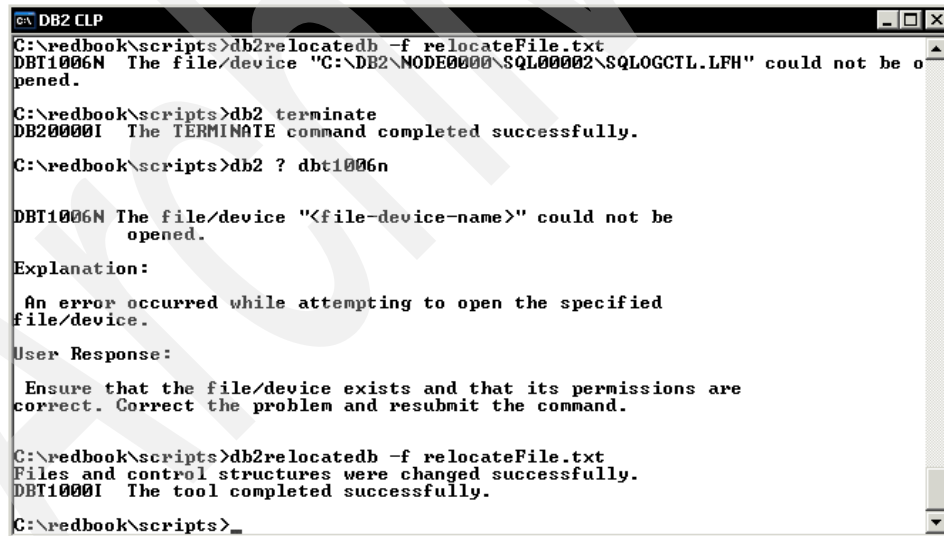
The configuration file for the **db2relocatedb** command would be as shown in Example 2-2.

Example 2-2 Relocate database new system configuration

```
DB_NAME=redDB
DB_PATH=E:,F:
INSTANCE=redInst
LOG_DIR="F:\db2\redDB\logs","G:\db2\redDB\logs"
CONT_PATH="E:\redbook\data","F:\redbook\data"
```

The information above (Example 2-2) would be placed into a file and then the following command would be run.

```
db2relocatedb -f configFile
```



```
DB2 CLP
C:\redbook\scripts>db2relocatedb -f relocateFile.txt
DBT1006N The file/device "C:\DB2\NODE0000\SQL00002\SQLLOGCTL.LFH" could not be o
pened.

C:\redbook\scripts>db2 terminate
DB20000I The TERMINATE command completed successfully.

C:\redbook\scripts>db2 ? dbt1006n

DBT1006N The file/device "<file-device-name>" could not be
opened.

Explanation:

  An error occurred while attempting to open the specified
file/device.

User Response:

  Ensure that the file/device exists and that its permissions are
correct. Correct the problem and resubmit the command.

C:\redbook\scripts>db2relocatedb -f relocateFile.txt
Files and control structures were changed successfully.
DBT1000I The tool completed successfully.

C:\redbook\scripts>
```

Figure 2-7 Executing the db2relocatedb command

Before you run the **db2relocatedb** command, be sure that there are no connections to the database and that it has not been activated. The use of the **db2relocatedb** command is illustrated in Figure 2-7. If any connections remain

open, you can force them off using the DB2 **force** command. The command syntax is:

```
force application {ALL | ( application-handle [ {,application-handle} ... ] )}
```

You can also use the **quiesce** command, which was introduced in V8.1. If you place a database into quiesce mode, then no further connections are allowed to the database until the mode is changed. This ensures that other users do not connect to the database when you force off the connections until you start your database relocation. The **quiesce** command has a built-in force option, which forces all applications off the database.

The command syntax is:

```
quiesce database {IMMEDIATE | DEFER} [FORCE CONNECTIONS]
```

Example 2-3 Preparing a database to be relocated

```
db2 list applications
>> Output:
DREWKB  db2bp.exe      7          *LOCAL.DB2.00B988234903      NETEDIT  1

db2 quiesce database FORCE
>> DB20000I  The QUIESCE DATABASE command completed successfully.
db2 list applications
>> SQL1611W  No data was returned by Database System Monitor.  SQLSTATE=00000
db2 relocatedb....
db2 unquiesce database
```

2.2.2 Migrating between versions of DB2 UDB

Migrating between versions of DB2 UDB has been simplified with the DB2 system commands that are included with DB2 UDB V8.2. You can migrate your DB2 UDB from V6 or V7 to V8.2 directly without going through V8.1. The migration of your databases is required if you want to continue using upgraded instances from DB2 UDB V6 or V7 with the V8.2 installation. There is a detailed description of the migration process in the DB2 UDB V8.2 manual *Quick Beginnings for DB2 Servers*, SG09-4836-01.

Restrictions

Before you begin your migration, make sure to review the following requirements:

- ▶ Migration is supported only from DB2 UDB V6 and V7.
- ▶ The migration command can be issued only from a DB2 UDB V8.x client or server, not from a V6 or V7 client or server machine.

- ▶ You cannot migrate a database between platforms. Only existing Windows databases can be migrated to Windows Server 2003.
- ▶ Migrating a partition database requires that DB2 UDB V8.1 must first be installed on all the computers with the DB2 partitions on them.
- ▶ Windows allows only one version of DB2 UDB on a machine. If you already have DB2 UDB V6 or V7 on the machine, then the V8.2 install process will uninstall the previous version.
- ▶ User objects cannot have V8.2 reserved schema names as qualifiers. These qualifiers include SYSCAT, SYSSTAT, and SYSFUN. It is not recommended that you use DB2INFO, because this schema is used by the DB2 UDB Version 8.1 and later Cube Views product.
- ▶ The database must also be in a stable state. It cannot be in any of the following states:
 - Backup pending
 - Roll forward pending
 - Table spaces not in a stable state
 - Transactions inconsistent

The following are the steps for a database migration:

1. Take a backup.

Make sure to take a full offline database backup before you begin the migration. After a database is migrated to DB2 UDB V8.2, it cannot be migrated back to the old release. The backup enables you to uninstall V8.2 and reinstall your previous release with the old database.

2. Confirm that you have sufficient space for the migration.

3. Record your database and instance configurations.

4. Check the diagnostic level.

It is recommended that you set the DB2 instance parameter DIAGLEVEL to 4 during the migration to ensure that all error messages are captured. Use this command to set the DB2 diagnosis level:

```
db2 update dbm cfg using diaglevel 4
```

5. Verify that the databases are ready to be migrated.

The command **db2ckmig** checks whether your database is ready to be migrated. The command confirms that:

- A database is not in an inconsistent state.
- A database is not in a backup pending state.
- A database is not in a rollforward pending state.

- Table spaces are all in a normal state.

The command syntax is:

```
db2ckmig databaseAlias -1 logFileLocation
```

Note: In Windows, the **db2ckmig** command must be run before DB2 UDB V8.2 is installed. **db2ckmig** can be found on the DB2 CD-ROM or install image under the directory images\DB2\Windows\utilities.

6. Take the database system offline.

The entire DB2 system will have to be taken offline. This includes stopping both the instance and the license server.

- a. Stop the DB2 license server: **db2licd -end**
- b. Force all applications off the databases: **db2 force applications all**
- c. Stop the instance: **db2stop**
- d. If the stop fails, then you may have to force it: **db2stop force**

7. Install DB2 UDB V8.2.

8. Migrate the database using the DB2 **migrate** command.

Migrating the database

The databases from previous versions are migrated using the DB2 command **migrate**. The command only migrates a database to the latest version, and does not migrate databases back to previous releases. If an error occurs during the use of a command, the DB2 **terminate** command must be issued before the command is used again. Additional documentation can be found in the *IBM DB2 Universal Database Command Reference Version 8*, SC09-4828.

Figure 2-8 shows the command syntax, and Example 2-4 on page 50 shows the command example.

```
>>-MIGRATE--+-DATABASE-+-database-alias----->
               '-DB-----'

>--+-----+-----><
   '-USER--username--+-----+-'
                   '-USING--password-'
```

Figure 2-8 DB2 migrate command syntax

```
db2 migrate database redDB user redTest using passwordTest
>> Files and control structures were changed successfully.
>>DBT1000I The tool completed successfully.
```

2.2.3 Migrating from 32-bit to 64-bit

With enhancements made in DB2 UDB V8.2, the process to move DB2 UDB V8 or V8.2 from 32-bit Windows Server 2003 OS to 64-bit Windows Server 2003 is not complicated. As discussed in “Using DB2 Backup and Restore” on page 41, the backup and restore process is not supported for major releases and bit-ness. Figure 2-6 on page 42 illustrates which combinations are supported.

If you are moving from V7.2 on Windows 2000 to V8.2 on Windows Server 2003 64-bit, you should refer to the next section.

Note: It is now possible to restore from a V7 backup to V8. This restoration is only supported for the same bit level of the restore.

2.2.4 Migrating everything at once

It is definitely not recommended that you try to move both your Windows release and your DB2 UDB version at the same time. However, it is quite likely that you may be moving from Windows Server 2000 with DB2 UDB V7.2 to Windows Server 2003 64-bit to DB2 UDB V8.1 64-bit. Two different methods are given to illustrate different migration strategies.

Method one

This method isolates the changes to the new target server and leaves your original server intact and operational. This method works well if you want to be able to easily fail back to your original server if there are problems with the new server.

The migration steps are:

1. Install Windows 2003 64-bit on the new target server.
2. Install DB2 UDB V8.2 on the target server.
3. Take an offline backup of the DB2 database on the original server.
4. Check that the database is ready to be migrated by using the **dbckmig** command.
5. Record all of the DB2 instance and database configuration parameters.

6. Copy the DB2 backup image to the target server from the original.
7. Take the database system offline on the original server.
8. On the target server, change the DB2 instance configuration parameter `diaglevel` to 4. This ensures that a full amount of information on the migration is captured.
9. Restore the database image on the target server. The restored database is migrated automatically, so the DB2 **migrate database** command is not required.

Method two

The second method has the migration performed on the original server. This reduces the complexity of the data movement and enables Windows Server 2003 to be installed in parallel on the target machine. The method does, however, make it difficult to fail back to the old configuration because the installation of V8.2 on the original server causes an uninstall of DB2 UDB V7.

Steps

The method two migration procedure is:

1. Take an offline backup of DB2 on V7 or V6 on the original server.
2. While working on the DB2 database backup, you could install Windows Server 2003 on the new target server.
3. Install DB2 UDB V8.2 on the target server.
4. Check that the database is ready to be migrated by using the **dbckmig** command.
5. Record all of the DB2 instance and database configuration parameters.
6. Install DB2 UDB V8.2 on the original server.
7. Migrate the database to V8.2 using the DB2 **migrate** command.
8. Take an offline backup of the database on the original machine on V8.2.
9. Copy the backup image to the target server.
10. Restore the backup on the target server.
11. Apply the database and instance configuration parameter from the original server to the target server.

Administration and management

DB2 UDB V8.2 has added great enhancements to its already rich autonomic database management functions and features. These autonomic functions and features can reduce time for DBA routine maintenance work and assist DBA in database design and performance tuning.

This chapter discusses the DB2 UDB V8.2 advancements in administration and management. The autonomic and management enhancements covered are:

- ▶ Automated backup and recovery
- ▶ Automated log file management
- ▶ Automated table maintenance
- ▶ Integrated Design Advisor
- ▶ Auto RUNSTATS/REORG

3.1 Backup, recovery, and logging

DB2 UDB V8.2 provides automated backup, backup compression, and self-tuning backup and restore advancements. In addition, DB2 provides automated log file management and backs up the logs along with the database backup.

3.1.1 Automatic backup

Database backup is a routine DBA task. Automatic database backup simplifies database backup management tasks by ensuring that a recent full backup of the database is performed properly and regularly, without the DBA having any knowledge of the backup command. DB2 takes online or offline database backups automatically, based on the maintenance objectives defined through the Configure Automatic Maintenance wizard in the Control Center or Health Center.

Use these steps of the Configure Automatic Maintenance wizard to set up the automatic database backup:

1. Start the Configure Automatic Maintenance wizard in one of three ways:
 - In the Control Center, right-click the database and select **Configure Automatic Maintenance** (Figure 3-1 on page 55).
 - In the Health Center, right-click the database and select **Configure Automatic Maintenance**.
 - Navigate to **Control Center** → **Tools** → **Wizards** → **Configure Automatic Maintenance**.

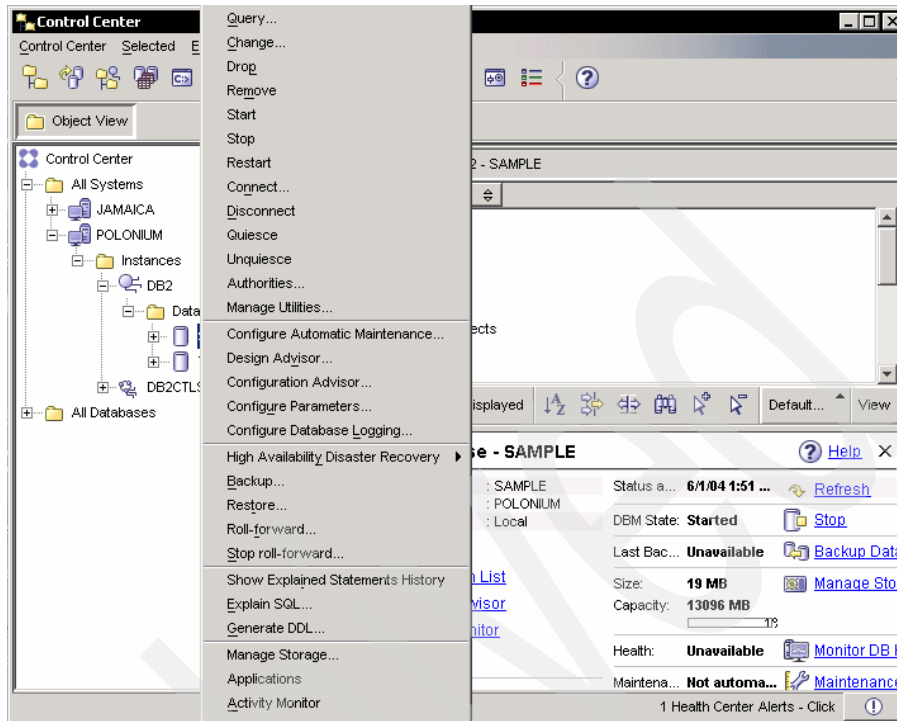


Figure 3-1 Starting configure Automatic Maintenance

2. At the Configure Automatic Maintenance window (Figure 3-2), click **Next**.

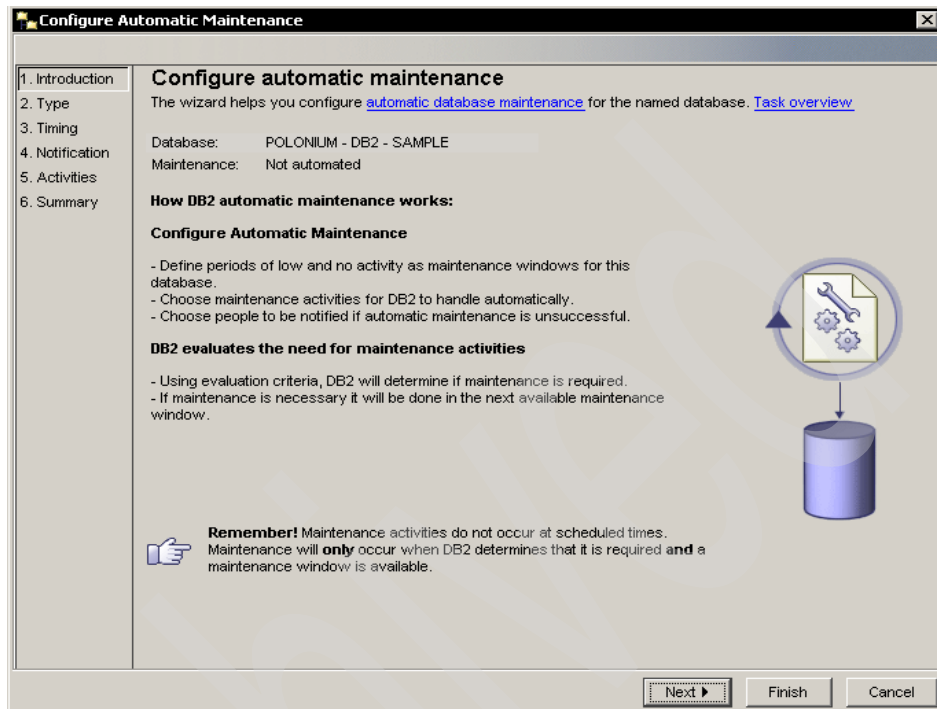


Figure 3-2 Configure automatic maintenance: Introduction

3. At the window shown in Figure 3-3, select the type of automatic maintenance for your database or disable the automatic database maintenance. To choose the type, select **Change automation setting** and click **Next**.

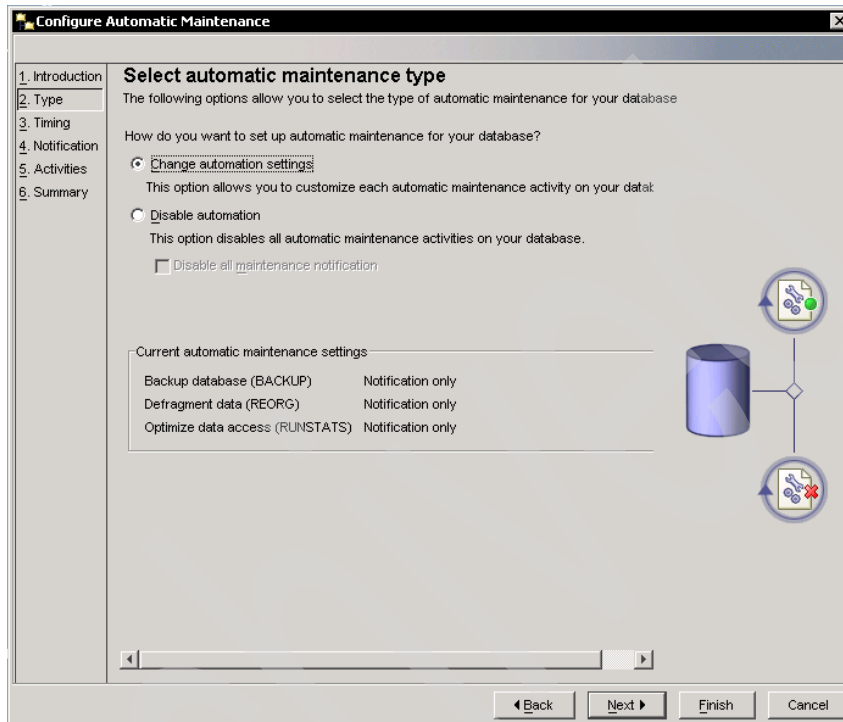


Figure 3-3 Select automatic maintenance type

4. In the dialog in Figure 3-4, specify the online and off-line maintenance periods. We want to back up the database online daily from 10:00 PM to 12:00PM, so we click **Change**.

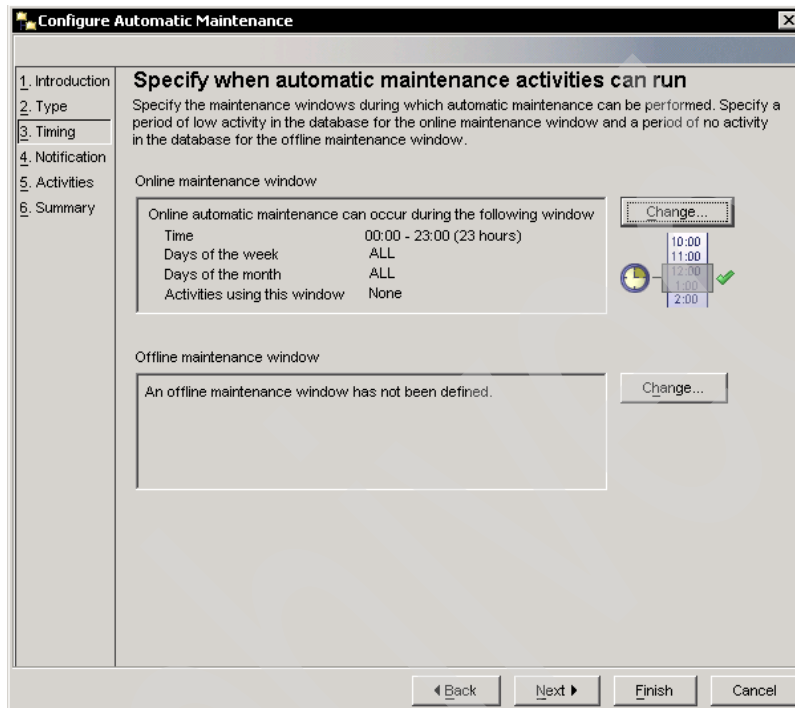


Figure 3-4 Specify when automatic maintenance will run

5. This opens a pop-up window to specify the start time and duration of the maintenance window and the day the maintenance should take place (Figure 3-5). Note that there is no default offline maintenance window.

Change Maintenance Window Specification - Online Activity

Specify when automatic maintenance can occur.

☒ During the specified time.

☐ Outside the specified time.

Specify the start time and the duration of the maintenance window. The start time is specified using a 24-hour clock.

Start time: 20:00

Duration: 2 hours

Specify how often this maintenance window occurs. A valid maintenance window must meet the conditions specified on both the Days of the Week tab and the Days of the Month tab.

Days of the week | Days of the month

☒ All

☐ Only on selected days

☐ Monday ☐ Friday

☐ Tuesday ☐ Saturday

☐ Wednesday ☐ Sunday

☐ Thursday

Preview

Online automatic maintenance can occur **during** the following window.

Time: 20:00 - 22:00 (2 hours)

Days of the week: ALL

Days of the month: ALL

Activities using this window: None

OK Cancel Help

Figure 3-5 Specify maintenance window

6. In the health notification contact list, you can select, add, or delete people who will receive job-status e-mails (Figure 3-6). When you are finished, click **Next**.

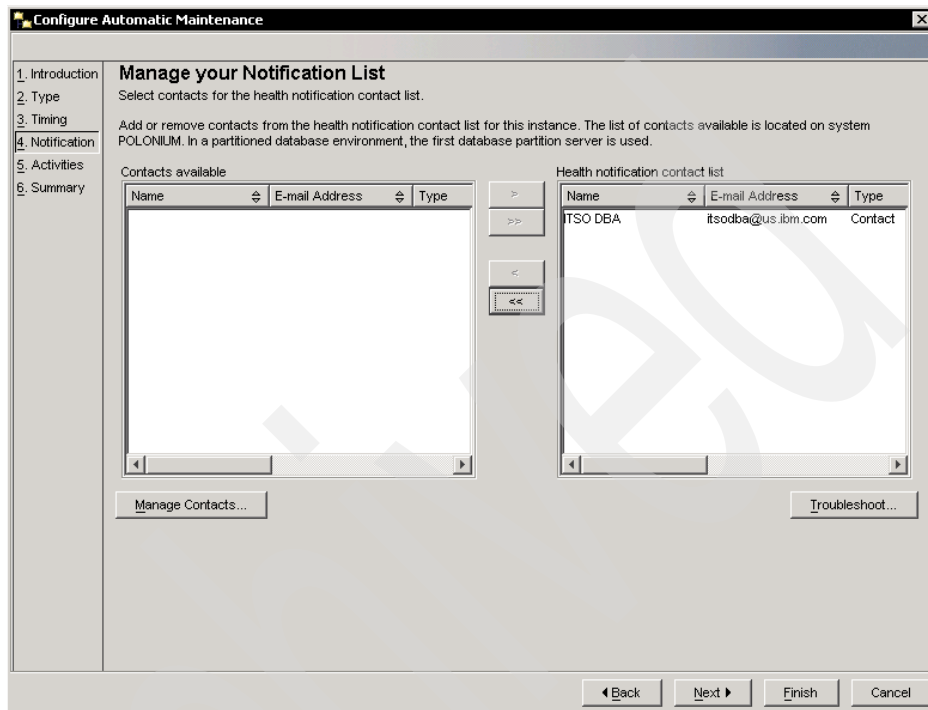


Figure 3-6 Manage the notification list

Note: The Configure Automatic Maintenance wizard can be used to configure automatic backup, RUNSTATS, and REORG. The maintenance windows and notification list as defined up to this point will be used for backup, RUNSTATS, and REORG if all three tasks are activated for automatic maintenance.

7. In this window, you can configure database activities: backup, REORG, and RUNSTATS (Figure 3-7). We configure database backup by selecting **Backup database (BACKUP)** and clicking **Configure settings**.

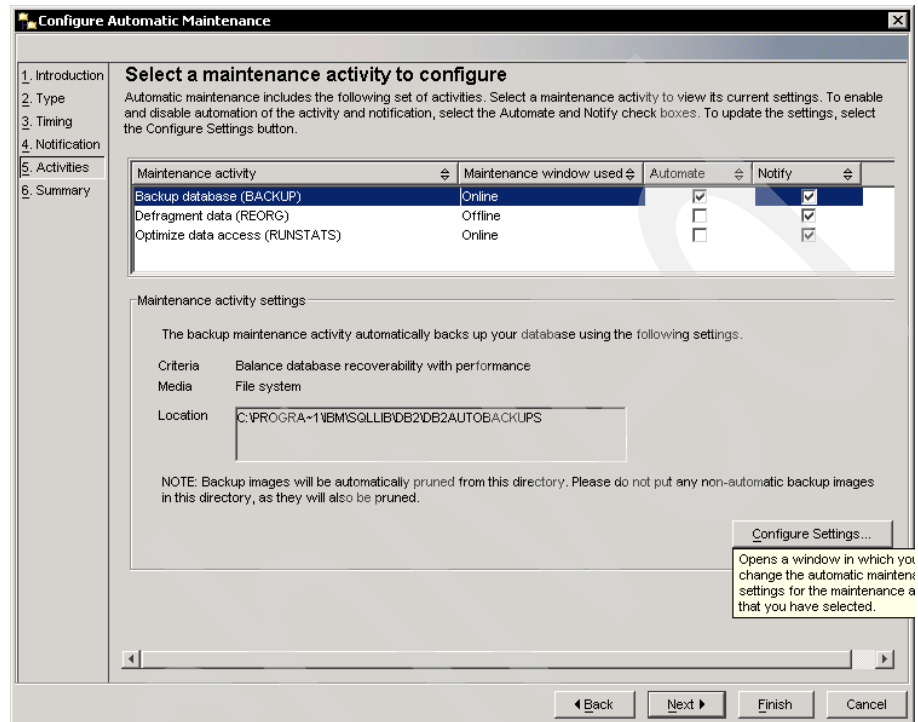


Figure 3-7 Select maintenance activity to configure

8. This opens the Configure Settings - Backup Database (BACKUP) window (Figure 3-8). There are three tabs:
- Backup criteria
The criteria determine how often the database will be backed up. You can let DB2 decide by selecting the provided options or use Customize to specify exactly how often to back up your database.
 - Backup location
Specify where to locate the database backup. The media types include file system, tape, TSM, XBSA, and vendor DDL.
 - Backup mode
Specify online or offline backup.

We choose **Optimize for database recoverability (more frequent backups)** for a daily database backup. We use the file system D:\db2backup as the backup location, and **online** for the backup mode. Click **OK**.

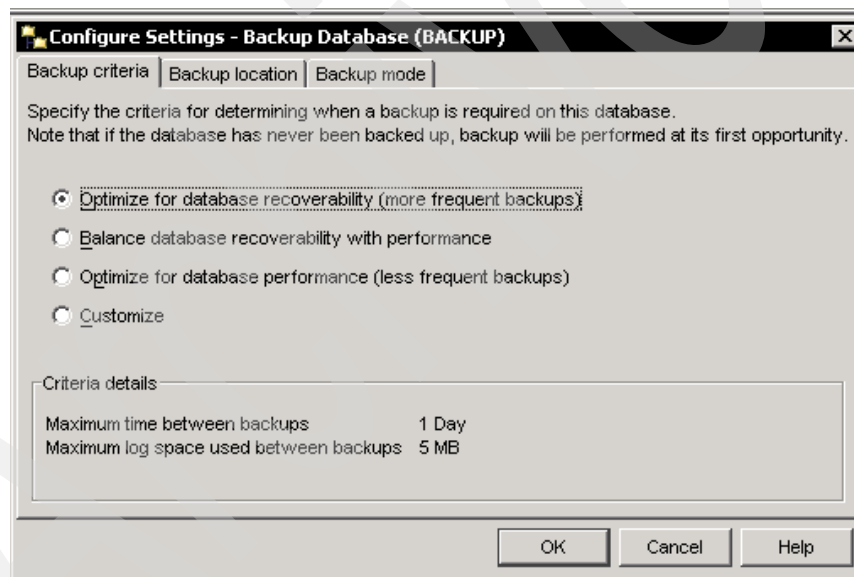


Figure 3-8 Configure settings: Backup Database (BACKUP)

9. The Review the automatic maintenance settings dialog is a summary of the automatic maintenance activity you just configured or changed (Figure 3-9).

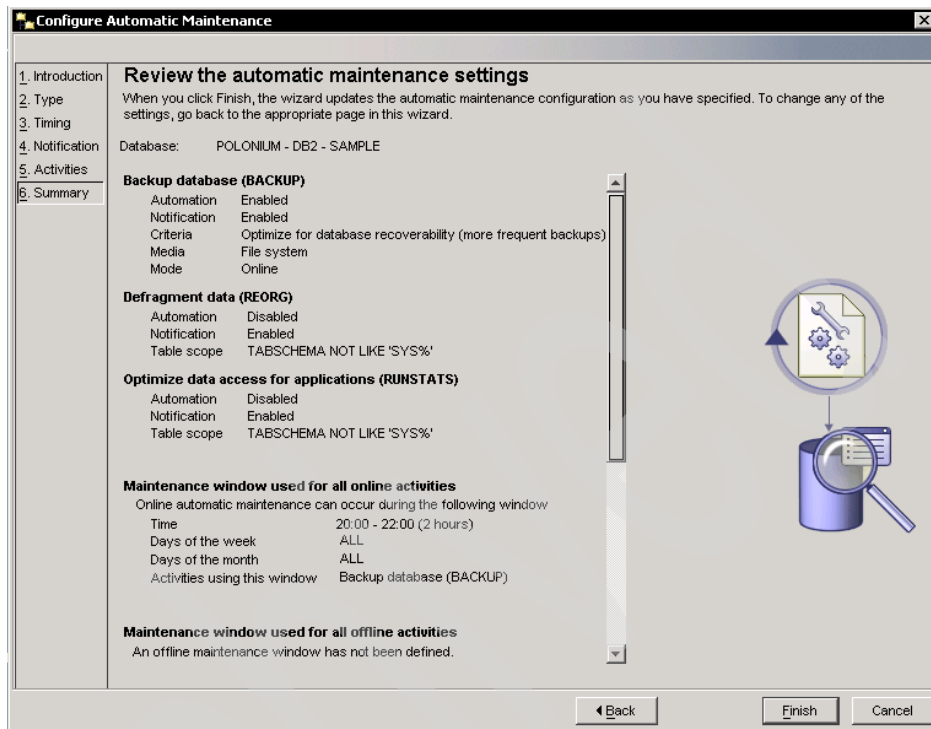


Figure 3-9 Review the automatic maintenance settings

A new database configuration parameter `AUTO_DB_BACKUP` is provided as the switch to turn the backup automation on and off. The parameter can be set by the following command:

```
db2 update database configuration for dbname using auto_db_backup on/off
```

Example 3-1 shows the database configuration after we configured the automatic database backup feature.

Example 3-1 Database configuration with automatic database backup on

Database Configuration for Database sample

Database configuration release level	= 0x0a00
Database release level	= 0x0a00
Database territory	= US
Database code page	= 1252
Database code set	= IBM-1252
Database country/region code	= 1

Database collating sequence	= UNIQUE
Alternate collating sequence	(ALT_COLLATE) =

...

Automatic maintenance	(AUTO_MAINT) = ON
Automatic database backup	(AUTO_DB_BACKUP) = ON
Automatic table maintenance	(AUTO_TBL_MAINT) = OFF
Automatic runstats	(AUTO_RUNSTATS) = OFF
Automatic statistics profiling	(AUTO_STATS_PROF) = OFF
Automatic profile updates	(AUTO_PROF_UPD) = OFF
Automatic reorganization	(AUTO_REORG) = OFF

Note: The automatic table maintenance database parameters have the hierarchical relationship. AUTO_MAINT is the parent of all the parameters. Turning a parent parameter OFF stops all children automatic activities but does not change children settings. Turning the parent parameter back on will reactivate previously automatic activity.

3.1.2 Self-tuning backup and restore

To minimize the amount of time that is required to complete backup and restore operations, DB2 UDB V8.2 automatically chooses the number of buffers, the buffer size, and the parallelism settings for both backup and restore operations. The chosen values are based on the amount of available memory, the number of available processors, the number of table spaces, and the extent size of each table space.

The BACKUP DATABASE and RESTORE DATABASE commands (Example 3-2) automatically choose an optimal value for the following parameters whenever they are not explicitly specified or if they are set to 0 in API:

- ▶ WITH num-buffers BUFFERS
- ▶ PARALLELISM n
- ▶ BUFFER buffer-size

The database manager configuration parameters BACKBUFSZ and RESTBUFSZ are ignored. If you wish to use the values, you must specify the value in the backup and restore command.

For a restore database operation, a multiple of the buffer size used for the backup operation will always be used.

Example 3-2 Backup and restore database command

RESTORE Command:

If you use a user-provided compression library, the library is stored in the backup image by default and will be used on restore. The command syntaxes are:

```
DB2 BACKUP DATABASE sample COMPRESS COMPLIB libname COMPROPTS options  
DB2 RESTORE DATABASE sample COMPLIB libname COMPROPTS options
```

In these examples, COMPROPTS is used to pass the options to the compression library.

Use of the automatic database function could create some CPU costs due to the compression computation. However, the media I/O time will be decreased because the image size is smaller. The overall backup/restore performance impact depends on whether CPU or media I/O is a bottleneck of the system.

3.1.4 Logs in backup images

DB2 transaction log files are essential for restoring and recovering a database or table space to a consistent point in time using online backup images. Prior to Version 8.2, the backup and restore utilities required shipping the backup image and corresponding log files as separate objects to the disaster recovery sites. This risks losing the log files required for recovery. The new enhancement enables you to include the logs on the online backup. This way, you can ship the backup images with the log files to ensure that the backup will be restorable if archived logs are misplaced. This feature supports all types of online backups such as database, table space, incremental, and compressed.

If the INCLUDE LOGS option is specified in the backup command, DB2 will truncate the tail log file and close it after the table space or database is backed up. All logs that are needed to restore the backup and roll forward to the time corresponding to the end of the backup are placed in the backup image. DB2 backs up log files in parallel with backing up the table space data. If database backup compress is specified, the log files will be compressed as well.

By default, the log will not be backed up with the database or table space backup image. To use this feature, specify the INCLUDE LOGS option in the backup command, such as:

```
DB2 BACKUP DB sample ONLINE TO d:\db2backup INCLUDE LOGS
```

When restoring the database, specify the directory where you want DB2 to place the log files. For example:

```
RESTORE DB sample ONLINE FROM d:\db2bckup LOGTARGET d:\db2logs  
ROLLFORWARD DB sample TO END OF LOGS AND STOP OVERFLOW LOG PATH d:\db2logs
```

You also can restore log files only from the integrated backup image:

```
RESTORE DB sample LOGS FROM d:\db2backup LOGTARGET d:\db2logs
```

3.2 Automated log file management

The new integrated and automated log file management system in DB2 UDB V8.2 provides another time saving feature for DBAs. You no longer need to customize sample userexit or code your own log management routine to archive and delete the DB2 logs. V8.2 introduces a new log manager, db2logmgr, which manages the use, archive, retrieval, and deletion of log files with minimal user interaction. You can archive the logs to TSM, tape, disks, or third-party vendor device. You can specify multiple archive targets for extra redundancy or change the archive targets online.

To set up, simply update the database configuration parameters based on your needs. The new database parameters used by the new log manager for controlling archive locations and handling archival problem are:

- ▶ Managing log archive location
 - LOGARCHMETH1: to specify the primary log archive. The syntax is:
 - DISK: <path>
 - TSM: [management class name]
 - Vendor: <vendor library>
 - USEREXIT
 - LOGRETAIN
 - LOGARCHMETH2: optional log archival target, same syntax as LOGARCHMETH1.
 - LOGARCHOPT1: for specifying optional archiving parameters
 - LOGARCHOPT2: for specifying optional archiving parameters

For example:

```
DB2 UPDATE DB CFG FOR sample USING LOGARCHMETH1 DISK:d:/db2logsbk
DB2 UPDATE DB CFG FOR sample USING LOGARCHMETH1 TSM
DB2 UPDATE DB CFG FOR sample USING LOGARCHOPT1 -SERVERNAME=ITSOTSM1
```

Note: LOGRETAIN and USEREXIT are deprecated.

- ▶ Handling log archival problems
 - FAILARCHPATH: Specifies the path to be used when any primary targets cannot be archived. The media for FAILARCHPATH can only be disk. This parameter can be set dynamically after a problem with a primary target. Any pending-retry log archival requests are switched to use this path.
 - NUMARCHRETRY: The number of retry attempts on primary target(s) before archiving to FAILARCHPATH. The default value is 5.

- ARCHRETRYDELAY: The number of seconds between retry attempts. The default value is 20 seconds.

Note: userexit is still supported as another log management option.

A new option, AND DELETE, is added to the PRUNE HISTORY command to prune the unnecessary logs.

```
>>-PRUNE----->
>--+HISTORY--timestamp--+-----+--+-----+--+<
|                               '-WITH FORCE OPTION-'   '-AND DELETE-' |
|                               '-LOGFILE PRIOR TO--log-file-name-----'
```

When the history file entry is removed, the associated log archives will be physically deleted. This option is especially useful for ensuring that archive storage space is recovered when log archives are no longer needed.

Note: If you are archiving logs via a user exit program or in-house developed routine, the logs cannot be deleted using this option.

3.3 RECOVER command

DB2 UDB V8.2 introduces the new RECOVER command, which simplifies the database recovery process. This command combines RESTORE and ROLLFORWARD into a single step and uses the information in Recovery history file to determine automatically which backup images and log files to use. The Recovery history file now keeps detailed information about log chains, so even when multiple logs chains exist due to past recoveries to a point in time, the RECOVER command can be used to restore a database to the point of failure. With this command, the user can recover a multiple-partition database in a single command. In the partition database environment, this command can only be invoked from the catalog partition.

Some examples using the RECOVER command follow. For the complete RECOVER command syntax, refer to *IBM DB2 UDB Command Reference Version 8.2*, SC09-4828-01.

- To recover a database to the end of logs from the best available backup images:

```
RECOVER DB sample
```

- To recover a database to a particular point in time:

```
RECOVER DB sample TO 2004-05-31-05.30.28
```

- To recover a point in time not represented in the current history:

```
RECOVER DB sample TO 2004-05-27-10.13.20 USING HISTORY FILE  
(d:/db2history/archive/db2rhist.asc)
```

3.4 Automated table maintenance

REORG and RUNSTATS are two table maintenance tasks the DBA must perform regularly. Determining the right time to collect statistics and to reorganize a table is not easy. The automated table maintenance feature in DB2 UDB V8.2 provides a solution for DBAs.

3.4.1 Automatic statistics collection

When dealing with table statistics collection, DBA faces the following challenges:

- When to collect statistics
- How often to collect statistics
- What table to collect statistics on
- Which statistics to collect for each table
- What parameters to use for RUNSTATUS on each table

The automated statistics maintenance features of DB2 UDB V8.2 provides the solution to all of these challenges with *automatic statistics collection (automatic runstats)* and *automatic statistics profiling*.

Automatic statistics collection introduces two automatic feedback loops into the DB2 system:

- Data-manipulation-driven feedback loop
This feedback loop monitors table activity. Whenever update, delete, insert (UDI), or load statements against a specified table have changed the data distribution so that the present statistics for that table are substantially outdated, it triggers statistics collection (RUNSTATS) on the table in a throttled background process.
- Query-driven feedback loop
This feedback loop monitors query activity on a table. Whenever it detects, based on query feedback, that the statistics parameters have been configured improperly, it modifies the RUNSTATS profile. The RUNSTATS profile has the proper settings for the statics configuration parameters for the table.

Figure 3-10 on page 71 depicts the architecture of the overall system for automatic statistics collection.

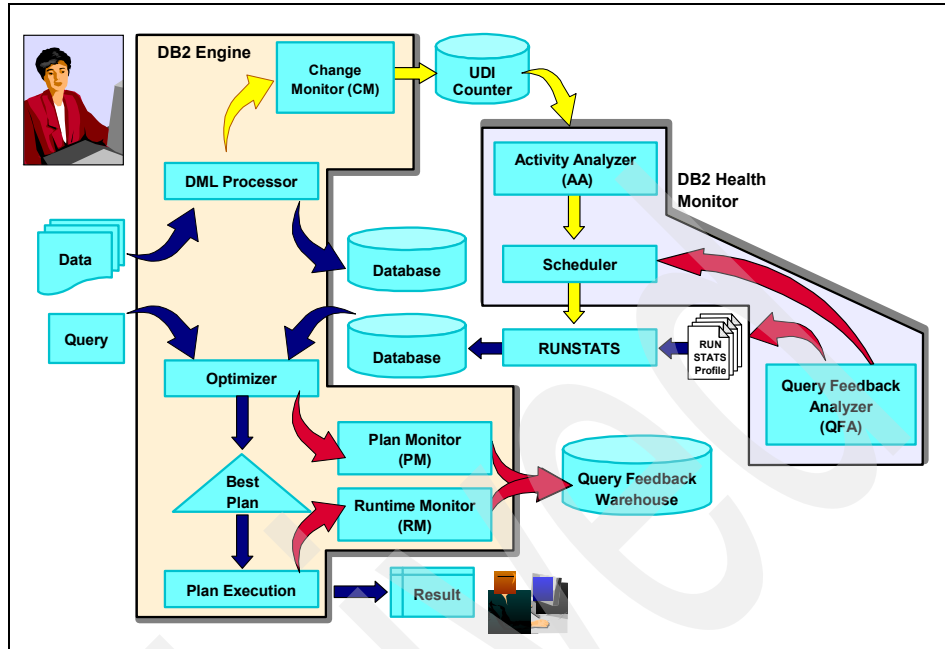


Figure 3-10 Automatic statistics collection overview

The automatic statistics collection system can be triggered by either data activity or query activity. The left part of the figure depicts functionality that is implemented in the DB2 engine, including the query processor with optimizer and plan execution, data manipulation language (DML) processor, and some monitors, which have been added to the DB2 engine to facilitate several automatic capabilities in DB2. The right part of the figure depicts several analyzers that have been added to the DB2 Health Monitor to realize automated statistics collection.

In case of data activity through DML or LOAD statements against a table in the database, the DML processor both modifies the database and employs a change monitor (CM) that records the number of changes against each table using a UDI counter. An activity analyzer (AA) uses this information to determine whether statistics on an active table have changed substantially, in order to justify statistics collection (RUNSTATS) in the background for this table. This data manipulation (data-centric) feedback loop provides input to a scheduler that prioritizes statistics collection based on the degree to which activity on a table has altered the data distribution.

In case of query activity, the DBMS optimizer determines the best plan with respect to its cost model, which is then executed by the runtime system. The optimizer uses statistics gathered by RUNSTATS to derive the best plan.

RUNSTATS collects the statistics using the statistics configuration of each table, which are stored and maintained in a RUNSTATS profile. The query-centric approach adds a feedback loop to statistics configuration. This feedback loop observes query activity by using a plan monitor, which stores the best plan together with the cardinalities estimates for each intermediate result as derived by the optimizer. During plan execution, a run-time monitor observes the estimated cardinalities. All of this compile-time and run-time information is stored in a query feedback warehouse. A feedback analyzer periodically analyzes the information in the warehouse. Based on the discrepancy between actual and estimated cardinalities, it derives modifications to the RUNSTATS profile for individual columns, column groups, or tables. The feedback analyzer communicates its findings about tables with modified RUNSTATS profiles and tables with outdated statistics to the scheduler, so that the scheduler can properly prioritize executing RUNSTATS.

Statistics collection, as with any other background maintenance task, must not have a significant impact on more important business-critical tasks that the DBMS is used for. In the automatic statistics collection implementation, the statistics collection is guaranteed to not affect the user workload beyond a certain threshold. As with the database backup utility, the RUNSTATS utility is implemented with the DB2 utility throttling. The DBA can set the table maintenance objectives for RUNSTATS. For example, the DBA can limit the scope of automated statistics collection to certain tables and specify the maintenance time window to define time intervals that RUNSTATS should be executed in.

A new database manager configuration parameter, `AUTO_RUNSTATS`, is introduced in Version 8.2 for automatic statistics collection (automatic RUNSTATS). Details of the parameter are discussed in next section.

The Configure Automatic Maintenance wizard can be used to activate the automatic statistics collection functions and set the table maintenance objectives. The table maintenance objectives defined using the wizard are for controlling the set of tables to be maintained automatically and when (or when not) the maintenance window collects statistics.

Activating automatic RUNSTATS

To configure the automatic runstats, use the Configure Automatic Maintenance wizard.

The steps for starting the Configure Automatic Maintenance wizard and defining maintenance windows and notification lists are the same as steps 1 on page 54 through 6 on page 60 of setting up automatic backup. See 3.1.1, “Automatic backup” on page 54.

After completing the first six steps, proceed as follows:

1. Under Select a maintenance activity to configure (Figure 3-11), select **Optimize data access (RUNSTATS)** and click **Configure Settings** for a pop-up window to identify the tables to be maintained by DB2. Click **Next**.

Configure Automatic Maintenance

1. Introduction
2. Type
3. Timing
4. Notification
5. Activities
6. Summary

Select a maintenance activity to configure

Automatic maintenance includes the following set of activities. Select a maintenance activity to view its current settings. To enable and disable automation of the activity and notification, select the Automate and Notify check boxes. To update the settings, select the Configure Settings button.

Maintenance activity	Maintenance window used	Automate	Notify
Backup database (BACKUP)	Offline	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Defragment data (REORG)	Offline	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Optimize data access (RUNSTATS)	Online	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Maintenance activity settings

The data access optimization maintenance activity automatically analyzes the data in tables to allow for applications to access the data as quickly as possible. This activity will occur using the following settings.

Table scope

All tables including system tables

[Configure Settings...](#)

[Back](#) [Next](#) [Finish](#) [Cancel](#)

Figure 3-11 Automated runstats: Select maintenance activity

2. Under Configure settings - Optimize Data Access (RUNSTATS), define the tables that you want to exclude from the automatic statistics collection process(Figure 3-12). Note that the volatile tables are always excluded. After the filter is defined, click **Refresh Resulting Tables** to display all of the tables to be maintained, and review the list. Click **OK** when the list is complete.

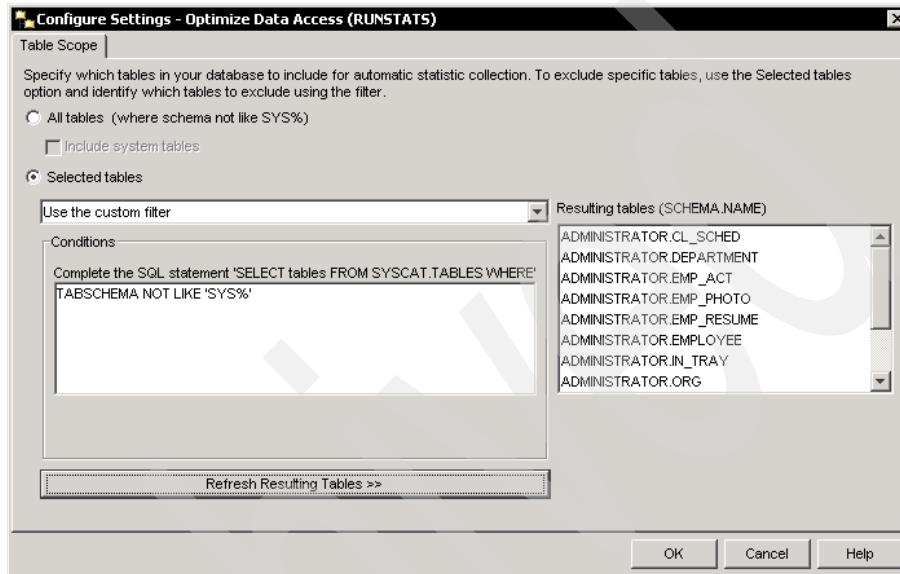


Figure 3-12 Automated runstats: Configure settings

- Under Review the automatic maintenance settings, review the summary of automatic maintenance settings for the database. In Figure 3-13, you can see that automated runstats is defined. Click **Finish** to process.

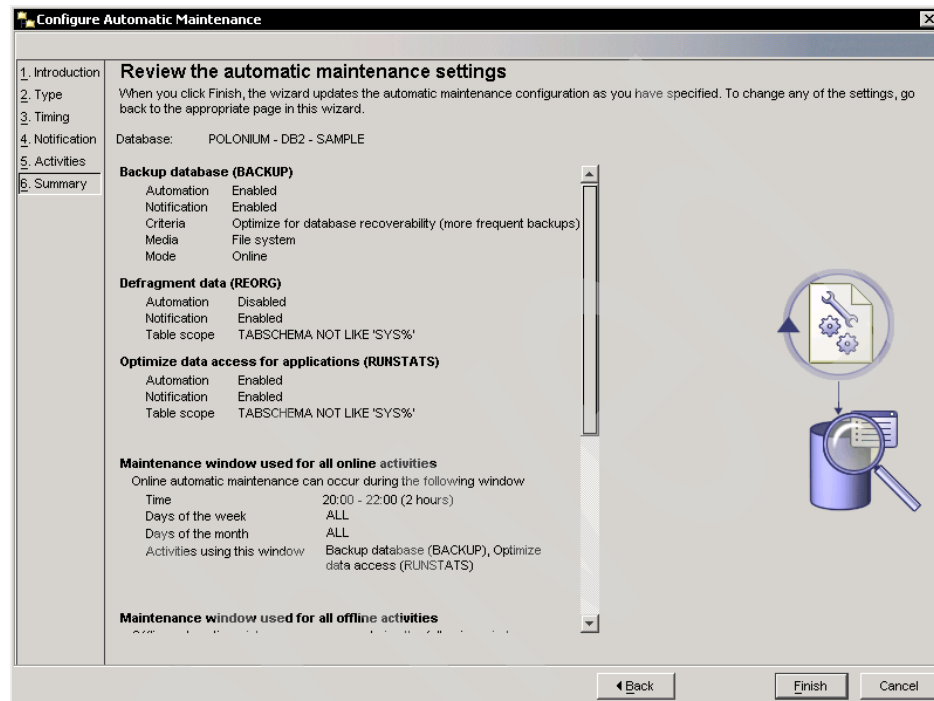


Figure 3-13 Automated runstats: summary

The database configuration parameter `AUTO_RUNSTATS` used for automatic statistics collection will be set by wizard if the tool is used to set up the automatic RUNSTATS. This parameter also can be set using the update database configuration command via command prompt. Note there is a hierarchical relationship in parameters `AUTO_MAINT`, `AUTO_TBL_MAINT`, and `AUTO_RUNSTATS`. The parent parameters have to be on before the child parameter can be set to on. Example 3-3 shows that these two parameters were turned on.

Example 3-3 Automated runstats database configuration parameters

Database Configuration for Database sample

Database configuration release level	= 0x0a00
Database release level	= 0x0a00
Database territory	= US
Database code page	= 1252
Database code set	= IBM-1252

Database country/region code	= 1
Database collating sequence	= UNIQUE
Alternate collating sequence	(ALT_COLLATE) =

...

Automatic maintenance	(AUTO_MAINT) = ON
Automatic database backup	(AUTO_DB_BACKUP) = ON
Automatic table maintenance	(AUTO_TBL_MAINT) = ON
Automatic runstats	(AUTO_RUNSTATS) = ON
Automatic statistics profiling	(AUTO_STATS_PROF) = OFF
Automatic profile updates	(AUTO_PROF_UPD) = OFF
Automatic reorganization	(AUTO_REORG) = OFF

3.4.2 Automatic statistics profiling

The RUNSTATS utility provides an option to register and use a statistics profile, which is a set of options that specify which statistics are to be collected on a particular table, such as table statistics, index statistics, or distribution statistics. This feature simplifies statistics collection by enabling you to store the options that you specify when you issue the RUNSTATS command so that you can collect the same statistics repeatedly on a table without having to re-type the command options.

Statistics profiles can also be generated automatically by the DB2 automatic statistics profiling feature. Automatic statistics profiles are the query-driven configuration of statistics collection. This feature uses empirical results from previously executed queries to validate statistics and assumptions, gather statistics that reflect the workload, and repair costing for the next query optimization round. Automatic statistics profiling can recommend which statistics to collect and when to collect them.

Figure 3-14 on page 77 illustrates the architecture of automatic statistics profiling.

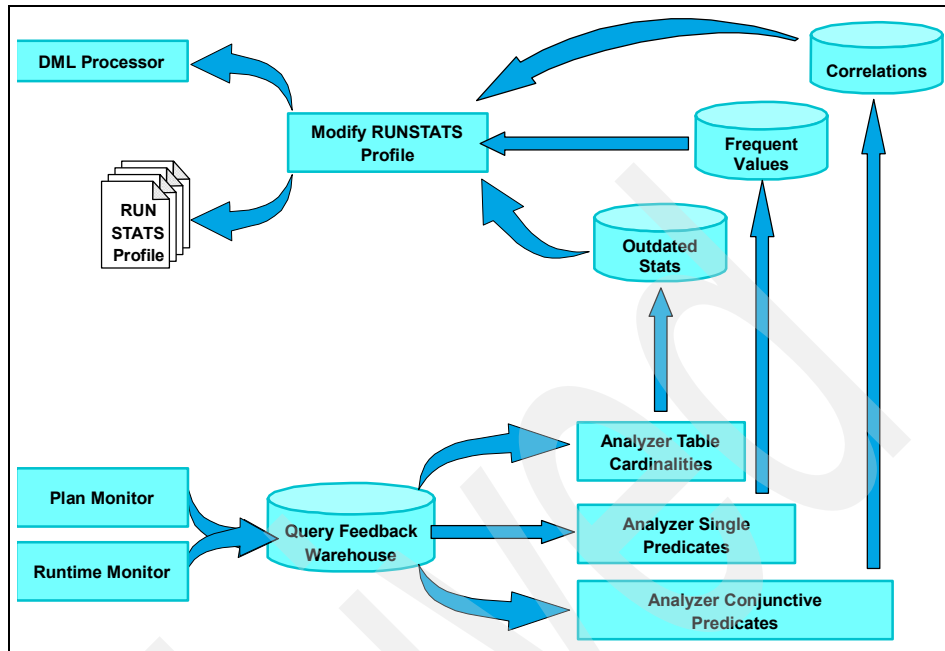


Figure 3-14 Automatic statistics profiling

The query-centric configuration of statistics collection uses a feedback loop that monitors query execution to populate a query feedback warehouse with estimation errors. It analyzes the warehouse to determine:

- ▶ Which tables have outdated statistics
- ▶ Whether and how frequently the values for columns on a particular table should be reconfigured
- ▶ Which correlation statistics should be created in order to reduce estimation errors in the future

The query feedback analyzer will produce:

- ▶ A prioritized list of tables that require statistics collection and communicate it to the scheduler.
- ▶ Information that indicates which statistics should be collected on a particular table and stores it in the RUNSTATS profile.

Note: In DB2 UDB V8.2, automated statistics profiling is only available for DB2 ESE without the database partitioning feature (DPF) enabled.

Activating automatic statistics profiling

Automatic statistics profiling can be activated using the following steps:

1. Create feedback warehouse tables.

The feedback warehouse consists of three tables in the SYSTOOLS schema that store information about the predicates encountered during query execution. The three tables are OPT_FEEDBACK_QUERY, OPT_FEEDBACK_PREDICATE, and OPT_FEEDBACK_PREDICATE_COLUMN.

These tables can be created using the SYSINSTALLOBJECTS stored procedure. This stored procedure is the common stored procedure for creating and dropping objects in the SYSTOOLS schema. Invoke the SYSINSTALLOBJECTS stored procedure as follows:

```
call SYSINSTALLOBJECTS ( toolname, action, tablespacename, schemaname )
```

In this example:

- *toolname* specifies the name of the tool whose objects are to be created or dropped (in our case, ASP or AUTO STATS PROFILING).
- *action* specifies the action to be taken: C for create, D for drop.
- *tablespacename* is the name of the table space in which the feedback warehouse tables will be created. This input parameter is optional. If it is not specified, the default user space will be used.
- *schemaname* is the name of the schema with which the objects will be created or dropped. This parameter is not used.

For example, to create the feedback warehouse using default table space:

```
F:\SQLLIB\BIN>db2 "call sysproc.sysinstallobjects('ASP','C','','')"

```

```
Return Status = 0

```

2. Set database configuration parameters.

The database configuration parameters for automatic statistics profiling are AUTO_STATS_PROF and AUTO_PROF_UPD. These two parameters have the following settings:

- AUTO_STATS_PROF:
 - OFF: This default value indicates that all features of statistics profiling are disabled.
 - ON: Statistics Profiling is enabled, performing runtime monitoring, feedback warehouse population, and feedback warehouse mining.
- AUTO_PROF_UPD (this is the child of AUTO_STATS_PROF)
 - ON: This indicates that the RUNSTATS profile for each table is updated with the recommendation.

- OFF: In this default value, recommendations are stored in the SYSTOOLS.OPT_FEEDBACK_RANKING table to be evaluated by the DBA for manually creating the profile.

The heretical relationship of the related database configuration parameter is:

Automatic maintenance	(AUTO_MAINT) = ON
Automatic database backup	(AUTO_DB_BACKUP) = ON
Automatic table maintenance	(AUTO_TBL_MAINT) = ON
Automatic runstats	(AUTO_RUNSTATS) = ON
Automatic statistics profiling	(AUTO_STATS_PROF) = OFF
Automatic profile updates	(AUTO_PROF_UPD) = OFF
Automatic reorganization	(AUTO_REORG) = OFF

The command to turn on automatic statistics profiling are

```
DB2 UPDATE DB CFG FOR sample USING AUTO_STATS_PROF on
DB2 UPDATE DB CFG FOR sample USING AUTO_PROF_UPD on
```

Note: AUTO_STATS_PROF and its child AUTO_PROF_UPD can only be activated in DB2 serial mode and are blocked for queries in federated, SMP, or MPP. Trying to activate these two parameters in SMP, MPP, or federated will result in an SQL5126N error.

In Version 8.2, new RUNSTATS options are introduced to improve performance:

► **USE PROFILE**

This enables RUNSTATS to employ a previously stored statistics profile to gather statistics for a table. The statistics profile is created using the SET PROFILE options and is updated using the UPDATE PROFILE options.

► **SET PROFILE**

Enables RUNSTATS to generate and store a specific statistics profile in the system catalog tables and executes the RUNSTATS command options to gather statistics.

► **SET PROFILE ONLY**

Enables RUNSTATS to generate and store a specific statistics profile in the system catalog tables without running the RUNSTATS command options.

► **SET PROFILE NONE**

Specifies that no statistics profile will be set for this RUNSTATS invocation.

► **UPDATE PROFILE**

Enables RUNSTATS to modify an existing statistics profile in the system catalog tables, and runs the RUNSTATS command options of the updated statistics profile to gather statistics.

► **UPDATE PROFILE ONLY**

Enables RUNSTATS to modify an existing statistics profile in the system catalog tables without running the RUNSTATS command options of the updated statistics profile.

The statistics profile is stored in a visible string format, which represents the RUNSTATS command, in the STATISTICS_PROFILE column of the SYSIBM.SYSTABLES system catalog table. This is the RUNSTATS command with SET PROFILE option:

```
runstats on table <db2user.tablename> tablesample bernoulli (20) set profile
```

Figure 3-15 shows the stored value in the STATISTICS_PROFILE column of the SYSIBM.SYSTABLES table resulting from the command:

```
select statistics_profile  
from sysibm.systables where name = 'EMPLOYEE' and creator = '<db2user>'
```

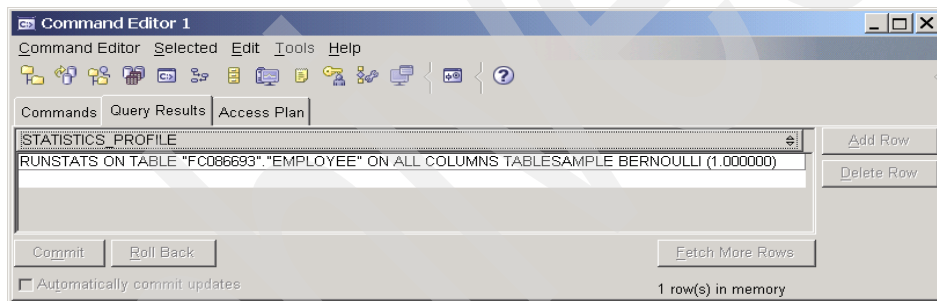


Figure 3-15 Stored RUNSTATS profile in SYSIBM.SYSTABLES

3.4.3 Automatic reorganization

Table reorganization is used to reclaim the fragmented spaces of a table and to ensure optimal I/O and query performance. The table reorganization process can consume resources and have an effect on performance. Determining the right time to reorganize a table is not a easy task for DBAs.

Prior to V8.2, DB2 had in-place reorg for both table and index high-availability features. The automatic table and indexing de-fragmentation and re-clustering features introduced in V8.2 enable DBAs to manage table and index reorg without manual intervention.

The automatic table maintenance objectives define the tables to be evaluated for the need of reorganization. Periodically, DB2 determines reorg candidates by evaluating the tables with fresh statistics or tables where an automatic reorg did not complete previously. Figure 3-16 on page 81 shows when the table or index reorg will take place after evaluation.

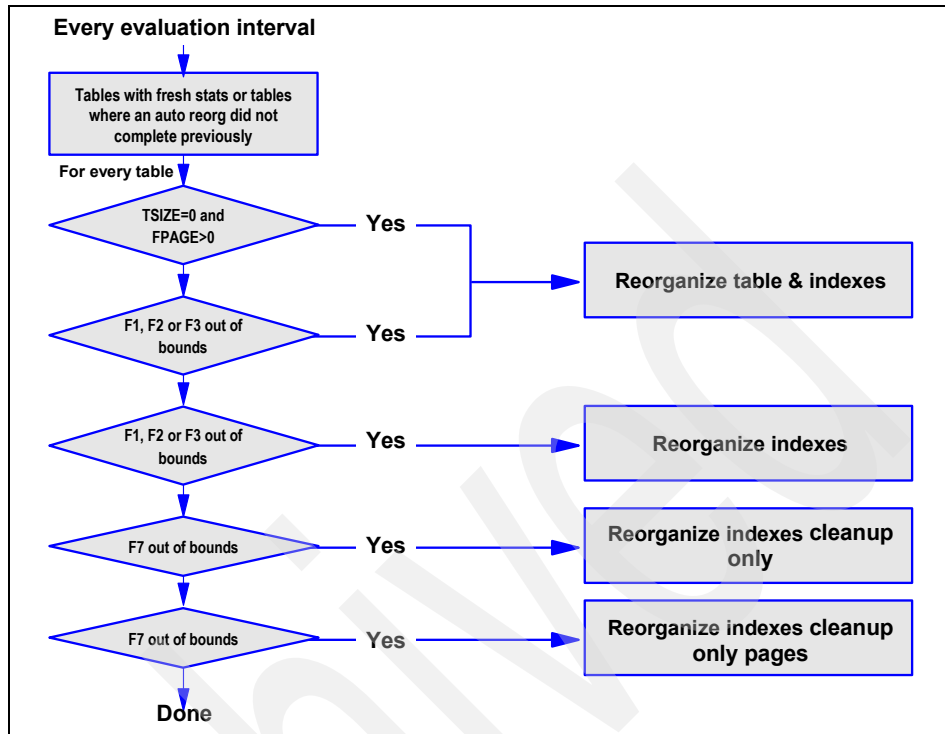


Figure 3-16 Automatic reorg evaluation

F1, F2, and F3 are the formulas used on table statistics; F4 to F8 are for index statistics. For details, refer to the REORGCHK command in *IBM DB2 Universal Database Command Reference*, SC09-4828-01.

The automatic table and index reorganization are run in the offline maintenance time period. It runs to completion even if the job goes beyond the specified time period. The internal scheduling mechanism learns over time and estimates job completion times. If the offline time period is too small for a reorganization activity, the scheduler will not start the job the next time around. A state-based collection health indicator in the Health Monitor provides notification of any reorganizing actions that cannot be handled using the defined maintenance objectives or any tables that require manual reorganization.

The process for setting up the automatic table reorganization using Configure Automatic Maintenance wizard is same as setting up automated RUNSTATS:

1. Refer to 3.1.1, “Automatic backup” on page 54 for steps 1 on page 54 to 4 on page 58. Only offline automatic reorg is supported in this release, so you must specify the offline maintenance window. In step 4 on page 58, click **Change** next to the Offline maintenance window to open a pop-up dialog.

2. In the Change Maintenance Window Specification - Offline Activity window, we specify that the offline maintenance window is every Sunday beginning at midnight for five hours (Figure 3-17). Click **OK** to close the window.

Change Maintenance Window Specification - Offline Activity

Specify when automatic maintenance can occur.

☒ During the specified time.

☐ Outside the specified time.

Specify the start time and the duration of the maintenance window. The start time is specified using a 24-hour clock.

Start time: 00:00

Duration: 5 hours

Specify how often this maintenance window occurs. A valid maintenance window must meet the conditions specified on both the Days of the Week tab and the Days of the Month tab.

Days of the week: Days of the month

☒ All

☒ Only on selected days

☐ Monday ☐ Friday

☐ Tuesday ☐ Saturday

☐ Wednesday ☒ Sunday

☐ Thursday

Preview

Offline automatic maintenance can occur **during** the following window.

Time	00:00 - 05:00 (5 hours)
Days of the week	Sunday
Days of the month	ALL
Activities using this window	None

OK Cancel Help

Figure 3-17 Automated reorg: Specify offline maintenance window

- Under Select a maintenance activity to configure, select **Defragment data (REORG)** and click **Configure Settings** to identify the tables to be maintained by DB2 (Figure 3-18).

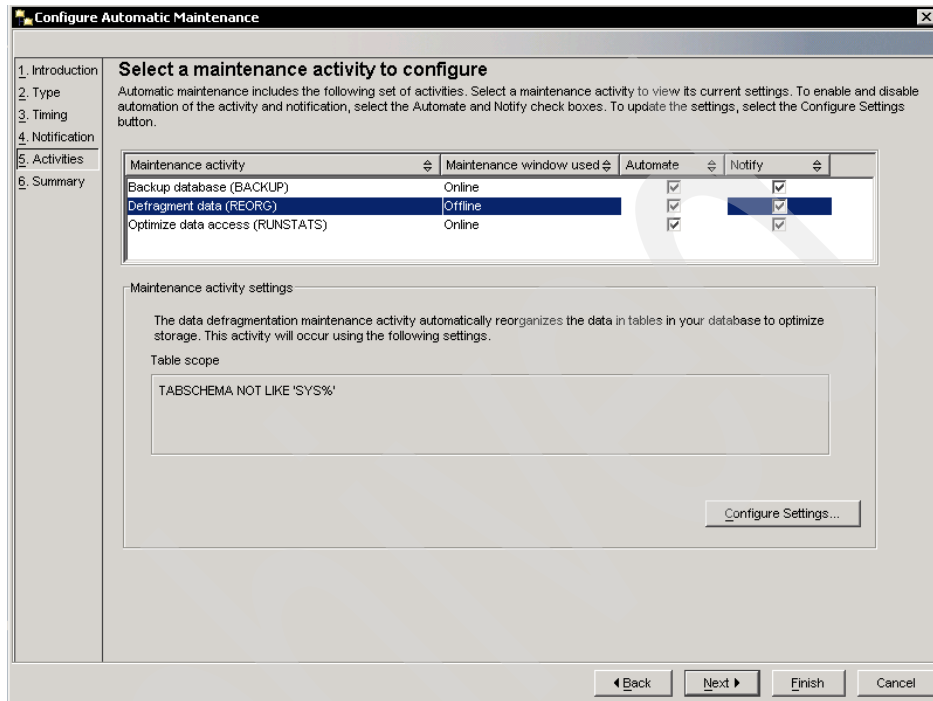


Figure 3-18 Automated reorg: select activity

4. This opens the Configure settings - Defragment data (REORG) window (Figure 3-19).

In this window, define the tables you want to exclude from the automatic data reorganization process. Note that the volatile tables are always excluded. When the filter is defined, click **Refresh Resulting Tables** to display all of the tables to be maintained. Review the list and, when it is complete, click **OK**.

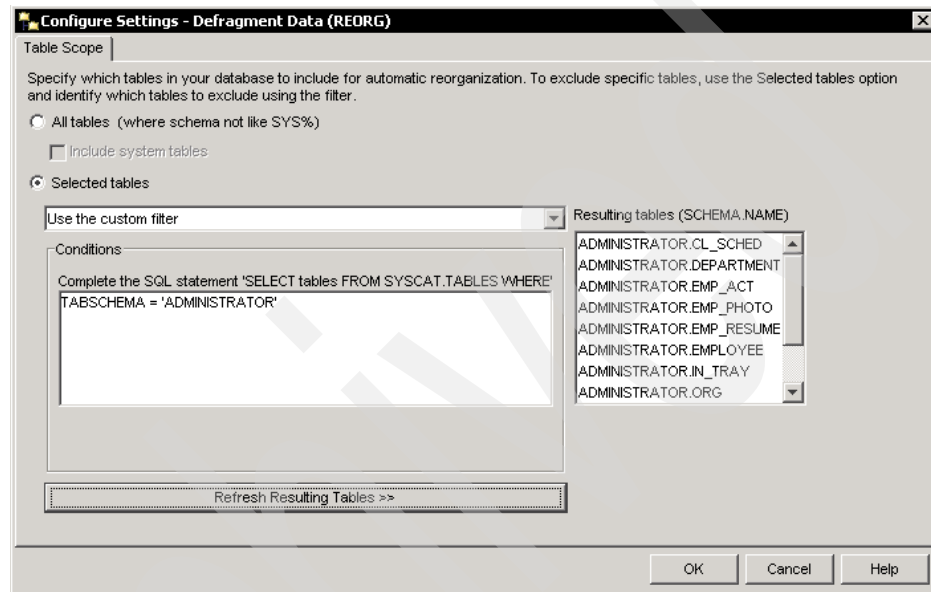


Figure 3-19 Automated reorg: configure setting

- Under Review the automatic maintenance settings, check the summary of the settings you have entered for the database. Figure 3-20 shows that reorg automation is enabled. Click **Finish** to process.

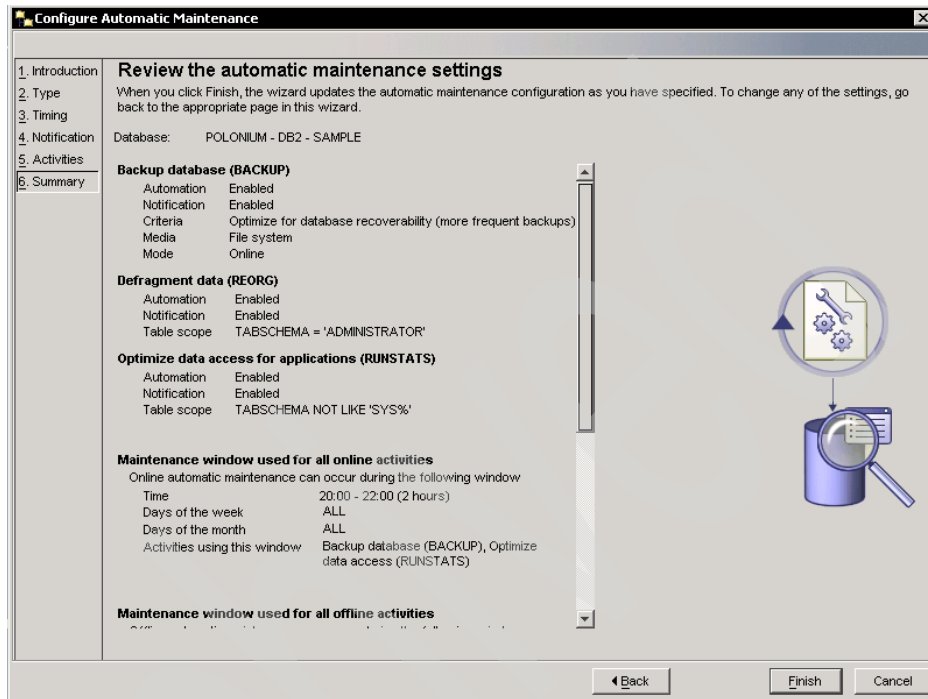


Figure 3-20 Automated reorg: summary

AUTO_REORG is the new database configuration parameter to enable or disable reorg automation. Example 3-4 is a snippet of the LIST DATABASE CONFIGURATION command output. It shows that the automatic table reorganization parameter AUTO_REORG has been turned on.

Example 3-4 Automated reorg database configuration parameter on

Database Configuration for Database sample

```
Database configuration release level      = 0x0a00
Database release level                  = 0x0a00
Database territory                      = US
Database code page                     = 1252
Database code set                      = IBM-1252
Database country/region code           = 1
Database collating sequence            = UNIQUE
Alternate collating sequence            (ALT_COLLATE) =
```

...

Automatic maintenance	(AUTO_MAINT) = ON
Automatic database backup	(AUTO_DB_BACKUP) = ON
Automatic table maintenance	(AUTO_TBL_MAINT) = ON
Automatic runstats	(AUTO_RUNSTATS) = ON
Automatic statistics profiling	(AUTO_STATS_PROF) = OFF
Automatic profile updates	(AUTO_PROF_UPD) = OFF
Automatic reorganization	(AUTO_REORG) = ON

3.5 Integrated Design Advisor

The new integrated Design Advisor can assist DBAs in making optimal and comprehensive database design decisions. Many DBAs would agree that the decisions made with respect to the design of a database are some of the most challenging, time consuming, and critical to make. This self-configuring tool greatly simplifies the design process by using workload, database, and hardware information to recommend specific performance acceleration options for routine design tasks.

The integrated Design Advisor consists of:

- ▶ Index advisor
- ▶ Materialized Query Table advisor
- ▶ Multidimensional Clustering advisor
- ▶ Partitioning advisor

The Design Advisor can help you significantly improve your workload performance. The common task of selecting which indexes, MQTs, clustering dimensions, or partitions to create for a complex workload can be quite daunting. The Design Advisor identifies all of the objects that are needed to improve the performance of your workload. Given a set of SQL statements in a workload, the Design Advisor will generate recommendations for:

- ▶ New indexes
- ▶ New materialized query tables (MQTs)
- ▶ Conversion to multidimensional clustering (MDC) tables
- ▶ Repartitioning of tables
- ▶ Deletion of indexes and MQTs unused by the specified workload

Also, you can order the Design Advisor to implement these recommendations immediately or schedule some or all of them for a later time.

The Design Advisor analyzes the characteristics (database and hardware resource) of your workload to consider the costs and advantages of each feature when formulating design recommendations. Then it weighs the performance improvements against the costs associated with implementing the recommendations.

Note: It is very important for the Design Advisor that you provide a representative query workload. If the workload that you submit is not representative of the overall database activity, the recommendations might not provide performance enhancements for queries not included in the workload. Also, it is critical that the statistics on any object that is involved in the workload are up-to-date. Inaccurate statistics might negatively affect the quality of the Design Advisor recommendations.

Because the Design Advisor performs the cost-benefit analysis for you, all features are selected by default for consideration. It is recommended that you leave this default setting and allow the Design Advisor to determine which features provide the greatest overall performance advantages for the workload that you specify. If you do not want to implement specific performance enhancement features, you can choose not to select those features for evaluation. Eliminating those features will reduce the amount of time that the Design Advisor spends evaluating different performance enhancement possibilities for your database.

Using either the Design Advisor GUI or the **db2adv** command-line tool, the Design Advisor can help simplify the following tasks:

► Planning for or setting up a new database or partitioning structure

While designing your database or database partitions, use the Design Advisor to:

- Generate design alternatives in a test environment for partitioning, indexes, MQTs, and MDC tables.
- Determine initial database partitioning before loading data into a database.
- Assist in migrating from a non-partitioned DB2 database to a partitioned DB2 database.
- Assist in migrating to DB2 in a partitioned environment from another database product.
- Evaluate indexes, MQTs, or partitions that have been generated manually.

► **Workload performance tuning**

After your database is set up, you can use the Design Advisor to help you meet the following tuning goals:

- Improve performance of a particular statement or workload.
- Improve general database performance, using the performance of a sample workload as a gauge.
- Improve performance of the most frequently executed queries, for example, as identified by the Activity Monitor.
- Determine how to optimize the performance of a new key query.
- Respond to Health Center recommendations regarding shared memory utility or sort heap problems in a sort-intensive workload.
- Find objects that are not used in a workload.

Optimizing workload performance with Design Advisor

A workload is a set of SQL statements that the database manager has to process during a given period of time. For example, during one month your database manager may have to process 1,000 INSERTs, 10,000 UPDATEs, 10,000 SELECTs, and 1,000 DELETEs. The Design Advisor analyzes a specified workload and considers factors such as the type of workload statements, the frequency with which a particular statement occurs, and characteristics of your database to generate recommendations that minimize the total cost to run the workload.

How to create a workload

For your workload you should have SQL statements that your applications run against the database that should be considered for optimization. You can paste these statements into a delimited text file, create a new workload file, or modify a previously existing file. To collect the SQL statements that run against your database, you can use several methods to get them from different sources:

- Extract them from an Event Monitor table.
- Take recently used SQL statements.
- Take them from a previously captured DB2 snapshot.
- Use the Query Patroller historical data tables.
- Get the explained statements from the EXPLAINED_STATEMENT table.

Showing this window (Figure 3-21) now is a little bit premature but it increases the pleasant anticipation. It can help you to save work and time to consider the selection possibilities of the Design Advisor.

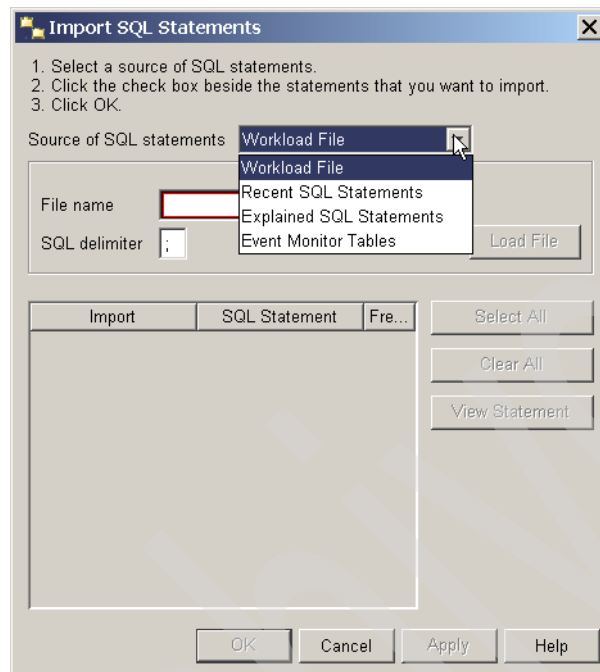


Figure 3-21 Show the different sources to get SQL statements for your workload

Using DB2 Design Advisor

We can start the GUI-based DB2 Design Advisor by the following ways:

- ▶ Start Control Center, right-click a database, then select **Design Advisor**, (Figure 3-22).

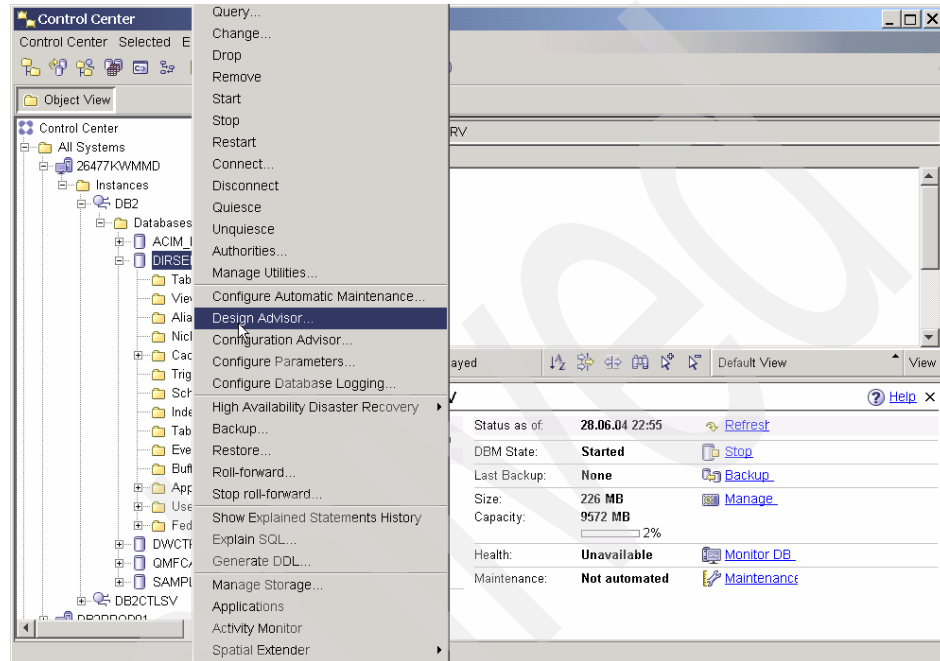


Figure 3-22 Selecting the Design Advisor from the DB2 Control Center

or

- Open **Tools** → **Wizards** → **Design Advisor** from the DB2 Control Center and select your database (Figure 3-23).

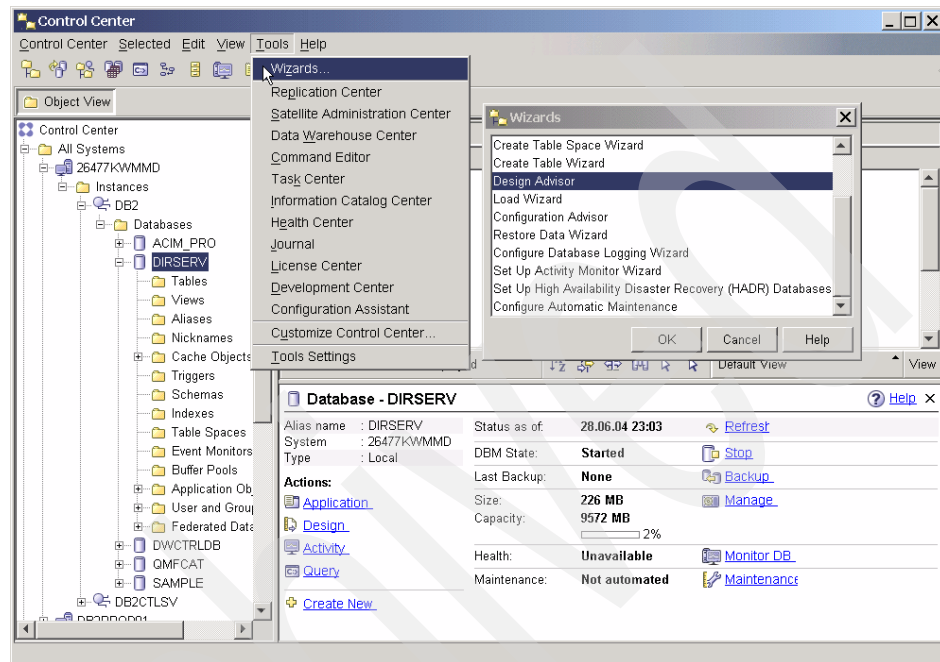


Figure 3-23 Select the DB2 Design Advisor from the Tools menu

There are 10 steps in Design Advisor:

1. In the first dialog of the Design Advisor, the Introduction (Figure 3-24), click the underlined items to see detailed information of the items in the DB2 Information Center.

Click **Next** or select the **Features** tab to continue.

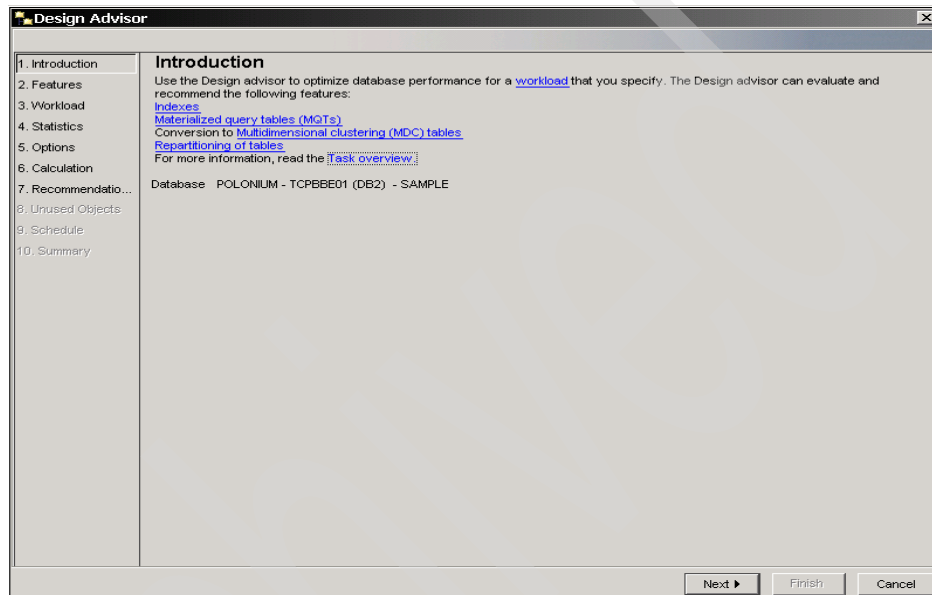


Figure 3-24 Introduction to the Design Advisor

2. Under Select performance features (Figure 3-25), specify the combination of features—Indexes, Materialized Query Tables (MQT), multidimensional clustering tables (MDC)—that you want the advisor to consider for workload optimization and click **Next**.

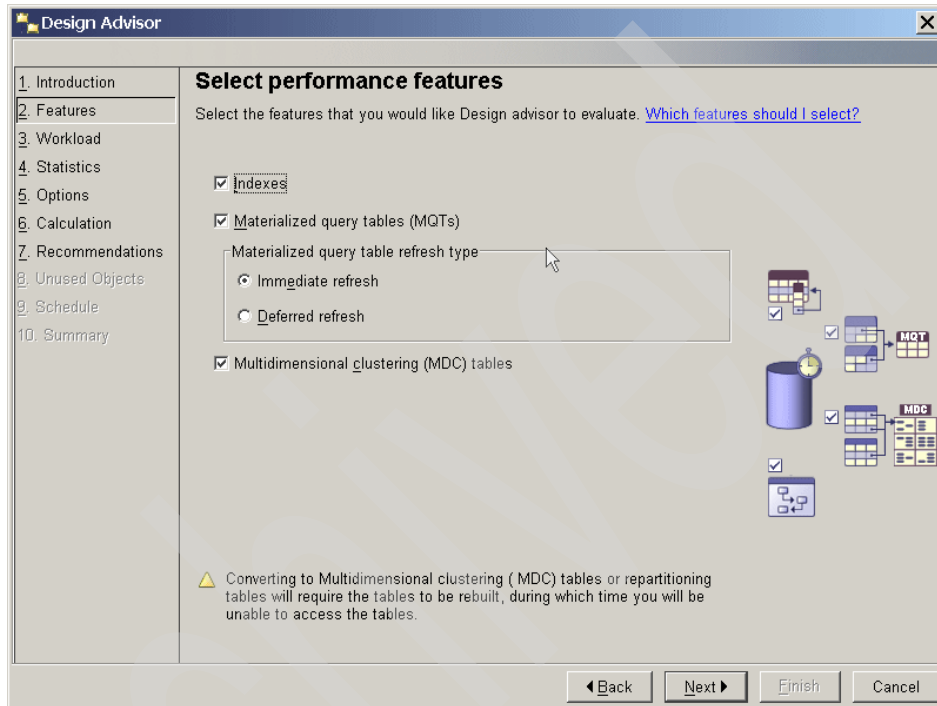


Figure 3-25 Design Advisor features

3. For Define a new workload (Figure 3-26), you can designate the workload that is used to calculate the recommendations. You can create a new workload or specify an existing one. After you have created and specified your workload, you can work with its SQL statements. You can add or import more statements, change the existing statements, or remove statements.

Consider whether you provided a representative query workload. If the workload that you submit is not representative of the overall database activity, the recommendations might not provide performance enhancements for queries not included in the workload.

Click **Next**.

The screenshot shows the 'Design Advisor' window with the 'Define a new workload' tab selected. The left sidebar contains a list of steps: 1. Introduction, 2. Features, 3. Workload (selected), 4. Statistics, 5. Options, 6. Calculation, 7. Recommendations, 8. Unused Objects, 9. Schedule, and 10. Summary. The main area is titled 'Define a new workload' and contains the following text: 'A workload is a set of SQL statements and their frequency. To add SQL statements to your new workload, click Import or Add. Recommendations are based on this workload'. Below this text are two input fields: 'Workload name' with an empty text box, and 'Schema to qualify the workload' with a dropdown menu showing '<Select>'. Under the heading 'SQL statements in workload', there is a large empty table with columns 'SQL Statement', 'Name', and 'Frequency'. To the right of the table are four buttons: 'Import...', 'Add...', 'Change...', and 'Remove'. At the bottom of the window are four buttons: 'Back', 'Next', 'Finish', and 'Cancel'.

Figure 3-26 Defining a new workload

4. The first time you start the Design Advisor, you will be prompted to create a new workload. You can now import your SQL statements that define your workload or add all of the statements individually. We discuss the importing workload feature first. To import the workload, click **Import**, and the Import SQL Statements window opens (Figure 3-27). This feature is very powerful and gives us the opportunity to easily extract the SQL statements from different sources.

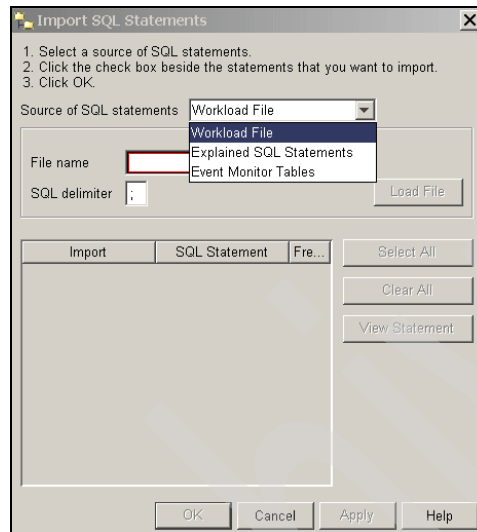


Figure 3-27 Select the import source for your workload

Figure 3-28 shows an example of selecting a predefined file with appropriate SQL statements.

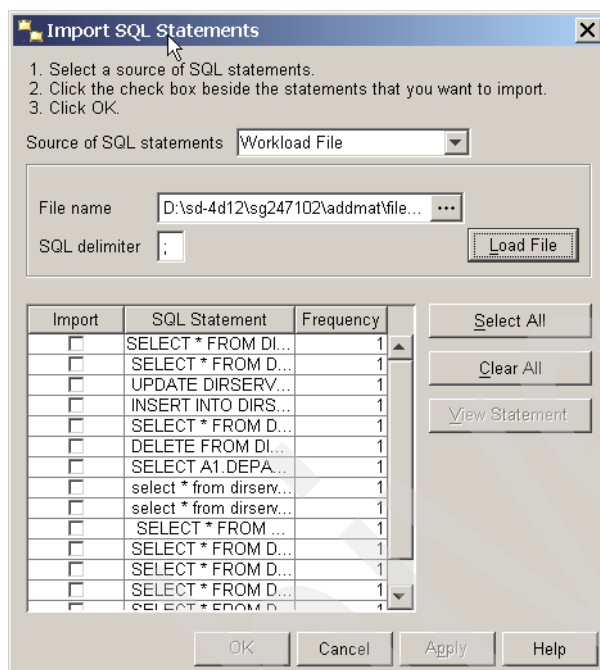


Figure 3-28 importing a workload file

When you click **Load File**, the Design Advisor Wizard reads the SQL statements from your specified file. You now can select all or specific statements. To adjust the frequency of these statements, press **OK** after selection, then **Change**.

You can also change the SQL statement (Figure 3-29).

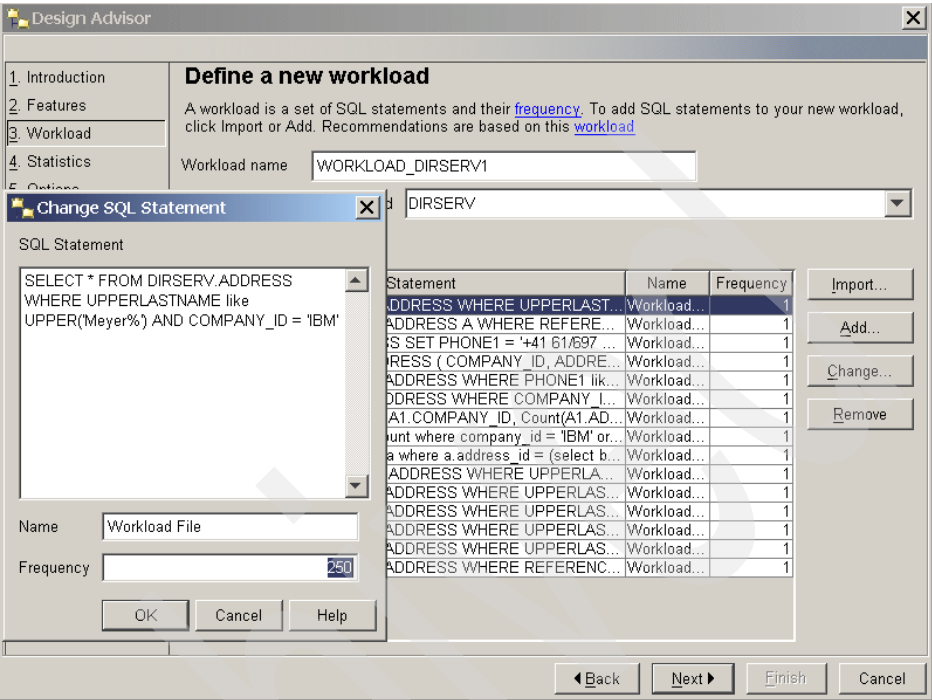


Figure 3-29 Adjust the frequency of your SQL statement

If you have already created a workload, you will see a different dialog window: Specify a workload. Your workloads are all stored in DB2 tables and very easy to retrieve after you have specified them (Figure 3-30).

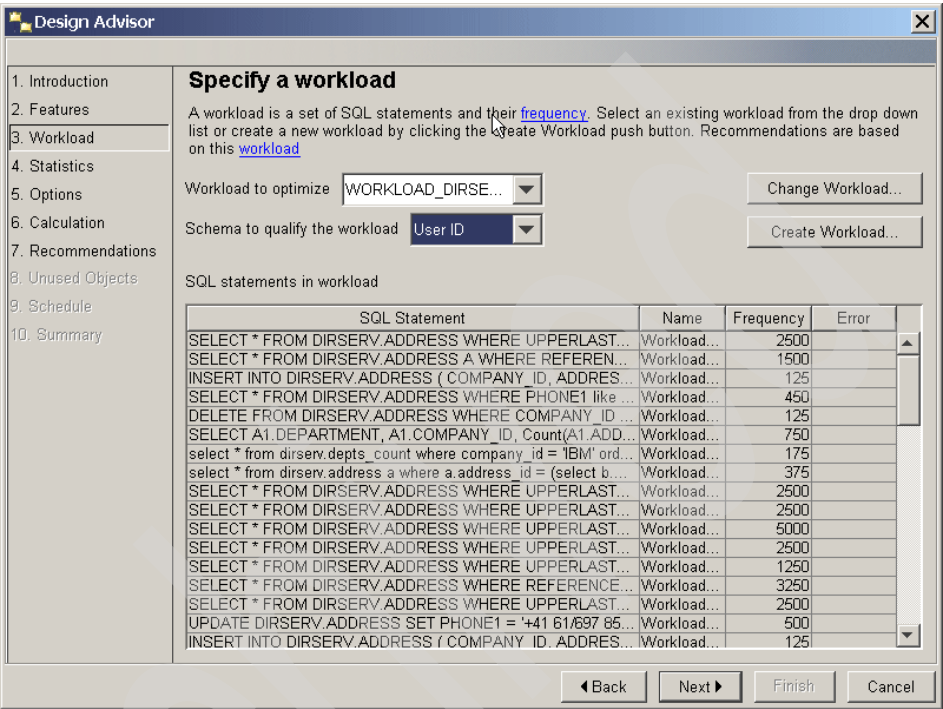


Figure 3-30 The workload dialog if there are already specified workloads available

Clicking **Change Workload** opens the dialog in Figure 3-31 on page 99.

In this dialog, you can adjust, add (also by importing new ones from different sources), or delete the SQL statements.

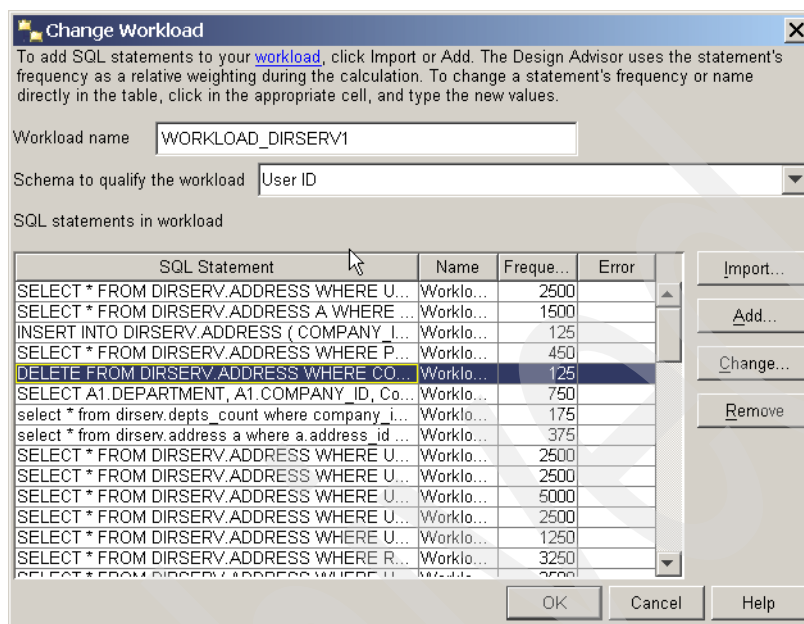


Figure 3-31 Change an already existing workload

We now return to selecting the SQL statements of your workload. Figure 3-32 shows an example of using SQL statements captured by the EXPLAIN feature of DB2. In the Import SQL Statements window, select **Explained SQL Statement** from the Source of SQL statement drop down menu.

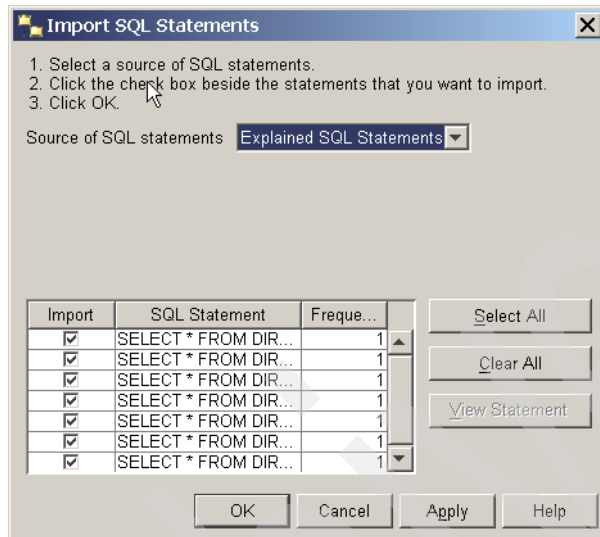


Figure 3-32 Use Explained SQL Statements for your workload

In this way, you can select SQL statements captured by the DB2 Event Monitor (Figure 3-33).

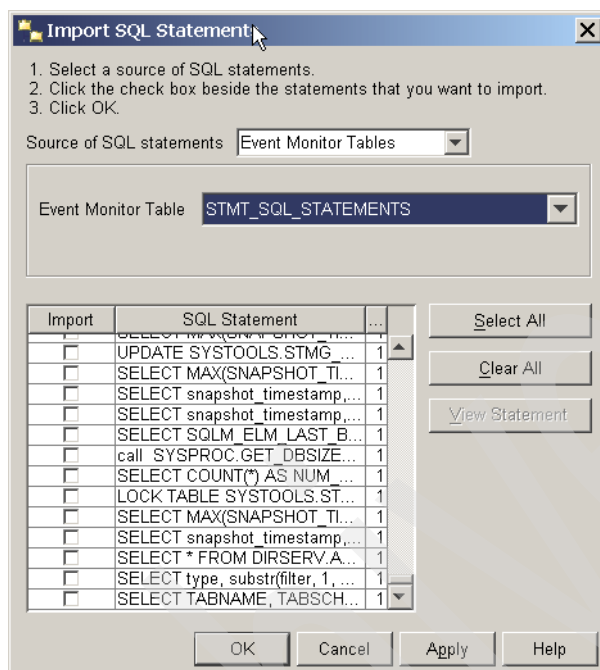


Figure 3-33 Using SQL statements for your workload captured by the Event Monitor

After specifying our workload, much of our preparation work has been done. The next dialog step recommends using actual statistics for our workload. You can speed up the process of the RUNSTATS commands by choosing only the tables and views specified in the workload.

5. Under Update Catalog Statistics (Figure 3-34), you can specify selected tables on which to update the table statistics. For the best recommendations from the advisor, your table statistics should be current. Select the tables that you want to update the statistics for. The update will occur just prior to the calculation of the recommendations.

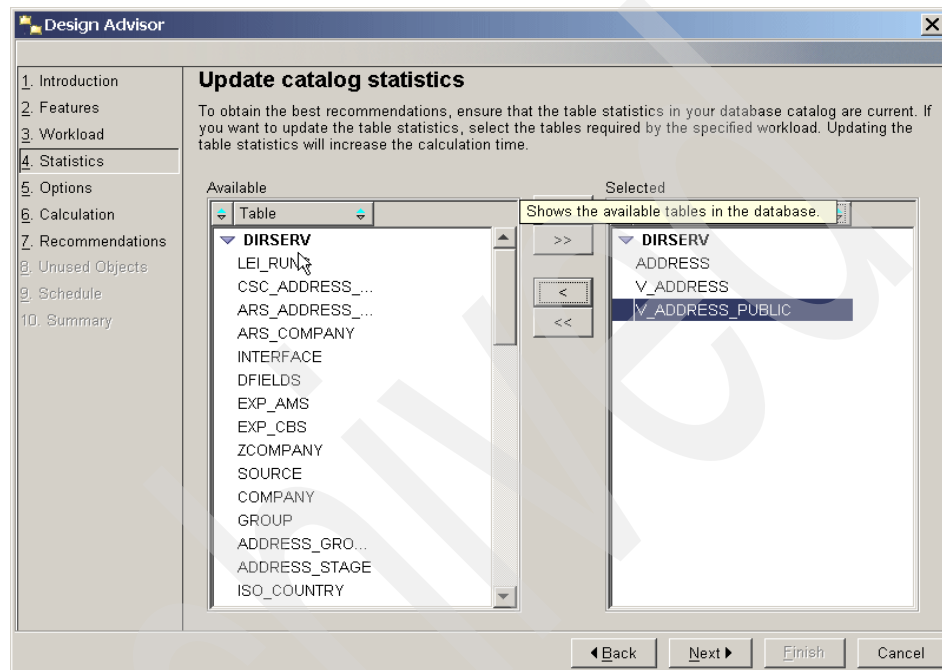


Figure 3-34 Update the catalog statistics by doing RUNSTATS before the run

Updating table statistics will increase the calculation time of the recommendations. (The time limit you set on the Selecting when to calculate the recommendations page does not include the time required for a runstats update.) Click **Next**.

6. Under Specify recommendation options (Figure 3-35), you can specify how much disk space may be needed for the recommended objects and which table spaces you can use for your MQTs. Use this page to set a maximum disk space usage for the creation of the recommended optimization objects and to specify a table space and schema for creating materialized query tables in. If disk space is not an issue and you want the best set of recommendations, do not set a disk space limit. Click **Next**.

The screenshot shows the 'Design Advisor' window with the 'Specify recommendation options' tab selected. On the left is a navigation pane with steps 1 through 10, where 'Options' is highlighted. The main area contains the following elements:

- Title:** Specify recommendation options
- Text:** You can choose to set and specify maximum disk space for recommended objects. For materialized query tables, you can select the table space and schema, and whether to derive statistics by sampling.
- Checkboxes:**
 - ☒ Set maximum disk space for the recommended objects: 500 MB
 - ☐ Derive statistics by sampling
- Hints:**
 - Hint: Specifying a disk limit might reduce the number of recommended objects to fit in the given disk space.
 - Hint: If you select to derive statistics by sampling, the Design advisor can produce better MQT recommendations, but the calculations will take longer to complete.
- Materialized query table options:**
 - Table space: AUTOMATIC SELECT
 - Schema name: AUTOMATIC SELECT
- Diagram:** A small diagram on the right shows a database icon with an arrow pointing to a grid of colored squares (purple, yellow, and grey).
- Buttons:** Back, Next, Finish, and Cancel.

Figure 3-35 Specify the recommendation options

7. Under Set when to calculate the recommendations (Figure 3-36), specify when you want DB2 to calculate the optimization recommendations. You can choose to have the calculations start immediately or schedule the calculation to start at a later time. You can also decide how long you want the advisor to calculate the recommendations by setting a maximum calculation time. (The time limit you set does not include the time required for a runstats update.).

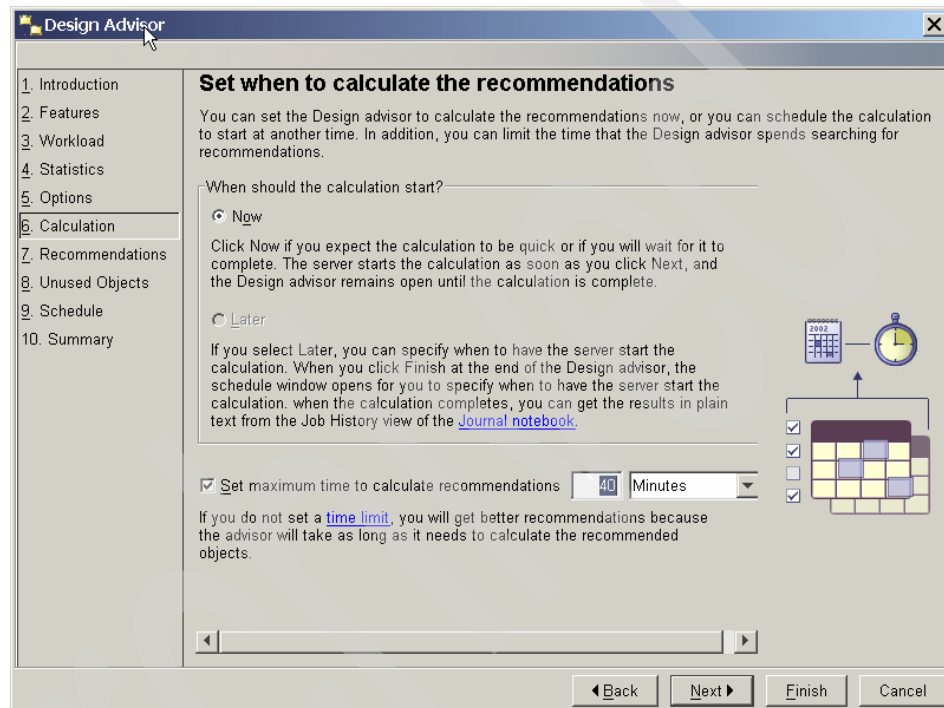


Figure 3-36 Adjust the calculation recommendations

8. Under Select the recommendations (Figure 3-37), review the recommended actions for performance improvement and select the changes that you want to have implemented. You can also see the estimated cost of implementing all of the recommendations in terms of disk space and the projected performance improvements as a percentage.

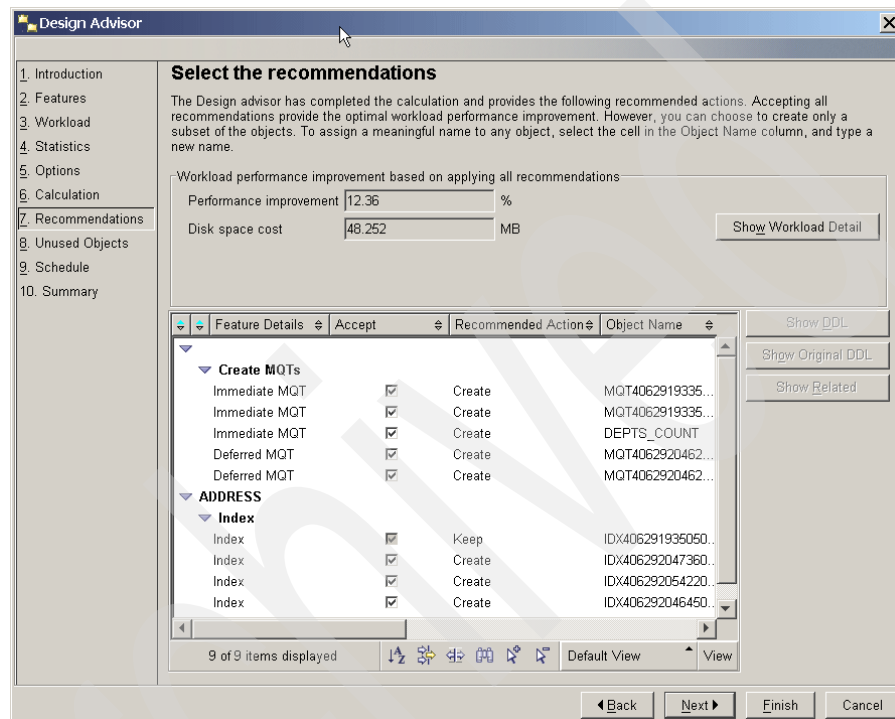


Figure 3-37 Design Advisor recommendations

We can see that selecting a later schedule for the calculations is not possible in our test case. The reason is that no tools database exists in which the schedules could be stored. We can create this database on the **Schedule** dialog step later. Click **Next** to proceed.

Figure 3-37 shows the Design Advisor recommendations for our example. By creating Materialized Query Tables and new indexes that use less than 50 MB of disk space, we can reach a performance improvement of more than 12% in our example.

The DB2 Design Advisor recommends that we create Materialized Query Tables and Indexes to optimize the table access.

After selecting a feature detail, the Show DDL button will be enabled and you can easily copy and paste the DDL for a recommendation (Figure 3-38).

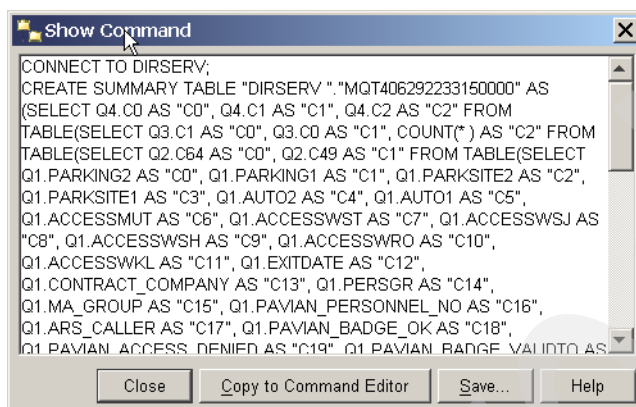


Figure 3-38 DDL statements generated by the Design Advisor

Click **Show Workload Details** to see the execution costs before and after implementing the recommendation of Design Advisor (Figure 3-39).

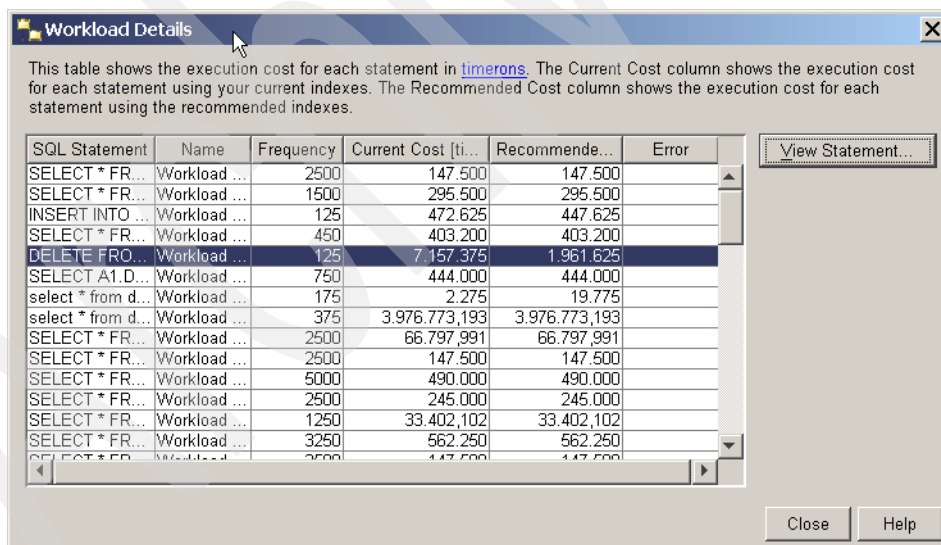


Figure 3-39 Details of the calculation before and after the run of the workload

If there are errors in SQL statements of a workload, the Design Advisor will skip such statements and inform you about it after calculating its recommendations (Figure 3-40).

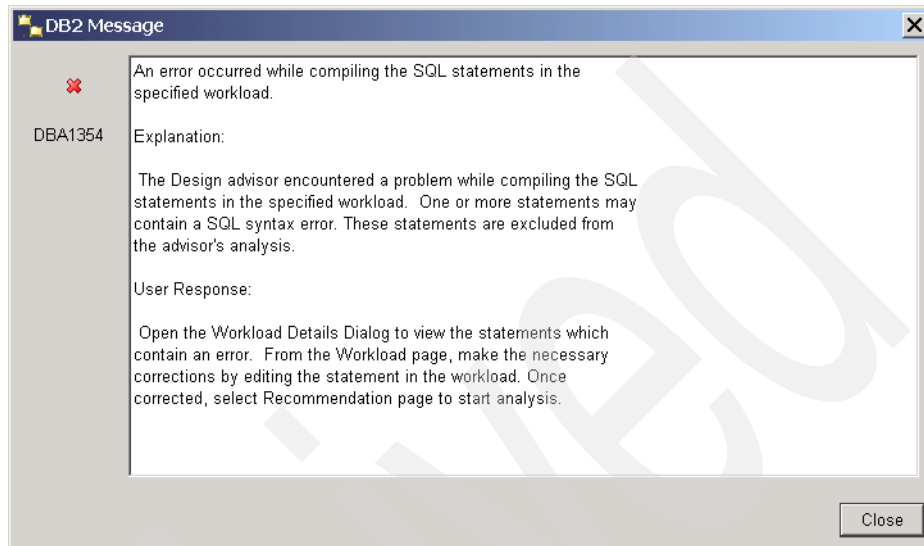


Figure 3-40 Errors in a workload's SQL statements

To see the SQL statements that caused these errors, click **Show Workload Detail**. To correct errors, return to the **Workload** tab (step 3 on page 94).

Click **Change Workload** in the Workload tab and scroll down the list of workload SQL statements to see which statement caused the error (Figure 3-41). Click **Change**, copy the statement and paste it in a DB2 command editor window, correct it, and return it to the workload design facility.

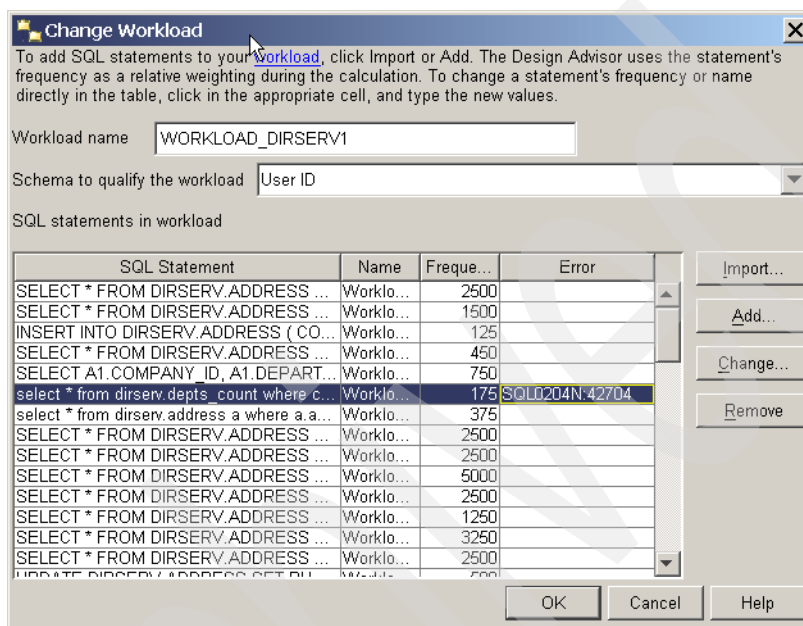


Figure 3-41 The SQL statement which caused the error

9. Review unused objects (Figure 3-42) shows the database objects that were identified by the Design Advisor's calculations as unnecessary in the specified workload. Unused database objects must be maintained by the database manager, and they consume CPU and disk space. Remember: An index slows down write operations on tables when doing Inserts, Updates, and Delete statements.

Note: Delete unused objects only if you are really sure that they are not needed by any SQL statements that are not included in your workload.

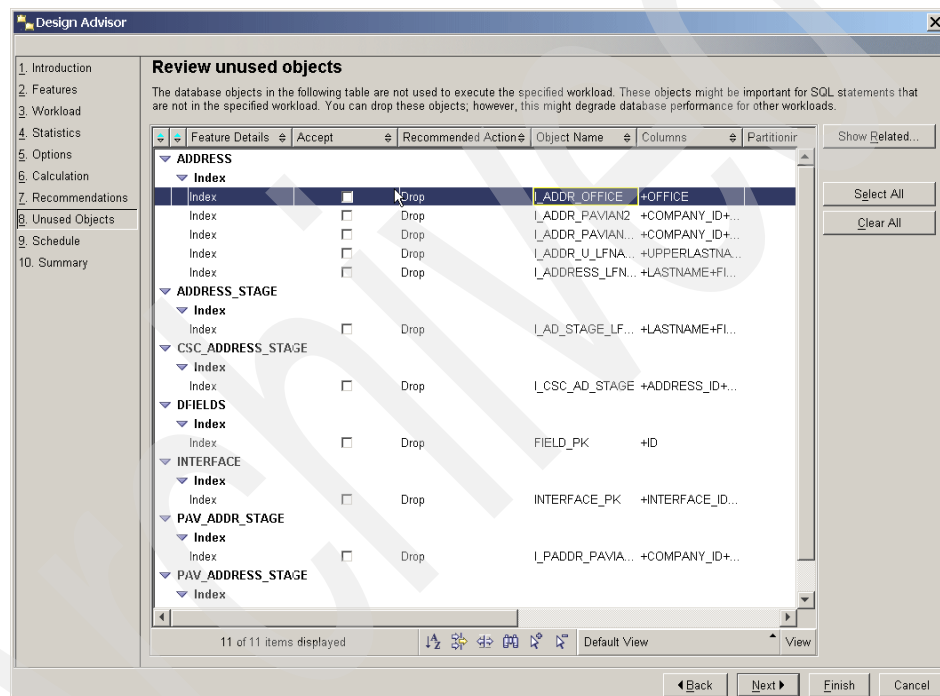


Figure 3-42 Review unused database objects

To drop these unused objects, select them to include them in the tuning process. All of these proposed changes in your database can be run now or done by the scheduler at a time when the current operations of the databases and the system will not be disturbed by the additional load of creating indexes and MQT tables. Click **Next**.

10. Under Scheduling tasks execution (Figure 3-43), you can choose to generate a recommendations report, execute the recommendations immediately, or create a task in the Task Center. Creating a task enables you to schedule task execution and to maintain its history.

The screenshot shows the 'Design Advisor' window with the 'Scheduling task execution...' dialog box open. The dialog box has a sidebar on the left with a list of steps: 1. Introduction, 2. Features, 3. Workload, 4. Statistics, 5. Options, 6. Calculation, 7. Recommendations, 8. Unused Objects, 9. Schedule (highlighted), and 10. Summary. The main area of the dialog box is titled 'Scheduling task execution...' and contains the following elements:

- A text box: 'You can select whether to execute the commands immediately or create a task in the Task Center. Creating a task allows you to schedule task execution and maintain its history.'
- A button: 'Generate recommendation report' with a 'Generate' button next to it.
- Two radio buttons: 'Run now without saving task history' (unselected) and 'Create this as a task in the Task Center' (selected).
- Two dropdown menus: 'Run System' (set to '26477KWMMD') and 'Scheduler System' (set to '26477KWMMD'), with an 'Advanced...' button to the right of the second dropdown.
- A text box: 'Task name' (set to 'Create Recommended Objects - 3').
- Three radio buttons: 'Save task only' (unselected), 'Save and run task now' (unselected), and 'Schedule task execution' (selected).
- A 'Details' section with a text box containing 'Run once, on 01.07.04 at 04:00:00.' and a 'Change...' button to its right.
- A 'Runtime authorization' section with a 'User ID' text box (set to 'Administrator') and a 'Password' text box (set to '*****').

At the bottom of the dialog box are four buttons: 'Back', 'Next', 'Finish', and 'Cancel'.

Figure 3-43 Scheduling to create the Design Advisor recommended objects

If you have not specified a Tools database where your scheduling can be stored, the dialog step shown in Figure 3-44 will appear.

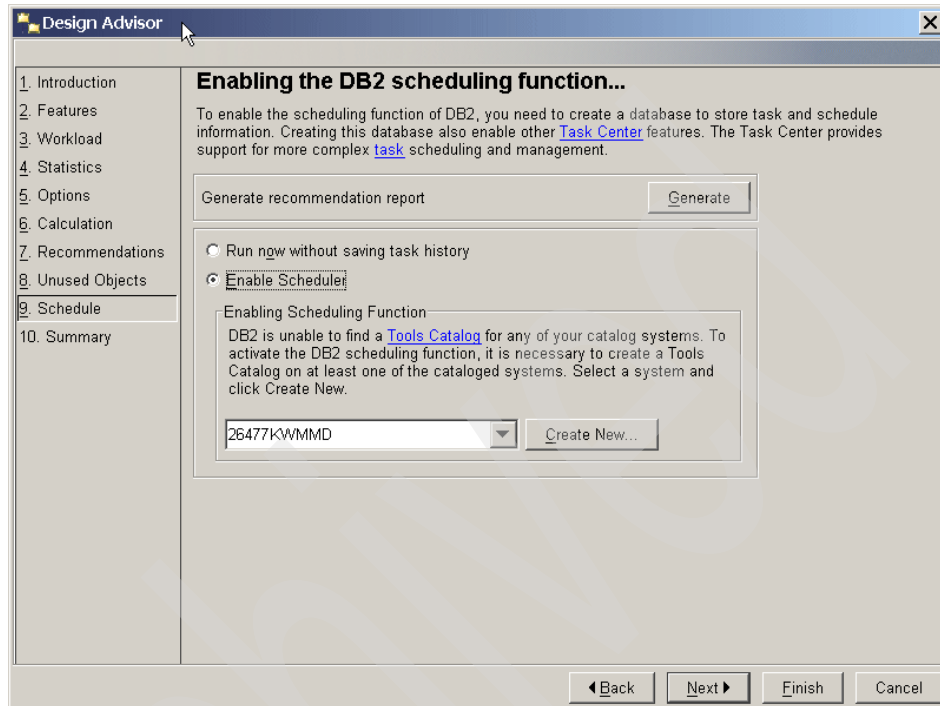


Figure 3-44 Create the Tools Catalog database to enable the DB2 Scheduling function

By clicking **Create New** under Enabling the DB2 scheduling function, you can specify the details for the new Tools Catalog database (Figure 3-45).

The screenshot shows a Windows-style dialog box titled "Create New Tools Catalog - 26477KWMMD". It contains the following fields and options:

- Instance:** A dropdown menu showing "DB2".
- Tools Catalog schema:** A text field containing "TOOLS".
- Database options:**
 - ☒ **Create Tools Catalog in a new database**
 - New database name:** A text field containing "TOOLS".
 - ☐ **Create Tools Catalog in an existing database**
 - Database:** A dropdown menu showing "DIRSERV".
 - ☐ **Use an existing table space** (disabled)
 - TB_ADDRESS:** A dropdown menu showing "TB_ADDRESS".
- ☐ **Force all applications for instance restart**
- ☒ **Activate Tools Catalog**
- Buttons:** "OK", "Cancel", "Show Command", and "Help".

Figure 3-45 Create the Tools Catalog database

After creating this database, you can schedule Design Advisor tasks in the Scheduling task execution dialog (Figure 3-43 on page 110). This window also gives you the option of creating a recommendation report, which is stored as a text file (Figure 3-46), by clicking **Generate**.

```

UltraEdit-32 - [D:\Temp\db2_AD_recsep.txt*]
File Edit Search Project View Format Column Macro Advanced Window Help
db2_AD_recsep.txt
Filter: [ ] Refresh
Open Files
D:\Temp\db2_AD_recsep.txt
1 -- DB2 Design Advisor Recommendation Report
2 -- Database Name : DIRSERV
3 -- Time : 2004-06-30 16:52:43.2
4 -- Workload Name : WORKLOAD_DIRSERV1
5 -- Workload Schema : null
6 -- Recommendation Features : IMC
7 -- Running Time : 0 Minutes
8 -- Update Catalog Statistics : false
9 -- Performance Improvement : 0.5
10 -- Disk Cost: 28.199312
11 -- ADDRESS      Index      Index      true      Create  IDX407010116540000      +REFERENCE
12 -- ADDRESS      Index      Index      true      Keep    I_ADDR_ADDRESS_ID      +ADDRESS_ID
13 -- Create MQTs   Immediate MQT      true      Create  DEPTS_COUNT
14 -- Create MQTs   Deferred MQT       true      Create  MQT407010039530000
15 -- ADDRESS      Index      Index      true      Drop    I_ADDR_OFFICE      +OFFICE
16 -- ADDRESS      Index      Index      true      Drop    I_ADDR_PAVIAN2      +COMPANY_ID+PERSONN
17 -- ADDRESS      Index      Index      true      Drop    I_ADDR_PAVIANKEY     +COMPANY_ID
18 -- CSC_ADDRESS_STAGE      Index      Index      true      Drop    I_CSC_AD_STAGE      +ADDRESS_ID
19 -- PAV_ADDR_STAGE      Index      Index      true      Drop    I_PADDR_PAVIANKEY    +CO
20 -- PAV_ADDRESS_STAGE      Index      Index      true      Drop    I_PAVADD_PAVIANKEY   +CO
21 -- INTERFACE      Index      Index      true      Drop    INTERFACE_PK         +INTERFACE_ID+INTER
22 CONNECT TO DIRSERV;
23 CREATE INDEX "DIRSERV " ."IDX407010116540000" ON "DIRSERV " ."ADDRESS" ("REFERENCE" AS
24 CREATE INDEX "DIRSERV " ."I_ADDR_ADDRESS_ID" ON "DIRSERV " ."ADDRESS" ("ADDRESS_ID" AS
25 ;
26 CREATE SUMMARY TABLE "DIRSERV " ."MQT407010039530000" AS (SELECT Q4.C0 AS "C0", Q4.C1
27 CREATE SUMMARY TABLE "DIRSERV " ."MQT407010039540000" AS (SELECT Q4.C0 AS "C0", Q4.C1
28 CREATE SUMMARY TABLE "DIRSERV " ."MQT407010039290000" AS (SELECT Q3.C0 AS "C0", Q3.C1
29 CREATE SUMMARY TABLE "DIRSERV " ."MQT407010039490000" AS (SELECT Q3.C0 AS "C0", Q3.C1
For Help, press F1      Ln 1, Col. 1, CW      UNIX      Mod: 30.06.2004 16:52 File Size: 8008      JMS

```

Figure 3-46 An excerpt of a DB2 Design Advisor recommendation report

Click **Advanced** in the Scheduling task execution dialog (see Figure 3-43 on page 110) to open the Advanced Scheduling Settings window (Figure 3-47). Here you can select centralized or local server scheduling, and create a Tools Catalog database. Make your selections and click **OK**.

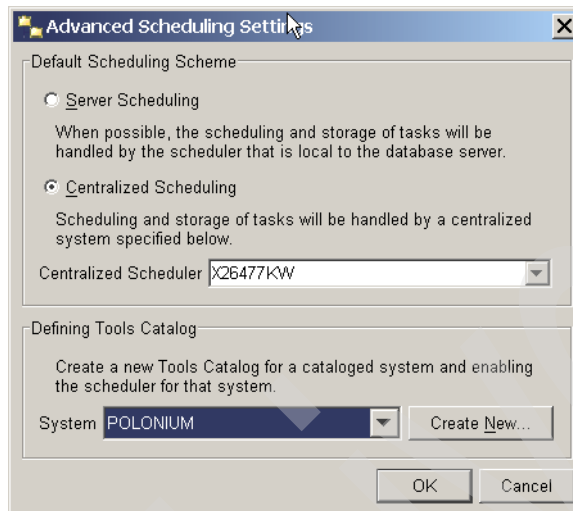


Figure 3-47 Advanced Scheduling

11. The Summary page that appears as last step of the Design Advisor wizard is dependent on your selections on the Set when to calculate the recommendations page (Figure 3-36 on page 104). If you chose to run the task immediately, the Summary page shows a list of objects (materialized query tables and indexes) that will be created and a list of objects that will be dropped. You can review the objects, go back to the Design Advisor and make any necessary changes, and then click **Finish** to execute the commands.
- If you have scheduled the recommended commands you still can review and change them in the DB2 Task Center. Open it from the DB2 Control Center with the **Tools** → **Task Center** menu (Figure 3-48) and select the Scheduler system on which the task will run.

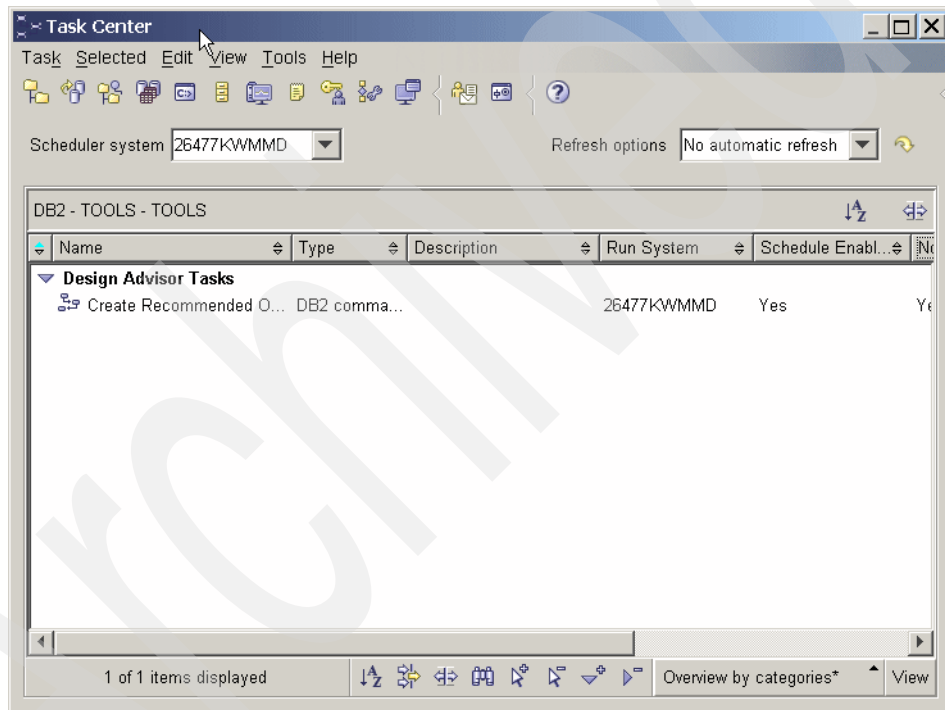


Figure 3-48 The created recommendation run task in the DB2 Task Center

Security

In this chapter, we discuss the new security enhancements of DB2 UDB V8.2 in the Windows environment. A detailed description of Windows Security mechanisms in connection with DB2 UDB V8.1 security functions can be found in the redbook *DB2 UDB Exploitation of the Windows Environment*, SG24-6893.

This chapter contains the following topics:

- ▶ More consistent user IDs and group names with the Windows standards
- ▶ Enhanced support for Active Directory domains
- ▶ Running Windows Services for DB2 under the local system account (LSA)
- ▶ Data encryption
- ▶ New security plug-in mechanisms

4.1 Windows Domain handling and user IDs

To improve DB2 UDB integration with the Windows Server environment and with Active Directory support, these limitations have been removed in Version 8.2:

- ▶ Limitation of 8-character group names
- ▶ Limitation of the allowed character set in user and group names
- ▶ Limitation on the support of the NETAPI32 application programming interface for group lookup that has no token support included

4.1.1 Improved Windows Domain and Active Directory support

DB2 UDB Version 8.2 now provides the following services:

- ▶ Support using cached credentials for DB2 authentication and group lookup
- ▶ Support for nested group semantics
- ▶ Support for Domain Local Groups
- ▶ Support for implicit trusts between Windows Domains

DB2 UDB V8.2 acquires Windows user's group information by using an access token that contains complete information about the security context of a process and its threads. It is a protected object that is generated by the Windows OS after a successful logon and contains identity and privileges held by a user account. After you have been authenticated to the system, all credentials are cached by the operating system. Among other things, it contains the following information:

User	The Security Identifier (SID) for this User account.
Group	The list of the SIDs of all groups the user belongs to. This includes local groups and various domain groups such as global groups, domain local groups, and universal groups.
Privileges	The list of the privileges held by the user and its groups.

Every process that runs on behalf of your user ID will be provided with this access token. It can be referenced in the systems cache when it is not possible to get in contact with the domain controller and contains the information of the last successful logon.

Access token support for group lookup is not the default lookup method. You must turn this on by setting DB2_GRP_LOOKUP=xxx,yyy, where xxx is the old lookup mode (LOCAL,DOMAIN or blank) and yyy is the access token lookup mode (TOKEN_LOCAL, TOKEN_DOMAIN, or TOKEN).

Note: If your DB2 client uses *client authentication* on a remote connection, then group lookup cannot be supported because the access token support is executed on the system the DB2 server runs. All other authentication types are supported:

- ▶ SERVER
- ▶ SERVER_ENCRYPT
- ▶ KRB_SERVER_ENCRYPT
- ▶ DATA_ENCRYPT
- ▶ DATA_ENCRYPT_CMP

Note: KERBEROS authentication (KRB_SERVER_ENCRYPT) can also support access tokens, as long as the system is homogeneously Kerberos (that is, the client and server are both Windows 2000, Windows Server 2003 and belong to a Windows 2000, Windows Server 2003 domain).

A complete detailed description about the Windows access control mechanism can be found in the Windows 2000 Server Resource Kit documentation. Information about the Window 2000 Server Resource Kit can be found at:

<http://www.microsoft.com/windows2000/techinfo/reskit/default.asp>

How to set up DB2 to use access token support

With the ability to set up access token support in DB2, DB2 administrators can define the range of groups a user belongs to that the DB2 UDB system will evaluate. Access Token support looks up groups for the user who is connecting to DB2. Other users group information cannot be obtained through the access token support, and DB2 will fall back to use the LANMAN API. For example, in a complex SQL statement that requires authority checking on multiple AUTHIDs, only the group information of the currently connected user can be obtained using Access Token. Other AUTHIDs still rely on the LANMAN API for group enumerations (because we do not have enough credentials to acquire the corresponding Access Token).

DB2 access token support must be activated by using of the **db2set** command to set the registry variable DB2_GRP_LOOKUP. This variable can have the following values:

- ▶ not set
- ▶ LOCAL
- ▶ DOMAIN
- ▶ TOKEN
- ▶ TOKENLOCAL
- ▶ TOKENDOMAIN

The last three values were introduced in DB2 UDB V8.2 and have to be combined with those that were available previously.

- ▶ **TOKEN:** for looking up all groups a user belongs to

This option forces DB2 to enumerate the groups and nested groups of the domain where the user was created and the local server users where DB2 runs.

This option uses the access token to look up groups the user belongs to by combining the membership of the local server and at the domain where the user account is defined if it is a domain account. To set the registry variable:

```
db2set DB2_GRP_LOOKUP=,TOKEN
```

- ▶ **TOKENLOCAL:** for looking up local groups on the DB2 UDB server's system

This option uses Access Token to look up local groups and fall back to use the LANMAN API to look up local groups. To set the registry variable:

```
db2set DB2_GRP_LOOKUP=LOCAL,TOKENLOCAL
```

- ▶ **TOKENDOMAIN:** for looking up all domain groups in the domain a user belongs to

This option uses the access token to look up domain groups and fall back to use the LANMAN API to look up domain groups. If the user is a local user (defined on DB2 SERVER), local groups are enumerated. This is the existing behavior of LANMAN API and token lookup follows this. To set the registry variable:

```
db2set DB2_GRP_LOOKUP=DOMAIN,TOKENDOMAIN
```

Note: The parameters of DB2_GRP_LOOKUP values are positional and take the TOKEN parameters as second parameter. Notice the comma in:

```
DB2_GRP_LOOKUP=,TOKEN
```

You must restart the DB2 instances after updating the DB2_GRP_LOOKUP variable.

Active Directory support examples

The Active Directory (AD) domain db2test.almaden.ibm.com consists of two Windows 2003 Domain Controllers and one Windows 2003 member server (Figure 4-1). All three servers have DB2 UDB V8.2 Enterprise Edition installed.

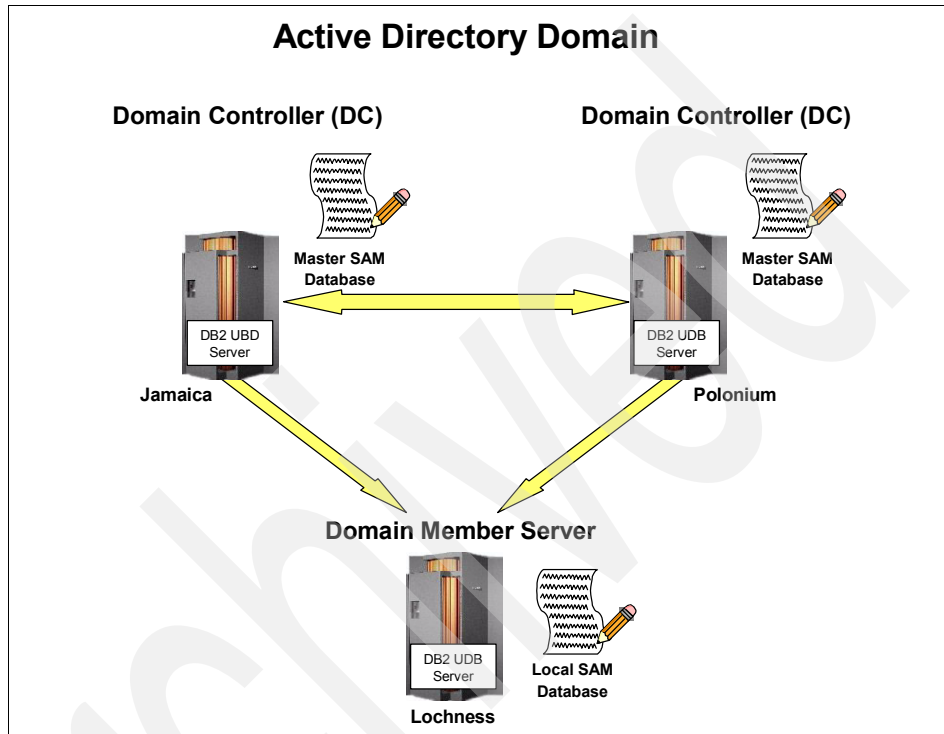


Figure 4-1 Topology of the Active Directory domain db2test.almaden.ibm.com

Nested group lookup example 1

All of the following groups and user IDs have been defined in the Active Directory domain db2test.almaden.ibm.com.

User Joe F. Miller is a member of the Windows global security group DB2_GLOBAL_GROUP (Figure 4-2).

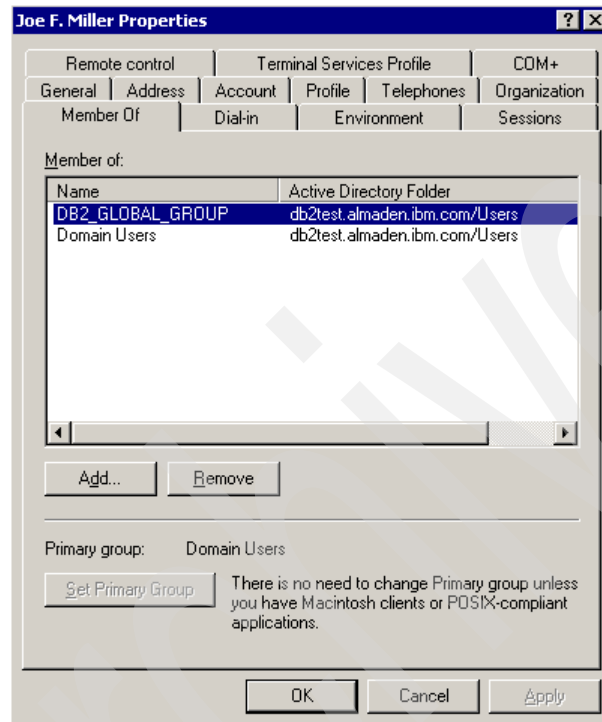


Figure 4-2 A user in a global group that is not directly authorized in DB2 UDB

This group is a member of the domain local security group DB2_SAMPLE_READERS (Figure 4-3). If Joe F. Miller logs on to his DB2 application, the group Joe F. Miller's account is in will be retrieved by DB2 UDB. All security groups that Joe F. Miller is in will be enumerated. DB2 UDB verifies whether one of the groups Joe F. Miller is in has the required authorizations to access the SAMPLE database.

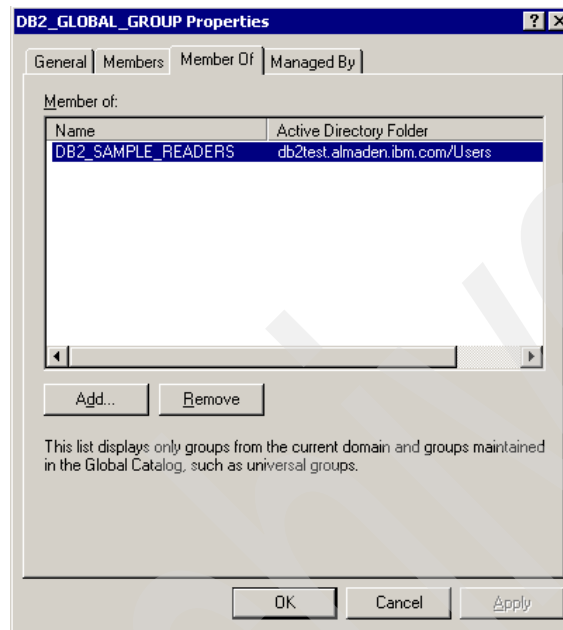


Figure 4-3 The DB2_SAMPLE_READERS group, which is authorized in DB2 UDB

Joe F. Miller is a member of the group DB2_GLOBAL_GROUP but not directly defined in the group DB2_SAMPLE_READERS.

By the nested group support, Joe F. Miller will be recognized as a member of the group DB2_SAMPLE_READERS, which is authorized to connect to the SAMPLE database and read its tables (Figure 4-4).

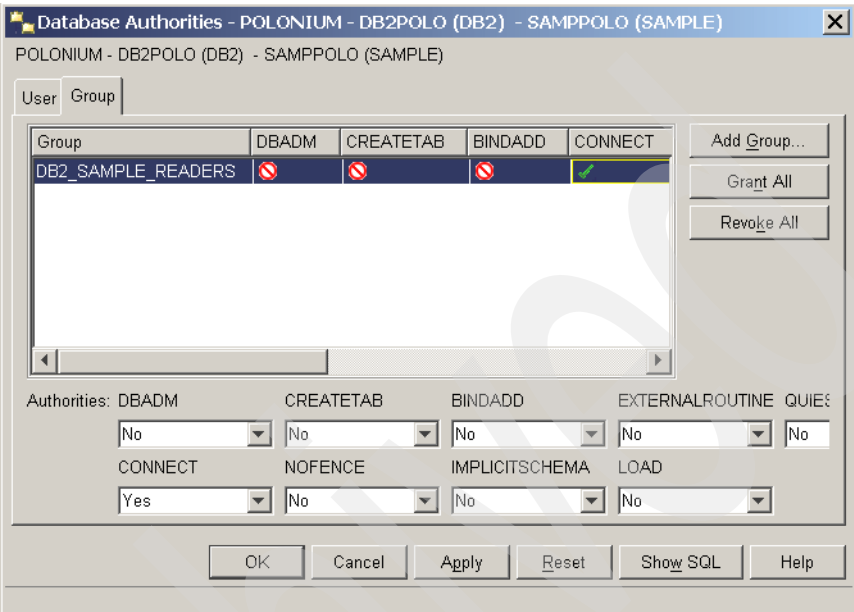


Figure 4-4 Connect authorizations for the DB2_SAMPLE_READERS group

Figure 4-5 shows the authorizations needed on the tables for read access.

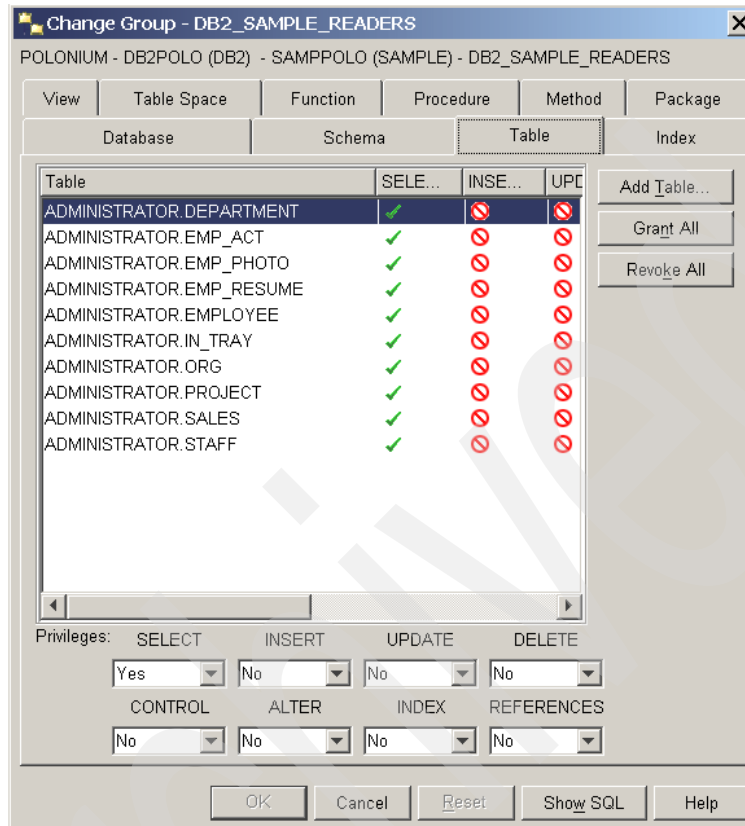


Figure 4-5 DB2_SAMPLE_READERS table authorizations

Nested group lookup example 2

All of the following groups and user IDs are defined in the Active Directory domain db2test.almaden.ibm.com, but the server DB2 UDB is installed only on member server LOCHNESS.

DB2_GRP_LOOKUP is set to TOKEN so that a user ID either at the domain or the local SAM will be searched.

Joe F. Miller's account is defined on the domain controller. He is a member of the global security group DB2_GLOBAL_GROUP.

On the LOCHNESS server, which is not a domain controller and has its own Security Accounts Management, we define the group DB2_SAMPLE_READERS and put the Group DB2_GLOBAL_GROUP of the domain DB2TEST

(DB2TEST\DB2_GLOBAL_GROUP) in as a member (Figure 4-6). Every permission assignment will be done in DB2 to this group.

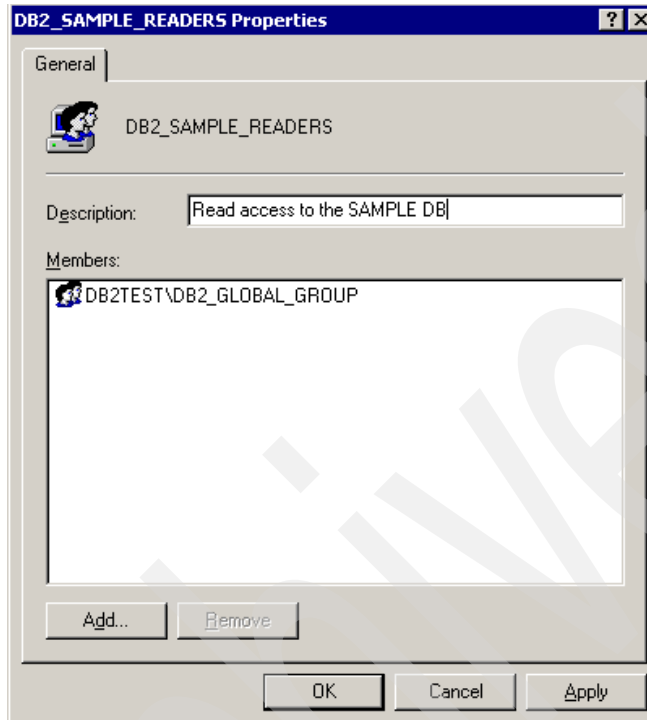


Figure 4-6 Local group DB2_SAMPLE_READERS on server LOCHNESS

As we see in Figure 4-7, DB2 does not list the members of the domain DB2TEST, but does list its local groups and users.

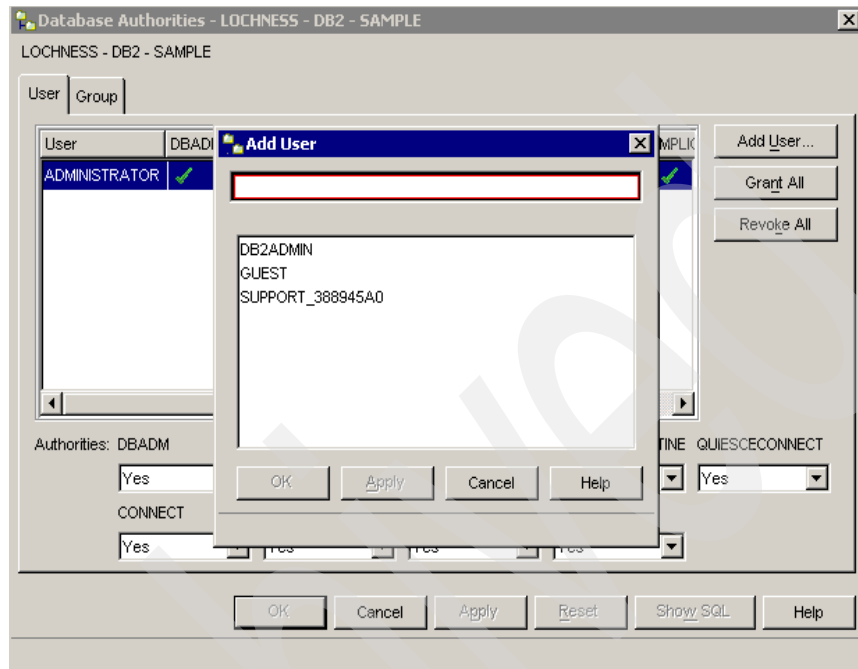


Figure 4-7 User IDs taken from the local SAM of the server LOCHNESS

4.1.2 User ID and group name enhancements

Naming rules now have the following enhancements:

- Additional special characters in user IDs and group names are allowed (!%&(){}~.^~ and space). For example:

Joe F. Miller, Hans-Peter Sonderegger

Figure 4-8 on page 128 shows connecting to a DB2 database with a user ID that has a special character.

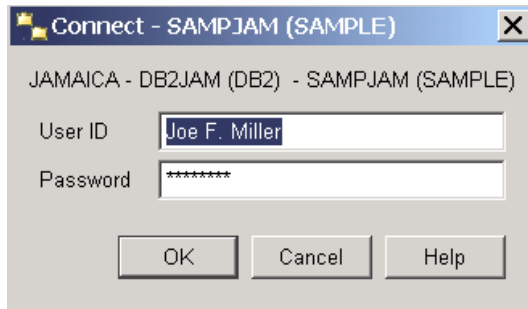


Figure 4-8 Logon with a user name with special characters in DB2 Control Center

Example 4-1 shows connecting to database from DB2 Command Processor using a user ID with a special character.

Example 4-1 User ID with special character

```
db2 => connect to sample user "Joe F. Miller"
Enter current password for Joe F. Miller:
```

Database Connection Information

```
Database server      = DB2/NT 8.2.0
SQL authorization ID = JOE F. M...
Local database alias = SAMPLE
```

Attention: Be sure to put the user name between single quotes in a DB2 Command Window. Double quotes work on the DB2 Command Processor and the DB2 Command Editor, but not on the DB2 Command Window.

- User names can contain up to 30 characters. Figure 4-9 on page 129 shows the user with long name.

Note: If you complete the name so that it matches the user name in the Windows Active Directory, you will not be able to connect to DB2 with this user ID. Leave it as proposed in the DB2 Control Center. Remember this also for GRANT statements, otherwise they will not have the desired effect. Truncate the user ID to 20 characters.

For example, this will work:

```
GRANT DBADM ON DATABASE TO USER 'HANS-PETER SONDEREGG';
```

This will not:

GRANT DBADM ON DATABASE TO USER 'HANS-PETER SONDEREGGER';

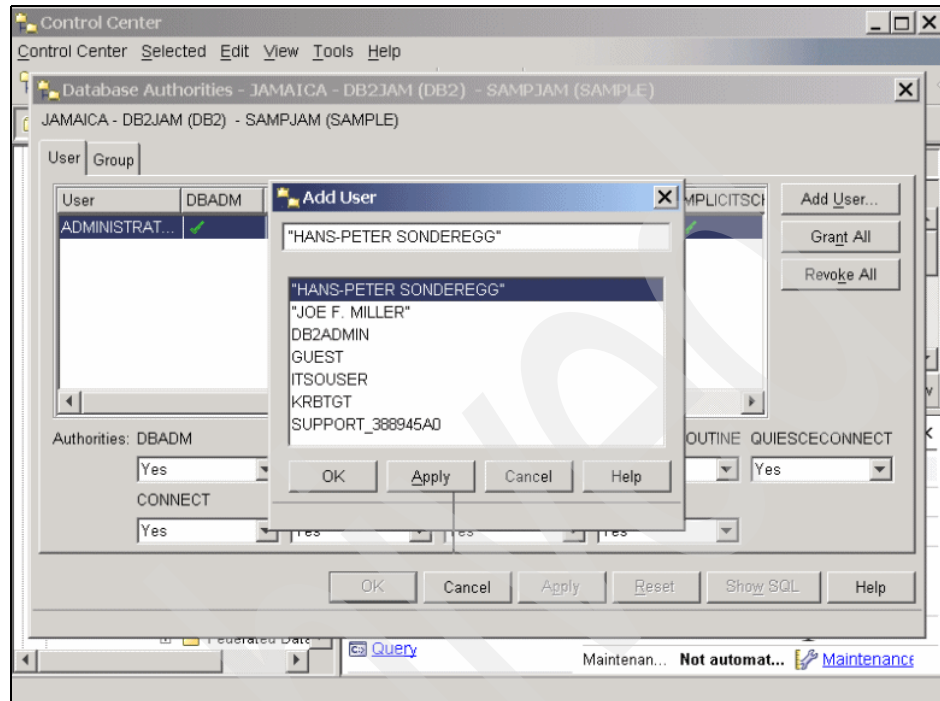


Figure 4-9 Long user names

- Group names can contain up to 30 characters

Security mechanism group names are no longer limited to eight characters and can contain special characters. This is also valid for SYSADM_GROUP, SYSCTRL_GROUP, SYSMANT_GROUP, and SYSMON_GROUP. For example:

A local or global Windows group can be called “Sales Representatives Global” or “Scrapping & Waste Disposal” (Figure 4-10 on page 130).

DB2 reads user ID and group memberships from the underlying operating system, so it does not need its own mechanism to maintain them.

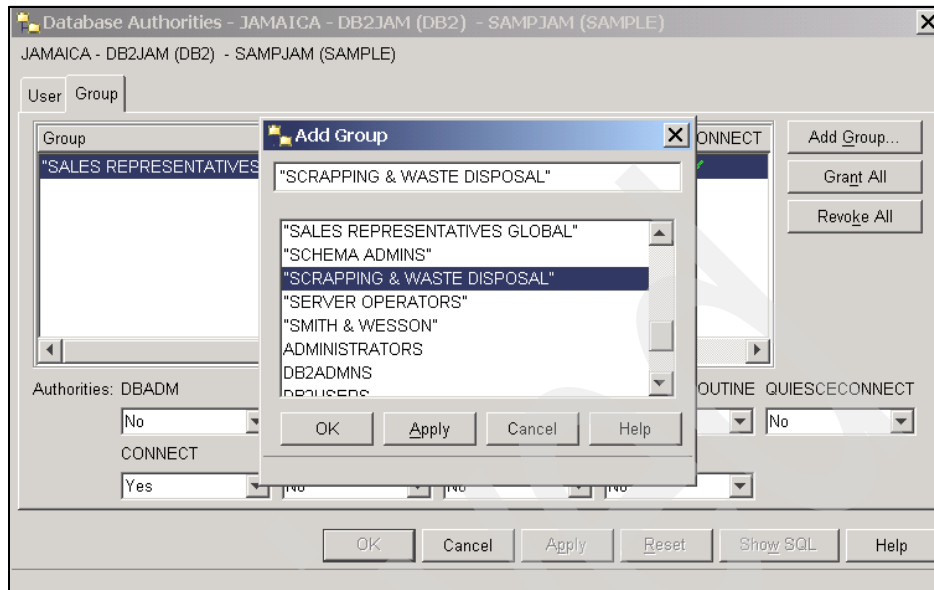


Figure 4-10 Long group names

- Two-part names on CONNECT and ATTACH that contain a Windows domain name and the user ID are allowed.

Figure 4-11 shows an example of logging on from Control Center with a long user name.

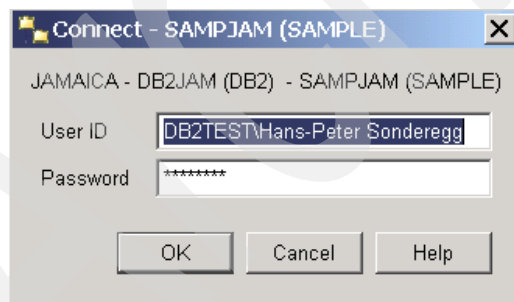


Figure 4-11 Logon with a long user name in DB2 Control Center specifying domain

- Specifying the Windows domain name avoids network traffic associated with looking up the user name in the trusted domain forest. This can result in faster connect time.

For example, Joe F. Miller's account was created in the Windows domain DB2TEST:

```
connect to sample user "DB2TEST\Joe F. Miller" using password
```

Using two-part names also avoids ambiguity as to which account is being used (for example, two accounts named Bob but in different domains and belonging to different groups).

Following are some examples of using enhanced user names. The examples with syntax that works are:

- In the GUI-based DB2 Command Editor and the DB2 Command Line Processor

```
connect to sample user "Joe F. Miller" using password
connect to sample user 'Joe F. Miller' using password
connect to sample user "DB2TEST\Joe F. Miller" using password
connect to sample user 'DB2TEST\Joe F. Miller' using password
```

- In the DB2 Command Window

```
db2 connect to sample user 'Joe F. Miller' using password
db2 connect to sample user 'DB2TEST\Joe F. Miller' using password
```

Examples that do not have valid syntax are:

- In the GUI based DB2 Command Editor and the DB2 Command Line Processor

```
connect to sample user Joe F. Miller using password
connect to sample user DB2TEST\Joe F. Miller using password
connect to sample user 'DB2TEST\Joe F. Miller' using password
```

- In the DB2 Command Window

```
db2 connect to sample user "DB2TEST\Joe F. Miller" using password
db2 connect to sample user DB2TEST\Joe F. Miller using password
```

4.2 Running DB2 under the local system account

In DB2 UDB Version 8.2, the various DB2 services can now run under the Windows Local System account (LSA).

Note: Security is very important in any aspect including DB2. Installing DB2 with the default user IDs (for example, db2admin) and providing a weak password or none can put a Windows domain or your local server at risk because this account has very high permissions. Be sure to set the proper password for DB2 accounts.

After a successful DB2 installation on your server, you can see the DB2 processes running via the Windows Task Manager. Be sure to click the **Show processes from all users** flag. By clicking the **Image Name** tab you can sort the processes alphabetically. See Figure 4-12.

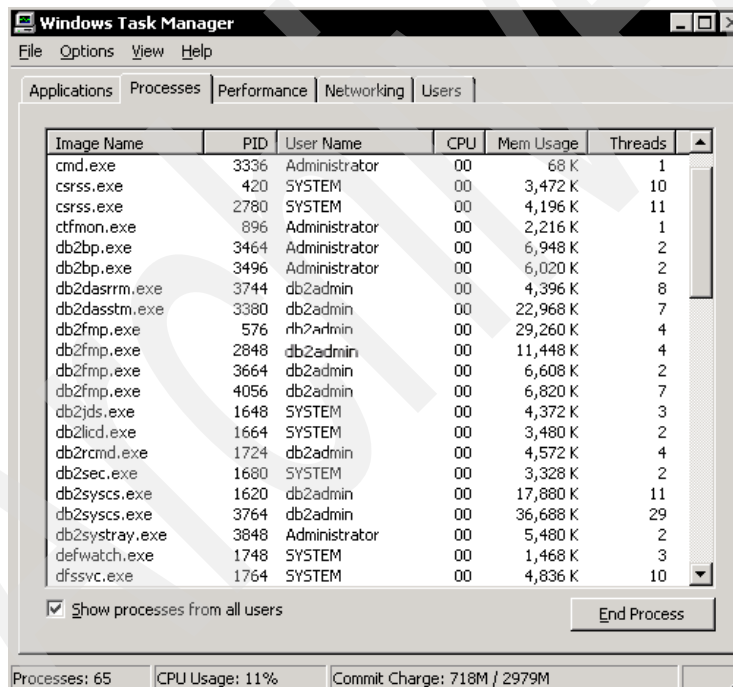


Figure 4-12 DB2 processes in Windows Task Manager

The *LocalSystem* account is the most-used for running services. It is a built-in Windows OS-level account that contains all of the required advanced user rights.

Using the LocalSystem account can be desirable because it does not require any password maintenance. If your company's account policy does not allow functional user accounts with non-expiring passwords, using the LocalSystem account can avoid circumstances such a policy can cause, such as changing the password for every DB2 service in the Windows Services administration, restarting the instance.

Before switching all the affected DB2 services from the dedicated user account to the LocalSystem account, you should consider that the LocalSystem account cannot have access to LAN shares, because it cannot be authenticated on another computer. So you have to ensure that your DB2 system does not have any of its file resources on another machine.

Changing the DB2 services to the LocalSystem account can be done via the Windows services MMC Snap-In:

1. Select **Start** → **Control Panel, Administrative Tools** → **Services** to open the window shown in Figure 4-13.

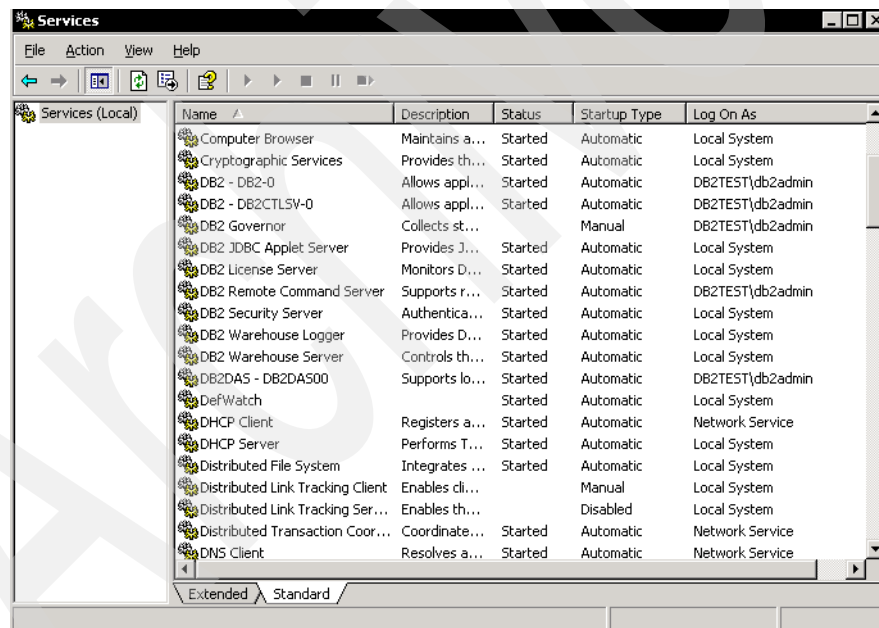


Figure 4-13 DB2 Services before changing to the LocalSystem account

2. Stop every service with a name starting with DB2 (Figure 4-14).

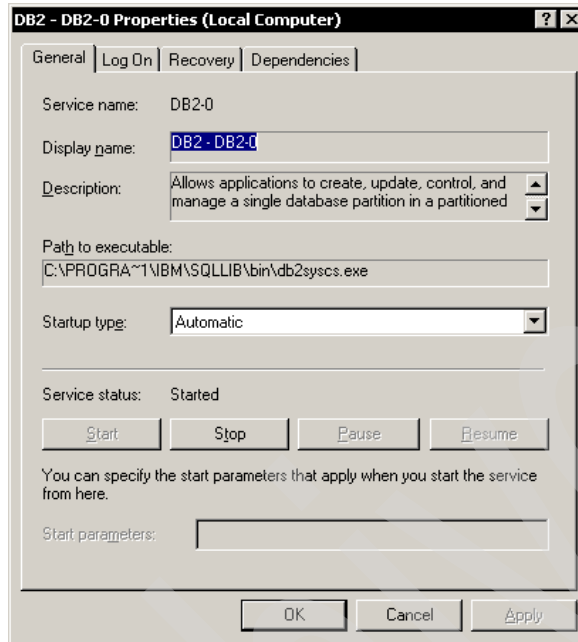


Figure 4-14 Stop the DB2 Services

3. Choose **Local System account** instead of using the assigned account, as shown in Figure 4-15.

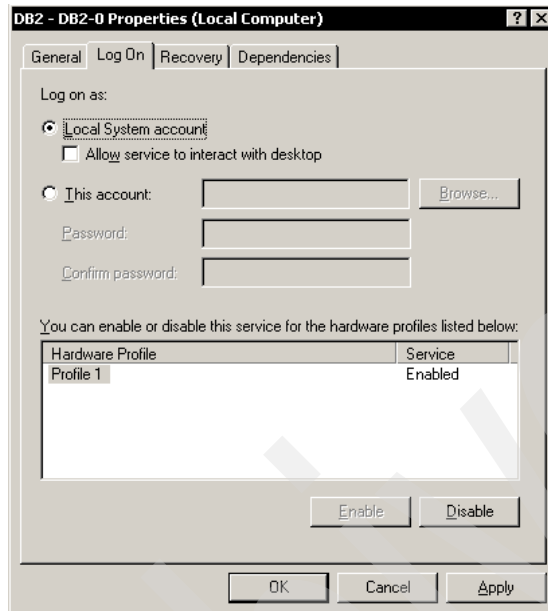


Figure 4-15 Changing DB2 Services to Local System account

4. Start all affected services and test your installation.

4.3 Protecting DB2 UDB system files

DB2 UDB V8.2 creates two groups, DB2ADMNS and DB2USERS, in a successful installation. Only users who are in one of these groups have access to the DB2 system files, registry keys, network shares, and DB2 services on the local machine where DB2 runs. See Figure 4-16 and Figure 4-17 on page 137.

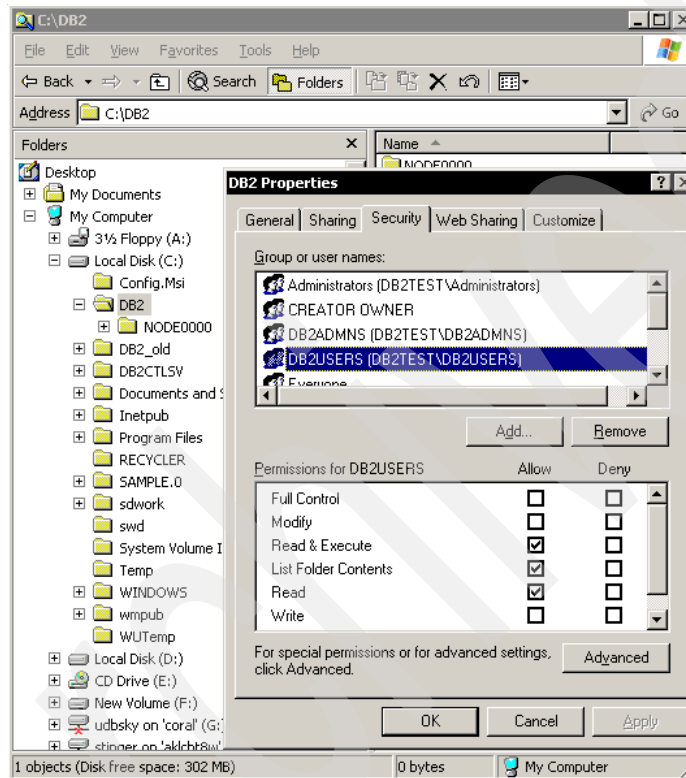


Figure 4-16 Protection of the DB2 instance directories

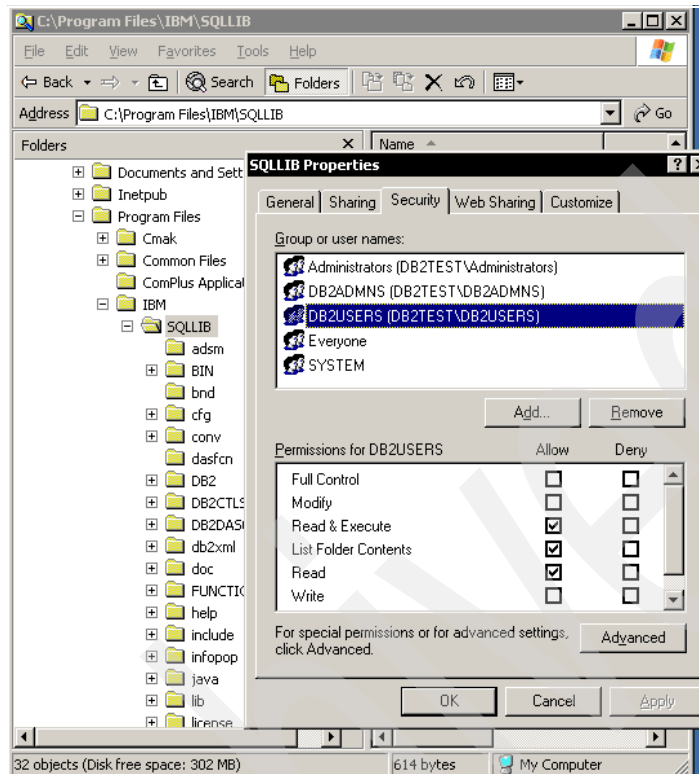


Figure 4-17 Protected DB2 system files

If you did not enable the new security feature during the installation, you still can activate this feature by running the `db2secv82.exe` program, which can be found in the `$DB2PATH\SQLLIB\BIN\` directory.

If these group names do not fit to your company's group name standards you can use different names by specifying parameters when using **db2secv82**:

```
db2secv82 /u usergroup /a admingroup
```

The new `DB2_EXTSECURITY` registry variable prevents unauthorized access to DB2 by locking the DB2 system files. The registry variable is set to YES by default.

4.4 Using DB2 with data encryption

DB2 UDB V8.2 introduces new authentication methods for your server. Remember, a user must be authenticated before accessing an instance or a database. DB2 knows different authentication types. Authentication types for each instance determine how and where a user will be verified. The authentication type is stored in the database manager configuration file at the server. It is set in the instance level and is applied for the instance and all the databases under it.

This section describes new data encryption features introduced in Version 8.2. For details and complete authentication types, consult the DB2 Information Center.

DB2 UDB V8.2 introduces two new authentication types for data encryption. The following objects will be encrypted when using these new authentication types:

- ▶ SQL statement
- ▶ SQL program variable data
- ▶ Output data from the server processing of an SQL statement including the description of the data
- ▶ The answer data set resulting from a query
- ▶ Large object (LOB) data streaming
- ▶ SQLDA descriptors

Note: The SERVER_ENCRYPT authentication type only encrypts the user ID and the password but not any application data.

The following options can be activated for encryption:

- ▶ DATA_ENCRYPT

The server accepts SERVER authentication schemes and the encryption of user data. The authentication works the same way as SERVER_ENCRYPT, except that the objects listed above are also encrypted.

Note: If a DB2 client has explicitly cataloged a database with the authentication type DATA_ENCRYPT, connection to DB2 servers prior to Version 8.2 will fail.

On the DB2 server, you can activate this authentication type by the command:

```
UPDATE DBM CFG USING AUTHENTICATION DATA_ENCRYPT
```


You can also activate it from the DB2 Control Center, as shown in Figure 4-18.

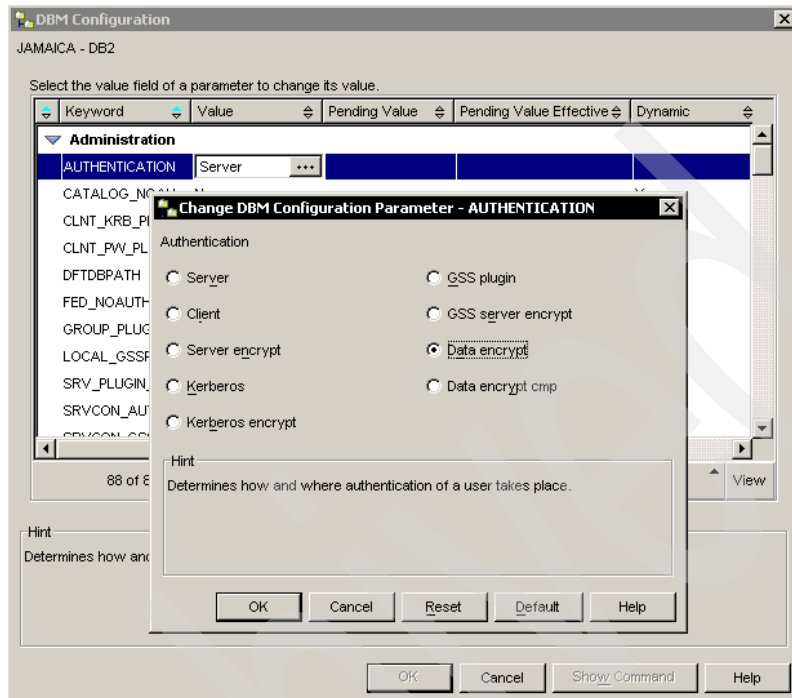


Figure 4-18 Activation the data encryption functionality on a DB2 server

Note: You have to restart the affected DB2 instance to take the new authentication type in effect.

On the DB2 client side, when you catalog a database, you can specify the DATA_ENCRYPT authentication type. You can catalog a database by:

- Using the DB2 command line interface

```
db2 catalog database SAMPLE at node DB2JAM authentication
DATA_ENCRYPT
```

To catalog without explicitly specifying DATA_ENCRYPT authentication:

```
db2 catalog database SAMPLE at node DB2JAM
```

- Using the GUI-based DB2 Configuration Assistant:

Start → Programs → IBM DB2 → Set-up Tools → Configuration Assistant → 7. Security Options. See Figure 4-19 on page 140.

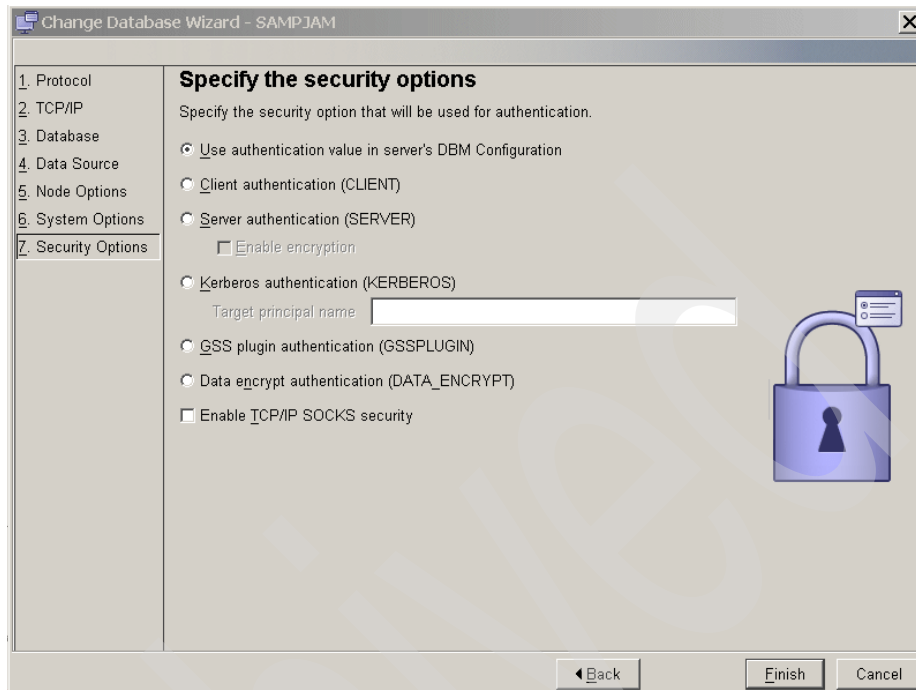


Figure 4-19 Setting security options when cataloging a database

Note: All other authentication types that are different from those described above will result in an SQL30082N error when DATA_ENCRYPT is specified on the DB2 server.

► DATA_ENCRYPT_CMP

This authentication type specifies that authentication takes place on the node containing the target database, and that connections must use data encryption. This option is provided for compatibility with downlevel products that do not support data encryption, in which case they will be allowed to connect with SERVER_ENCRYPT and not encrypt user data. Any product that does support data encryption will be forced to use it.

For example:

You can connect to a DB2 server with authentication type DATA_ENCRYPT_CMP using a V8.1 client that catalogues database with the option SERVER_ENCRYPT.

The DATA_ENCRYPT_CMP authentication type cannot be specified by the CATALOG DATABASE command.

Note: Because these options are valid on the instance level, the authentication type is valid for every database of the affected DB2 instance. Administrators have to think of it when cataloging clients' databases. Explicitly specified authentication types (for example, SERVER in the database catalog on the DB2 Client's security options can cause connection attempts to fail when the server's DBM CFG switched to DATA_ENCRYPT or DATA_ENCRYPT_CMP.

4.5 The new DB2 security exit

To protect your data and database server resources, DB2 uses a combination of operation system security facility and DB2-provided security system. To access a DB2 database server and its database objects, you have to pass security checks.

The first security check is called *authentication*. During this, you are confirmed to be who you say you are.

The second step is the proof of your *authorization*. This mechanism checks whether you are allowed to execute the commands you want to, and whether you have the required permissions to access database objects such as tables, view, procedures, and the operations you want to do with them.

This section discusses the authentication part of DB2.

To explain the new authentication mechanism of DB2, we first take a short look at authentication in DB2 before Version 8.2.

4.5.1 DB2 authentication before Version 8.2

DB2 does not have its own mechanism to maintain user IDs and passwords and the group memberships of these users.

- ▶ Authentication of a user is managed directly by the underlying OS or an external security system such as Kerberos (on the Windows platform).
- ▶ Group enumeration can be influenced by setting the DB2_GRP_LOOKUP registry variable to get user IDs and groups from the local server or the domain. Authentication of domain users is dependent of the availability of a Windows domain controller.
- ▶ Local authorizations and connections use the same authentication method.
- ▶ The user or group name to the authorization ID (AUTHID) is performed by DB2:
 - By stripping the blank characters from the name.

- By using uppercase letters for the Authorization ID.

This modified Authorization ID is verified against the DB2 name rules.

4.5.2 The new DB2 authentication model in Version 8.2

The new authentication model of DB2 is based on external security plug-ins. A security plug-in is a dynamically loadable library that will be loaded if required. Now, DB2 provides plug-in interfaces for the identification, authentication, group and authorization mapping tasks. There are three types of DB2 security plug-ins, which provide the following functionality:

- ▶ Group retrieval plug-in: retrieves the group membership information for the given user
- ▶ Client authentication plug-in: manages authentication of a user on a DB2 client
- ▶ Server authentication plug-in: manages authentication of a user on a DB2 server

The pre-Version 8.2 functionality of the OS-based authentication support and Kerberos support are still available but re-implemented as security plug-ins.

As described in 4.1.2, “User ID and group name enhancements” on page 127, the name space DB2 supports was enhanced: for example, Domain\username (“DB2TEST\Joe F. Miller”).

Advantages of the security plug-in technology

The new security plug-in technology provides the following advantages:

- ▶ Improved flexibility and extensibility: Customers can implement security mechanisms customized for their own security policy.
- ▶ Extended authentication mechanism technologies can be used or developed and connected with the plug-in technology, such as by buying plug-ins from a third party.
- ▶ Multiple different authentication types can be used concurrently; that is, the server can support multiple GSS-API plug-ins concurrently along with encrypted userid/password if using GSS_SERVER_ENCRYPT.
- ▶ Local authorization and connections can be configured to use different authentication methods.

Applications for the security plug-in technology

Examples of what you can realize with the DB2 security plug-ins:

- ▶ A user ID remapping function can be implemented in order to remap several users to another user ID so that the need for the administration of individual user IDs on the database system or the OS can be reduced significantly.
- ▶ You can write your own plug-in to authenticate a user's intranet userid/password for your site against the internal LDAP directory for accessing your company's DB2 databases.
- ▶ Previously unavailable user logon restricting logic can now be easily implemented:
 - You can restrict users to logging on only from particular workstations or from a particular IP address range.
 - You can restrict users to logging on only during business hours so that they will not interfere with nightly batch jobs and cause locking problems.
- ▶ Bypassing the authentication mechanism: Use the connection details to allow specific users to bypass authentication if connecting locally, or if coming in from a particular IP address. For example, the DBA does not have to be authenticated when connecting locally or from the IP address of his or her personal desktop.
- ▶ When you run instance-level operations such as **db2start**, **db2stop**, or **db2trc**, DB2 performs authorization checking for the operation using authentication plug-ins.

4.5.3 DB2 security plug-ins

DB2 security plug-ins are realized under Windows as dynamically loadable libraries (DLLs) and have to be placed and developed as described in the later sections: 4.5.4, "Developing security plug-ins" on page 152 and 4.5.5, "Deploying a userid/password plug-in" on page 157.

The DB2-provided plug-ins are located in the directory
\$DB2PATH\SQLLIB\security\plugin\IBM (Figure 4-20).

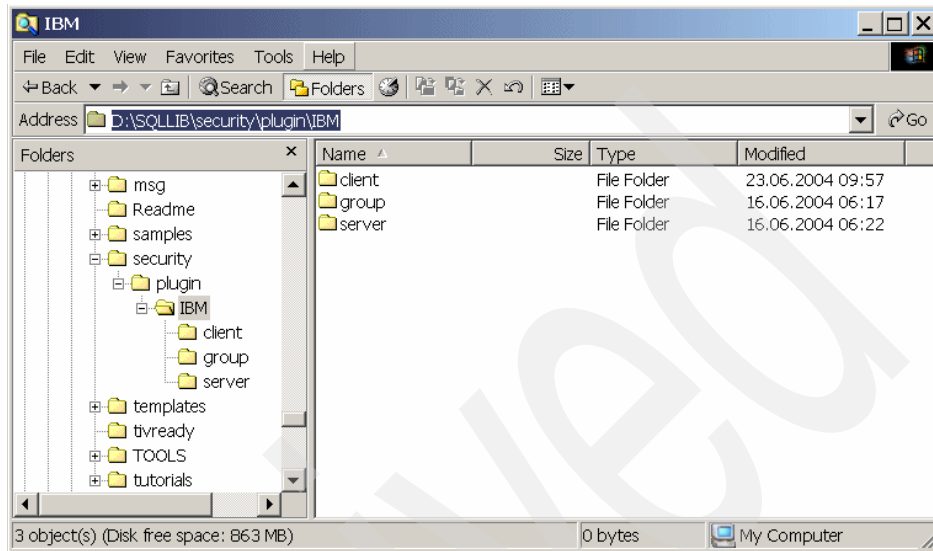


Figure 4-20 The IBM directory for DB2 default security plug-ins

Customer-developed security plug-ins must use the following directory conventions:

```
$DB2PATH\SQLLIB\security\plugin\<instname>\client  
$DB2PATH\SQLLIB\security\plugin\<instname>\server  
$DB2PATH\SQLLIB\security\plugin\<instname>\group
```

These user directories should be created manually. DB2 will search and load the default plug-ins in the IBM directory first, then look in the instance's plug-in directories.

There are two types of mechanism DB2 supports for plug-in:

- ▶ Userid / password authentication plug-ins
- ▶ GSS-API plug-ins

For userid/password, the customer may use his own for client, server, or both. For GSS-API, compatible client and server plug-ins are both required.

Userid / password authentication plug-ins

Authentication is done by userid/password authentication. The authentication types on the database manager configuration parameter AUTHENTICATION

determine where and how the authentication takes place. The authentication types implemented to use the userid/password plug-ins are:

- ▶ CLIENT
- ▶ SERVER
- ▶ SERVER_ENCRYPT
- ▶ DATA_ENCRYPT
- ▶ DATA_ENCRYPT_CMP

GSS-API plug-ins

The GSS-API (Generic Security Service Application Program Interface Version 2) provides security services in a generic fashion, supportable with a range of underlying mechanisms and technologies and enabling source-level portability of applications to different environments.

Detailed information about the Generic Security Service Application Programming Interface Version 2 (IETF RFC2743) and C-Bindings (IETF RFC2744) can be found under <http://www.ietf.org/rfc/rfc2743.txt> and under <http://www.ietf.org/rfc/rfc2744.txt>.

The Kerberos authentication facility is also implemented as a GSS-API plug-in. The authentication types for the DB2 server's database manager configuration parameter AUTHENTICATION are:

- ▶ KERBEROS
- ▶ GSSPLUGIN
- ▶ KRB_SERVER_ENCRYPT
- ▶ GSS_SERVER_ENCRYPT

Both KRB_SERVER_ENCRYPT and GSS_SERVER_ENCRYPT authentication types direct the server to accept both GSS-API/Kerberos authentication and SERVER_ENCRYPT authentication.

Both the userid/password and the GSS-API plug-ins can exist on a DB2 client and on a DB2 server.

DB2 does not provide a default GSS-API plug-in.

Client-side plug-ins

On the DB2 client, you can have different types of security plug-ins. Depending on the authentication type, they may be used independently or in conjunction with one or more of the other plug-ins.

For example, you can use your own server plug-in, and for client and group authentication you use the DB2 default plug-ins. You can have the following plug-ins on a DB2 client (Figure 4-21 on page 146):

- ▶ Userid/password client plug-in
- ▶ Group plug-in
- ▶ GSS-API client plug-in
- ▶ Kerberos client plug-in

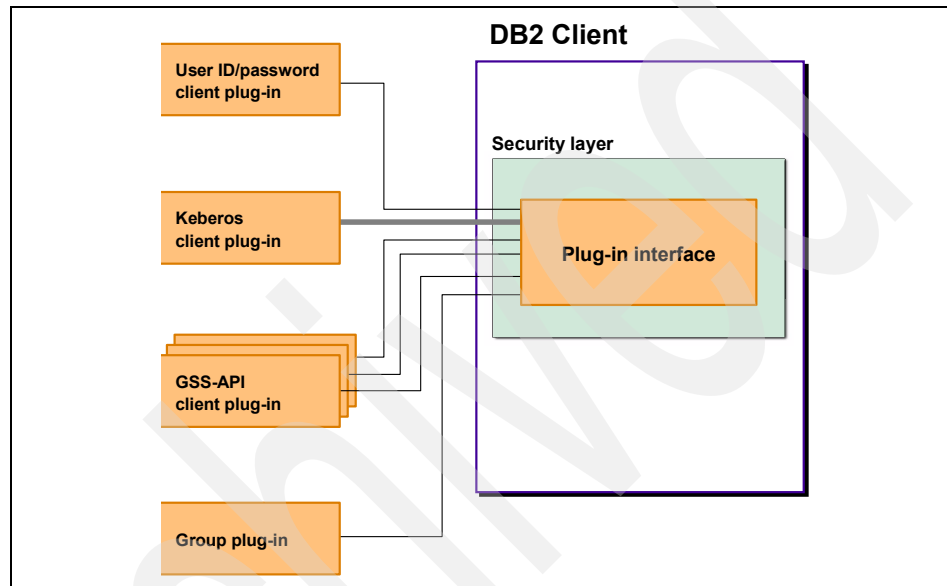


Figure 4-21 Security plug-in architecture on the DB2 Client

The client-side or client authentication plug-in manages the authentication on a client's side. Whenever an application uses the CONNECT or ATTACH command, the client plug-in is used. The client plug-in is also used whenever a user is identified for instance-level local authorization checks (for example, db2start, db2 update dbm cfg, db2audit). The client plug-in is unloaded upon termination of the application.

Figure 4-22 on page 147 shows the functionality of client-side security plug-ins. If the authentication type has not been cataloged, or the authentication type is CLIENT, SERVER, SERVER_ENCRYPT, or DATA-ENCRYPT, then the default behavior of DB2 UDB Version 8.2 is to use the built-in userid/password plug-in, which uses the operating system's authentication mechanism. DB2 UDB Version 8.2 also provides the group and the Kerberos plug-in which is enabled by default on the Windows platform.

This is the default behavior only if the authentication type has not been cataloged or the authentication type is client/server/server_encrypt/data_encrypt.

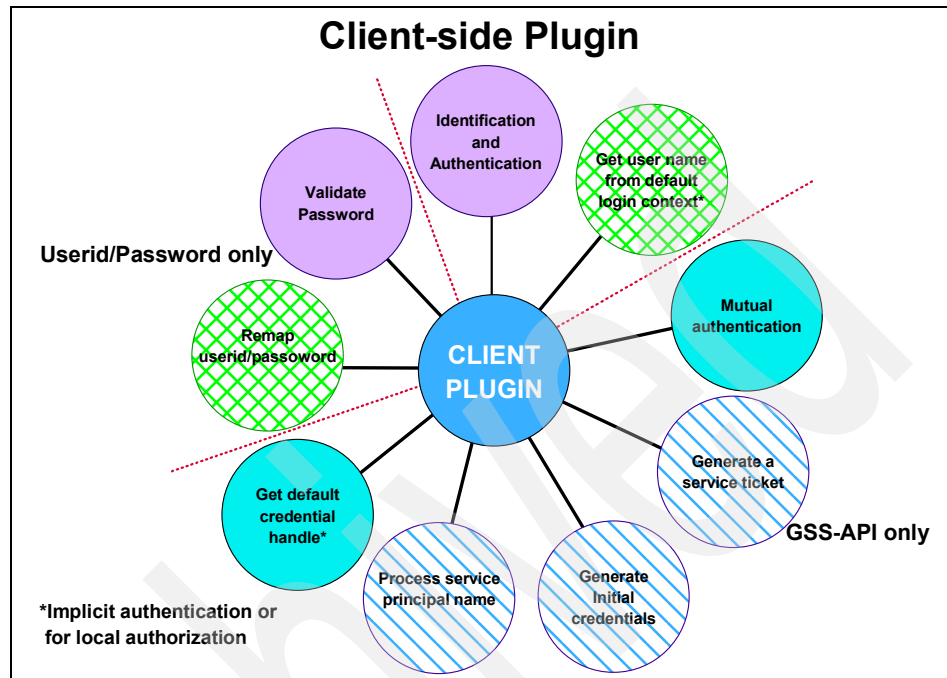


Figure 4-22 Client-side plug-in functionality

If you browse the file directory structure of the machine where your DB2 software (client or server) is installed, you will find the client plug-ins under \$DB2PATH\SQLLIB\security\plugin\IBM\client (Figure 4-23).

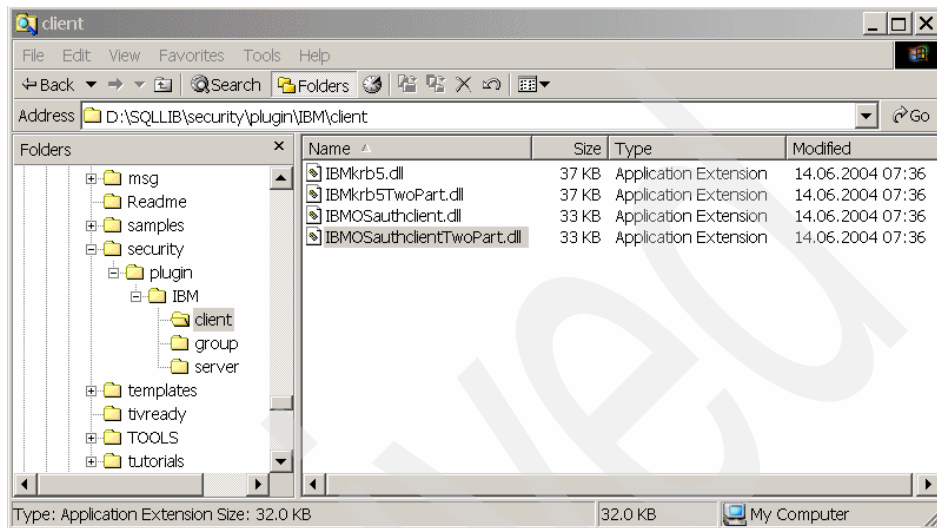


Figure 4-23 The IBM client plug-in directory

Server-side plug-ins

The server-side plug-in is required for authentication during the CONNECT/ATTACH command. Figure 4-24 shows the security plug-in architecture on the DB2 server.

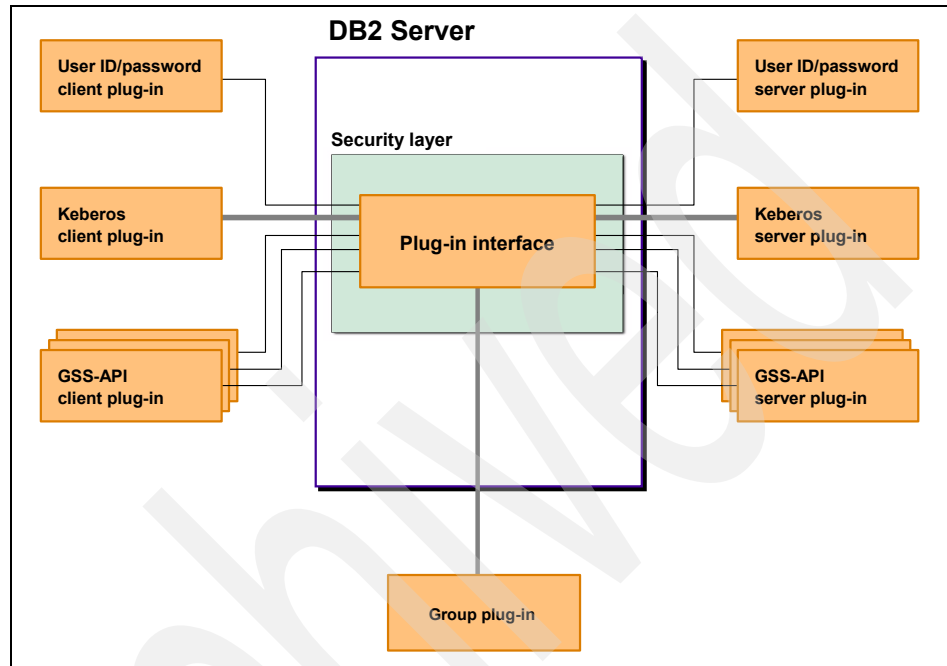


Figure 4-24 Security plug-in architecture on the DB2 Server

On a DB2 server you can have these plug-ins:

- ▶ Userid/password client plug-in
- ▶ Userid/password server plug-in
- ▶ Group plug-in
- ▶ GSS-API client plug-in
- ▶ GSS-API server plug-in
- ▶ Kerberos client plug-in
- ▶ Kerberos server plug-in

Figure 4-25 on page 150 shows the functionality of the server-side plug-ins.

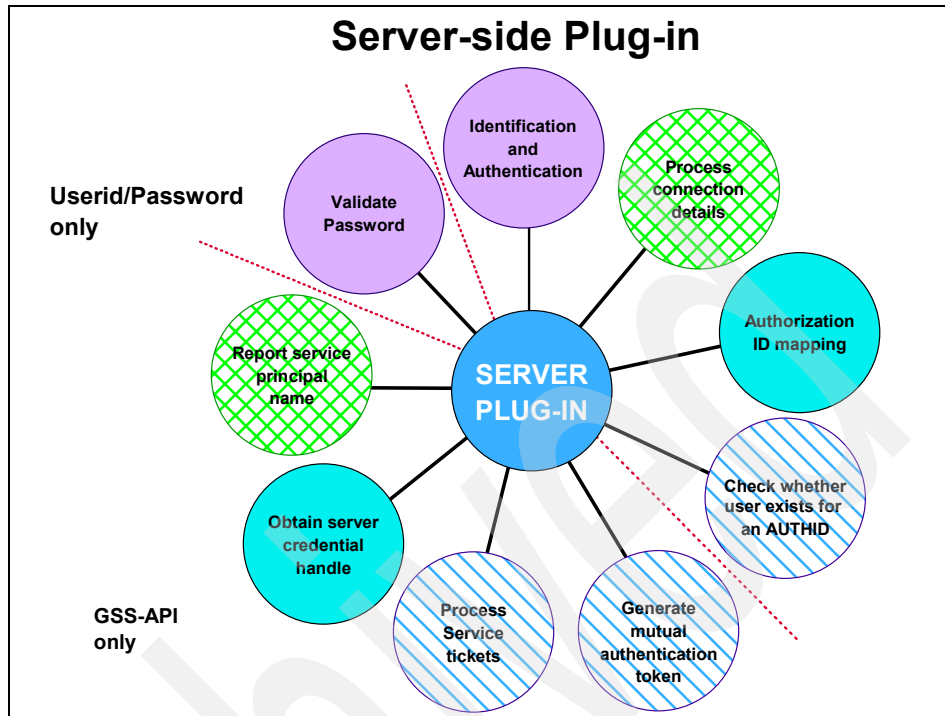


Figure 4-25 Functionality of the Server-side plug-in

At the DB2 server side you can implement multiple authentication plug-ins concurrently: one group plug-in, one userid/password plug-in, and multiple GSS-API plug-ins. Only one GSS-API plug-in may be a Kerberos plug-in.

Note: Because instance-level commands such as **db2start** and **db2stop** perform authorization checking, the client authentication plug-ins are also required on the server.

Group plug-ins

This plug-in can get a list of groups to which a user belongs. It is needed on the client side to verify the local authorization for client-side commands. The group plug-in is also used on the server to obtain the secondary AUTHIDs for the connecting/attaching user.

- ▶ Only one group function on each of client and server is allowed.
- ▶ Multiple group lookup method implementation is possible. DB2 provides the authentication plug-in name to the group plug-in if available so that the group plug-in may determine group membership in a different way, if desired.

Note: The authentication plug-in name is only known for a CONNECT/ATTACH command and will not be provided outside these commands, such as in a GRANT/REVOKE statement.

- ▶ Verification of secondary AUTHIDs is done against the naming rules and the following limits: 30 bytes maximum user/group name length.

CONNECT authentication flow

In a CONNECT authentication, the authentication information is communicated between the client and server. The simplified authentication flow is as follows.

- ▶ First, at the client side, DB2 loads and initializes client plug-in if the plug-in is not already loaded.

At the server side, all supported plug-ins should have been loaded already, as they are loaded at db2start time.

- ▶ After the plug-ins are loaded, these activities take place at the client side:
 - Obtain the default user name/credentials (implicit connect).
 - Remap userid/password (u/p only).
 - Validate password if necessary (u/p only and for client authentication only).
 - Generate service ticket (GSS-API only).
- ▶ The authentication process then flows to the server side. These activities are performed:
 - Validate client credentials:
 - Process connection details

At any time during the validation of the client's credential, the server plug-in may choose to inspect the connection details to refuse the connection.
 - Generate mutual authentication token (GSS-API only)

For GSS-API, the mutual authentication token, if required, is generated in the process of validating the client's credential.
 - Perform AUTHID mapping and check against naming rules.
 - Obtain group list for AUTHID and check against naming rules.

Any group name in the group list that is returned by the plug-in that fails the DB2 naming restrictions is simply discarded.
 - Check authority to connect.

- ▶ Optionally, the process then flows back to the client side to perform mutual authentication (GSS-API only).
- ▶ Authentication complete, both server and client side perform cleanup activities. Server plug-ins are not unloaded until the instance is stopped (**db2stop**). The client plug-in is unloaded upon termination of the application.

4.5.4 Developing security plug-ins

There is a detailed description about the development of DB2 security plug-ins in the *Application Development Guide: Programming Client Applications*, SC09-4826, which is also included in the DB2 Information Center.

Sample DB2 security plug-ins, which can be easily extended and customized to your needs, can be found in the \$DB2PATH\SQLLIB\samples\security\plugins directory.

This section explains basic logic for the development of the DB2 security plug-ins.

Prerequisites

The prerequisites for developing security plug-ins are:

- ▶ You must create plug-in directories for the client, server, and group plug-ins:

```
$DB2PATH\security\plugin<instance name>\client
$DB2PATH\security\plugin<instance name>\server
$DB2PATH\security\plugin<instance name>\group
```

- ▶ C or C++ Compiler: Plug-in libraries can only be implemented in C or C++.

- ▶ Header files:

```
$DB2PATH\include\db2secPlugin.h
$DB2PATH\include\gssapiDB2.h
```

- ▶ Plug-in names have to correspond to the library names.
- ▶ Libraries must have the *dll* file extension under Windows.
- ▶ Libraries must be written re-entrant so that they are thread-safe.
- ▶ C-linkage must be used at least for the *init* functions.

DB2 Database Manager configuration parameters

Under the plug-in model, in this release we have more parameters available now for the authentication. The security plug-in related database manager parameters are:

```
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

The AUTHENTICATION database manager configuration parameter works as previously described if none of the other parameters are used.

The values GSSPLUGIN and GSSPLUGIN_SERVER_ENCRYPT authentication types were newly introduced for the AUTHENTICATION parameter. The database or database manager configuration parameters should be set to the proper value when you deploy the plug-ins.

Under the DBM configuration wizard, all of these parameters are grouped under Administration (Figure 4-26).

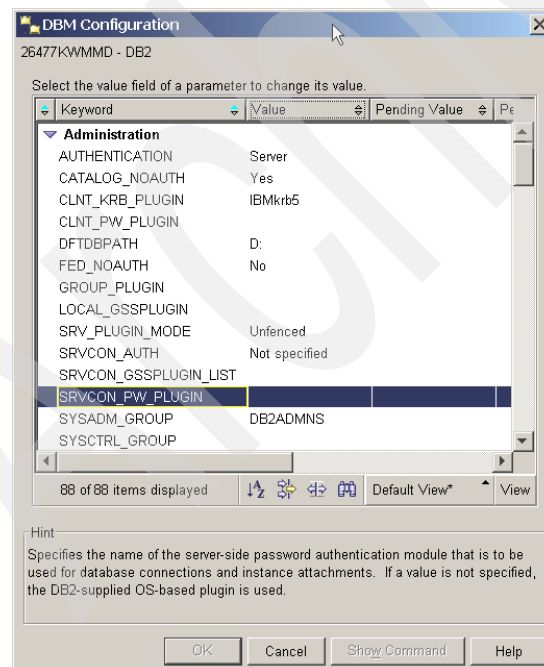


Figure 4-26 New authentication plug-in parameters in database configuration manager

Basic steps

Follow these steps to develop the security plug-ins:

1. Decide whether you want to create userid/password or GSS-API plug-ins.

Note: A good understanding of the GSS-API is strongly recommended before writing a GSS-API plug-in.

2. Create the appropriate initialization function for the plug-in.

- Server authentication plug-in initialization:

```
db2secServerAuthPluginInit (
    int version,
    void *server_fns,
    db2secGetConDetails *getConDetails_fn,
    char **errorMsg,
    int *errorMsgLen )
```

- Client plug-in initialization function:

```
db2secClientPluginInit ()
```

- Group plug-in initialization function:

```
db2secGroupPluginInit ()
```

3. Populate the function pointer structure before returning to DB2:

- Structure also indicates plug-in API version used by the plug-in.
- Structure also indicates the plug-in type (userid/password, GSS-API, Kerberos).

For example:

```
SQL_API_RC SQL_API_FN db2secGroupPluginInit(db2int32 version,
                                             void *group_fns,
                                             db2secLogMessage *msgFunc,
                                             char **errorMessage,
                                             db2int32 *errorMessageLength)
{
    db2secGroupFunction_1 *p;

    p = (db2secGroupFunction_1 *)group_fns;

    p->version = DB2SEC_API_VERSION;
    p->plugintype = DB2SEC_PLUGIN_TYPE_GROUP;
    p->db2secGetGroupsForUser = &LookupGroups;
    p->db2secDoesGroupExist = &DoesGroupExist;
    p->db2secFreeGroupListMemory = &FreeGroupList;
    p->db2secFreeErrorMsg = &FreeErrorMessage;
    p->db2secPluginTerm = &PluginTerminate;
```



```

logFunc = msgFunc;

*errorMessage = NULL;
*errorMessageLength = 0;
return(DB2SEC_PLUGIN_OK);
}

```

4. Compile the plug-in source and create a shared library.
Compile as 32-bit or 64-bit to correspond with the application/server instance.
5. Place the library in the appropriate directory.
Copy the libraries to the client, server, or group directory under
\$DB2PATH\security\plugin\<instance name>.

Client functions

Table 4-1 shows client plug-in API functions. For details, consult the *Application Development Guide: Programming Client Applications*, SC09-4826-01.

Table 4-1 client API functions

Userid/password	GSS-API
db2secRemapUserid	db2secGetDefaultLoginContext
db2secGetDefaultLoginContext	db2secGenerateInitialCred
db2secValidatePassword	db2secProcessServerPrincipalName
db2secFreeToken	db2secFreeToken
db2secFreeErrorMsg	db2secFreeErrorMsg
db2secClientAuthPluginTerm	db2secFreeInitInfo
	db2secClientAuthPluginTerm
	gss_init_sec_context
	gss_delete_sec_context
	gss_display_status
	gss_release_buffer
	gss_release_cred
	gss_release_name

Server functions

Table 4-2 shows server plug-in API functions. For details, consult the *Application Development Guide: Programming Client Applications*, SC09-4826-01.

Table 4-2 Server plug-in API

Userid/password	GSS-API
db2secValidatePassword	db2secGetAuthIDs
db2secGetAuthIDs	db2secDoesAuthIDExist
db2secDoesAuthIDExist	db2secFreeToken
db2secFreeToken	db2secFreeErrorMsg
db2secFreeErrorMsg	db2secServerAuthPluginTerm
db2secServerAuthPluginTerm	gss_accept_sec_context
	gss_display_name
	gss_delete_sec_context
	gss_display_status
	gss_release_buffer
	gss_release_cred
	gss_release_name

Group functions

Table 4-3 shows group plug-in API functions. For details, consult the *Application Development Guide: Programming Client Applications*, SC09-4826-01.

Table 4-3 Group plug-in API

Userid/password and GSS-API
db2secGetGroupsForUser
db2secDoesGroupExist
db2secFreeGroupListMemory
db2secFreeErrorMsg
db2secPluginTerm

4.5.5 Deploying a userid/password plug-in

To deploy a user ID/password authentication plug-in, perform the following steps on the database server and client.

Deploying on the database server

To deploy the server security plug-in:

1. Place the user ID/password authentication plug-in library in the server's plug-in directory.
2. Update the database manager configuration parameter **SRVCONS_PW_PLUGIN** with the name of the server plug-in.

This plug-in is used now by the server when it handles connection (CONNECT) and attachment (ATTACH) requests.

3. Either:
 - Set the database manager configuration parameter **SRVCON_AUTH** to the CLIENT, SERVER, SERVER_ENCRYPT, DATA_ENCRYPT, or DATA_ENCRYPT_CMP authentication type.
 - Or:
 - Set the database manager configuration parameter **SRVCON_AUTH** to NOT_SPECIFIED, and set **AUTHENTICATION** to CLIENT, SERVER, SERVER_ENCRYPT, DATA_ENCRYPT, or DATA_ENCRYPT_CMP authentication type.

Note: If SRVCON_PW_PLUGIN is left blank, it will default to the IBM-provided plug-in IBMOSauthserver.

Deploying the database client

To deploy the client security plug-in:

1. Place the user ID/password authentication plug-in library in the client plug-in directory on the client.
2. Update the database manager configuration parameter **CLNT_PW_PLUGIN** with the name of the client plug-in.

This plug-in is loaded and called regardless of where the authentication is being done (that is, not only when client authentication is enabled).

Note: If CLNT_PW_PLUGIN is left blank, it will default to the IBM-provided plug-in IBMOSauthclient.

Local authorization on a client, server, or a gateway

1. Place the user ID/password authentication plug-in library in the client plug-in directory on the client, server, or gateway.
2. Update the database manager configuration parameter **CLNT_PW_PLUGIN** with the name of the plug-in.
3. Set the authentication database manager configuration parameter to **CLIENT**, **SERVER**, **SERVER_ENCRYPT**, **DATA_ENCRYPT**, or **DATA_ENCRYPT_CMP**.

4.5.6 Deploying a group retrieval plug-in

To deploy a group retrieval plug-in in your database environment, perform the following steps:

Deploying on the database server

1. Place the group retrieval plug-in library in the server's group plug-in directory.
2. Update the database manager configuration parameter **GROUP_PLUGIN** with the name of the plug-in.

Deploying on the database client

1. Place the group retrieval plug-in library in the client's group plug-in directory.
2. Update the database manager configuration parameter **GROUP_PLUGIN** with the name of the plug-in.

4.5.7 Deploying a GSS-API plug-in

For GSS-API or Kerberos plug-ins, you must have matching plug-in types on the client and the server. The plug-ins on the client and server need not be from the same vendor, but they must generate and consume compatible GSS-API tokens. For example, any combination of Kerberos plug-ins deployed on the client and the server is okay because Kerberos plug-ins are standardized; however, different implementations of less-standardized GSS-API mechanisms, such as x.509 certificates, might not be completely compatible.

All GSS-API authentication plug-ins must be placed in either the client plug-in directory or the server plug-in directory, depending on the intended use of the plug-ins. If a plug-in is placed in the client plug-in directory, it will be used for local authorization checking and when a client attempts to connect with the server. If the plug-in is placed in the server plug-in directory, it will be used for handling incoming connections to the server and for checking whether an AUTHID exists and is valid whenever the GRANT statement is issued without specifying either the keyword **USER** or **GROUP**.

Deploying on the database server

1. Place the GSS-API authentication plug-in library in the server plug-in directory on the server. You can copy numerous GSS-API plug-ins into this directory.
2. Update the database manager configuration parameter **SRVCON_GSSPLUG_LIST** with an ordered, comma-delimited list of the names of the plug-ins installed in the GSS-API plug-in directory.
3. Either:
 - Set the database manager configuration parameter **SRVCON_ATUH** to **GSSPLUGIN**. Or:
 - Set the database manager configuration parameter **SRVCON_ATUH** to **NOT_SPECIFIED** and set authentication to **GSSPLUGIN**.

Deploying on the database client

1. Place the GSS-API authentication plug-in library in the client plug-in directory on the client. You can copy numerous GSS-API plug-ins into this directory. The client selects the appropriate GSS-API plug-in for authentication during **CONNECT/ATTACH** by picking the first GSS-API plug-in contained in the server's plug-in list on the client.
2. Optional: Catalog the databases that the client will access, indicating that the client will only accept a GSS-API authentication plug-in as the authentication mechanism (Example 4-2).

Example 4-2 Catalog the database for using a GSS-API plug-in

```
db2 UNCATALOG DATABASE testdb
db2 CATALOG DATABASE testdb AT NODE testnode AUTHENTICATION GSSPLUGIN
```

Local authorization on a client, server, or gateway

1. Place the GSS-API authentication plug-in library in the client plug-in directory on the client, server, or gateway.
2. Update the database manager configuration parameter **LOCAL_GSSPLUGIN** with the name of the plug-in.
3. Set the authentication database manager configuration parameter to **GSSPLUGIN**, or **GSS_SERVER_ENCRYPT**.

4.5.8 Error handling

For error handling in the plug-in functions, there was a need to introduce the following new SQL codes with new reason codes:

- ▶ New SQL codes introduced for plug-in errors:

SQL1365N Server-side local plug-in error

SQL1366N Client-side local plug-in error

- ▶ SQL30082 expanded to include plug-in error reason codes.

Reason codes 25 - 40 indicate plug-in specific errors generated by server or client during CONNECT or ATTACH commands.

- ▶ The Administration notification log uses the Windows event log.

Administration notification log error codes 13000 - 13006 indicate plug-in errors. 13001 and 13001 in particular display the plug-in name, error code, API name, and the plug-in generated error message for the failing GSS-API or db2sec function.

Performance and monitoring

This chapter provides information about the new performance enhancements of the DB2 UDB Version 8.2. The monitoring tool Activity Monitor is also discussed.

For building up a comprehensive understanding of performance and performance tuning, we recommend that you read the following redbooks, guides, and DB2 Information Center chapters:

- ▶ Redbooks:
 - *DB2 UDB Performance Expert for Multiplatforms: A Usage Guide*, SG24-6436
 - *DB2 UDB Exploitation of the Windows Environment*, SG24-6893
- ▶ DB2 UDB guides:
 - *Administration Guide: Performance*, SC09-4820-01
 - *System Monitor Guide and Reference*, SC09-4847-01
- ▶ The DB2 Control Center:
 - Monitoring performance using the Windows Performance Monitor
 - Developing a performance improvement process
 - Database System Monitor

5.1 SQL query optimization enhancements

This section discusses the SQL query optimization enhancements that have been added for this release of DB2 UDB.

5.1.1 Native SQL procedures

In Version 8.2 of DB2 Universal Database, a C or C++ compiler is no longer required for creating SQL procedures; therefore C or C++ compiler setup is not required. When you create a SQL procedure, its procedural statements are converted to a native representation that is stored in the database catalogs, as is done with other SQL statements. When a SQL procedure is called, the native representation is loaded from the catalogs and the DB2 engine executes the procedure.

5.1.2 SQL statement size limit increased to 2 MB

To support very complex queries and the enhanced functionality of the DB2 SQL programming language for creating triggers and procedures, DB2 provides an increased SQL statement size limit of 2 MB.

This new extended limit is also very useful when you migrate triggers or stored procedures from another database system.

5.1.3 Data sampling in SQL queries

The amount of data in databases can grow very large, and this often results in long-running and complex queries. In some cases, if a user is interested in finding overall trends or patterns, approximate answers within some margin of error will suffice.

For the database application developers, it would often be an advantage not to run the queries against the entire amount of data. The reasons are:

- ▶ Often the large amount of data does not allow mobile development and testing of an application either because it would cross the limits of file space and processing capacity or it requires the development of time-consuming export procedures to have an excerpt of the data that corresponds to reality and gives appropriate results.
- ▶ Queries often have to be run several (hundred) times in development and test until they satisfy our demands. It is better not to sum up the time we wait behind the computer.

Users of database applications (for which this facility was developed primarily) often have the same problems and requirements:

- ▶ Queries against the whole amount of data in large tables increases the load of system resources
- ▶ Results of the queries that are accurate could be obtained by a subset of data, but this facility is not known.

One way to speed up such queries is to perform the query on a random sample of the database. DB2 UDB Version 8.2 now enables you to do efficient sampling of data in SQL queries, potentially improving performance of large queries by orders of magnitude while maintaining a high degree of accuracy.

For database end users, the most common application of sampling is for aggregate queries such as AVG, SUM, and COUNT, where reasonably accurate answers of the aggregates can be obtained from a sample of the data.

Sampling can also be used to obtain a random subset of the actual rows in a table for auditing or testing purposes or to speed up data mining and analysis tasks.

DB2 provides two methods of sampling: *row-level Bernoulli sampling* and *block-level sampling*. You can obtain detailed information about these two methods in the DB2 Information Center, Data Sampling in SQL queries, and *Administration Guide: Performance*, SC09-4821.

The facility of table sampling can be specified in a SQL query by the `tablesample` clause after the table name. The following examples work on the DB2 SAMPLE database. If it is not already installed, you can install it easily by invoking the `db2samp1` command in the DB2 command window or by the GUI-based DB2 First Steps Launchpad (**Start** → **Programs** → **IBM DB2** → **Set-up Tools** → **First Steps**).

Example 5-1 computes the total sales revenue in the Manitoba region for each salesperson using a random 10% SYSTEM sample of the Sales table. The semantics of SUM are for the sample itself, so to extrapolate the sales to the entire Sales table, the query must divide that SUM by 2 the sampling rate (0.1).

Example 5-1 Data sampling

```
SELECT Sales.Sales_Person, SUM( Sales.Sales ) / (0.1)
FROM Sales TABLESAMPLE SYSTEM( 10 )
WHERE Sales.Region = 'Manitoba'
GROUP BY Sales.Sales_Person ;
```

If you run the query several times in a row, you will notice that the results are different (Figure 5-1 on page 164).

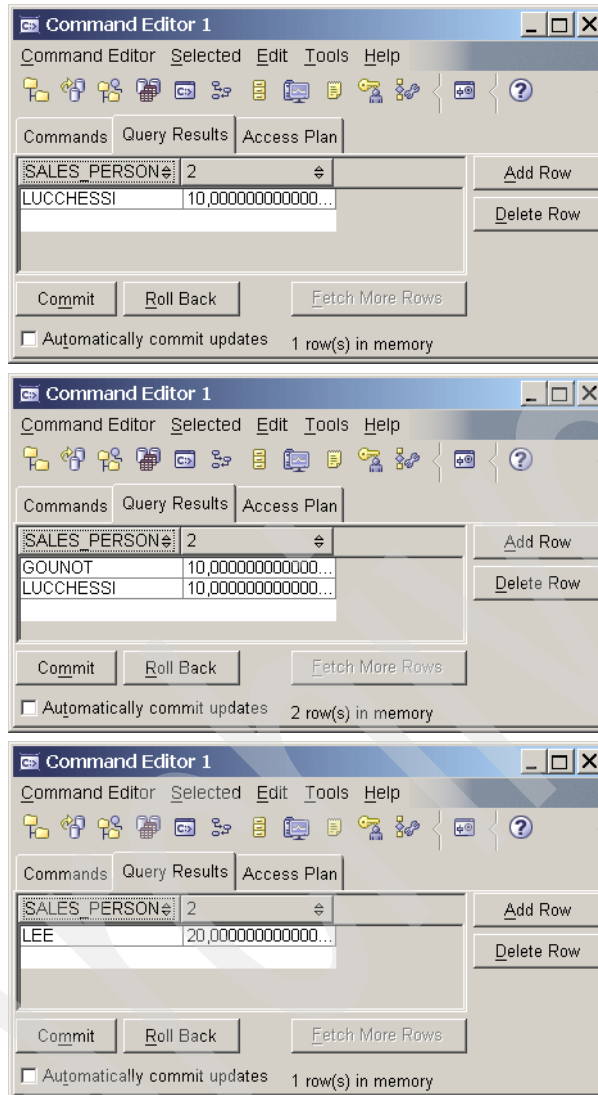


Figure 5-1 Different results in data sampling

If it is not desirable to have different results from the same example, you can force DB2 to evaluate the same data by specifying the REPEATABLE option, as shown in Example 5-2.

Example 5-2 Data sampling with resulting the same output

```
SELECT Sales.Sales_Person, SUM( Sales.Sales ) / (0.1)
FROM Sales TABLESAMPLE SYSTEM( 10 ) REPEATABLE (12345)
```

```
WHERE Sales.Region = 'Manitoba'  
GROUP BY Sales.Sales_Person ;
```

The value of the REPEATABLE constant enclosed in parentheses is arbitrary. The query will return different results only when you change this constant.

5.1.4 Dynamic SQL re-optimization

To enable query (re)optimization of static and dynamic SQL statements that have host variables, special registers, or parameter markers, bind the package with the REOPT bind option. You can set the bind option REOPT to one of the following three values:

- | | |
|---------------|--|
| NONE | The access path for a given SQL statement containing host variables, parameter markers, or special registers will not be optimized using real values for these variables. The default estimates for these variables will be used instead, and this plan is cached and used subsequently. This is the default behavior. |
| ONCE | The access path for a given SQL statement will be optimized using the real values of the host variables, parameter markers, or special registers when the query is first executed. This plan is cached and used subsequently. |
| ALWAYS | The access path for a given SQL statement will always be compiled and reoptimized using the values of the host variables, parameter markers, or special registers known at each execution time. |

Explicit rebinding can be accomplished using the REBIND command with the REOPT option. Following are REOPT examples (in a DB2 command window):

```
db2 bind package <package-filename> reopt once  
db2 rebind package <package-name> reopt always
```

If the bind option REOPT is used, the access path for a SQL statement belonging to that package and containing host variables, parameter markers, or special registers will be optimized using the values of these variables rather than default estimates chosen by the compiler. This optimization takes place at query execution time when the values are available.

Information about access plans for static SQL is stored in the system catalog tables. When the package is executed, the database manager will use the information stored in the system catalog tables to determine how to access the data and provide results for the query. This information is used by **db2exp1n**.

5.1.5 Specifying a lock wait mode strategy

An individual session can now specify a lock wait mode strategy, which is used when the session requires a lock that it cannot obtain immediately. The strategy indicates whether the session will:

- ▶ Return a SQLCODE and SQLSTATE when it cannot obtain a lock.
- ▶ Wait indefinitely for a lock.
- ▶ Wait a specified amount of time for a lock.
- ▶ Use the value of the LOCKTIMEOUT database configuration parameter when waiting for a lock.

The lock wait mode strategy is specified through the new **SET CURRENT LOCK TIMEOUT** statement, which changes the value of the **CURRENT LOCK TIMEOUT** special register. The **CURRENT LOCK TIMEOUT** special register specifies the number of seconds to wait for a lock before returning an error indicating that a lock cannot be obtained.

Traditional locking approaches can result in applications blocking each other. This happens when one application must wait for another application to release its lock. Strategies to deal with the impact of such blocking usually provide a mechanism to specify the maximum acceptable duration of the block. That is the amount of time that an application will wait before returning without a lock. Prior to V8.2, this was only possible at the database level by changing the value of the **LOCKTIMEOUT** database configuration parameter.

Whereas the value of the **LOCKTIMEOUT** parameter applies to all locks, the lock types that are affected by this new function include row, table, index key, and multidimensional clustering (MDC) block locks.

The command **SET CURRENT LOCK TIMEOUT -1** advises the database manager to wait until the lock is released or a deadlock has been detected.

SET CURRENT LOCK TIMEOUT 30 gives the applications the possibility to wait up to 30 seconds until a SQL error is returned, **SET CURRENT LOCK TIMEOUT NULL** and **SET CURRENT LOCK TIMEOUT** use the predefined value in **LOCKTIMEOUT** of the database configuration parameter. This parameter can be obtained by the command **GET DATABASE CONFIGURATION FOR <dbname>** or **GET DB CFG FOR <dbname>**.

More information about the **SET CURRENT LOCK TIMEOUT** can be obtained in the *SQL Reference, Volume 2*, SC09-4845. You can also find this information by searching in the DB2 Information Center.

5.1.6 Improved query execution plans

Improved query execution plans were realized by better cardinality estimation (the number of distinct values) where the optimizer determines the number of qualifying rows after filtering operations. Accurate data distribution statistics are needed to produce accurate cardinality estimates. Query execution plans with accurate cardinality estimates perform much faster.

In DB2 Version 8.2, the optimizer can exploit distribution statistics from materialized query tables as well as detect correlation from column group statistics to ensure the most accurate cardinality estimates.

5.1.7 Multipage file allocation on SMS table spaces

Multipage file allocation is enabled by default and allocates disk space in an extent at a time instead of a page. You can enable multipage file allocation with the command:

```
db2empfa <database-name>
```

This command changes the value of the database configuration parameter `MULTIPAGE_ALLOC` to YES.

Note: Multipage file allocation cannot be disabled after it has been enabled. If multipage file allocation is not desired, the `DB2_NO_MPFA_FOR_NEW_DB` DB2 registry variable must be set appropriately before the database is created.

5.1.8 Automatic setting of table space prefetch size

When the prefetch size is not specified for a table space, DB2 now uses the value for the `DFT_PREFETCH_SZ` configuration parameter as the default. This parameter can now be set to `AUTOMATIC`, which enables DB2 to calculate an appropriate prefetch size for a table space based on the extent size, the number of containers, and the number of physical spindles per container.

For more information about the table space prefetch size monitor element, see the *System Monitor Guide and Reference*, SC09-4847, which is also in the DB2 Help Center.

5.2 Multidimension clustering tables

Multidimensional clustering (MDC) provides an elegant method for flexible, continuous, and automatic clustering of data along multiple dimensions. This

results in significant improvement in the performance of queries, as well as significant reduction in the overhead of data maintenance operations, such as reorganization, and index maintenance operations during insert, update, and delete operations. Multidimensional clustering is primarily intended for data warehousing and large database environments, and it can also be used in online transaction processing (OLTP) environments.

MDC enables a table to be physically clustered on more than one key (or dimension) simultaneously. Prior to Version 8, DB2 only supported single-dimensional clustering of data, through clustering indexes. Using a clustering index, DB2 maintains the physical order of data on pages in the key order of the index, as records are inserted and updated in the table. Clustering indexes greatly improves the performance of range queries that have predicates containing one or more keys of the clustering index. With good clustering, only a portion of the table must be accessed and, when the pages are sequential, more efficient prefetching can be performed.

With MDC, these benefits are extended to more than one dimension, or clustering key. In terms of query performance, range queries involving any combination of specified dimensions of the table will benefit from clustering. Not only will these queries access only those pages having records with the correct dimension values, these qualifying pages will be grouped by extents. Furthermore, although a table with a clustering index can become unclustered over time as space fills up in the table, an MDC table is able to maintain its clustering over all dimensions automatically and continuously, thus eliminating the need to reorganize the table to restore the physical order of the data.

Regular indexes are row-based (Figure 5-2 on page 169); MDC clustered indexes are block-based (Figure 5-3 on page 169). They are smaller than record-based indexes and take up less disk space and are faster to scan.

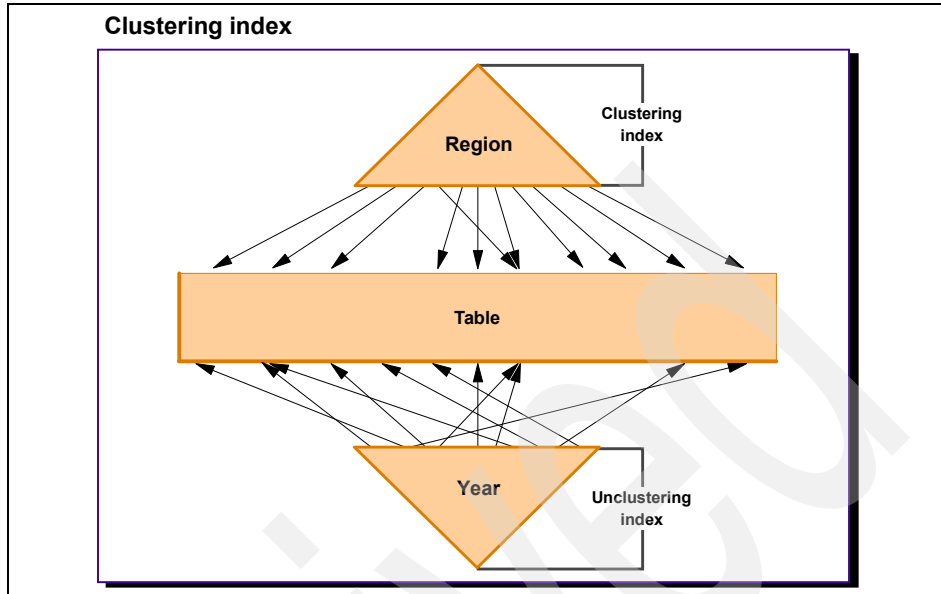


Figure 5-2 A regular table with a clustering index

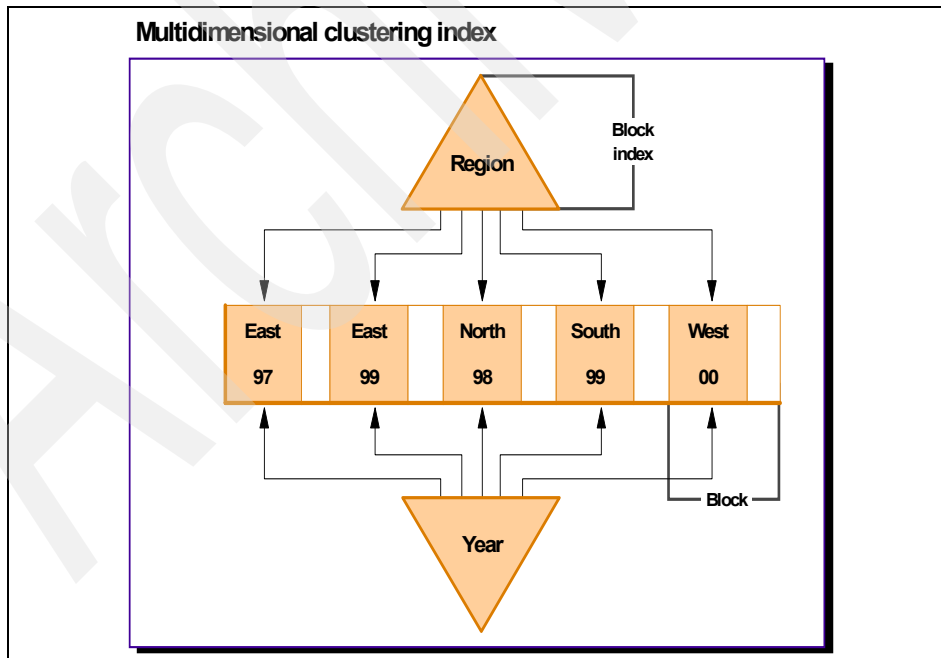


Figure 5-3 A multidimensional clustering table

MDCs have the following advantages over regular tables:

- ▶ Dimension block index lookups can identify the required portions of the table and quickly scan on the required blocks.
- ▶ Because block indexes are smaller than row-based indexes, lookups are faster.
- ▶ Index ANDing and ORing can be performed at the block level and combined with row-based indexes.
- ▶ Data is guaranteed to be clustered on extents, which makes retrieval faster.

Consider the following simple example (Example 5-3) for an MDC table named SALES with dimensions defined on the region and month columns.

Example 5-3 MDC index

```
SELECT * FROM SALES
WHERE YEAR=99 AND REGION='South'
```

For this query, the optimizer can perform a dimension block index lookup to find blocks in which the year '99 and the South region occur. Then it can quickly scan only the resulting blocks of the table to fetch the result set.

A very detailed description of MDC tables, their design, and a comparison of regular and MDC tables can be found in the *Administration Guide: Planning*, SC09-4822.

The new DB2 Design Advisor (formerly known as Index Advisor), either GUI-based or as the `db2adv` command, has now a built-in MDC feature.

5.2.1 MDC performance recommendations

Although the big advantage of MDC tables lies in physically clustering on more than one key, an MDC table defined with even a single dimension can benefit from these MDC attributes and can be a viable alternative to a regular table with a clustering index. Remember: The block-based indexes of an MDC table are drastically smaller.

The decision should be based on many factors, including the queries that make up a workload, the columns by which it is searched, and the nature and distribution of data in the table. Refer to “Considerations when choosing dimensions” and “MDC Advisor Feature on the DB2 Advisor” in the DB2 Information Center.

Note: The usefulness of block index during query processing depends on the order of its key parts. The key part order is determined by the order of the columns encountered by the parser when parsing the dimensions specified in the ORGANIZE BY clause of the CREATE TABLE statement. Refer to “Block index considerations for MDC tables” for more information.

Multidimensional clustering (MDC) tables in SMS table spaces

If you plan to store MDC tables in an SMS table space, we strongly recommend that you use multipage file allocation. See 5.1.7, “Multipage file allocation on SMS table spaces” on page 167.

The reason for this recommendation is that MDC tables are always extended by whole extents, and it is important that all pages in these extents are physically consecutive. Therefore, there is no space advantage to disabling multipage file allocation; and furthermore, enabling it will significantly increase the chances that the pages in each extent are physically consecutive.

Note: Multipage file allocation is the default for newly created databases in Version 8.2 and later.

Improving the performance of the load utility

Increase the UTIL_HEAP_SZ of the database parameter. The load algorithm will perform significantly better when loading MDC tables because this will reduce disk I/O during the clustering of data that is performed during the load phase. Also choose bigger DATA BUFFER sizes when using this option. Because of extended logging requirements, you should also increase the LOGBUFSZ database configuration parameter.

5.3 Improvement of the RUNSTATS utility

The RUNSTATS utility has been improved for performance with a sampling feature that uses only a subset of its data and with the possibility to throttle its execution.

5.3.1 Improved RUNSTATS performance through sampling

Table statistics are used by the query optimizer in selecting the best access plan for any given query, so it is important that statistics remain current to accurately reflect the state of a table at any given time. As the activity against a table increases, so should the frequency of statistics collection. With the increasing size of databases, it is becoming more important to find efficient ways to collect

statistics. Random sampling of table data on which the statistics are to be collected can reduce the amount of time that it takes to collect statistics. For I/O-bound or CPU-bound systems, the performance benefits can be enormous. The smaller the sample, the faster statistics collection completes.

Starting in Version 8.2, the RUNSTATS command provides the option to collect statistics on a sample of the data in the table by using the TABLESAMPLE option. This feature can increase the efficiency of statistics collection because sampling uses only a subset of the data. At the same time, the sampling methods ensure a high level of accuracy.

You can specify the table sampling methods Bernoulli or System the same way you do in SQL queries (5.1.3, “Data sampling in SQL queries” on page 162). The new parameters are:

- ▶ TABLESAMPLE BERNOULLI
- ▶ TABLESAMPLE SYSTEM

These two options can be specified by the REPEATABLE parameter if there is a need for exactly the same results. Example 5-4 shows the RUNSTATS command using new options.

Example 5-4 Runstats using the data sampling method with repeatable results

```
RUNSTATS ON TABLE db2user.employee WITH DISTRIBUTION  
TABLESAMPLE BERNOULLI(30) REPEATABLE(4196)
```

5.3.2 Throttling of the RUNSTATS utility

Executing resource-intensive utilities like RUNSTATS can negatively impact overall database performance. However, in order to maintain efficient database operation, statistics must be collected regularly, leaving database administrators with the task of identifying periods of time when the impact of utility execution is most tolerated. In many environments, there are no regular windows of reduced database activity.

Throttling of the RUNSTATS utility limits the amount of resources consumed by the utility, based on the current level of database activity. When database activity is low, the utility runs more aggressively; when database activity increases, the resources allocated to executing RUNSTATS are reduced.

You can now specify the RUNSTATS command with the priority parameter UTIL_IMPACT_PRIORITY. You can specify values from 1 to 100. A value of 100 means no throttling of the RUNSTATS command and has the same effect as not specifying the UTIL_IMPACT_PRIORITY, because the default is not throttling the command.

Note: You also have the choice to limit a performance degradation by throttling utilities with the UTIL_IMPACT_LIM database manager configuration parameter.

The DBA can then run online utilities during critical production periods and be guaranteed that the performance impact on production work will be within acceptable limits.

Automatic statistics profiling of the RUNSTATS utility is another performance improvement feature. 3.4.2, “Automatic statistics profiling” on page 76 has a detailed discussion.

5.4 DB2 performance elements in Windows

Windows Task Manager has a Performance tab that enables you to see the graph of current CPU and memory usage in real time (Figure 5-4).

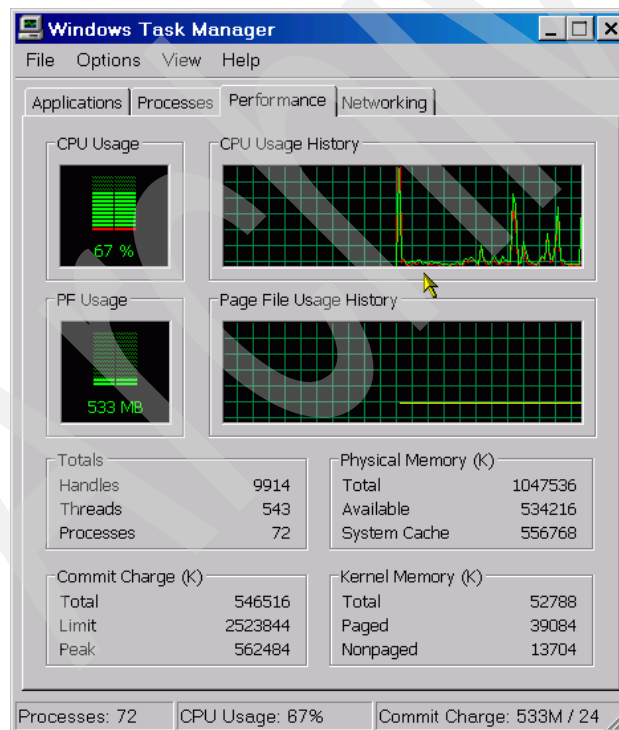


Figure 5-4 Performance screen of the Task Manager

The only option available is to split CPU times between the Kernel and User portions by enabling **View** → **Show Kernel times** in the main menu.

This monitor does not allow adding any additional performance-related counters and can be used for a quick glance at CPU and memory utilization.

Windows has another monitor: the System Performance Monitor, which is available from **Start Menu** → **Control Panel** → **Administrative Tools** → **Performance**. This performance monitor enables monitoring of local and remote machines and allows selection of many performance counters.

DB2 UDB V8.2 is fully integrated with System Performance Monitor. You can select Database Instance related counters, Active Database related counters, or Active Database Application related counters. Figure 5-5 shows the Add Counters window from which you add the database-related counters.

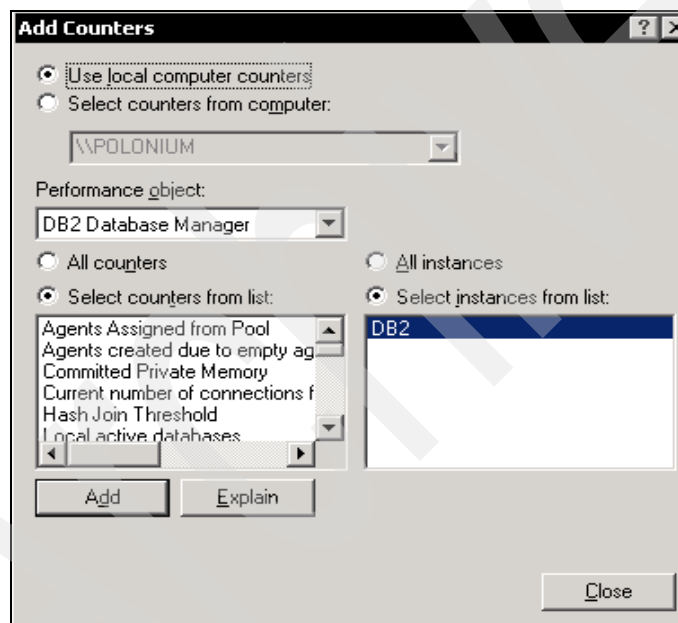


Figure 5-5 Database Manager Instance Performance counters.

Notice that the instance list in the right pane only shows active (started) instances. Similarly, if you choose **DB2 Applications** from the **Performance object** pull-down list in the left pane, you will see only active applications.

You may also choose **DB2 Databases** from the **Performance object** pull-down list and you will be able to choose from active databases, for which at least one connection exists or which have been activated with the **activate database** command.

The list of performance counters is very detailed; for example, the list of counters that are available for active instance is quite large, as shown in Figure 5-6.

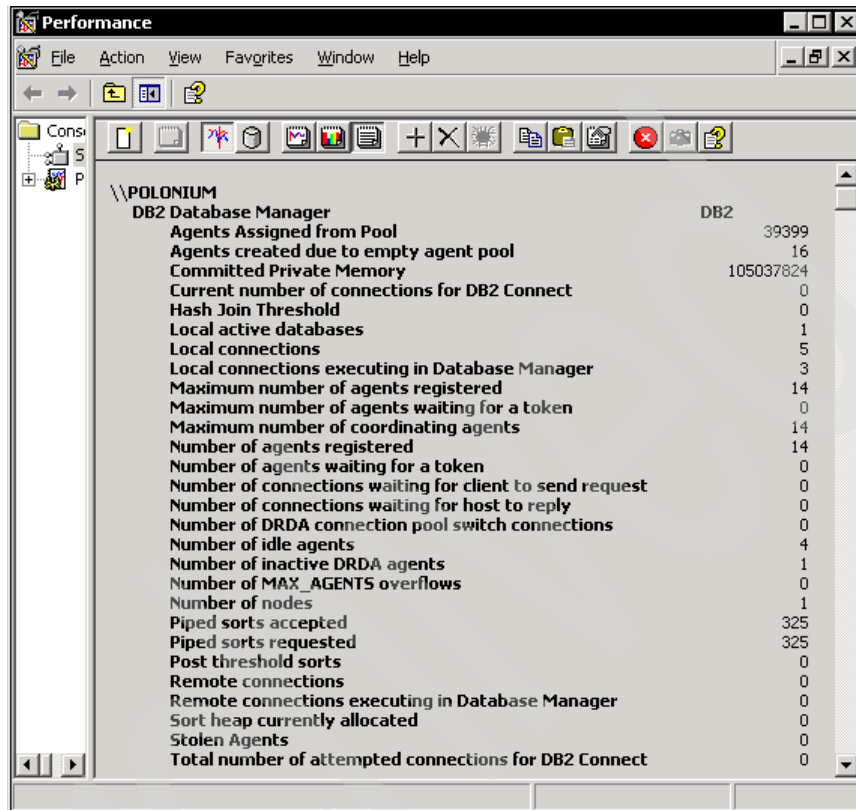


Figure 5-6 Database Manager instance performance counters

There is a much longer list of performance counters available for active database and active applications.

If you are not a performance expert, you may find the amount of information overwhelming for day-to-day use. This is precisely why you will find a new feature of DB2 UDB V8.2, Activity Monitor, so attractive.

5.5 DB2 Activity Monitor

Activity Monitor is a new tool that helps database administrators in resolving application performance issues, application tuning, and SQL tuning tasks.

Activity Monitor can be started as a task from Control Center or from the Start menu: **Programs** → **IBM DB2** → **Monitoring Tools** → **Activity monitor**.

Suppose that a DBA receives calls that simple SQL queries which query DEPARTMENT table and return department number and manager number of a given department number sometimes are returning result after a very long time.

We start Activity Monitor to see what is happening, using this simple example.

1. Select a database from the pull-down list, which opens a System, Instance, and Database selection window (Figure 5-7).

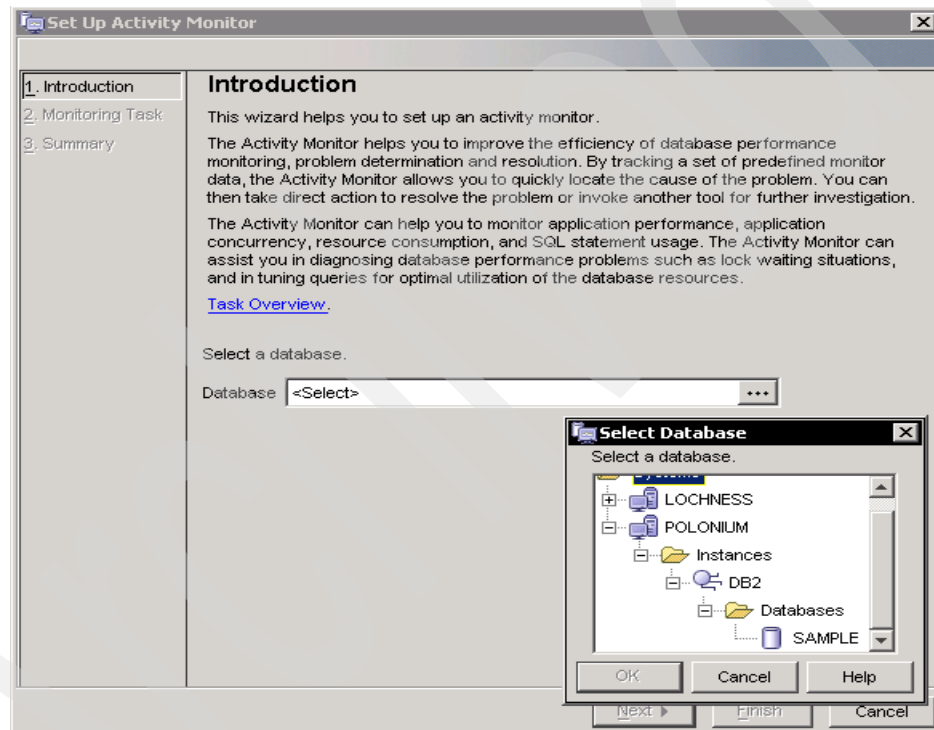


Figure 5-7 Activity Monitor Setup Wizard

Figure 5-8 shows the Introduction dialog with the selected database.

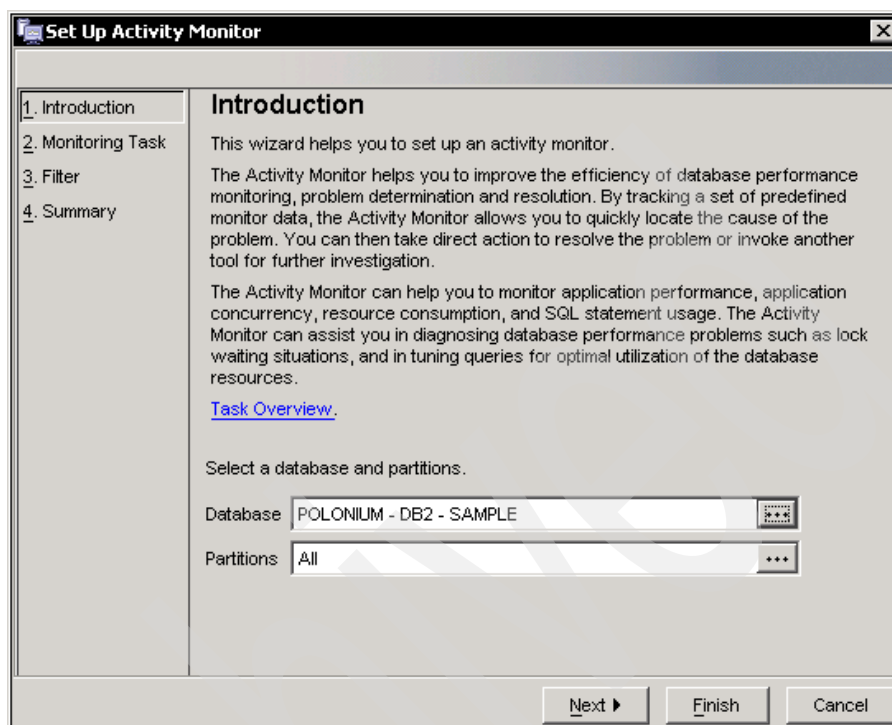


Figure 5-8 Activity Monitor: Introduction

2. Choose from predefined monitoring tasks or create a new one (Figure 5-9). When creating a new one, we can use as a base one of the IBM-supplied predefined tasks. For our example, we use one of existing monitoring tasks, **Resolving the performance degradation of an application**, which most closely resembles our problem. Click **Next**.

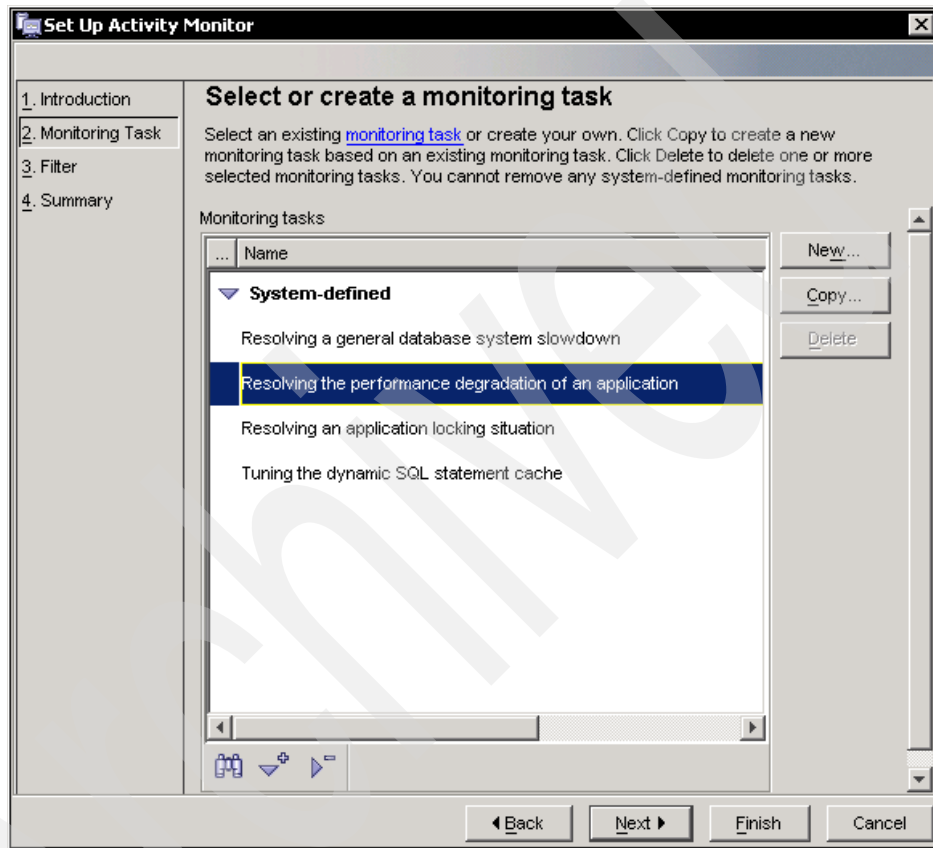


Figure 5-9 Selecting a monitoring task

3. Specify a filter for applications to be monitored (Figure 5-10). We are not sure of the reason for application performance degradation or even which application is causing this, so we leave the filter at the default selection of **All applications** and click **Next**.

Set Up Activity Monitor

1. Introduction
2. Monitoring Task
3. Filter
4. Summary

Specify a filter for applications to be monitored

Select the applications that you want to monitor. You can accept the default setting and monitor all applications, or you can specify a filter by selecting applications that meet specific conditions, which enables you to focus on a particular group of applications.

☒ All applications

☐ Applications that meet the following conditions

Attribute	Comparison	Values
Authorization ID	=	
Application Name	=	
Application Handle (agent ID)	=	

Clear

☒ Meet all conditions ☐ Meet any Conditions

Back Next Finish Cancel

Figure 5-10 Filter selection screen

4. We see five applications on the list of top CPU-time-consuming applications (Figure 5-11). Some are not relevant; for example, javaw.exe is a DB2 Activity Monitor itself. We see that there are three applications that did not consume any CPU time; perhaps they are waiting for something? We therefore concentrate on the CLP application (db2bp.exe) with application handle 173.

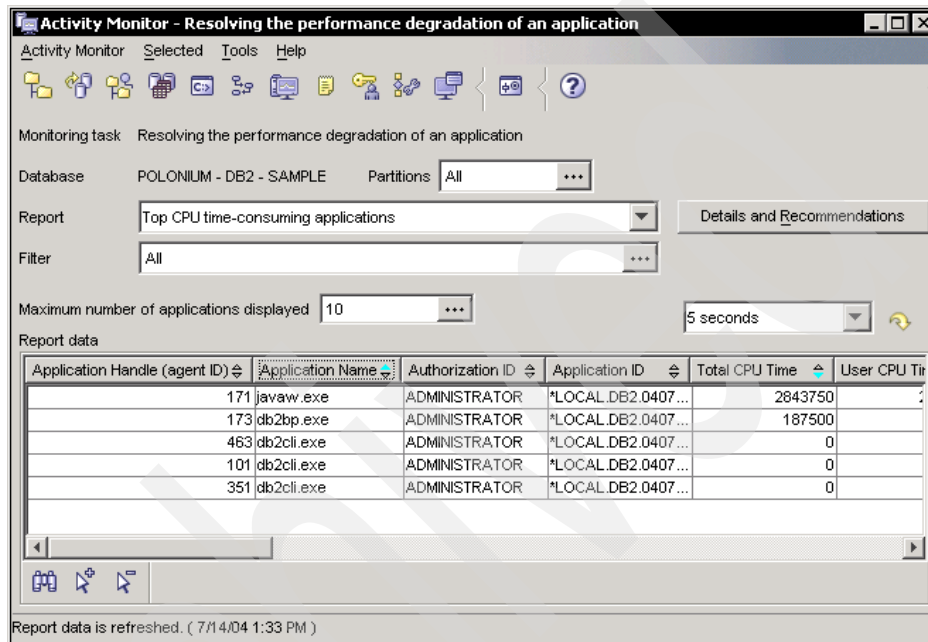


Figure 5-11 Top CPU-time-consuming applications

5. Right-click the application with handle 351, which we know is slow, to open its context menu (Figure 5-12). Select **Show Lock Chains**. By our theory, other applications are waiting for something, so we see whether this could be Lock Waits.

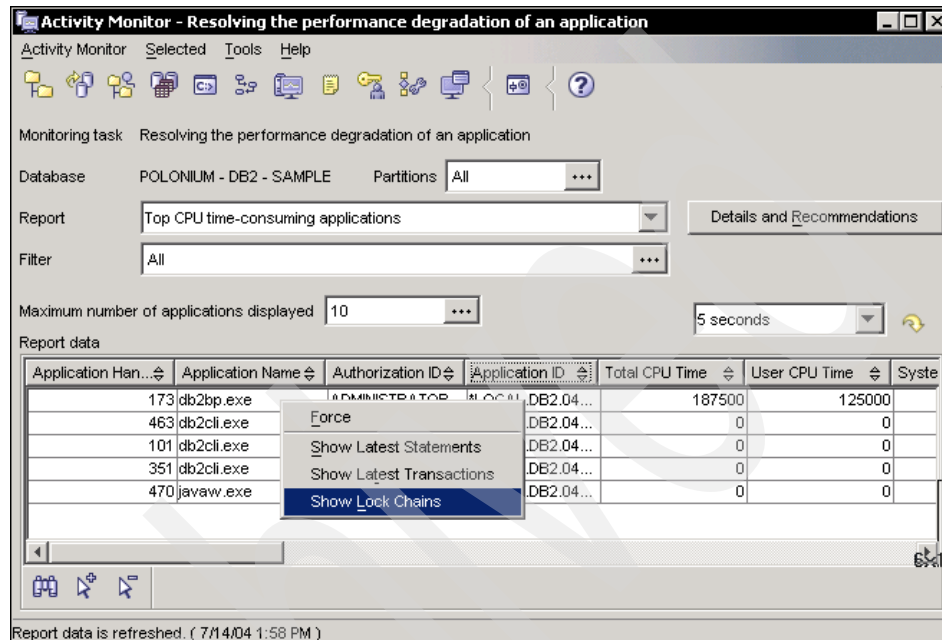


Figure 5-12 Selecting lock chains for an application

6. The answer is that the applications we received complaints about are indeed in lock wait, waiting for a lock acquired by another application: not the application with handle 173 but the application with handle 463 (Figure 5-13). We could come to the same conclusion using Snapshot monitor and getting snapshots of the database for all applications and for locks on a database, but Activity Monitor helped us find an answer quickly with the ease of a GUI.

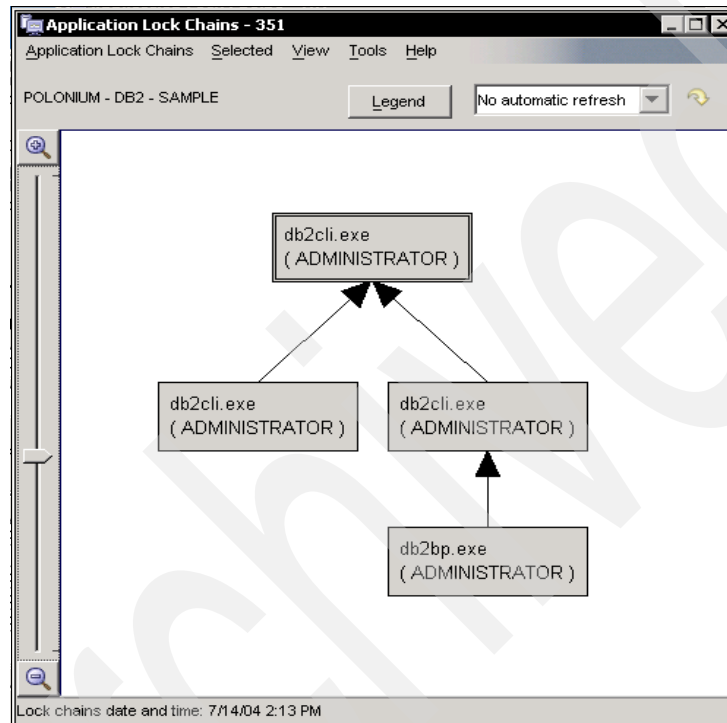


Figure 5-13 Application lock chains

7. But why are all applications waiting for locks and why is the application with handle 463 holding some locks? Examine lock details for this application by right-clicking the application in the lock chains chart and selecting **Show Lock Details** from the context menu(Figure 5-14).

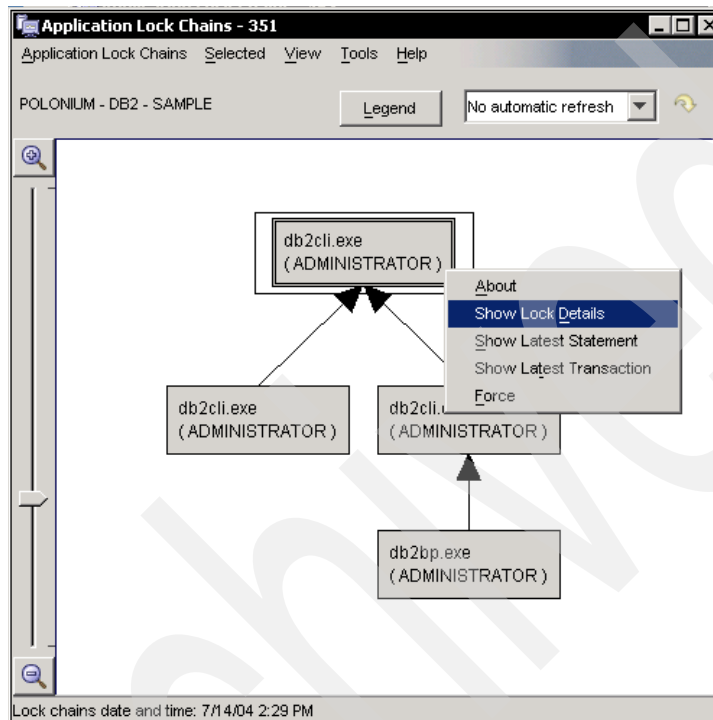


Figure 5-14 Show Lock Details

8. We can see that our culprit has an IX (intent exclusive) lock on table and X (eXclusive) lock on the row (Figure 5-15). It appears that this application is updating the DEPARTMENT table by inserting or deleting a row and holding the transaction, which may be not committing often enough or may be waiting for user input to confirm change.

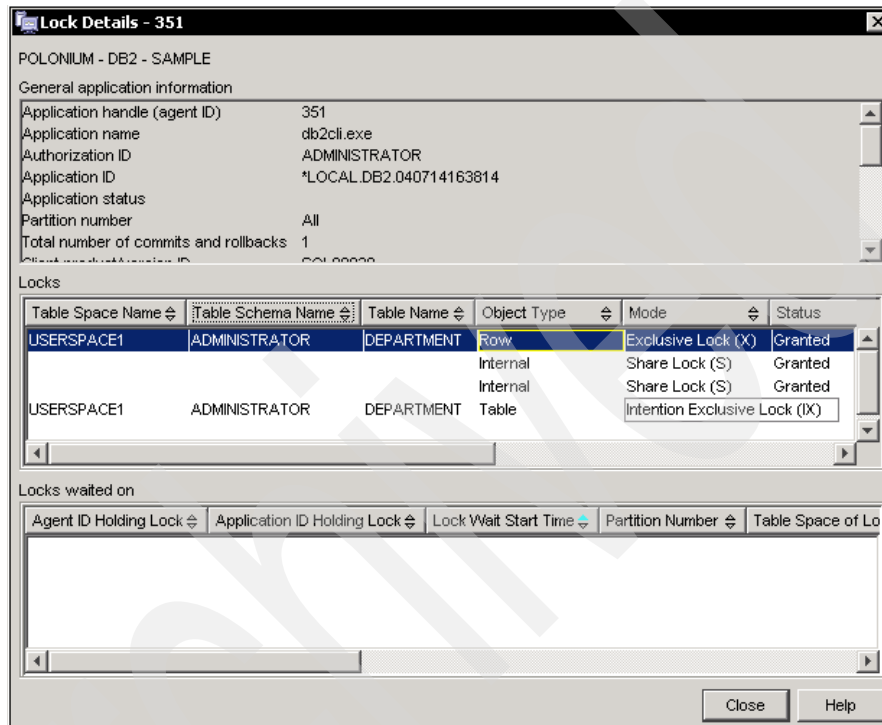


Figure 5-15 Application lock details

9. The Show Latest Statement option shows the exact SQL statement for which application is running and holds the lock. Right-click the statement and select **Show Statement Text** displays the whole SQL statement (Figure 5-16).

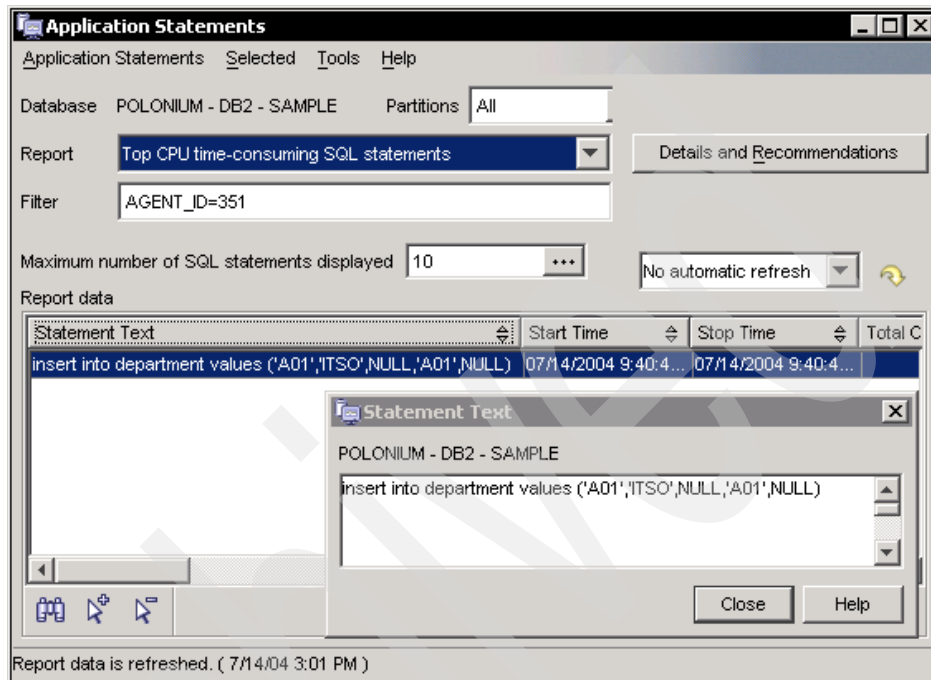


Figure 5-16 Show application statement

10. Now you can use the Force option to force off the application that holds the lock, or take the opportunity to find whether there is other reason for the application's slowdown. For example, application 173 is a simple select statement. We would like to know whether there is a reason other than lock wait that makes the query run slowly. In the Application Statements dialog, right-click the statement and select **Explain SQL** (Figure 5-17).

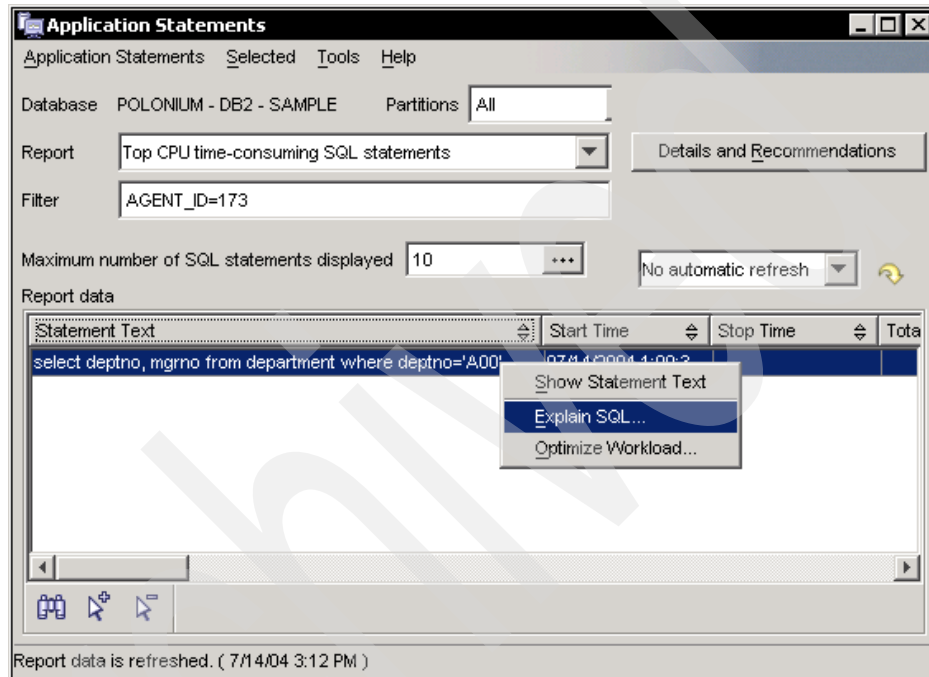


Figure 5-17 Explain SELECT SQL statement

Figure 5-18 shows the Explain SQL setup screen. You also can get a different query from a file you previously saved. Click **OK** to proceed.

Explain SQL Statement - SAMPLE

POLONIUM - DB2 - SAMPLE

SQL text

select deptno, mgrno from department where deptno='A00'

Get

Save

Query number 1

Query tag

Optimization class 5

☒ Populate all columns in Explain tables

OK Cancel Help

Figure 5-18 Explain SQL Statement setup

11. DB2 displays the query access plan (Figure 5-19). From here you can find details of the access plan. If you find out that the query ran slowly because the statistics were not updated, you can invoke the Control Center and run RUNSTATS on the table. For more about using the DB2 UDB Explain tool, refer to *System Monitor Guide and Reference*, SC09-4847, and *Guide to GUI Tools for Administration and Development*, SC09-4851.

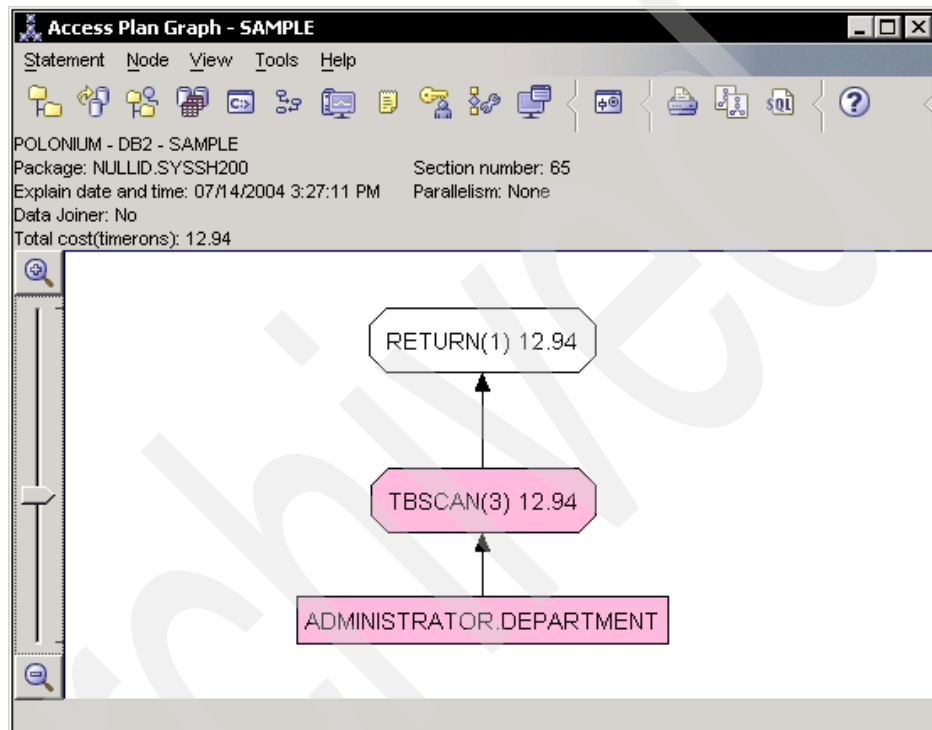


Figure 5-19 Access plan

High availability

Today's e-business enterprises are facing the challenge of 24 x 7 availability. These businesses require protection against both localized failures and entire site disasters. The disaster recovery system should be simple to set up and administer, with minimal impact on performance. When a system failure occurs, the failover and fallback between the primary and backup system should be fast and transparent for applications. DB2 UDB V8.2 offers a High Availability Disaster Recovery (HADR) feature, which provides a solution that meets all of the requirements. This chapter provides the detailed setup process of HADR. In addition, high availability features Automatic Client Reroute and Index Logging are also discussed.

6.1 High Available Disaster Recovery

High Availability Disaster Recovery (HADR) is a replication that takes place at the database level. The HADR requires two active systems: *primary* and *standby*. All database transactions take place at the primary system. The transaction log entries are continuously shipping to the standby machine via TCP/IP. The standby system receives and stores log entries from the primary system and applies the transactions. If the primary failed, the standby can take over the transactional workload and become the new primary machine. Implementing with the DB2 client reroute feature, the application will be rerouted to the new primary machine by DB2. When the failed machine becomes available again, it can be resynchronized and restored as the primary. Figure 6-1 illustrates the basic principles of HADR.

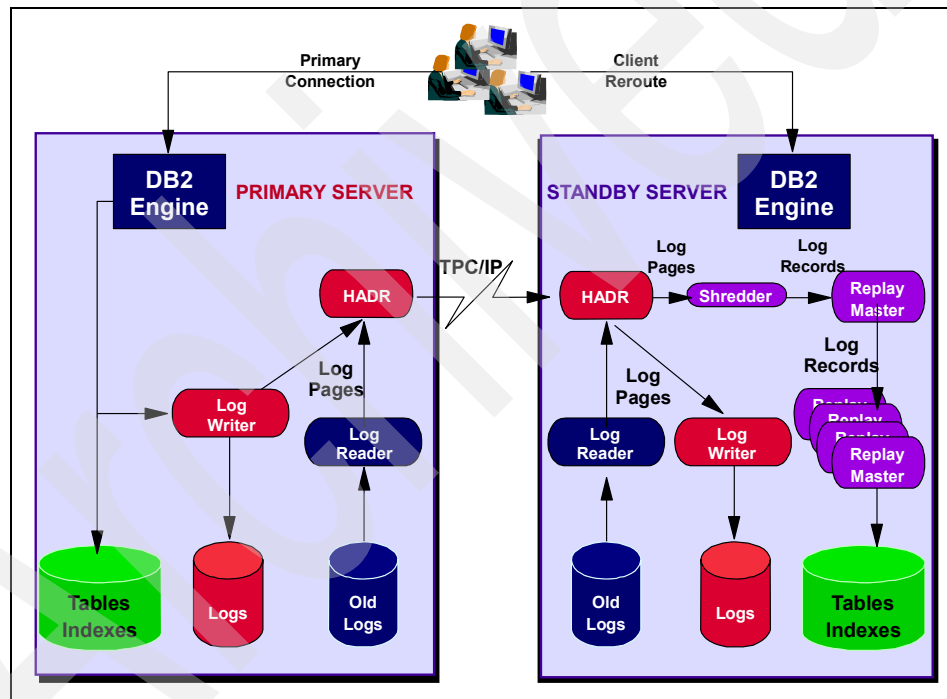


Figure 6-1 HADR implementation

Setup

Setting up HADR is simple. DB2 UDB V8.2 provides both GUI and commands to set up HADR. The setup procedure is as follows

- Prepare the primary stem: Set up the database logging to archive logging.

- ▶ Prepare the standby by cloning the primary:
 - Install DB2 UDB V8.2 on standby system, if not already done so.
 - Use the DB2 restore facility, flash copy, or split mirror to create the database in the standby system.
- ▶ Start the standby.
- ▶ Start the primary.

Our lab environment has the following system:

- ▶ Primary: xSeries® named Polonium
- ▶ Standby: Netfinity® named Jamaica

All of the disk drives are attached; there is no SAN system. We use the HADR Setup wizard to perform the configuration:

1. From the primary system, start the Control Center. There are two ways to invoke the HADR setup wizard:
 - Select **Tools** → **Wizards** → **Set Up High Availability Disaster Recovery (HADR) Databases** (Figure 6-2).

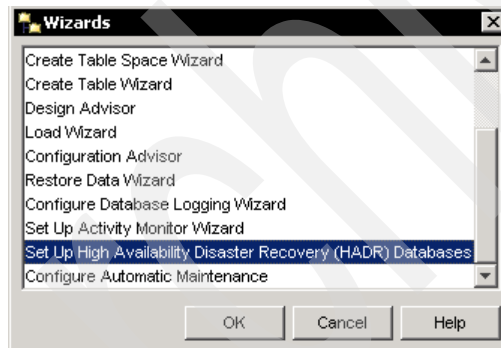


Figure 6-2 Invoking HADR Setup wizard from Tools bar

- Right-click the database and select **High Availability Disaster Recovery** → **Set Up** (Figure 6-3 on page 192).

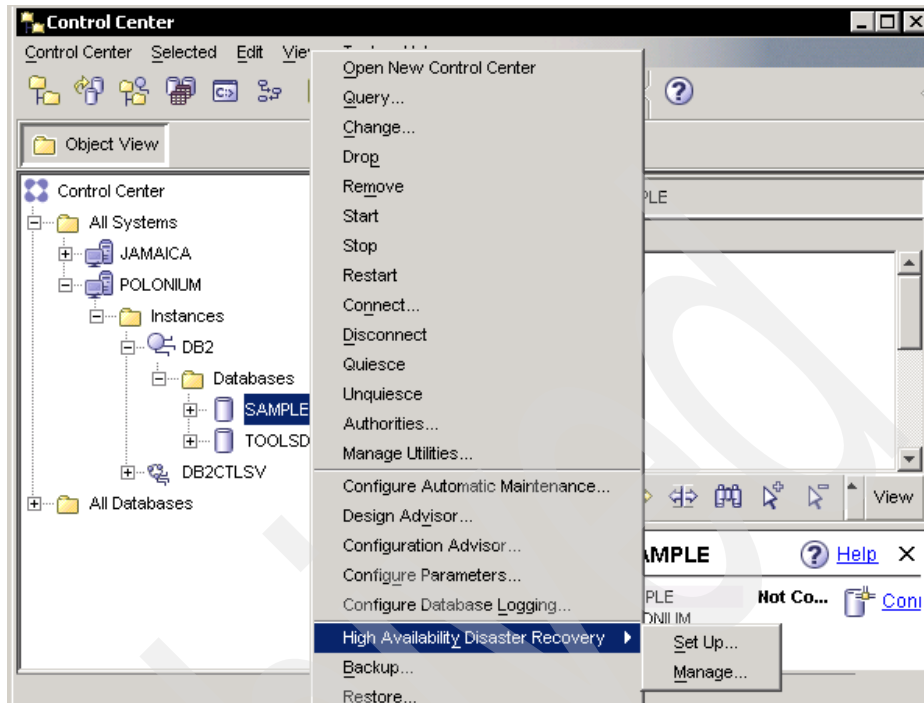


Figure 6-3 Invoking HADR Setup wizard

2. The Database Selection window shows the systems, instances, and databases that are cataloged on the database server. Select the system, instance, and database of the primary system (Figure 6-4).

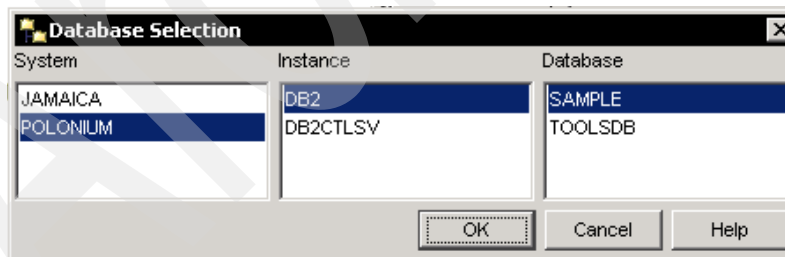


Figure 6-4 HADR Database Selection

3. The first HADR setup window introduces the HADR functions, a total of 10 setup steps (Figure 6-5). Click **Next**.

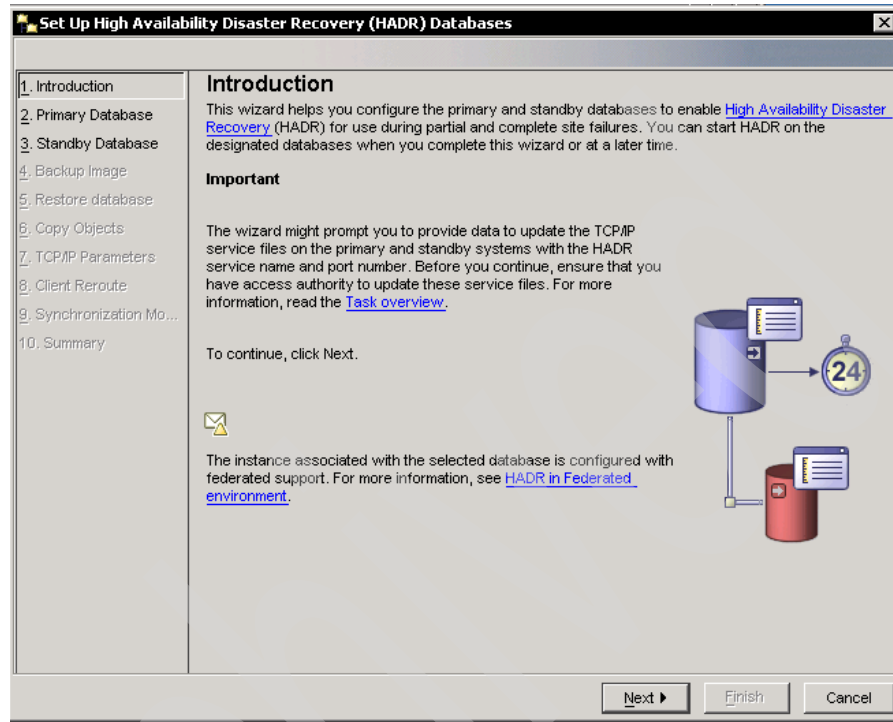


Figure 6-5 HADR Introduction

4. The HADR setup wizard checks whether the primary database has archive logging configured. Click **Next**.

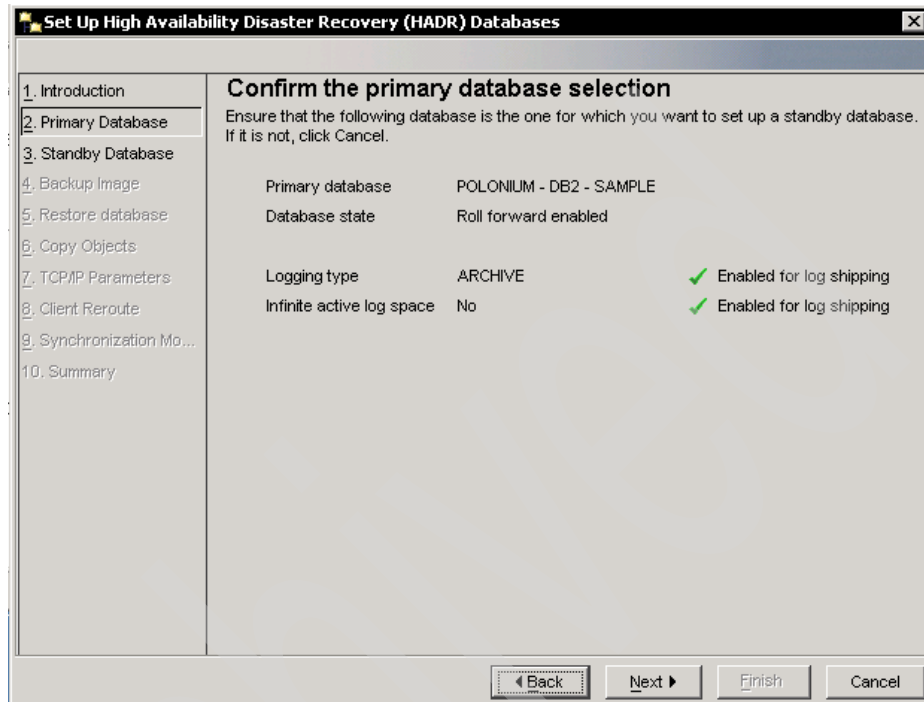


Figure 6-6 HADR: Confirm the primary database selection

5. Under Identify a standby database (Figure 6-7), specify the standby system and instance and click **Add**. A pull-down list shows all instances and databases cataloged in the primary machine. There are three options to initialize the standby database, and we select the first: **Use the backup image of the primary database to initialize a standby database**.

The screenshot shows a wizard window titled "Set Up High Availability Disaster Recovery (HADR) Databases". On the left is a navigation pane with steps 1 through 10. Step 3, "Standby Database", is selected. The main area is titled "Identify a standby database" and contains the following elements:

- A sub-header: "Identify a standby database".
- Instructions: "Select the standby system and instance. If the standby system and instance are not currently cataloged, click Add to catalog them."
- Form fields:
 - "System name": A dropdown menu showing "JAMAICA" and an "Add..." button.
 - "Instance name": A dropdown menu showing "JAMDB2 (DB2)" and an "Add..." button.
- A section titled "Standby database initialization options:" with three radio buttons:
 - ☒ "Use a backup image of the primary database to initialize a standby database". Below this is explanatory text: "You can select from a list of existing backup images or create a new backup image. The following pages will help you to initialize the standby database using the DB2 UDB [backup](#) and [restore](#) utilities."
 - ☐ "Use another existing database as the standby database". Below this is the instruction: "Enter a database alias name to catalog the standby database." followed by a "Database name" field with the value "SAMPLE", a "Database alias name" text box, and an "Add Database..." button.
 - ☐ "Use suspended I/O and online split mirror to initialize a standby database". Below this is explanatory text: "To create a split mirror, consult the storage vendor documentation applicable for your device. Exit from this wizard to perform the split mirror procedure. Once you have created the mirror database, return to this wizard to configure the new database as the standby database. For more information, see [Suspended I/O and online split mirror](#)."
- At the bottom are four buttons: "Back", "Next", "Finish", and "Cancel".

Figure 6-7 HADR: Identify a standby database

6. The next window gives three options for you to specify the database backup image for initializing the standby database:
 - **Select a backup image from the list provided:** HADR setup wizard lists all database backup images in the primary system for you to choose.
 - **Back up the primary database:** This option takes you to the database backup wizard to back up the database.
 - **Enter the backup image information:** Enables you to specify the backup image source (file system, tape, TSM, XBSM) on either the primary or standby machine.

We select the first option and choose the offline backup image taken previously (Figure 6-8).

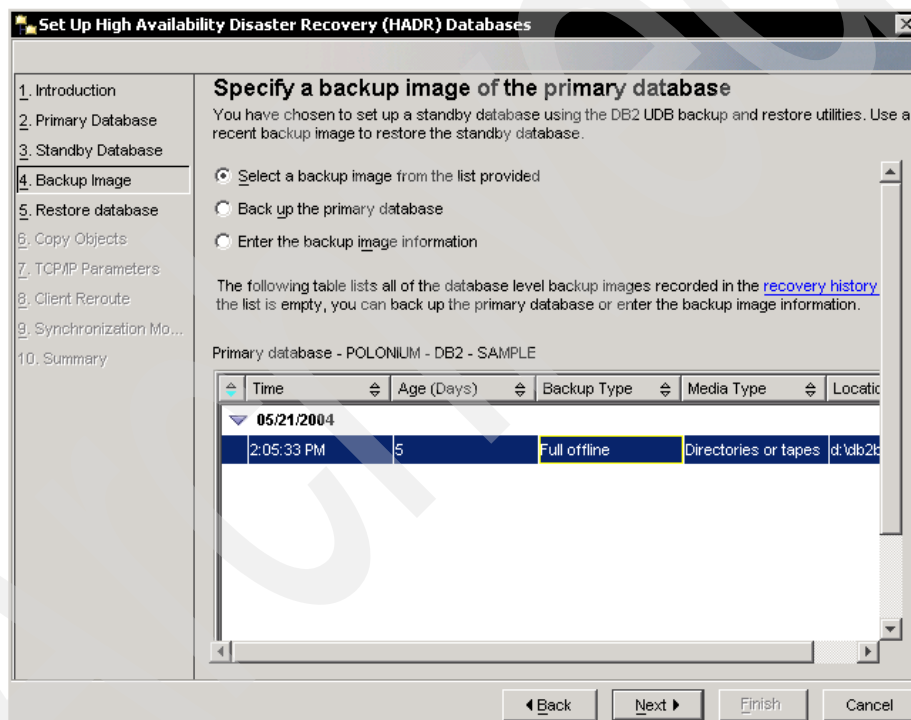


Figure 6-8 HADR: Specify a backup image of the primary database

7. You can let HADR setup wizard ships the database image copy to the standby system to the specified location as shown in Figure 6-9.

Set Up High Availability Disaster Recovery (HADR) Databases

1. Introduction
2. Primary Database
3. Standby Database
4. Backup Image
5. Restore database
6. Copy Objects
7. TCP/IP Parameters
8. Client Reroute
9. Synchronization Mo...
10. Summary

Restore database on standby system

Standby database - JAMAICA - JAMDB2 (DB2)

The following values show the cataloged alias name identical to the primary database name.

Database name	SAMPLE
Database alias name	JAMDB

Copy backup image - Directories or tapes - 05/28/2004 6:47:27 PM

Before you restore the database on the standby system, you might need to copy the backup image.

☒ Copy the backup image from the primary system to the standby system.
☐ Do not copy. The backup image location is already accessible on the standby system.

Backup image location

Primary system - POLONIUM	Standby system - JAMAICA
d:\db2backup	D:\db2backup

Restore standby database

To set advanced restore options, click the browse push button.

RESTORE DATABASE SAMPLE FROM "D:\db2backup" TAKEN AT 20040528184727 REPLACE HISTORY FILE WITHOUT PROMPTING

Back Next Finish Cancel

Figure 6-9 HADR: Restore database on standby system

8. Objects that are stored externally (such as stored procedures and UDFs) are not included in the database backup image. This step enables you to specify those objects to be copied from the primary system to the standby system (Figure 6-10). We do not have any external objects, so we click **Next**.

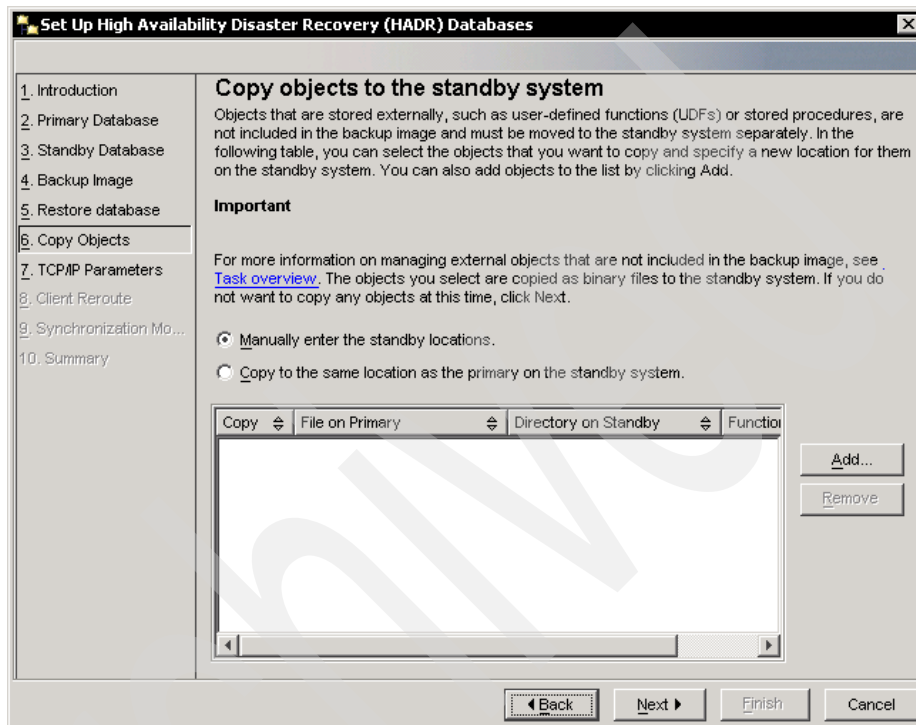


Figure 6-10 HADR: Copy objects to the standby system

9. Primary and standby systems communicate through TCP/IP. You can specify the service name and port number for HDAR in primary and standby systems (Figure 6-11). Click **View Service File** to view the service names and port numbers that are already defined in the service file.

Set Up High Availability Disaster Recovery (HADR) Databases

1. Introduction
2. Primary Database
3. Standby Database
4. Backup Image
5. Restore database
6. Copy Objects
7. TCP/IP Parameters
8. Client Reroute
9. Synchronization Mo...
10. Summary

Specify TCP/IP communication parameters

TCP/IP connections are used between the primary and standby systems. When HADR makes a connection between the primary and standby databases, it looks up the TCP/IP service name and host name for each of the database configurations.

Important

If the [HADR service name](#) and [port number](#) pair is not specified in the service file, ensure that you have necessary file permission before you continue. Without permission, the service file update action will fail.

Primary database - POLONIUM - DB2 - SAMPLE

Host name: [View IP Information...](#)

HADR service name: [View Service File...](#)

HADR port number:

Standby database - JAMAICA - JAMDB2 (DB2) - JAMDB (SAMPLE)

Host name: [View IP Information...](#)

HADR service name: [View Service File...](#)

HADR port number:

◀ Back Next ▶ Finish Cancel

Figure 6-11 HADR: Specify TCP/IP communication parameters

10. The next window is an optional configuration that sets the client reroute information for DB2 to reroute the application after HADR failover has been manually issued. We select this option (Figure 6-12).

Set Up High Availability Disaster Recovery (HADR) Databases

1. Introduction
2. Primary Database
3. Standby Database
4. Backup Image
5. Restore database
6. Copy Objects
7. TCP/IP Parameters
8. Client Reroute
9. Synchronization Mo...
10. Summary

Configure databases for automatic client reroute

[Client reroute](#) permits a client application to continue its work with minimal interruption after an [HADR failover](#) has been manually issued to the database that it is accessing.

☒ Specify alternate server for databases

Primary database - POLONIUM - DB2 - SAMPLE

Alternate host name: JAMAICA
Alternate port number: 50000

Standby database - JAMAICA - JAMDB2 (DB2) - JAMDB (SAMPLE)

Alternate host name: POLONIUM
Alternate port number: 50000

◀ Back Next ▶ Finish Cancel

Figure 6-12 HADR: Configure databases for automatic client reroute

11. The next window for specifying the synchronization mode for peer state log writing. The synchronization mode specifies the log write behavior between primary and standby when they are in peer state. There are three synchronization modes:
- Synchronous (zero data loss): In this mode, DB2 gives commit acknowledgement to an application only when the transaction is written to disk on both primary and standby. Therefore, no transaction will be lost even if both primary and standby failed.

- Near synchronous: In this mode, the log write at primary and the send to the standby are performed in parallel at the primary. A commit succeeds when the log data is on disk at the primary and it has been received by the standby. This is a no transaction loss in single site failure scenario.
- Asynchronous: In this mode, a commit succeeds when the log data is on disk at the primary and it has been sent to the standby. The log data transmission is guaranteed by TCP/IP sockets. There is no acknowledgement of log data transmission between primary and standby.

There is some performance trade-off between modes. You should choose the synchronization mode based on the business requirement. We use **Near synchronous** mode, see Figure 6-13.

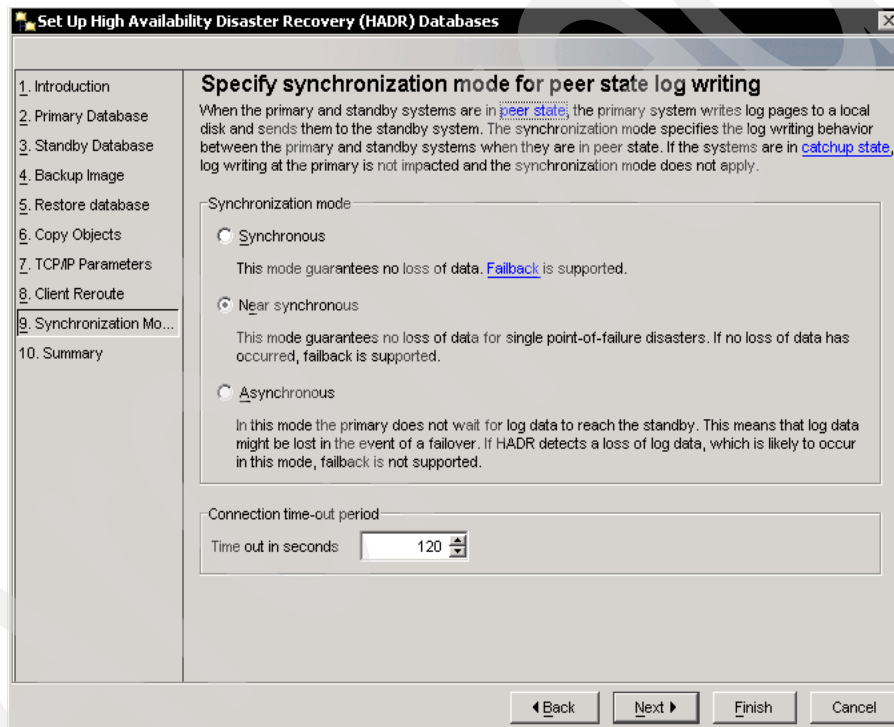


Figure 6-13 HADR: specify synchronization mode

12. The last window shows a summary of the HADR setup information. You can specify whether you want to start HADR immediately or later (Figure 6-14).

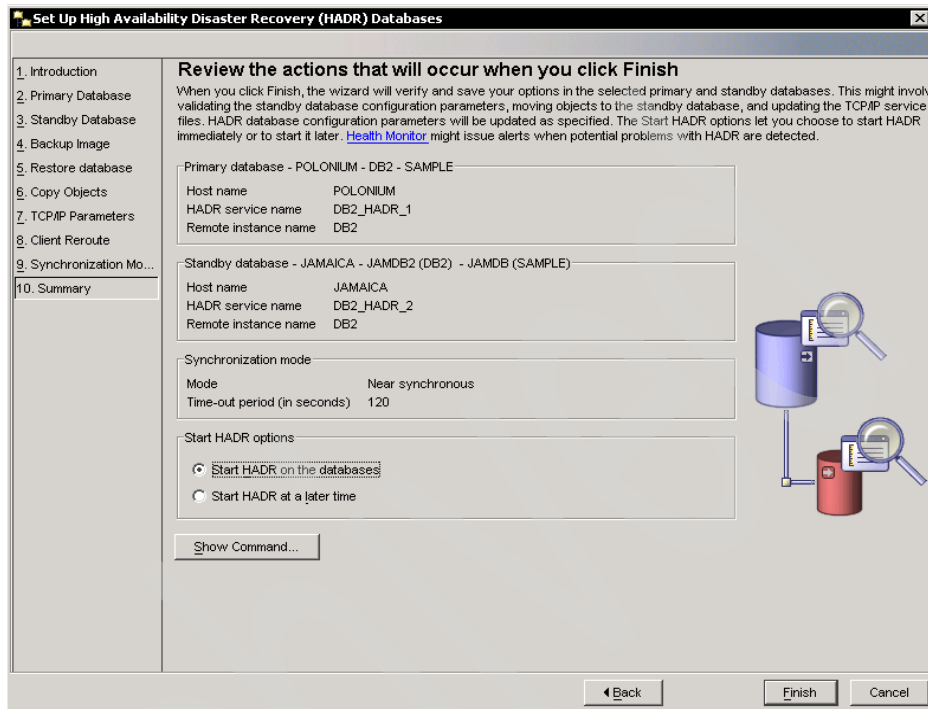


Figure 6-14 HADR: Summary

Click **Show Command** to display the entire HADR setup command script, which is shown in Example 6-1.

Example 6-1 HADR setup script

```
--
-- Update configuration parameters on primary database - POLONIUM - DB2 -
SAMPLE
--
UPDATE DB CFG FOR DB SAMPLE USING LOGINDEXBUILD ON
UPDATE DB CFG FOR DB SAMPLE USING INDEXREC RESTART
--
-- Copy backup images from primary to standby system.
--
-- Location on primary system : d:\db2backup
-- Location on standby system : D:\db2backup
--
-- Restore database on standby system - JAMAICA - JAMDB2 (DB2) - JAMDB
(SAMPLE)
```



```

--
RESTORE DATABASE SAMPLE FROM "D:\db2backup" TAKEN AT 20040528164636 REPLACE
HISTORY FILE WITHOUT PROMPTING
--
-- Configure databases for client reroute - POLONIUM - DB2 - SAMPLE
--
UPDATE ALTERNATE SERVER FOR DATABASE SAMPLE USING HOSTNAME JAMAICA PORT 50000
--
-- Configure databases for client reroute - JAMAICA - JAMDB2 (DB2) - JAMDB
(SAMPLE)
--
UPDATE ALTERNATE SERVER FOR DATABASE SAMPLE USING HOSTNAME POLONIUM PORT 50000
--
-- Update service file on primary system - POLONIUM
-- Service name : DB2_HADR_1
-- Port number : 55001
-- Service name : DB2_HADR_2
-- Port number : 55002
--
-- Update service file on standby system - JAMAICA
-- Service name : DB2_HADR_1
-- Port number : 55001
-- Service name : DB2_HADR_2
-- Port number : 55002
--
-- Update HADR configuration parameters on primary database - POLONIUM - DB2 -
SAMPLE
--
UPDATE DB CFG FOR SAMPLE USING HADR_LOCAL_HOST POLONIUM
UPDATE DB CFG FOR SAMPLE USING HADR_LOCAL_SVC DB2_HADR_1
UPDATE DB CFG FOR SAMPLE USING HADR_REMOTE_HOST JAMAICA
UPDATE DB CFG FOR SAMPLE USING HADR_REMOTE_SVC DB2_HADR_2
UPDATE DB CFG FOR SAMPLE USING HADR_REMOTE_INST DB2
UPDATE DB CFG FOR SAMPLE USING HADR_SYNCMODE NEARSYNC
UPDATE DB CFG FOR SAMPLE USING HADR_TIMEOUT 120
CONNECT TO SAMPLE
QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS
UNQUIESCE DATABASE
CONNECT RESET
--
-- Update HADR configuration parameters on standby database - JAMAICA - JAMDB2
(DB2) - JAMDB (SAMPLE)
--
UPDATE DB CFG FOR SAMPLE USING HADR_LOCAL_HOST JAMAICA
UPDATE DB CFG FOR SAMPLE USING HADR_LOCAL_SVC DB2_HADR_2
UPDATE DB CFG FOR SAMPLE USING HADR_REMOTE_HOST POLONIUM
UPDATE DB CFG FOR SAMPLE USING HADR_REMOTE_SVC DB2_HADR_1
UPDATE DB CFG FOR SAMPLE USING HADR_REMOTE_INST DB2
UPDATE DB CFG FOR SAMPLE USING HADR_SYNCMODE NEARSYNC

```

```

UPDATE DB CFG FOR SAMPLE USING HADR_TIMEOUT 120
--
-- Start HADR on standby database - JAMAICA - JAMDB2 (DB2) - JAMDB (SAMPLE)
--
DEACTIVATE DATABASE SAMPLE
START HADR ON DATABASE SAMPLE AS STANDBY
--
-- Start HADR on primary database - POLONIUM - DB2 - SAMPLE
--
DEACTIVATE DATABASE SAMPLE
START HADR ON DATABASE SAMPLE AS PRIMARY
;

```

13. Click **Finish**, and the HADR setup wizard displays a list of setup activities and starts performing each step. If you chose to start HADR later, you will not see the last step, shown in Figure 6-15.

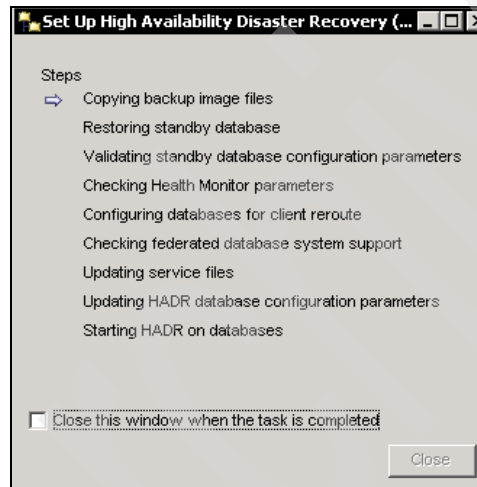


Figure 6-15 HADR: Perform HADR setup

An error message will pop up and setup is stopped if HADR encounters errors. A success message will be displayed at the end when all tasks have completed successfully (Figure 6-16).

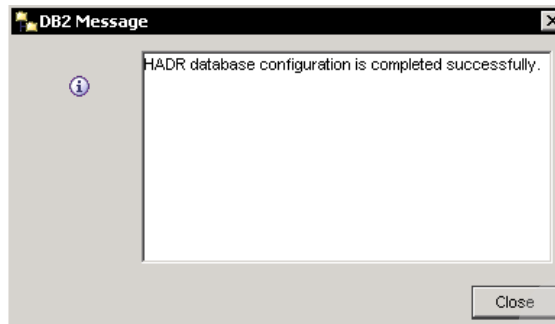


Figure 6-16 HADR setup completed successfully

Managing HADR

After the HADR is set up, you can manage it using the DB2-provided GUI tool. To invoke the HADR manager, start Control Center, right-click the primary database, select **High Availability Disaster Recover**, then **Manage**. Figure 6-17 on page 206 shows the HADR managing window, which displays the status of the HADR pair and the log position. The status is updated based on the specified refresh interval.

The HADR managing window can be used to start and stop HADR, and to perform a takeover action.

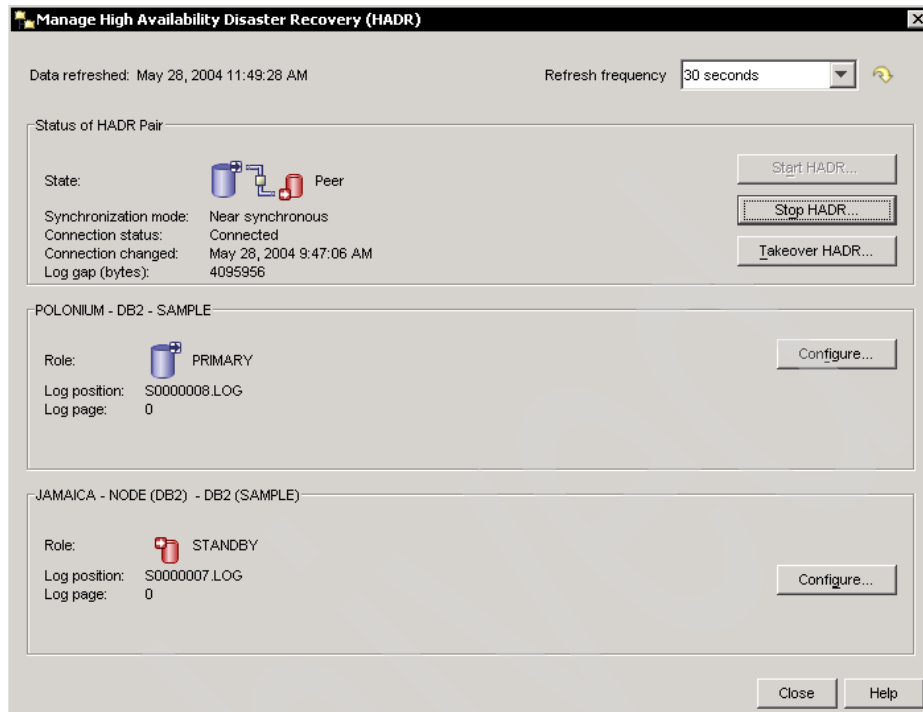


Figure 6-17 HADR: Managing HADR

You can use the DB2 registration parameter `DB2_HADR_BUF_SIZE` to set the HADR buffer size. The database configuration file shows the HADR setup information:

HADR database role	= PRIMARY
HADR local host name	(HADR_LOCAL_HOST) = POLONIUM
HADR local service name	(HADR_LOCAL_SVC) = DB2_HADR_1
HADR remote host name	(HADR_REMOTE_HOST) = JAMAICA
HADR remote service name	(HADR_REMOTE_SVC) = DB2_HADR_2
HADR instance name of remote server	(HADR_REMOTE_INST) = DB2
HADR timeout value	(HADR_TIMEOUT) = 120
HADR log write synchronization mode	(HADR_SYNCMODE) = NEARSYNC

Note: Setting these parameters neither enables HADR on a database nor indicates whether a database is primary or standby. This separation allows users to disable HADR without losing its configuration.

Monitoring

You can use the DB2 snapshot to view the log position of primary and standby databases. When a snapshot is taken on a HADR node, it includes log positions

of both primary and standby (Example 6-2). The two machines will exchange their log positions via messaging.

In case of a broken connection, remote log position and log gap will not be updated, but the old values will still be reported so that users have some idea of the state of the system when the connection was lost.

Example 6-2 HADR: monitoring using snapshot

Database Snapshot

```
Database name           = SAMPLE
Database path           = C:\DB2\NODE0000\SQL00002\
Input database alias    = SAMPLE
Database status         = Active
Catalog database partition number = 0
Catalog network node name =
Operating system running at database server= NT
Location of the database = Local
First database connect timestamp = 06/01/2004 10:25:13.423638
Last reset timestamp    =
Last backup timestamp   = 05/28/2004 18:47:27.000000
Snapshot timestamp      = 06/01/2004 11:35:44.290858
```

...

HADR Status

```
Role                   = Primary
State                  = Peer
Synchronization mode   = Nearsync
Connection status      = Connected , 06/01/2004 10:25:13.765488
Heartbeats missed      = 0
Local host             = POLONIUM
Local service          = DB2_HADR_1
Remote host            = JAMAICA
Remote service         = DB2_HADR_2
Remote instance        = DB2
timeout(seconds)       = 120
Primary log position(file, page, LSN) = S0000001.LOG, 0, 0000000000FA0060
Standby log position(file, page, LSN) = S0000001.LOG, 0, 0000000000FA0060
Log gap running average(bytes) = 0
```

Failover

HADR enables the standby database to take over as the primary database with full DB2 functionality. The failover process is ultra fast. With the TAKEOVER command, you can change the standby into primary in response to a failure of primary or switch the roles of a healthy primary-standby pair to update software

on the fly. The TAKEOVER command can be embedded in very simple heartbeat scripts.

There are two types of TAKEOVER:

► Emergency TAKEOVER (by force)

In an emergency situation such as primary system hardware failure, you can force the standby system to take over the primary role. By issuing the TAKEOVER command, the standby sends a notice asking the primary to shut itself down. The standby does *not* wait for any acknowledgement from the primary to confirm that it has received the takeover notification or that it has shut down. The standby system stops receiving logs from the primary, finishes replaying the logs it has already received, and becomes a primary.

► Normal TAKEOVER (no force)

The normal TAKEOVER can be used in situations such as software upgrade. For this type of takeover, the primary and standby switch roles. The following steps are carried out:

- a. Standby tells primary that it is taking over.
- b. Primary forces off all client connections and refuses new connections.
- c. Primary rolls back any open transactions and ships remaining log, up to the end of log, to standby.
- d. Standby replays received log, up to end of the log.
- e. Primary becomes new standby.
- f. Standby becomes new primary.

The HADR TAKEOVER command syntax is as follows:

```
>>-TAKEOVER HADR ON--+-DATABASE-+---database-alias----->
                        '-DB-----'
>-+-----+-----+-----+-----+-----+-----+----->
  '-USER--user-name--+-----+-' '-BY FORCE-'
                        '-USING--password-'
```

In addition to the TAKEOVER command, the HADR takeover action can be performed via the HADR manager GUI tool. After the HADR manager is invoked, click **Takeover HADR** (Figure 6-17 on page 206). The TAKEOVER HADR window opens (Figure 6-18 on page 209).

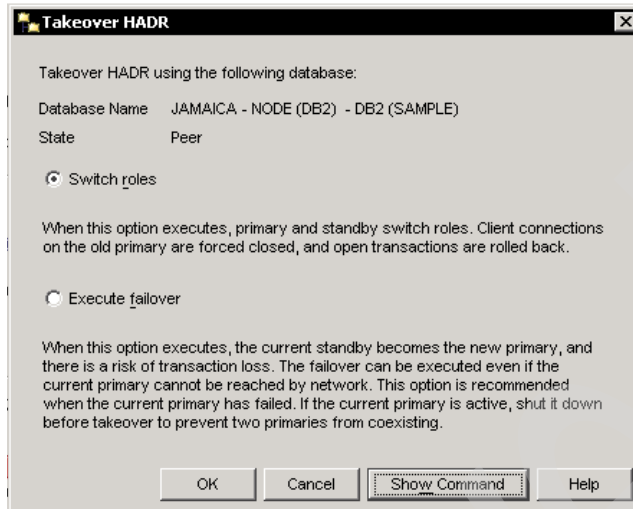


Figure 6-18 HADR: TAKEOVER

This window has two options:

► **Switch roles**

This action is a normal takeover. It issues the following TAKEOVER command:

```
TAKEOVER HADR ON DATABASE DB2 USER db2admin USING *****;
```

► **Execute failover**

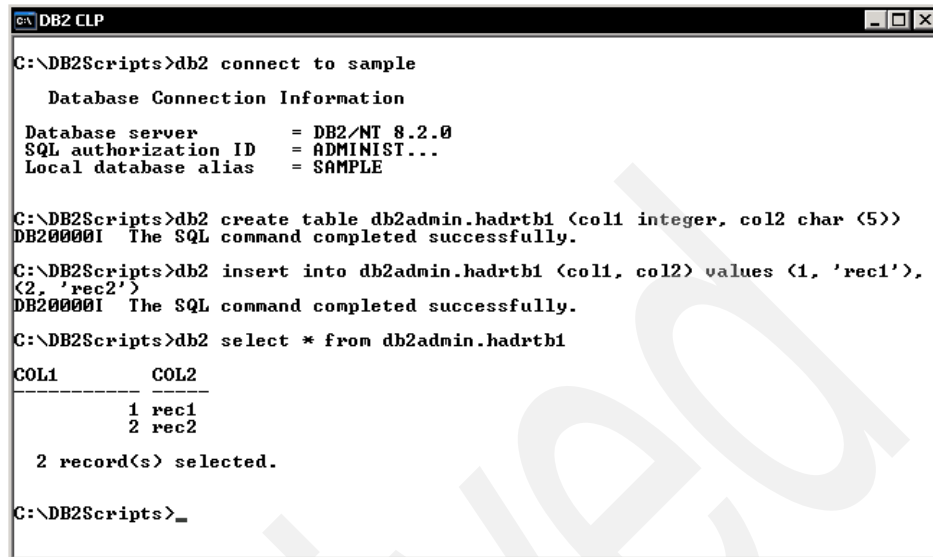
This is the emergency takeover situation. It issues the following command:

```
TAKEOVER HADR ON DATABASE DB2 USER db2admin USING ***** BY FORCE;
```

TAKEOVER example

To demonstrate the HADR function, we simulate the system crash situation by shutting down the primary system while DB2 is up and running. We performed these activities in our lab with our primary system (Polonium) and standby system (Jamaica):

1. Create a sample table db2admin.hadrtb1 in the primary system (Polonium) and insert two records (Figure 6-19 on page 210).

A screenshot of a Windows command prompt window titled "DB2 CLP". The window shows a series of commands and their outputs. The first command is "db2 connect to sample", which returns "Database Connection Information" with details: "Database server = DB2/NT 8.2.0", "SQL authorization ID = ADMINIST...", and "Local database alias = SAMPLE". The second command is "db2 create table db2admin.hadrth1 (col1 integer, col2 char (5))", which returns "DB20000I The SQL command completed successfully.". The third command is "db2 insert into db2admin.hadrth1 (col1, col2) values (1, 'rec1'), (2, 'rec2')", which also returns "DB20000I The SQL command completed successfully.". The fourth command is "db2 select * from db2admin.hadrth1", which returns a table with two columns, COL1 and COL2, and two rows of data: (1, rec1) and (2, rec2). Below the table, it says "2 record(s) selected.". The prompt ends with "C:\DB2Scripts>_".

```
C:\DB2Scripts>db2 connect to sample

Database Connection Information

Database server      = DB2/NT 8.2.0
SQL authorization ID = ADMINIST...
Local database alias = SAMPLE

C:\DB2Scripts>db2 create table db2admin.hadrth1 (col1 integer, col2 char (5))
DB20000I The SQL command completed successfully.

C:\DB2Scripts>db2 insert into db2admin.hadrth1 (col1, col2) values (1, 'rec1'),
(2, 'rec2')
DB20000I The SQL command completed successfully.

C:\DB2Scripts>db2 select * from db2admin.hadrth1

COL1      COL2
-----
      1 rec1
      2 rec2

      2 record(s) selected.

C:\DB2Scripts>_
```

Figure 6-19 HADR Example: Create table

2. Shut down the primary system Polonium.
3. Because the primary system crashed, we bring up the standby system as the primary. In standby system Jamaica, open a DB2 command line window and issue the takeover command:

```
db2 takeover hadr database sample force
```

After the takeover has completed successfully, we connect to the database and insert one record to the table adb2admin.hadrth1 (Figure 6-20 on page 211).


```

C:\Program Files\IBM\SQLLIB\BIN>db2 takeover hadr on database sample by force
DB20000I The TAKEOVER HADR ON DATABASE command completed successfully.

C:\Program Files\IBM\SQLLIB\BIN>db2 connect to sample

      Database Connection Information

Database server      = DB2/NT 8.2.0
SQL authorization ID = ADMINIST...
Local database alias = SAMPLE

C:\Program Files\IBM\SQLLIB\BIN>db2 "select * from db2admin.hadrthb1"

COL1          COL2
-----
      1 rec1
      2 rec2

      2 record(s) selected.

C:\Program Files\IBM\SQLLIB\BIN>db2 insert into db2admin.hadrthb1 values(3,'rec3')
DB20000I The SQL command completed successfully.

C:\Program Files\IBM\SQLLIB\BIN>_

```

Figure 6-20 HADR example: take over the primary system

- When the old primary Polonium is up, the database comes online as a primary. You will see two primaries in the HADR Manage wizard but it will not complete crash recovery until it is switched as a standby. The DBA must issue the **START HADR AS STANDBY** command to initiate the re-integration process.

At the Polonium system, open a DB2 command line process window. If DB2 is not already open, start it with the **db2start** command, then enter the following commands:

```

db2 deactivate database sample
db2 start hadr on database sample as standby

```

Figure 6-21 shows the execution of the commands.

```

C:\Program Files\IBM\SQLLIB\DB2>db2 deactivate database sample
SQL1496W Deactivate database is successful, but the database was not
activated.

C:\Program Files\IBM\SQLLIB\DB2>db2 start hadr on database sample as standby
DB20000I The START HADR ON DATABASE command completed successfully.

C:\Program Files\IBM\SQLLIB\DB2>_

```

Figure 6-21 HADR example: Start the old primary as standby

Now, Polonium becomes the standby and transactions will be shipped from Jamaica to Polonium (Figure 6-22 on page 212).

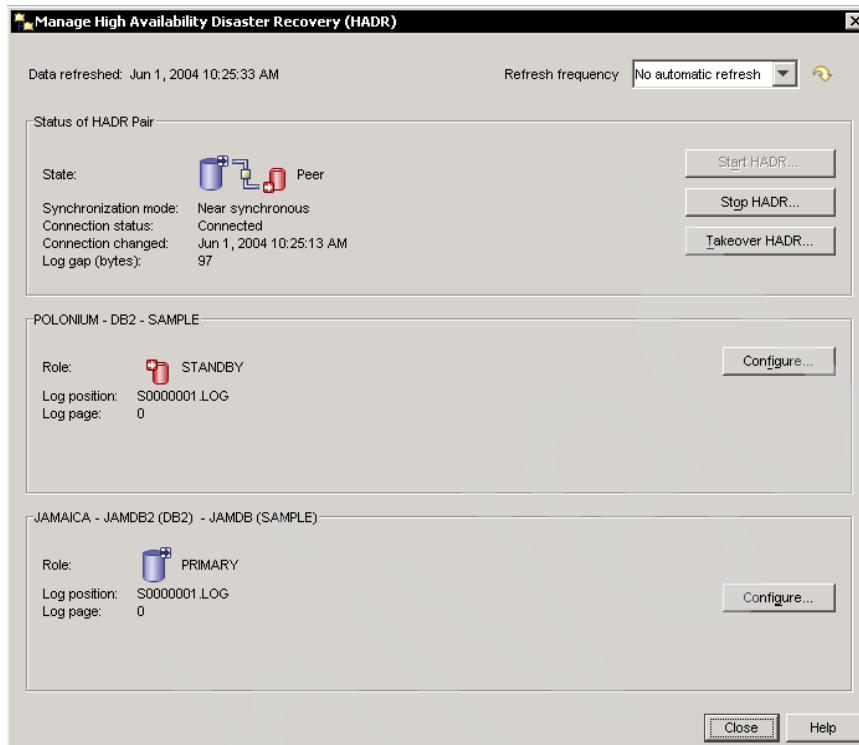


Figure 6-22 HADR example: Old primary now is the standby

4. To return the system roles to their pre-crash status, use the HADR switch role function from the wizard, or use a command.

To use the wizard, in Polonium, open **HADR Manage wizard** → **Takeover HADR** → **Switch Roles**.

The command is:

```
TAKEOVER HADR ON DATABASE SAMPLE
```

The roles are switched and the transactions that took place on Jamaica are now on Polonium (Figure 6-23 on page 213).

A screenshot of a DB2 Command Line Processor (CLP) window. The window title is "DB2 CLP". The command prompt shows the user at the C:\Program Files\IBM\SQLLIB\DB2 directory, having just entered "db2 connect to sample". The output displays "Database Connection Information" with details: Database server = DB2/NT 8.2.0, SQL authorization ID = ADMINIST..., and Local database alias = SAMPLE. The user then enters "db2 select * from db2admin.hadrth1". The output shows a table with two columns, COL1 and COL2, containing three rows of data: (1, rec1), (2, rec2), and (3, rec3). Below the table, it states "3 record(s) selected." The command prompt is ready for the next input.

```
C:\Program Files\IBM\SQLLIB\DB2>db2 connect to sample

Database Connection Information

Database server      = DB2/NT 8.2.0
SQL authorization ID = ADMINIST...
Local database alias = SAMPLE

C:\Program Files\IBM\SQLLIB\DB2>db2 select * from db2admin.hadrth1

COL1      COL2
-----
1 rec1
2 rec2
3 rec3

3 record(s) selected.

C:\Program Files\IBM\SQLLIB\DB2>
```

Figure 6-23 HADR example: Switch the roles back to the original setup

Software upgrades on the fly

In the HADR environment, you can upgrade software without system outage by utilizing the normal TAKEOVER process. The procedure is as follows:

1. Make sure that HADR is in peer state.
2. Suspend the standby by stopping HADR.
3. Upgrade the standby.
4. Start the standby again and let it catch up with the primary.
5. Issue a normal TAKEOVER to switch the role of primary and standby.
6. Suspend the new standby and upgrade it.
7. Reactivate the new standby and let it catch up with the primary.
8. To switch the primary and standby back to their original roles, issue the TAKEOVER again.

Considerations

In this section, we discuss the performance, backup/restore, and other considerations of HADR.

Restrictions

The current release of HADR has the following restrictions:

- HADR is supported on DB2 UDB Enterprise Server Edition (ESE) only. However, DB2 ESE with DPF is not supported.

- ▶ The primary and standby databases must have the same operating system version and the same version of DB2 UDB, except for a short time during a rolling upgrade.
- ▶ The DB2 UDB release on the primary and standby databases must be the same bit size (32-bit or 64-bit).
- ▶ Backup operations are not supported on the standby database.
- ▶ Reads on the standby database are not supported. Clients cannot connect to the standby database.
- ▶ Log archiving can only be performed by the current primary database.
- ▶ Load operations with the COPY NO option specified are not supported.
- ▶ Only one standby for one primary, multiple-target system is not supported.
- ▶ Some operations are not supported:
 - Data links
 - Log file backups on the standby
 - Database or table space restore
 - Non-logged operations, such as changes to database configuration parameters and to the recovery history file, are not replicated to the standby database.

Performance

Theoretically, the standby can process transactions faster than the primary because it does not perform locking, manage connections, or generate logs. On the HADR primary, the commit processing may be slightly delayed if the system is configured to use a synchronization mode of SYNC or NEARSYNC.

If the connection is lost, a transaction may experience a commit delay of about the length of the configured HADR_TIMEOUT.

The performance of DB2 functionality immediately after failover may not be exactly the same as it was just prior to the failover; it depends on how much catch-up is required. A ramp-up time should be similar to what is experienced for an application when a DB2 database is first started.

The best way to describe the network requirements is to monitor logging activity and determine the rate of logging. Whatever the rate of logging is, that would be the required network transfer rate.

Alter table log index will increase the logging requirements. If the user chooses to not log index builds, then those indexes will have to be rebuilt on the new HADR primary after a takeover.

Backup and restore

Database backup is possible on the primary, but not supported on the standby database in the current HADR release. However, database-level restores are allowed against existing HADR primary and HADR standby databases. As part of the restore operation, however, the database will be changed from its existing HADR role into a “standard” DB2 database. As a result, after a restore the database will not automatically reconnect with its HADR partner.

Table-space-level restores on the primary and standby will be disabled. The redirected restores will not be supported in any form. HADR does not replicate the recovery history file.

Others

Some points that should be noted regarding HADR:

- ▶ The HADR database must not use circular logging.
- ▶ Log archiving methods supported by DB2 are: disk, TSM, vendor, userexit, and logretain. However, log archiving is done only from the primary database.
- ▶ The HADR standby does not archive log files; it deletes them when they are not longer needed. (The primary says when.)
- ▶ Log mirroring (MIRRORLOGPATH) works as usual on both the primary and the standby.
- ▶ HADR does *not* replicate configuration parameters.
- ▶ HADR does *not* replicate the shared libraries or DLLs of the UDFs or the Stored Procedures.
- ▶ The DB2 administrator must copy the executable in the proper directories.

6.2 Automatic client reroute

Uninterrupted service is the goal of high availability. The automatic client reroute feature enables client applications to recover from a loss of communication with the server and to continue working with minimal interruption. To enable DB2 to reroute the client applications, an alternate database location must first be specified at the server. This information then will be passed to the client at database connection time. If the communication is lost, the DB2 client will use the information to route the application to the alternate database. Automatic client reroute is only supported with TCP/IP protocol.

The automatic client reroute feature could be used within the following configured environments:

- ▶ Enterprise Server Edition (ESE) with the data partitioning feature (DPF)

- ▶ Data Propagator (DPROPR)-style replication
- ▶ High availability disaster recovery (HADR)
- ▶ Microsoft Cluster Server (MSCS)

The UPDATE ALTERNATE SERVER FOR DATABASE command can be used to establish the alternate database server. The alternate host name and port number is given as part of the command. The location is stored in the system database directory file at the server. You can specify an alternate location through the command line processor by invoking an application programming interface (API), or when adding a database using the Control Center or the advanced view of the Configuration Assistant. For example, we can use the Automatic Client Reroute feature with the HADR example shown in the previous section. The alternate server will be the standby server. This is the command:

```
UPDATE ALTERNATE SERVER FOR DATABASE sample USING jamaica PORT 50000
```

In the HADR setup example, we utilize the GUI to establish the client reroute for both servers. See “Setup” on page 190.

In general, if an alternate server is specified, automatic client reroute will be enabled when a communication error (SQLCODE -30081) or a SQLCODE -1224 is detected. However, in a HADR environment, it will also be enabled if SQLCODE -1776 is returned from the HADR standby server. If communication between the client and the server is lost for any reason, the DB2 UDB client code will attempt to re-establish the connection by using the alternate server information. The DB2 UDB client will attempt to reconnect with the original server and the alternate server, alternating the attempts between the two servers. When a connection is successful, SQLCODE -30108 is returned to indicate that a database connection has been re-established following the communication failure. The host name/IP address and service name/port number are returned. The client code only returns the error for the original communications failure to the application if the re-establishment of the client communications is not possible to either the original or alternative server.

Limitations

There are some limitations with use of the automatic client reroute feature:

- ▶ Communication protocol

Automatic client reroute is only supported when the communications protocol used for connecting to the DB2 UDB server, or to the DB2 Connect server, is TCP/IP. This means that if the connection is using a protocol other than TCP/IP, the automatic client reroute feature will not be enabled. Even if DB2 UDB is set up for a loopback, TCP/IP communications protocol must be used in order to accommodate the automatic client reroute feature.

► New connections

After the connection to the alternate server location is re-established, any new connection to the same database alias will be connected to the alternate server location. If you want any new connection to be established to the original location in case the problem on the original location is fixed, there are a couple of options from which to choose:

- Take the alternate server offline and allow the connections to fail back over to the original server. (This assumes that the original server has been cataloged using the `UPDATE ALTERNATE SERVER` command such that it is set to be the alternate location for the alternate server.)
- Catalog a new database alias to be used by the new connections.
- Uncatalog the database entry and recatalog it again.

► OS support

DB2 UDB for Linux, UNIX, and Windows operating systems will support the automatic client reroute feature in both the client and the server side. Other DB2 UDB families do not currently support this feature.

► DB2 version support

The DB2 UDB server installed in the alternate host server must be the same version (but could have a higher fix pack) when compared to the DB2 UDB installed on the original host server.

► Alternate server information

Regardless of whether you have authority to update the database directory at the client machine, the alternate server information is always kept in memory. In other words, if you did not have authority to update the database directory (or because it is a read-only database directory), other applications will not be able to determine and use the alternate server, because the memory is not shared among applications.

► Authentication

The same authentication is applied to all alternate locations. This means that the client will be unable to re-establish the database connection if the alternate location has a different authentication type than the original location.

► Session resources

When there is a communication failure, all session resources such as global temporary tables, identity, sequences, cursors, server options (`SET SERVER OPTION`) for federated processing and special registers are lost. The application is responsible to re-establish the session resources in order to continue processing the work. You do not have to run any of the special register statements after the connection is re-established, because DB2 UDB will replay the special register statements that were issued before the

communication error. However, some of the special registers will not be replayed. They are:

- SET ENCRYPTPW
- SET EVENT MONITOR STATE
- SET SESSION AUTHORIZATION
- SET TRANSFORM GROUP

► z/OS

A datasharing SYSPLEX setup on DB2 UDB for z/OS may return a list of servers that are available to be connected. However, the list is only kept in memory, when a communication failure happens. At that time, the DB2 UDB client will use the list to determine the location of the appropriate alternate server for connection.

Note:

If the client is using CLI or JCC Type 2 or Type 4 drivers, after the connection is re-established, then for those SQL statements that have been prepared against the original server, they are implicitly reprepared with the new server. However, for embedded SQL routines (for example, SQC or SQX applications), they will not be reprepared.

An alternate way to do Automatic Client Reroute is to use the DNS entry to specify an alternate IP address for a DNS entry. The idea is to specify a second IP address (an alternate server location) in the DNS entry; the client would not know about an alternate server but at connect time DB2 UDB would alternate between the IP addresses for the DNS entry.

6.3 Index logging

The index logging level is controlled by the database configuration parameter LOGINDEXBUILD and the LOG INDEX BUILD option in CREATE TABLE and ALTER TABLE commands. DB2 performs different levels of logging when it creates, re-creates, or reorg index on a table based on the setting of the parameter and option. LOGINDEXBUILD can be set to ON or OFF. The LOG INDEX BUILD option has three parameters:

► NULL

Specifies that the value of the LOGINDEXBUILD database configuration parameter will be used to determine whether index build operations are to be completely logged. This is the default when the table is created.

► OFF

Specifies that any index build operations on this table will be logged minimally. This value overrides the setting of the LOGINDEXBUILD database configuration parameter.

► ON

Specifies that any index build operations on this table will be logged completely. This value overrides the setting of the LOGINDEXBUILD database configuration parameter.

The index logging provides recoverability but can have performance tradeoff. In the HADR environment, consider the following recommendations when setting index logging level:

► Using the LOGINDEXBUILD database configuration parameter

For HADR databases, set the LOGINDEXBUILD database configuration parameter to ON to ensure that complete information is logged for index creation, re-creation, and reorganization. Although this means that index builds might take longer on the primary system and that more log space is required, the indexes will be rebuilt on the standby system during HADR log replay and will be available when a failover takes place. If index builds on the primary system are not logged and a failover occurs, any invalid indexes that remain after the failover is complete will have to be rebuilt before they can be accessed. While the indexes are being re-created, they cannot be accessed by any applications.

You might choose to set the LOG INDEX BUILD table attribute to OFF on one or more tables for either of the following reasons:

- You do not have enough active log space to support logging of the index builds.
- The index data is very large and the table is not accessed often; therefore, it is acceptable for the indexes to be re-created at the end of the takeover operation. In this case, set the INDEXREC configuration parameter to RESTART. Because the table is not frequently accessed, this setting will cause the system to re-create the indexes at the end of the takeover operation instead of waiting for the first time the table is accessed after the takeover operation.

If the LOG INDEX BUILD table attribute is set to OFF on one or more tables, any index build operation on those tables might cause the indexes to be re-created any time a takeover operation occurs. Similarly, if the LOG INDEX BUILD table attribute is set to its default value of NULL, and the LOGINDEXBUILD database configuration parameter is set to OFF, any index build operation on a table might cause the indexes on that table to be

re-created any time a takeover operation occurs. You can prevent the indexes from being re-created by taking one of the following actions:

- After all invalid indexes are re-created on the new primary database, take a backup of the database and apply it to the standby database. As a result of doing this, the standby database does not have to apply the logs used for re-creating invalid indexes on the primary database, which would mark those indexes as “rebuild required” on the standby database.
- Set the LOG INDEX BUILD table attribute to ON, or set the LOG INDEX BUILD table attribute to NULL and the LOGINDEXBUILD configuration parameter to ON on the standby database to ensure that the index re-creation will be logged.

► Using the INDEXREC database configuration parameter

Set the INDEXREC database configuration parameter to RESTART (the default) on both the primary and standby databases. This will cause invalid indexes to be rebuilt after a takeover operation is complete. If any index builds have not been logged, this setting enables DB2 to check for invalid indexes and to rebuild them. This process takes place in the background, and the database will be accessible after the takeover operation has completed successfully.

If a transaction accesses a table that has invalid indexes before the indexes have been rebuilt by the background re-create index process, the invalid indexes will be rebuilt by the first transaction that accesses it.

Using DB2 with Java

IBM DB2 UDB V8.2 has strong support to Java, the world's most popular Web application development language. DB2 UDB V8.2 provides driver support for client applications and applets that are written in Java using Java Database Connectivity (JDBC) and Structured Query Language Java (SQLJ). The DB2 UDB V8.2 has improved the JDBC driver, including distributed transaction support and JDBC 3.0 compliance. Other new features increase the flexibility of iterativor use in SQLJ and enhance interactivity with host and iSeries databases. In this chapter, we discuss various ways to use JDBC to perform operations on database and SQLJ.

DB2 UDB V8.2 supports Java on various platforms. In order to use Java and DB2 on Windows, you must have:

- ▶ IBM Developer Kit and Runtime Environment for Window, Java 2 Technology Edition, Version 1.3.1. This product gets installed with DB2 if you use DB2 Setup.
 - ▶ DB2 Java Enablement, provided on DB2 UDB Version 8.2 for Windows clients and servers.
- or
- ▶ Java Development Kit (JDK) 1.3.1 for Win32 from Sun Microsystems. (Version 8.2 supports JDK 1.4.)

7.1 Java Database Connectivity (JDBC)

JDBC is a bridge between application and database that provides a standard way to communicate with Java code. DB2 has support to JDBC. Note that an application written using JDBC supports static as well as dynamic SQL.

7.1.1 Types of JDBC driver:

The JDBC drivers are classified mainly into the following types:

- ▶ JDBC Type 1

The JDBC-ODBC Bridge driver is an example of this type of driver. Type 1 generally is dependent on a native library, which limits its portability.

- ▶ JDBC Type 2

This type of driver is suitable for a Java console or a desktop application that resides on a user machine. To use JDBC connectivity, the DB2 client must be installed on the user machine. This type of driver is not suitable for creating Java applets.

- ▶ JDBC Type 3

This type of driver is especially used for creating database-enabled Java applets. This type of driver does not have to be installed on a user's machine. The applet automatically downloads the Java class files and the DB2 JDBC driver that is required for communicating with the database.

Note: The JDBC type 3 driver is deprecated for Version 8.

- ▶ JDBC Type 4

The JDBC type 4 driver was introduced in Version 8 and can be used to create both Java applications and applets. This type of driver does not require the DB2 client to be installed on the user's machine; instead the application or applet based on this type of driver requires only a db2jcc.jar file.

DB2 Universal JDBC Driver (Type 2 and Type 4)

The DB2 Universal JDBC Driver is a single driver that includes JDBC Type 2 and JDBC Type 4 behaviors, as well as SQLJ support. When an application loads this driver, a single driver instance is loaded for Type 2 and Type 4 implementations, and the application can make connections (concurrently, if needed) for both types using this single driver instance. DB2 Universal JDBC Driver Type 2 driver behavior is referred to as DB2 Universal JDBC Driver Type 2 connectivity. DB2 Universal JDBC Driver Type 4 driver behavior is referred to as DB2 Universal JDBC Driver Type 4 connectivity.

This is an entirely new driver, rather than a follow-on to any other DB2 JDBC drivers. Therefore, you can expect some differences in behavior between this driver and other drivers.

7.1.2 Writing a JDBC application

In order to use JDBC, you must install proper drivers on your machine. For example, to use a JDBC type 4 driver, your system must have db2jcc.jar and sqlj.zip files and specify them in the system's CLASSPATH. The DB2 UDB Version 8.x for Windows automatically does this when you install it on your machine. The basic steps that are required to connect to the database using JDBC are:

1. Import necessary packages that contain JDBC methods.
2. Connect to database.
3. Execute SQL statements.
4. Perform operation on result after executing the SQL statement.
5. Disconnect from the database.

Each of these steps is discussed briefly in following sections.

Importing necessary packages that contain JDBC methods

To perform an operation on a database using JDBC you must invoke JDBC methods. By importing various Java Packages, these methods can be invoked. Table 7-1 shows various Java Packages and their description.

Table 7-1 Java packages

Packages	Description
java.sql	This package has the core JDBC APIs.
javax.naming	Contains classes and interfaces for Java Naming and Directory Interface (JNDI), which is often used for implementing a DataSource.
javax.sql	Contains JDBC 2.0 standard extensions.
javax.transaction	Contains JDBC support for distributed transactions for the DB2 JDBC Type 2 driver.
com.ibm.db2.jcc	Contains the DB2-specific implementation of JDBC for the DB2 Universal JDBC Driver.
com.ibm.db2.jdbc	Contains the DB2-specific implementation of the JDBC for the DB2 JDBC Type 2 driver.

Connecting to the database

Before performing any operation on a database, the application should be able to connect to it. This can be achieved either by using the *DriverManager* interface or the *DataSource* interface.

Using the DriverManager interface

To connect to a database using the DriverManager interface, first perform some initialization: load the driver and make the actual connection.

► Loading the driver

To use DriverManager interface, the JDBC driver has to be loaded by the Java Virtual Machine classloader. The following code shows how to load and register the driver by calling the Class.forName method.

```
try
{
    // Load the DB2 Universal JDBC Driver
    Class.forName("com.ibm.db2.jcc.DB2Driver");
}
catch (ClassNotFoundException eR) // driver not found
{
    System.err.println ("Unable to load database driver -" + eR);
}
```

After the driver is loaded, the application is ready to connect to the database. This is discussed in the next section.

► Making a connection with the database

The second step in establishing a connection is to have the appropriate driver for connecting to the database. The DriverManager class maintains a list of driver classes that are currently available. When the method *getConnection* is called, the DriverManager attempts to locate a suitable driver from the list of available drivers that are loaded explicitly using the same classloader. After such a driver is found, the DriverManager attempts to use it to connect to the database.

This code shows how to get a connection using the DriverManager *getConnection* method:

```
Connection con = DriverManager.getConnection(url, "myLogin",
"myPassword");
```

Here, *url* indicates the data source to connect (for example: jdbc:db2:sample).

The *getConnection* method can be written in several ways depending on the driver type and passing arguments, as shown:

- Using URL with user ID and password

The URL is created using the machine name, port, user ID, and password:

```
// Set URL/ID/Password for data source
String url =
"jdbc:db2://mymachine.ibm.com:1521:sample;user=db2adm;password=db2adm;";
getConnection(String url);
```

- Using the properties object

The properties object can be used to set various connection properties and then pass it while creating the connection:

```
// Create Properties object
Properties properties = new Properties();
// Set user ID for connection
properties.put("user", "db2adm");
// Set password for connection
properties.put("password", "db2adm");
// Set URL for data source
String url = "jdbc:db2://mymachine.ibm.com:1521:sample";
// Create connection
Connection con = DriverManager.getConnection(url, properties);
getConnection(String url, java.util.Properties info);
```

Using the DataSource interface

The application can be made portable among data sources using the *DataSource* interface. Using *DriverManager* to connect to the data source reduces portability because the application must identify a specific JDBC driver class name and driver URL. JDBC, starting with Version 2.0, provides the *DataSource* interface for connecting to the data source.

Table 7-2 shows various *DataSource* implementations provided by the DB2 JDBC Driver.

Table 7-2 DataSource provided by DB2 JDBC driver

DataSource implementations	Features	Driver type
com.ibm.db2.jcc.DB2SimpleDataSource	Does not support connection pooling.	- Universal Type 2 Connectivity - Universal Type 4 Connectivity

DataSource implementations	Features	Driver type
COM.ibm.db2.jdbc.DB2DataSource	<ul style="list-style-type: none"> - Built-in support for connection pooling. - Connection pooling is handled internally and is transparent to the application. 	DB2 JDBC Type 2 driver
COM.ibm.db2.jdbc.DB2XADDataSource	<ul style="list-style-type: none"> - Does not have built-in support for distributed transactions and connection pooling. - Distributed transactions and connection pooling managed in code or by using a tool such as WebSphere Application Server. 	DB2 JDBC Type 2 driver

When you create and deploy the DataSource object, perform following tasks:

- Import the necessary packages.

For using the DataSource object, JNDI naming service packages are required with other packages.

```
import java.sql.*;           // JDBC base
import javax.naming.*;       // JNDI Naming Services
import javax.sql.*;          // JDBC 2.0 standard extension APIs
import com.ibm.db2.jcc.*;    // DB2 implementation of JDBC 2.0
```

- Create an instance of the appropriate *DataSource* implementation:

```
DB2SimpleDataSource db2ds = new com.ibm.db2.jcc.DB2SimpleDataSource();
```

- Set the properties of the DataSource object:

```
db2ds.setDatabaseName("sampleloc1");
db2ds.setDescription("Sample Database");
db2ds.setUser("db2admin");
db2ds.setPassword("db2admin");
```

- Register the object with the Java Naming and Directory Interface (JNDI) naming service:

```
Context ctx=new InitialContext();
Ctx.bind("jdbc/sample",db2ds);
```


Executing the SQL Statements

In JDBC, the *Statement* object is used to execute queries against a database. The *Statement* object is responsible for sending the SQL statement and returning the result from the query. The result can be a data set or simple integer indicating rows affected by performing the SQL operation.

The *Statement* object supports two main methods: an *executeUpdate* method that is normally used for operations that do not generate a response, and an *executeQuery* method that returns data:

- The *Statement.executeUpdate* method is used to create and modify DB2 objects.

The *executeUpdate* method is used for INSERT, UPDATE, and DELETE SQL operations. This method takes an SQL statement as an argument and returns a number of rows updated after executing the SQL.

To use the *executeUpdate* method, first create the *Statement* using the *createStatement* method of *Connection* object, and then execute the SQL as shown:

```
// create a Statement object using already created connection cnn
stmt = cnn.createStatement();
// Perform the update
int rowUpd = stmt.executeUpdate("UPDATE EMPLOYEE SET PHONENO='1232467890'
WHERE EMPNO='000020'");
//close the statement and connection object
stmt.close();
cnn.close();
//print the total rows updated
System.out.println ("Total rows updated – “ + rowUpd);
```

- The *Statement.executeQuery* method is used to retrieve data from DB2 tables.

The *executeQuery* method is used with SELECT SQL statements. This method takes a SELECT SQL statement as an argument and returns results from the query. This code snippet shows how to use the *executeQuery* method to get all Employee IDs:

```
// create a Statement object using already created connection cnn
stmt = con.createStatement();
// Get the result from the query
ResultSet rs = stmt.executeQuery("SELECT EMPNO FROM EMPLOYEE");
//populate the result
//close the statement and connection object
stmt.close();
cnn.close();
```

The *PreparedStatement* and *CallableStatement* interfaces are inherited from the *Statement* interface.

► **PreparedStatement**

The *PreparedStatement* interface supports SQL statements with or without input parameters but returns no output parameter. For example, the *PreparedStatement* interface can be used to execute a stored procedure that has input parameters and that returns no output parameters or resultsets.

- The *PreparedStatement.executeUpdate* method is used to create and modify DB2 objects.

First, prepare your statement before performing any operation. This helps to create dynamic SQL that takes a parameter before executing the SQL:

```
// create a PreparedStatement object
PreparedStatement pstmt = cnn.prepareStatement( "UPDATE EMPLOYEE SET
PHONENO=? WHERE EMPNO=?");
// Assign value to parameters
pstmt.setString(1,"12345");
pstmt.setString(2,"000020");
// Perform the update
int rowUpdated = pstmt.executeUpdate();
//close the PreparedStatement
pstmt.close();
```

Note that the *PreparedStatement.executeUpdate* can be used to execute SQL statements directly without taking any parameters as shown:

```
PreparedStatement pstmt = con.prepareStatement("UPDATE EMPLOYEE SET
PHONENO='12345' WHERE EMPNO=' 000020'");
```

- The *PreparedStatement.executeQuery* method is used to retrieve data from DB2.

The *PreparedStatement.executeQuery* method can be used to execute a SQL statement that takes one or more input parameters and returns results in a *ResultSet* object.

```
// Create a PreparedStatement object
PreparedStatement pstmt = cnn.prepareStatement( "SELECT * FROM EMPLOYEE
WHERE EMPNO=?");
// Assign value to input parameter
pstmt.setString(1,"000020");
// Get the result table from the query
ResultSet rs = pstmt.executeQuery();
/* POPULATE RESULT */
// Close the ResultSet
rs.close();
// Close the PreparedStatement
pstmt.close();
```

► CallableStatement

The *CallableStatement* interface is used to execute SQL stored procedures that take input parameters and return one or multiple resultsets.

The *ResultSet* interface can be used to access the results that the query generates.

The *CallableStatement* interface has three methods:

- The *CallableStatement.executeUpdate* method is used where the stored procedure does not return resultsets.
- The *CallableStatement.executeQuery* method is used where the stored procedure returns one resultset.
- The *CallableStatement.execute* method is used where the stored procedure returns multiple resultsets.

In all three methods, the basic steps are the same as discussed in the previous section, but you must register input or output parameters as shown:

```
CallableStatement cstmt;  
cstmt = con.prepareCall("CALL EMPINFO(?,?,?)");  
// Set input parameter  
cstmt.setString (1, "000020");  
// Register output parameters  
cstmt.registerOutParameter (2, Types.STRING); cstmt.registerOutParameter  
(3, Types.LONG);  
//Call the Stored Procedure  
cstmt.executeUpdate();
```

Performing operations on a result after executing the SQL statement

The previous section showed how to get a result using the *executeQuery* statement. The *ResultSet* object maintains the cursor pointing to its current row of data. The *next* method of *ResultSet* object moves the cursor to the next row. It returns *false* when there are no more rows in the *ResultSet* object. This code snippet shows how to populate the result using the *ResultSet* object.

```
// populate data in ResultSet  
while (rs.next())  
{  
    empNo = rs.getString(1);  
    System.out.println("Employee ID = " + empNo);  
}  
// Close the ResultSet  
rs.close();
```

Disconnecting from the database

The last step includes closing the *Connection* object, which can be performed using the *close* method:

```
cnn.close();
```

Example 7-1 shows a simple JDBC application that displays an Employee number from the Employee database.

Example 7-1 JDBC application example

```
import java.sql.*;
public class simpleJDBC
{
    public static void main(String[] args)
    {
        String empNo;
        Connection cnn;
        Statement stmt;
        ResultSet rs;
        try
        {
            // Load the DB2 Universal JDBC Driver
            Class.forName("com.ibm.db2.jcc.DB2Driver");
            // Create the connection using the DB2 Universal JDBC Driver
            cnn = DriverManager.getConnection
('jdbc:db2://db2host:5001/sample','userid','password');
            // Create the Statement
            stmt = cnn.createStatement();
            // Execute a query and generate a ResultSet instance
            rs = stmt.executeQuery("SELECT EMPNO FROM EMPLOYEE");
            // Print all of the employee numbers to standard output device
            while (rs.next()) {
                empNo = rs.getString(1);
                System.out.println("Employee number = " + empNo);
            }
            // Close the ResultSet
            rs.close();
            // Close the Statement
            stmt.close();
            // Close the connection
            cnn.close();
        }
        catch (ClassNotFoundException e)
        {
            System.err.println("Could not load JDBC driver");
            System.out.println("Exception: " + e);
        }
        catch(SQLException ex)
```

```
    {  
        System.out.println("SQLException information " + ex);  
    }  
}  
}
```

7.2 SQL Java (SQLJ)

SQL Java (SQLJ) is a programming interface that enables SQL statements to be expressed at a high level in a Java program or a standard way to embed SQL statements in the Java programs. By writing SQL-like statements within Java, an SQLJ preprocessor generates static SQL, providing improved performance over dynamic SQL. SQLJ also generates iterator Java classes that enable developers to navigate through query results. By embedding SQL in Java programs, SQLJ enables programming tools to perform the following tasks:

- ▶ Compile time analysis of SQL in Java
- ▶ Syntax checking of SQL statements
- ▶ Type checking: determines whether the data exchanged between Java and SQL have compatible types and proper type conversions
- ▶ Schema checking: determines whether the SQL constructs are well-formed and valid in the database schema where they will be executed

The SQLJ performs these tasks by using SQLJ specifications, which are in several parts. For example:

- ▶ SQLJ: Embedded SQL
Has specifications for embedding SQL statements in Java methods.
- ▶ SQLJ: SQL Routines
Has specifications for calling Java static methods as SQL stored procedures and user-defined functions.
- ▶ SQLJ: SQL Types
Has Java classes as SQL user-defined data types.

Note that the SQL statements that are embedded in Java are static, as they are textually evident in the Java program and therefore “precompiled” when the Java program is compiled. The SQLJ precompiler translates the SQLJ clauses written in the file or SQLJ clauses into their equivalent JDBC calls.

Advantages of SQLJ over JDBC

SQLJ offers some distinct advantages over JDBC:

- ▶ Easy for debugging, because SQLJ programs require fewer lines of code than JDBC programs.
- ▶ Performs strong type checking of query at compile time.
- ▶ Performs syntactic and semantic checking on the code using database connections at compile time; therefore, easy to check program for errors at compile-time rather than at run-time.

However, JDBC is the only option if you want to execute dynamic SQL. For example, a program that builds queries on-the-fly or has an interactive component requires JDBC.

Table 7-3 shows differences between SQLJ and JDBC.

Table 7-3 Differences between SQLJ and JDBC

	SQLJ	JDBC
SQL Statements	Static	Dynamic
Pre-compile	Yes	No
Type checking	Strong (performed during development)	Weak (performed during run-time)
Schema checking	Strong (performed during development)	Weak (performed during run-time)
Standard	ANSI	SUN
Performance	Faster	Slower
Security	Strong	Average
SQLJ and JDBC interoperability	Yes	N/A

7.2.1 Writing SQLJ applications

Writing an SQLJ application has much in common with writing an SQL application in any other language. A static SQL statement in an SQLJ application, known as an SQLJ clause, begins with the token `#sql` and ends with a semicolon. An example of a SQL statement in SQLJ clauses enclosed in braces is:

```
#sql [ctxt] { UPDATE EMPLOYEE SET PHONENO=:x WHERE EMPNO=:y };
```

The SQLJ markers help the compiler identify which statements are SQLJ clauses and which are Java statements. The above statement can be written within Java code as:

```
void updateEmp (String x, String y) throws SQLException
{
    #sql [ctxt] {UPDATE EMPLOYEE SET PHONENO=:x WHERE EMPNO=:y};
}
```

The main steps required to write an SQLJ application are:

1. Import the necessary Java packages that contain SQLJ and JDBC methods.
2. Declare variables for sending data to or retrieving data from DB2 tables.
3. Connect to the database.
4. Execute SQL statements.
5. Disconnect from the database.

These steps are described in the following section.

Importing the necessary Java packages

The Java Packages that are required for writing an SQLJ application are:

- ▶ `sqlj.runtime`
- ▶ `java.sql`
- ▶ `com.ibm.db2.jcc`
- ▶ `javax.naming`
- ▶ `javax.sql`

Except for `sqlj.runtime`, we discussed all of the packages in the previous section. The `sqlj.runtime` package contains SQLJ runtime API. The `com.ibm.db2.jcc` package contains the DB2-specific implementation of JDBC and SQLJ.

Declare variables

In an SQLJ application, host expressions are used to pass the data. A host expression can be a simple Java identifier or a complex expression. Every host expression must start with a colon when it is used in an SQL statement. Host expressions are case-sensitive.

The following example demonstrates how *empName* is used to get the last name of an employee:

```
String empName;
#sql [ctxt]
{SELECT LASTNAME INTO :empName FROM EMPLOYEE WHERE EMPNO='000020'};
```

Connect to the database

In an SQLJ application, there are various ways to connect to the database either explicitly or implicitly:

- ▶ Explicitly by using the JDBC DriverManager interface
- ▶ Explicitly by using the JDBC DataSource interface
- ▶ Implicitly by creating the connection

Explicitly by using the JDBC DriverManager interface

The steps to create a connection using the JDBC DriverManager interface:

1. Create the SQLJ connection context.

The SQLJ application requires a connection context for performing operations on the database. This can be done using:

```
#sql context context-class-name;
```

2. Load the JDBC driver by invoking the *Class.forName* method:

This was discussed in 7.1.2, “Writing a JDBC application” on page 223. The *Class.forName* can be called depending on driver type, as shown.

For the DB2 Universal JDBC Driver:

```
Class.forName("com.ibm.db2.jcc.DB2Driver");
```

For the DB2 JDBC Type 2 Driver:

```
Class.forName("COM.ibm.db2.jdbc.app.DB2Driver");
```

3. Invoke the constructor for the connection context class that you created in step 1.

Doing this creates a connection context object that you specify in each SQL statement. You can execute it at the associated datasource.

```
Ctx myConnCtx= new Ctx("jdbc:db2://myServer:5021/dbs",  
userid,password,false);
```

You can also create a connection context using the *getConnection* method:

```
//get the connection using getConnection method  
Connection jdbccon=  
DriverManager.getConnection("jdbc:db2://myServer:5021/dbs",  
userid,password);  
//create context  
Ctx myConnCtx=new Ctx(jdbccon);
```

Explicitly by using the JDBC DataSource interface

This way, the connection object is created using the JDBC *DataSource* interface:

```
// create connection context class CtxSqlj  
#sql context CtxSqlj;  
// create context object
```



```
Context ctx=new InitialContext();
//Create datasource object and get the connection
DataSource ds=(DataSource)ctx.lookup("jdbc/sample");
Connection con=ds.getConnection();
CtxSqlj myConnCtx=new CtxSqlj(con);
```

Implicitly by creating the connection

In this technique, the default connection is used to connect to the datasource. Use the default connection by specifying SQL statements without a connection context object. There is no need to load the JDBC driver unless you explicitly use JDBC interfaces in the program.

To create a default connection context, SQLJ does a JNDI lookup for `jdbc/defaultDataSource`. If nothing is registered, a null context exception is issued when SQLJ attempts to access the context.

The following example shows how to use this technique:

```
// Use default connection for executing a SQL statement
#sql {SELECT LASTNAME INTO :empname FROM EMPLOYEE WHERE EMPNO='000020'};
```

Execute SQL statements

The created context object is used to execute SQL statements as shown:

```
#sql [myConnCtx] {SELECT LASTNAME INTO :empname FROM EMPLOYEE WHERE
EMPNO='000010'};
```

Disconnect from the database

The `close` method of context object is used to close the connection as shown:

```
myConnCtx.close();
```

Example 7-2 shows all of the tasks that we have discussed.

Example 7-2 Using all tasks

```
//import necessary packages
import sqlj.runtime.*;
import java.sql.*;
//create SQL context and iterator
#sql context mySqljCtx;
#sql iterator sqljNameIter (String LASTNAME);
public class demoSQLJ
{
    public static void main(String args[])
    {
        try
        {
            Class.forName("com.ibm.db2.jcc.DB2Driver");
```

```

Connection cnn = DriverManager.getConnection(jdbc:db2:xx);

ctx = new mySqljCtx(cnn);

sqljNameIter iter;
int count=0;
// Create result table of the SELECT
#sql [ctx] iter = {SELECT LASTNAME FROM EMPLOYEE};
while (iter.next())
{
    //Print last name
    System.out.println(iter.LASTNAME());
}
}
catch( Exception e )
{
    System.out.println("JAVA exception    = " + e);
    e.printStackTrace();
}
catch( SQLException e )
{
    System.out.println("SQL exception    = " + e);

}
// Close the iterator
iter.close();
ctx.close();
}

```

Using DB2 with .NET framework

DB2 UDB Version 8.2 is a significant release for .NET application developers with significant new features. The DB2 .NET Data Provider has much-improved performance and now supports DB2 UDB server on iSeries. DB2 UDB servers also support stored procedures written in Microsoft CLR (Common Language Runtime) compliant languages such as Visual Basic, .NET, and C#. DB2 tools add-in to Microsoft Visual Studio .NET are further augmented with schema operation capabilities, wizards for developing and deploying CLR Stored Procedures as well as WORF (Web Services Object Runtime Framework) web services.

In this chapter, we discuss various features available for .NET application developers:

- ▶ Overview of ADO.NET
- ▶ Various Providers for DB2
- ▶ Application development using DB2 .NET Provider
- ▶ IBM DB2 Development Add-In overview for Visual Studio .NET
- ▶ Writing DB2 Stored Procedure using CLR

8.1 An overview of ADO.NET

ActiveX Data Object for .NET (ADO.NET) is a suite of data-access technologies included in the .NET Framework class libraries. ADO.NET helps applications connect to a database and has been designed to be the single data access model used by all server processes and applications running on the Microsoft platform. Figure 8-1 shows the relationship between various applications based on the .NET framework, ADO.NET, and a database.

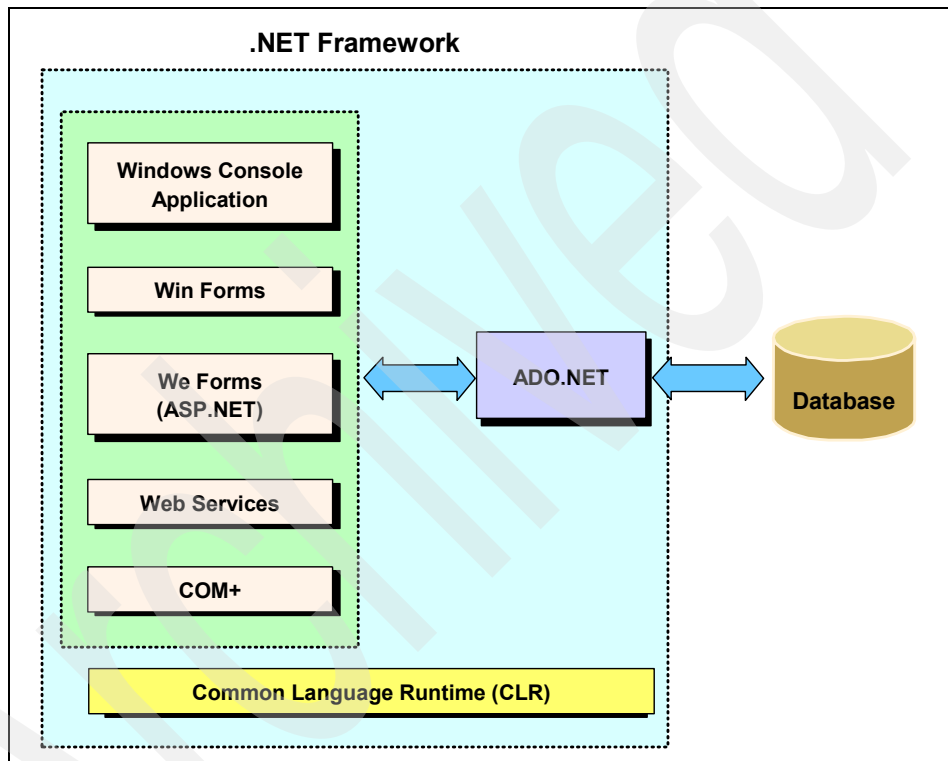


Figure 8-1 ADO.NET

8.2 ADO.NET architecture

The ADO.NET suite of database access technologies, which includes various classes, methods, and attributes to interface with a database, consists of objects such as Connection, Command, DataAdapter, and DataReader to perform operations on a database. Figure 8-2 shows the various components of ADO.NET.

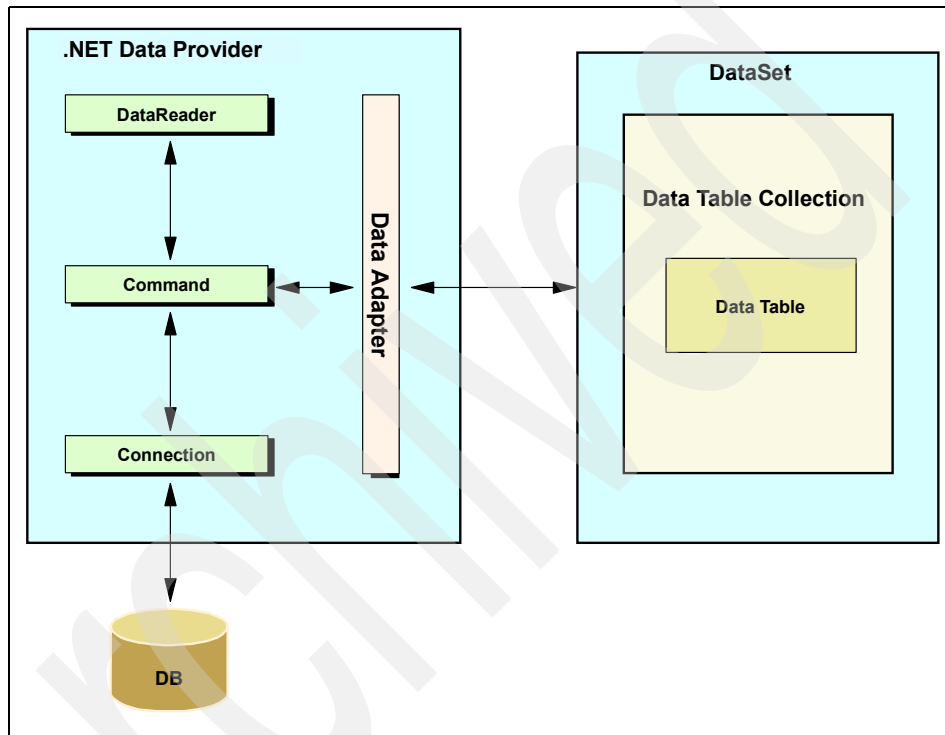


Figure 8-2 ADO.NET architecture

8.2.1 Connection

The Connection object in ADO.NET is used to connect to a database and can be used to control the transactions. The Connection objects are different for different providers but they serve one purpose. For example, DB2 .NET Provider has *DB2Connection*, and OLE DB provider uses *OleDbConnection* object to connect to the database.

Connections can be opened in two ways:

- ▶ Explicitly by calling the *Open* method on the connection
- ▶ Implicitly when using a DataAdapter

Some important interfaces of the Connection object are described in Table 8-1 and Table 8-2.

Table 8-1 Connection object properties

Public property	Description
ConnectionString	This is required for making a connection with a database. It requires the database source name and other parameters. For example, for DB2 .NET Provider you can specify a ConnectionString property with Connection cnn as: <code>cnn.ConnectionString ="Database=SAMPLE";</code>

Table 8-2 Connection object methods

Public method	Description
Open	Opens a database connection that is specified in a ConnectionString property, such as: <code>cnn.Open();</code> The Connection object throws an exception if it fails to open a database connection.
Close	Used to close the database connection. For example: <code>cnn.Close();</code>
CreateCommand	Returns a Command object associated with the connection, which can be used to perform SQL operations on a database. For example: <code>DB2Command cmd = cnn.CreateCommand();</code>
BeginTransaction	Begins a transaction at the local level. Transactions are discussed in 8.5.3, "Controlling transaction" on page 263.

8.2.2 Command

The *Command* object is used to execute SQL statements or Stored Procedures on a database (data source). Example 8-1 shows how to create a command object and assign connection.

Example 8-1 Command object

```
//Create command object
DB2Command cmd = new DB2Command();

//assign connection for where to perform operation
cmd.Connection = cnn;

//assign query
cmd.CommandText = "SELECT * FROM ORG";

//open command
cnn.Open();

//read the data by executing CommandText against the data source
DB2DataReader reader = cmd.ExecuteReader();
```

Some important interfaces of the *Command* object are described in Table 8-3 and Table 8-4.

Table 8-3 Command object properties

Public property	Description
CommandType	Describes whether the Command object will execute an SQL statement or Stored Procedure.
CommandText	Used to set or get an SQL statement or Stored Procedure to execute at a database. The default value of the <i>CommandType</i> property is <i>CommandType.Text</i> . For example: cmd.CommandText = "select * from STAFF"; cmd.CommandType = CommandType.Text;

Table 8-4 Command object methods

Public method	Description
CreateParameter	Used for handling parameters. The parameter could be input-only, output-only, bidirectional, or a stored procedure return value parameter.

Public method	Description
ExecuteNonQuery	Can be used to perform UPDATE, INSERT, or DELETE SQL operations on a database. This method returns the number of rows affected after executing the SQL statement. For example: <code>cmd.Connection.Open();</code> <code>cmd.ExecuteNonQuery();</code>
ExecuteReader	Used for reading results by executing a SELECT statement on a database.
ExecuteScalar	Used for retrieving a single value from a database. This reduces overhead required for the ExecuteReader method. For example: <code>cmd.CommandText = "select count(*) from STAFF";</code> <code>Int32 count = (int32) cmd.ExecuteScalar();</code>

8.2.3 DataReader

The *DataReader* class is used for populating the data on a client side by reading a forward-only stream of rows from a database.

The code snippet in Example 8-3 shows how to create a *DB2DataReader* object and populate it to display the result on the Windows console.

Example 8-2 Create DB2DataReader object

```
DB2DataReader reader;
reader = cmd.ExecuteReader();
// Populate result
while (reader.Read())
{
    Console.WriteLine(reader.GetString(1) + ", " + reader.GetString(2));
}
// close reader
reader.Close();
```

Table 8-5 and Table 8-6 on page 243 describe important properties and methods of the *DataReader* class.

Table 8-5 DataReader class properties

Public property	Description
FieldCount	Returns the number of columns in the current row.
HasRows	Indicates whether DataReader has one or more rows.

Table 8-6 *DataReader methods*

Public method	Description
Read	Used to read records one by one. This method automatically advances the cursor to the next record and returns true or false, indicating whether DataReader has any rows to read.
Close	Closes DataReader.
Getxxxx	Used to get data of type xxxx. For example, the <i>GetBoolean</i> method is used to get Boolean records, and the <i>GetChar</i> method is used to get <i>char</i> -type data.

8.2.4 DataAdapter

DataAdapter is used between *DataSet* and a database such as DB2 UDB. *DataAdapter* performs SELECT, UPDATE, DELETE, and INSERT operations for loading or unloading the data. Example 8-3 shows how to create *Db2DataAdapter*.

Example 8-3 *Create Db2DataAdapter*

```
Db2Connection cnn = new Db2Connection("Database=Sample");
DataSet ds = new DataSet();
Db2DataAdapter adpt = new Db2DataAdapter();
adpt.SelectCommand = new Db2Command("select * from staff", cnn);
adpt.Fill(ds);
//--- Code to perform further Operations on a dataset---
```

Table 8-7 and Table 8-8 on page 244 show some important *DataAdapter* public properties and methods.

Table 8-7 *DataAdapter properties*

Public property	Description
DeleteCommand	Gets or sets a SQL statement or Stored Procedure to delete records from the data set. For example: <pre>DB2DataAdapter adpt = new DB2DataAdapter (); DB2Command cmd; cmd = new DB2Command("DELETE FROM Customers WHERE CustomerID = '", cnn); adpt.DeleteCommand = cmd;</pre>
InsertCommand	Inserts new records into a database using SQL or Stored Procedure.
SelectCommand	Selects records in a database using SQL or Stored Procedure.

Public property	Description
UpdateCommand	Used to update records in a database using SQL or Stored Procedure.

Table 8-8 *DataAdapter methods*

Public method	Description
Fill	Used to fill records in DataSet. For example: adpt.Fill(dataset); //fills dataset
Update	Used to update rows in DataSet and a database by performing INSERT, DELETE, or UPDATE operations.

8.2.5 DataSet

The *DataSet* class represents a “in-memory cache of data” retrieved from a database. DataSet is used to improve overall performance of the application as it minimizes server trips to a database. Example 8-4 is sample code using DataSet.

Example 8-4 *DataSet class*

```
Db2Connection cnn = new Db2Connection("Database=Sample");
DataSet ds = new DataSet();
Db2DataAdapter adpt = new Db2DataAdapter();
adpt.SelectCommand = new Db2Command("select * from staff", cnn);
adpt.Fill(ds);
//---Code to perform further Operations on dataset---
```

Table 8-9 and Table 8-10 show some important DataSet public properties and methods.

Table 8-9 *DataSet properties*

Public property	Description
DataSetName	Gets or sets DataSet name.

Table 8-10 *DataSet methods*

Public method	Description
AcceptChanges	Used to commit changes made to the DataSet.
Clear	Clears the contents in DataSet.
GetXML	Gets XML representation of data in DataSet.

Public method	Description
ReadXml	Reads XML schema and XML into DataSet.
WriteXml	Writes XML schema and XML into DataSet.

More about using DataAdapter and DataSet can be found in 8.5.1, “Using DataAdapter and DataSet (Disconnected model)” on page 258.

8.3 Connecting to the database

In every database application, the first step is to connect to the database. To do this, the application needs an interface. A data provider is an interface that responds to various calls to provide information to a data consumer, which is an application or component that uses the interfaces of a provider to access the database. A data consumer can be any application that requires access to data.

Figure 8-3 on page 245 shows the relationship between Data Provider, Data Consumer, and a database.

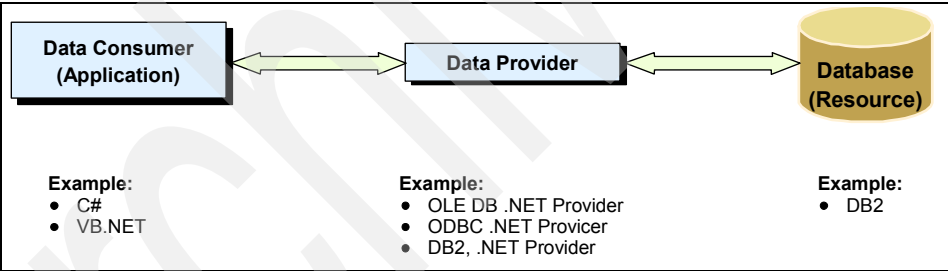


Figure 8-3 Relation between Data Consumer, Data Provider, and database

8.4 Data providers for DB2

There are several ways to connect to DB2 in .NET, depending on the type of the provider. Figure 8-4 shows various providers and available interfaces.

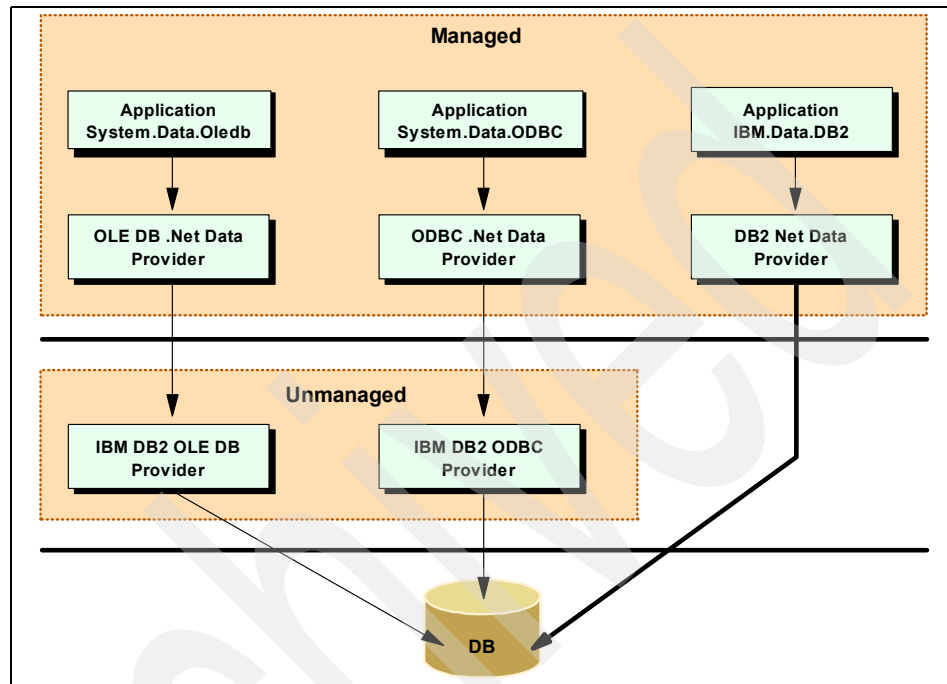


Figure 8-4 Providers in .NET to connect DB2

As shown in Figure 8-4, a .NET application can be connected to the DB2 database by using various providers. These providers are categorized as:

- ▶ OLE DB .NET Provider
- ▶ ODBC .NET Provider
- ▶ DB2 .NET Provider

8.4.1 OLE DB .NET Data Provider

OLE DB is Microsoft's strategic low-level application program interface (API) for accessing different data sources. OLE DB interfaces provide consistent access to SQL and non-SQL data sources.

The OLE DB .NET provider is a managed provider that can be used in any .NET-supported language (such as VB.NET or C#), but OLE DB provider is an unmanaged provider and can be used with unmanaged code (VB, VC++).

Note that the OLE DB .NET Data Provider uses the IBM DB2 OLE DB Driver to access DB2 databases, and the IBM OLE DB Provider for DB2 (whose provider name is IBMDADB2) enables OLE DB consumers to access data on the DB2 Universal Database server. If DB2 Connect is installed, these OLE DB consumers can also access data on a host DBMS such as DB2 for MVS™, DB2 for VM/VSE, or SQL/400®.

Using OLE DB .NET Provider

Here is an example to understand more about how to use DB2 .NET Provider. First, create a new Windows Console application in C# using Visual Studio .NET to open a new default class file. Add the proper name spaces required for programming as shown:

```
using System;
using System.Data ;
using System.Data.OleDb;
```

In this example, System.Data.OleDb is a namespace that has all required references for Ole Db interfacing. The next step shows declaring a connection, command, and data reader instances for Ole Db and then opens a new connection as shown:

```
OleDbConnection cnn =null;
OleDbCommand cmd=null;
OleDbDataReader reader=null;
try
{
    cnn= new OleDbConnection ();
```

In order to use OLE DB .NET Provider, we create a connection by specifying a connection string that describes how to connect to a database. For example:

```
cnn.ConnectionString="Provider=IBMDADB2.1;UserID=db2admin;Password=db2admin
;Data Source=SAMPLE";
```

Here, the Provider specifies which provider is used for making the connection, which is IBMDADB2.1.

In the following steps, a command object is created and used for executing an SQL statement specified in the CommandText property, and the result is populated using the while loop, as shown in Example 8-5.

Example 8-5 Using CommandText property

```
cmd = new OleDbCommand();
cmd.Connection =cnn;

cmd.CommandText = "SELECT * FROM ITSODBA.ORG";
cmd.CommandTimeout=20;
```

```

cnn.Open();
reader = cmd.ExecuteReader();
while (reader.Read()==true)
{
    Console.WriteLine (reader.GetString(1)); //Display Dept Name
}

```

Example 8-6 shows the entire code.

Example 8-6 Using OLE DB .NET provider

```

using System;
using System.Data ;
using System.Data.OleDb;

namespace oleDB
{
    class testOLEDB
    {
        [STAThread]
        static void Main(string[] args)
        {
            OleDbConnection cnn =null;
            OleDbCommand cmd=null;
            OleDbDataReader reader=null;
            try
            {
                cnn= new OleDbConnection ();
                cnn.ConnectionString ="Provider=IBMDADB2.1;User
ID=db2admin;Password=db2admin;Data Source=SAMPLE";
                cmd = new OleDbCommand();
                cmd.Connection =cnn;

                cmd.CommandText = "SELECT * FROM ADMINISTRATOR.ORG";
                cmd.CommandTimeout=20;
                cnn.Open();
                reader = cmd.ExecuteReader();
                while (reader.Read()==true)
                {
                    Console.WriteLine (reader.GetString(1)); //Display Dept Name
                }

                if (reader!=null)
                    reader.Close();
                if (cnn !=null)
                    cnn.Close();
            }
        }
    }
}

```

```

        catch (OleDbException oe)
        {
            Console.WriteLine (oe.Message);
        }
        Console.ReadLine ();
    }
}

```

Build and run the program, which displays the Department names on the Windows console.

8.4.2 ODBC .NET Data Provider

Open DataBase Connectivity (ODBC) is an open-standard API for accessing a database. This API is independent of DBMS or the operating system. With ODBC, applications can access databases from multiple database vendors. For example, an application can use ODBC to access a DB2 UDB, Microsoft SQL server, or an Oracle database server. The ODBC API is based on the CLI specifications from X/Open and ISO/IEC.

The ODBC .NET Data Provider makes ODBC calls to a DB2 data source using the DB2 CLI Driver. Therefore, the connection string keywords supported by the ODBC .NET Data Provider are same as those supported by the DB2 CLI driver.

Using ODBC .NET Provider

To use ODBC provider, you must create a Data Source Name (DSN) using the Windows built-in tool for ODBC Data Source Administrator.

To create DSN:

1. Go to **Start** → **Settings** → **Control Panel** → **Administrative Tools** → **Data Sources (ODBC)** which opens ODBC Data Source Administrator, as shown in Figure 8-5.

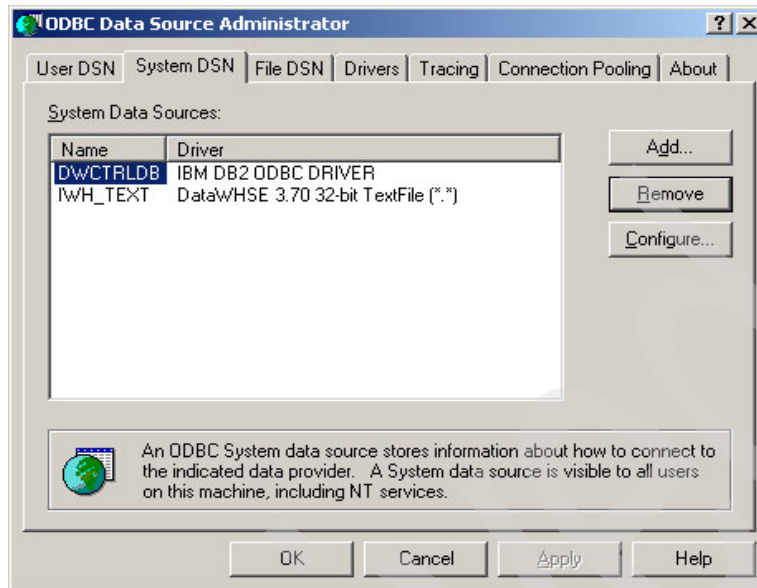


Figure 8-5 ODBC Data Source Administrator

- Click the **System DSN** tab, then click **Add**. A window with all database drivers pops up. Select **IBM DB2 ODBC DRIVER** and click **Finish** (Figure 8-6).

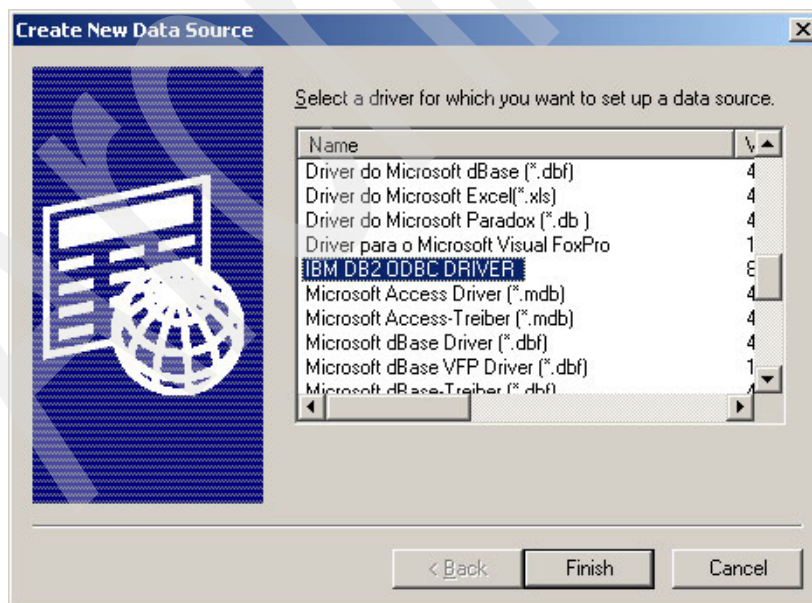


Figure 8-6 Create New Data Source

- When you click **Finish**, a window for adding database details pops up. Enter the appropriate information, as shown in Figure 8-7, and click **OK**.

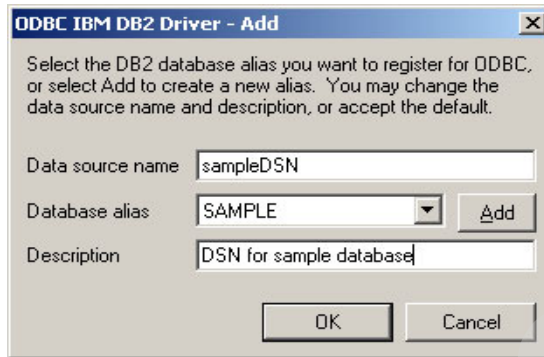


Figure 8-7 Add ODBC IBM DB2 driver

This adds a new DSN name in System Data Sources, as shown in Figure 8-8.

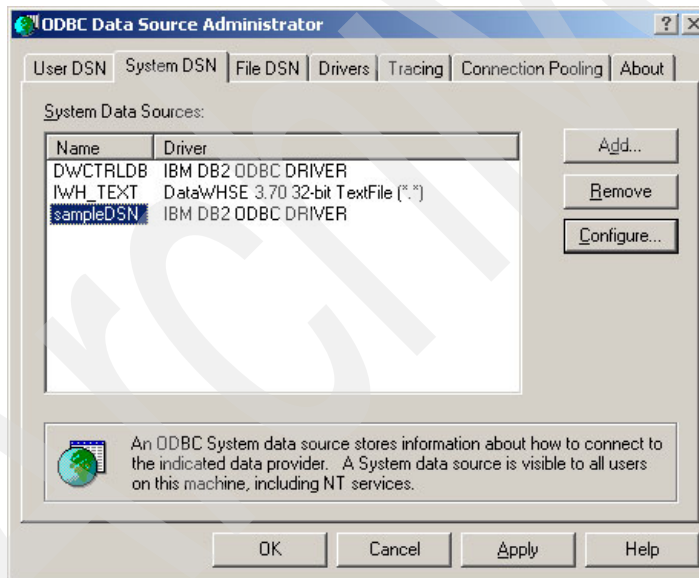


Figure 8-8 ODBC Data Source Administrator

- You can test the connectivity by clicking **Configure** to open the CLI/ODBC Settings window (Figure 8-9 on page 252).

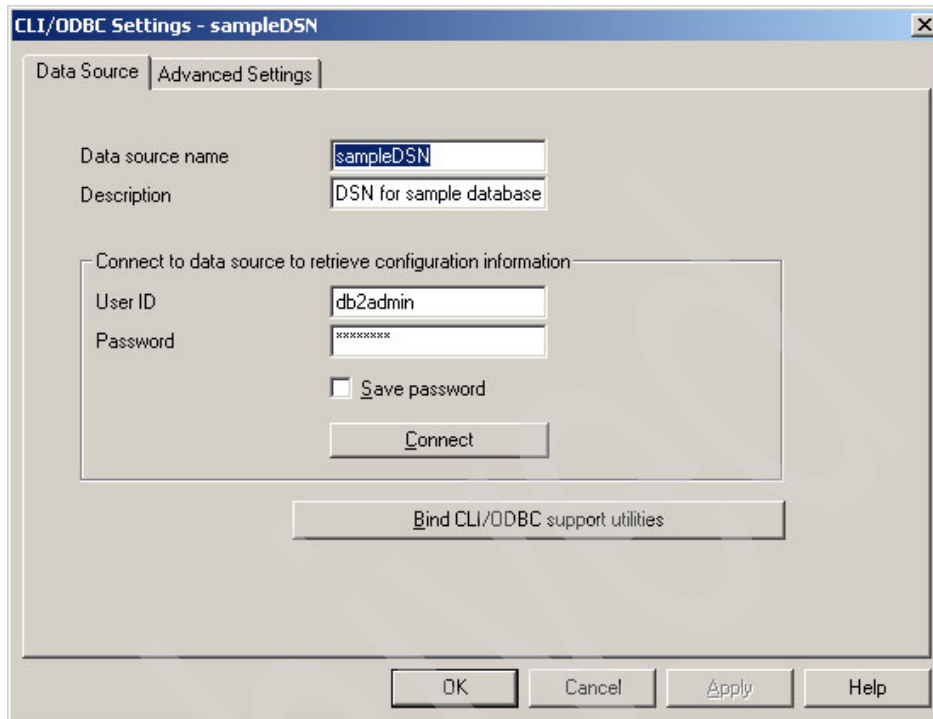


Figure 8-9 Test connection

5. Enter the proper user ID and password, and click **Connect** to test whether the DSN has proper connectivity with a database.

Next, we discuss how to use this DSN with ODBC .NET Provider to access a DB2 database table.

Use the same code as in the OLE DB section, except for the declaration of the namespace and how the connection string is specified to use ODBC .NET provider.

1. To start, add the namespace as shown:

```
using System.Data.Odbc;
```

2. The DSN that we created earlier is used to create a connection string:

```
cnn.ConnectionString = "DSN=sampleDSN";
//another way
// "DSN=sampleDSN;UID=db2admin;PWD=db2admin;Driver={IBM DB2 ODBC
Driver}";
```

Example 8-7 shows the entire code, which demonstrates how to use ODBC .NET Provider.

Example 8-7 Using ODBC .NET provider

```
using System;
using System.Data;
using System.Data.Odbc;

namespace odbcNet
{
    class odbcTest
    {
        [STAThread]
        static void Main(string[] args)
        {
            OdbcConnection cnn = null;
            OdbcCommand cmd = null;
            OdbcDataReader reader = null;
            try
            {
                cnn = new OdbcConnection ();
                cnn.ConnectionString = "DSN=sampleDSN";
                //"DSN=sampleDSN;UID=db2admin;PWD=db2admin;Driver={IBM DB2 ODBC
Driver}";
                cmd = new OdbcCommand();
                cmd.Connection = cnn;

                cmd.CommandText = "SELECT * FROM ITSODBA.ORG";
                cmd.CommandTimeout = 20;
                cnn.Open();
                reader = cmd.ExecuteReader();
                while (reader.Read() == true)
                {
                    Console.WriteLine (reader.GetString(1)); //Display Dept Name
                }
                if (reader != null)
                    reader.Close();
                if (cnn != null)
                    cnn.Close();
            }
            catch (OdbcException oe)
            {
                Console.WriteLine (oe.Message);
            }
        }
    }
}
```

```
        Console.ReadLine ();  
    }  
}  
}
```

Build and run the program, which displays the Department names on the Windows console.

8.4.3 DB2 .NET Data Provider

The DB2 .NET Data Provider is an extension of the ADO.NET interface that enables .NET applications to access a DB2 database through a secure connection, execute commands, and retrieve results. The DB2 .NET Data Provider delivers high-performing, secure access to DB2 data.

The DB2 .NET Data Provider enables your .NET applications to access the following database management systems:

- ▶ DB2 Universal Database V8 for Windows, UNIX, and Linux-based computers
- ▶ DB2 Universal Database V6 (or later) for OS/390 and z/OS, through DB2 Connect
- ▶ DB2 Universal Database V5, Release 1 (or later) for AS/400 and iSeries, through DB2 Connect
- ▶ DB2 Universal Database V7.3 (or later) for VSE & VM, through DB2 Connect

Using DB2 .NET Provider

To use DB2 .NET Provider, you must add a reference for it. Create a new Windows Console project in Visual Studio .NET and to add the reference select project name in the Project Explorer window. Right-click **Add References** to open the Add Reference window shown in Figure 8-10 on page 255.

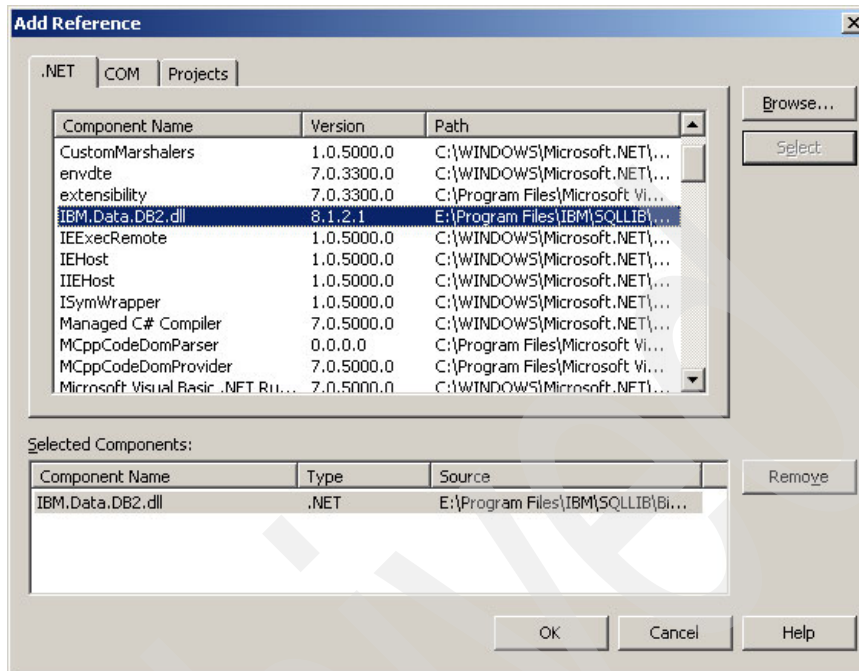


Figure 8-10 Add Reference

From the .NET tab, select **IBM.Data.DB2.dll** and click **Select**. This adds the component to the Selected Components list. Click **OK** to add the reference.

In a class, insert code to add the reference to the DB2 name space as shown:

```
using IBM.Data.DB2 ;
```

Using the DB2 .NET Provider, you can specify a connection string by database name as shown:

```
cnn.ConnectionString = "Database=SAMPLE";
```

The rest of the coding is the same as in the OLE DB .NET Provider section. Example 8-8 shows the complete code for using DB2 .NET provider.

Example 8-8 Using OLE DB .NET provider

```
using System;
using IBM.Data.DB2 ;

namespace db2Net
{
    class db2NetTest
```

```

{
    [STAThread]
    static void Main(string[] args)
    {
        DB2Connection cnn =null;
        DB2Command cmd=null;
        DB2DataReader reader=null;
        try
        {
            cnn= new DB2Connection ();
            cnn.ConnectionString ="Database=SAMPLE";

            cmd = new DB2Command();
            cmd.Connection =cnn;

            cmd.CommandText = "SELECT * FROM ORG";
            cmd.CommandTimeout=20;
            cnn.Open();
            reader = cmd.ExecuteReader();
            while (reader.Read()==true)
            {
                Console.WriteLine (reader.GetString(1)); //Display Dept Name
            }
            if (reader!=null)
                reader.Close();
            if (cnn !=null)
                cnn.Close();
        }
        catch (DB2Exception oe)
        {
            Console.WriteLine (oe.Message);
        }
        Console.ReadLine ();
    }
}

```

Build and run the program, which displays the Department names on the Windows console.

8.4.4 Comparison between providers

In previous sections, we saw various providers that are available to connect a DB2 database using .NET Framework. Although the steps for all providers look the same, they have some differences, which are discussed in the next section.

Comparison at code level

Table 8-11 summarizes the comparison at code level between all three providers.

Table 8-11 Provider code-level comparison

	OLE .NET Provider	ODBC .NET Provider	DB2 .NET Provider
Namespace	System.Data.OleDb	System.Data.ODBC	IBM.Data.DB2
Connection Object (used to create connection with database)	System.Data.ODBC	ODBCConnection	DB2Connection
Command Object (used to execute command)	OleDbCommand	ODBCCommand	DB2Command
Data Reader Object (used to read retrieved data)	OleDbDataReader	ODBCDataReader	DB2DataReader
DataAdapter	OleDbDataAdapter	ODBCDataAdapter	DB2DataAdapter
ConnectionString	IBMDADB2.1;User ID=db2admin;Password=db2admin;Data Source=SAMPLE	DSN=sampleDSN;UID=db2admin;PWD=db2admin;Driver={IBM DB2 ODBC Driver}	Database= SAMPLE
Transaction Object	OleDbTransaction	ODBCTransaction	DB2Transaction

Functionality comparison

Table 8-12 shows a functionality comparison of the three providers.

Table 8-12 .NET provider functionality comparison

	OLE DB .NET Provider	ODBC .NET Provider	DB2 .NET Provider
Pass-thru SQL	Y	Y	Y
Simple stored procedures	Y	N	Y
In, Out, and InOut parameters	N	N	Y
Dates and Currency	Y	Y	Y
LOB	N	Y	Y

8.5 Performing operations on a DB2 database

As DB2 .NET managed provider is much better than other providers in functionality and performance, we discuss only the DB2 .NET provider for the rest of the chapter:

- ▶ How to use DataAdapter and DataSet (disconnected model)
- ▶ Calling a Stored Procedure
- ▶ Controlling local transactions
- ▶ Using large objects (LOBs)
- ▶ Accessing DB2 data objects on Win Forms (binding data controls)

8.5.1 Using DataAdapter and DataSet (Disconnected model)

As we discussed, DataAdapter and DataSet are used to improve performance by reducing database server trips from the application. In this section we show an example of how to use DataAdapter and DataSet as a disconnected model.

Open and create a Windows Console project in C#, adding proper namespaces. In the Main method, create a connection and command object (Example 8-9).

Example 8-9 Using DataAdapter and DataSet (disconnected model)

```
static void Main(string[] args)
{
    DB2Connection cnn = null;

    try
    {
        cnn = new DB2Connection("Database=Sample");
        DB2Command cmd = new DB2Command();
        cmd.Connection = cnn;
        cmd.CommandText = "Select * from staff";
        cnn.Open();
    }
}
```

Create DataSet and DB2DataAdapter objects. The DB2DataAdapter object is created using CommandText and Connection objects and acts as bridge between DataSet and a database.

```
//create dataset
DataSet ds = new DataSet();
//create Data adapter
DB2DataAdapter db2Adpt = new DB2DataAdapter(cmd.CommandText,cnn);
```


The Fill method of DB2DataAdapter fills DataSet by executing the query assigned to CommandText:

```
db2Adpt.Fill(ds);  
cnn.Close ();
```

As you have data in DataSet, from here you do not need connection. You can perform various SQL operations on the data in a data set. The connection object is then required only to synchronize changes with the database.

In the example, the DataSet is populated using DataTable and DataRow objects. The DataTable object represents tables in DataSet, and DataRow represents rows. The data in DataSet is populated as shown:

```
foreach (DataRow r in ds.Tables[0].Rows)  
{  
    for (int i=0; i<r.ItemArray.Length; i++)  
    {  
        if (i>0) Console.Out.Write(", ");  
        Console.Out.Write(r.ItemArray[i]);  
    } // for  
    Console.Out.WriteLine();  
} // foreach
```

Figure 8-11 on page 260 shows some output of the code.

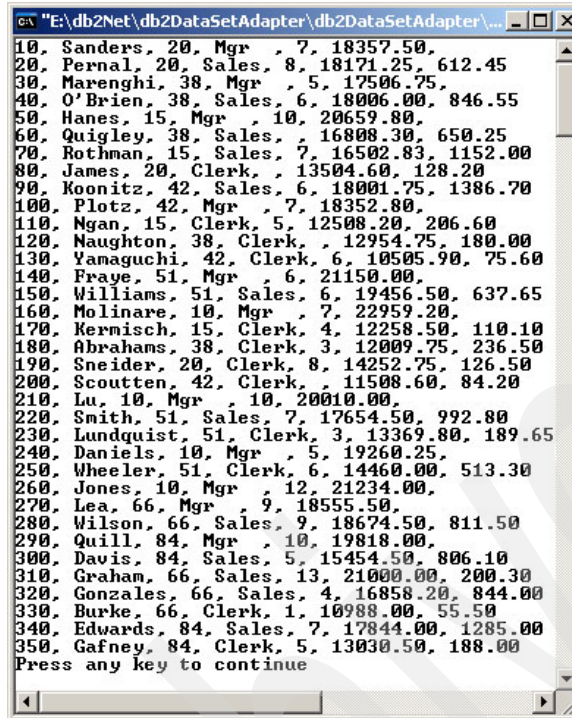


Figure 8-11 Output of using DataTable and DataRow to populate data

You can also get XML representation of data in DataSet using the GetXml method of DataSet as shown:

```

// xml output of dataset
string xmlstr = ds.GetXml();
Console.Out.WriteLine(xmlstr);
}
catch (Exception e)
{
    Console.Out.WriteLine(e.Message);
}
}

```

Figure 8-12 on page 261 resembles the XML output returned by the GetXml method.



```
<Table>
  <ID>20</ID>
  <NAME>Pernal</NAME>
  <DEPT>20</DEPT>
  <JOB>Sales</JOB>
  <YEARS>8</YEARS>
  <SALARY>18171.25</SALARY>
  <COMM>612.45</COMM>
</Table>
<Table>
  <ID>30</ID>
  <NAME>Marenghi</NAME>
  <DEPT>38</DEPT>
  <JOB>Mgr </JOB>
  <YEARS>5</YEARS>
  <SALARY>17506.75</SALARY>
</Table>
<Table>
  <ID>40</ID>
  <NAME>O' Brien</NAME>
  <DEPT>38</DEPT>
  <JOB>Sales</JOB>
  <YEARS>6</YEARS>
  <SALARY>18006.00</SALARY>
  <COMM>846.55</COMM>
</Table>
<Table>
  <ID>50</ID>
  <NAME>Hanes</NAME>
  <DEPT>15</DEPT>
  <JOB>Mgr </JOB>
  <YEARS>10</YEARS>
  <SALARY>20659.80</SALARY>
</Table>
<Table>
  <ID>60</ID>
  <NAME>Quigley</NAME>
```

Figure 8-12 XML returned by GETXML() method

8.5.2 Calling stored procedures

A stored procedure is a set of SQL statements and optional control-of-flow statements that are stored under a unique name in a database.

In DB2 UDB V8.2, stored procedures can be written in various languages:

- ▶ SQL stored procedures

In an SQL stored procedure, the source code is written in SQL and is a part of the CREATE PROCEDURE statement.

- ▶ Java stored procedures

Java stored procedures can access DB2 data using either Java Database Connectivity (JDBC) API calls or SQLJ statements.

- ▶ Common Language Runtime (CLR) stored procedures

A CLR stored procedures references a .NET assembly as its external code body.

Advantages of stored procedures

The advantages of using stored procedures include:

- ▶ Reduced network usage between client and server
- ▶ Enhanced hardware and software capabilities
- ▶ Improved security
- ▶ Reduced development cost and increased reliability
- ▶ Centralized security, administration, and maintenance for common routines

Using DB2 .NET Provider, you can call stored procedures, which takes input parameters and returns results in output parameters.

Before starting actual coding, make sure DB2 UDB has stored procedure and cataloged on it. To demonstrate, we used the following stored procedure, which takes an employee ID as an input and returns his salary, name, and job, as shown in Example 8-10.

Example 8-10 Create stored procedure EMP_INFO

```
CREATE PROCEDURE EMP_INFO (IN idno INT,  
                           OUT empsalary DOUBLE,  
                           OUT empname VARCHAR(9),  
                           OUT empjob CHAR(5))  
  
  BEGIN  
    DECLARE year INT DEFAULT 0;  
    DECLARE c1 CURSOR FOR  
    SELECT CAST(salary AS DOUBLE), years, job, name FROM staff WHERE ID =  
      idno;  
    OPEN c1;  
    FETCH c1 INTO empsalary, year, empjob, empname;  
    CLOSE c1;  
    RETURN year;  
  END
```

In .NET code, we assume that a connection is already created. To create a command object whose `CommandType` property is set to `CommandType.StoredProcedure` and `CommandText` to the name of the stored procedure:

```
DB2Command cmd = cnn.CreateCommand();  
cmd.CommandText = "EMP_INFO";  
cmd.CommandType = CommandType.StoredProcedure;
```

The input and output parameters can be assigned using the `Db2Parameter` object. Here you can define details of each parameter including direction and type, as shown in Example 8-11 on page 263.

Example 8-11 Create in and out parameters

```
// create in and out parameters
    DB2Parameter parm = cmd.Parameters.Add("@empyears", DB2Type.Integer);
    parm.Direction = ParameterDirection.ReturnValue;
    parm = cmd.Parameters.Add("@empid", DB2Type.SmallInt);
    parm.Direction = ParameterDirection.Input;
    parm.Value = empId;
    parm = cmd.Parameters.Add("@empsalary", DB2Type.Double);
    parm.Direction = ParameterDirection.Output;
    parm = cmd.Parameters.Add("@empname", DB2Type.VarChar,9);
    parm.Direction = ParameterDirection.Output;
    parm = cmd.Parameters.Add("@empjob", DB2Type.Char,5);
    parm.Direction = ParameterDirection.Output;
```

Execute the stored procedure using the `ExecuteNonQuery` method of command object.

```
// Call the stored procedure
cmd.ExecuteNonQuery();
```

Finally, retrieve output parameters using command object and display the output as shown in Example 8-12.

Example 8-12 Retrieve output parameter using command object

```
// Retrieve output parameters
    Double salary = (Double)cmd.Parameters["@empsalary"].Value;
    Int32 years = (Int32)cmd.Parameters["@empyears"].Value;
    String name = (String)cmd.Parameters["@empname"].Value;
    String job = (String)cmd.Parameters["@empjob"].Value;

    // Display details of the employee
    Console.WriteLine("    Employee Name : " + name);
    Console.WriteLine("    Employee Job : " + job);
    Console.WriteLine("    Employee Salary : " +
String.Format("{0:f2}",salary));
    Console.WriteLine("    Years Served : " + years);
```

8.5.3 Controlling transaction

DB2 .NET provider has the capability to handle transactions at the local level. You can define where to start a transaction using the `BeginTransaction` method of the connection object, and you can control where to commit and roll back results using the `Commit` and `RollBack` methods.

The `DB2Transaction` class is used for controlling local transactions. In the following example, the `UPDATE` SQL query updates staff table, a transaction

object commits the update using the Commit method, and the Rollback method rolls back the entire transaction if any errors occur.

In order to perform operations on the database, necessary objects are declared:

```
DB2Connection cnn = null;
DB2Transaction trans = null;
DB2Command cmd = null;
```

Next, the connection and command object are created with CommandText for updating the record in the table:

```
try
{
    cnn = new DB2Connection("Database=Sample");
    cmd = new DB2Command();
    cmd.Connection = cnn;
    cmd.CommandText = "update staff set salary = 20000 where id =20";

    cnn.Open();
```

The BeginTransaction method of connection object is used to initiate the transaction object, then it is assigned to the Transaction property of the command object:

```
trans = cnn.BeginTransaction ();
md.Transaction = trans;
```

The ExecuteNonQuery method executes update statements on the database and returns the total number of updated rows:

```
int rowsUpdated = cmd.ExecuteNonQuery();
Console.WriteLine ("Rows Updated = " + rowsUpdated);
```

If everything runs as planned, the Commit method of transaction object commits updates in the staff table:

```
trans.Commit ();
cnn.Close ();
}
```

But if any errors occur, the transaction is rolled back using the Rollback method:

```
catch (Exception e)
{
    trans.Rollback();
    Console.Out.WriteLine(e.Message);
}
```

8.5.4 Using large objects (LOBs)

An LOB is a large block of data that is stored in a database, such as an image or sound file. LOBs are not stored in table rows but in separate pages referenced by a pointer in the row.

The three types of LOB data types in DB2 are:

- ▶ Character large objects (CLOBs)

These are typically used to store blocks of text items, such as ASCII or PostScript files.

- ▶ Double-byte character large objects (DBCLOBs)

A DBCLOB is used to store large DBCS character-based data such as documents written with a single character set. A DBCLOB value can be up to 1 073 741 823 double-byte characters long.

- ▶ Binary large objects (BLOBs)

BLOB has no structure that can be interpreted by the database management system but is known only by its size and location.

The DB2 .NET Provider has the capability to handle LOBs. By using DB2 .NET provider, code can be written to execute SELECT, INSERT, UPDATE, or DELETE the LOBs from a database. For an example we use the Employee Information application, which gets an employee ID from the user and displays user information and his photo. The employee photo is stored in the database as an LOB. To start, open a new Visual C# Windows Application project and put controls on the default form, as shown in Figure 8-13 on page 266.

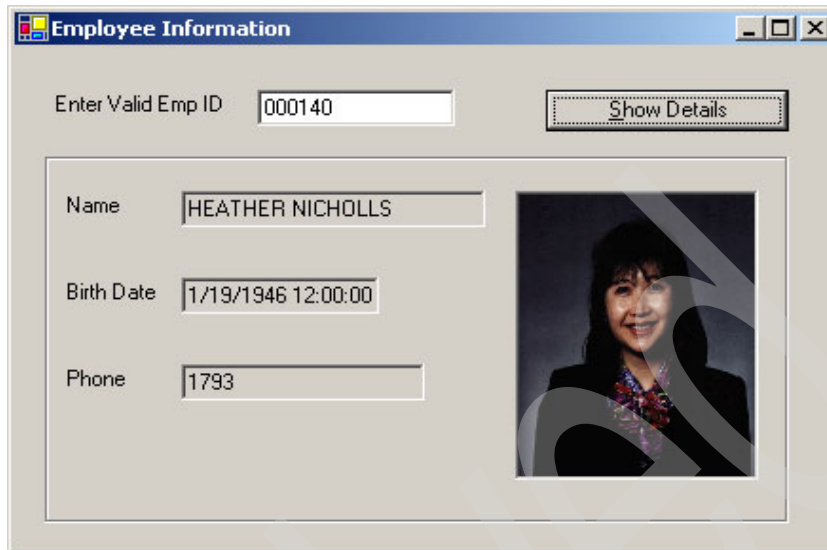


Figure 8-13 Employee Information result

Give proper names to the controls on a form and open a code window. Add references to IBM.Data.DB2 as in Example 8-8 on page 255. The code window consists default namespaces such as:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
```

In addition, add necessary namespaces for the database and other operations:

```
using System.Data;
using IBM.Data.DB2;
using System.IO ;
```

Add the `getCnn` function, which creates and returns the connection object as shown in Example 8-13.

Example 8-13 Add `getCnn`

```
private DB2Connection getCnn ()
{
    DB2Connection conn = null;
    try
    {
        conn=new DB2Connection("Database=SAMPLE");
        conn.Open ();
    }
```



```

    }
    catch (DB2Exception e)
    {
        MessageBox.Show (e.Message);
        conn.Close ();
    }
    return conn;
}

```

From the form design window, double-click the **Show Details** button and add code to retrieve data from the database, using the connection created in the `getCnn` method:

```

private void button1_Click(object sender, System.EventArgs e)
{
    DB2Connection cnn = null;
    cnn=getCnn();

```

Create a command object and assign a query that will return the employee's information, including his picture, as shown below:

```

    DB2Command cmd = cnn.CreateCommand();
    cmd.CommandText =
        "SELECT a.firstnme, a.lastname, a.phoneno, a.birthdate," +
        "b.picture FROM ADMINISTRATOR.EMPLOYEE a , " +
        "ADMINISTRATOR.emp_photo b " +
        "where a.EMPNO = b.empno and a.empno='" + txtEmpID.Text + "'";

```

To get the employee information, execute the SQL statement using the `ExecuteReader` method:

```

    DB2DataReader reader;
    reader = cmd.ExecuteReader(CommandBehavior.SingleResult);

    if(reader.Read())
    {

```

Populate the result in text boxes as shown:

```

        txtName.Text = reader.GetString(0) + " " + reader.GetString(1);
        txtPh.Text = reader.GetString (2);
        txtDOB.Text = reader.GetDate(3).ToString();
    }
}

```

To get the employee picture (LOB) data, create an array of type `Byte` and read the bytes using the `GetBytes` method of the reader. To display the employee's picture, convert the bytes into a stream using `MemoryStream` and assign the stream to the picture object as shown:

```

        // Obtain the LOB object

```

```
int maxSize = 102400;
Byte[] out_picture = new Byte[maxSize];
reader.GetBytes(4,0,out_picture,0,maxSize);
MemoryStream mem = new MemoryStream(out_picture);
picEmp.Image =Image.FromStream(mem);
```

Finally, do the cleanup of used objects:

```
}
reader.Close();
cnn.Close();
}
```

8.5.5 Binding Data Controls with ADO.NET

In .NET, Windows front-end or Win Form can interact with a database in two ways: either by writing code for each functionality as we have seen in previous sections or by simply dragging and dropping (binding) database-related controls. Dragging and dropping controls for database interaction requires few lines of code: an easy and fast way for developing the applications.

In this section, we give one simple example, which displays the staff table on a Windows form.

1. Open a new Windows Application project with a proper project name.

To the left of the new project, you will find all data-related controls under the Toolbox window. These data controls have the same properties and methods that we discussed in previous sections. Figure 8-14 shows various data controls that are available for database binding.

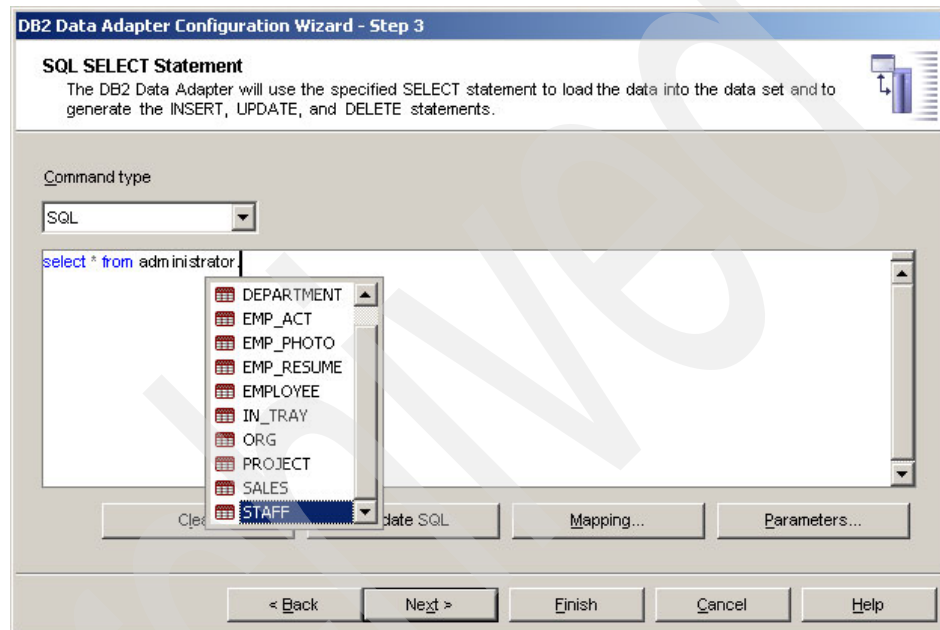


Figure 8-14 dbControls

2. From the Data window, drag and drop the **DB2DataAdapter** object onto the form.

3. This opens the DB2 Data Adapter Configuration Wizard (Figure 8-15). Click **Next**.

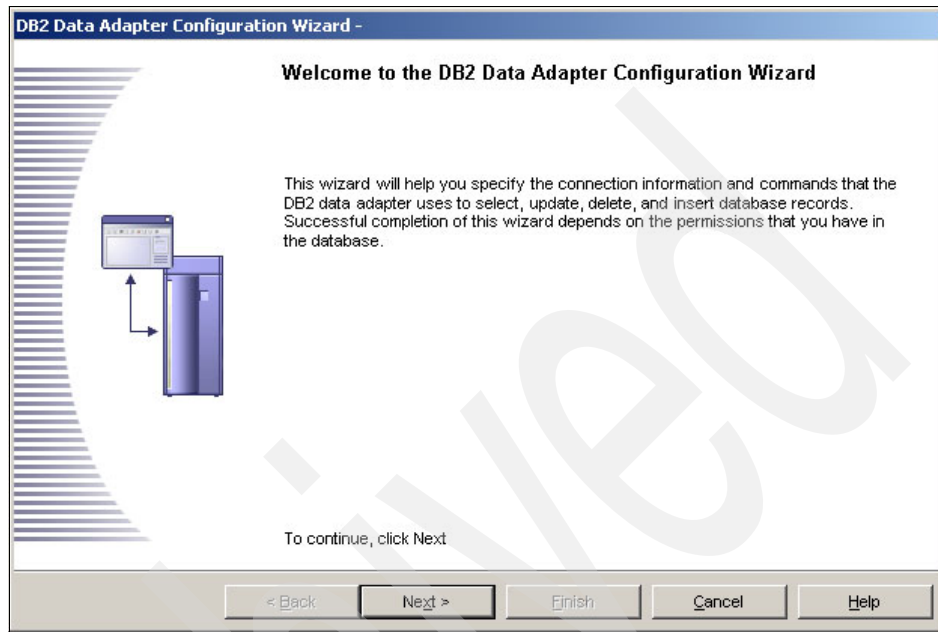


Figure 8-15 DB2 Data Adapter Configuration Wizard

4. In the next step of the wizard, enter the connection name, user name, and password. The **Test Connection** button provides the capability to check the connection with the database (Figure 8-16). Click **Next**.

DB2 Data Adapter Configuration Wizard - Step 1

DB2 Data Connection

In order to load and update data, DB2 Data Adapter will use this database connection information.

Connection name
SAMPLE:db2admin(localhost) [New Connection]

Server: localhost Database: SAMPLE

User name: db2admin Password: ***** [Test Connection]

< Back Next > Finish Cancel Help

Figure 8-16 DB2 Data Adapter Configuration Wizard: Step 1

5. In the SQL Statement Options window (Figure 8-17), you can select query types to perform operations on a data source.

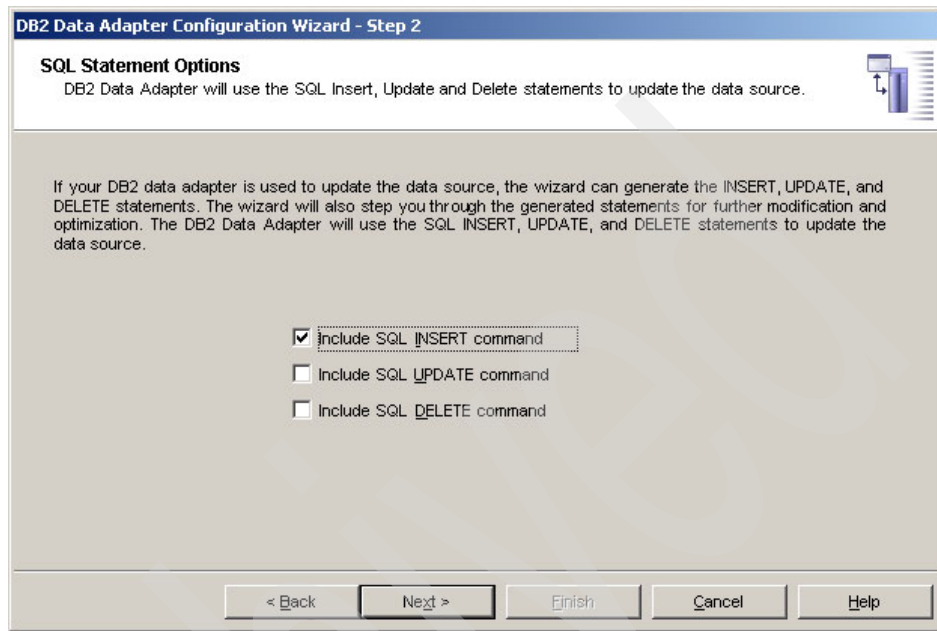


Figure 8-17 DB2 Data Adapter Configuration Wizard: Step 2

6. In the next window (Figure 8-18), select the command type, which can be SQL or stored procedure. Select **SQL** and enter `select SQL ()`. The wizard automatically shows all table names after the schema name. Select the **STAFF** table from the list and click **Next**.

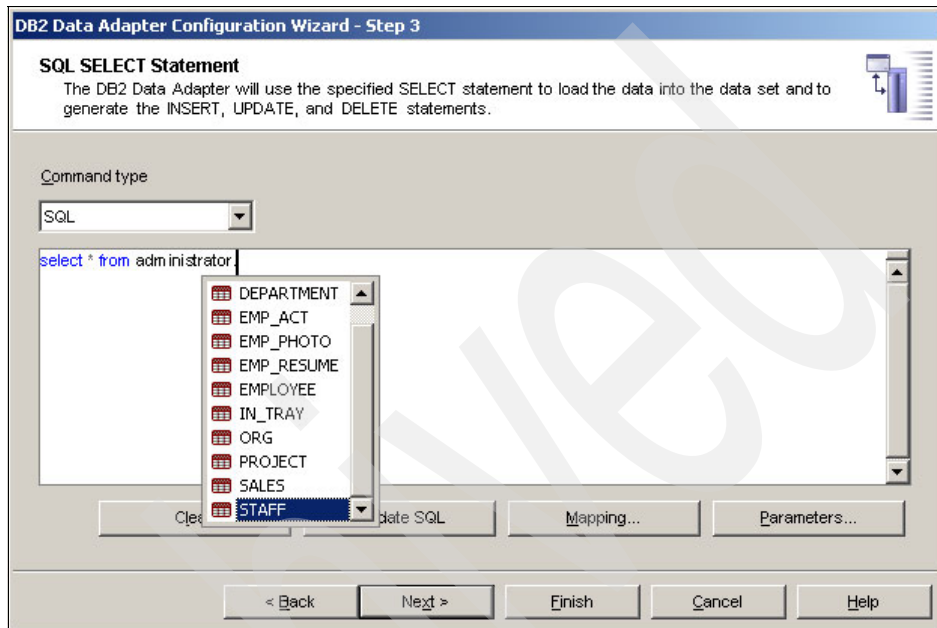


Figure 8-18 DB2 Data Adapter Configuration Wizard: Step 3

7. The wizard shows a summary of all options you chose (Figure 8-19).

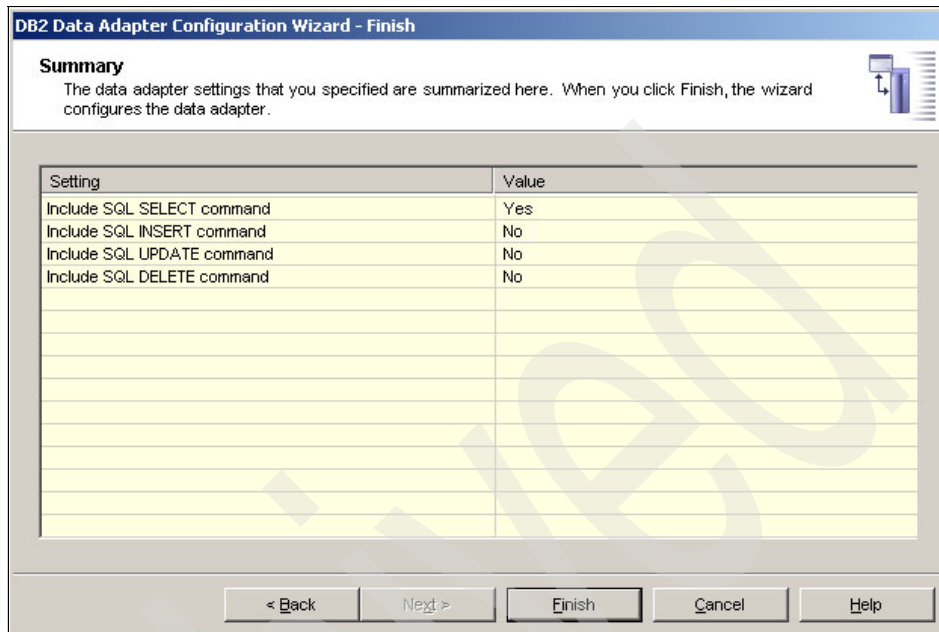


Figure 8-19 DB2 Data Adapter Configuration Wizard: Finish

8. When you click **Finish**, the Studio automatically creates DB2DataAdapter and Db2Connection objects as shown in Figure 8-20. Right-click **db2DataAdapter1** and select **Generate Data Set** from the pull-down menu.

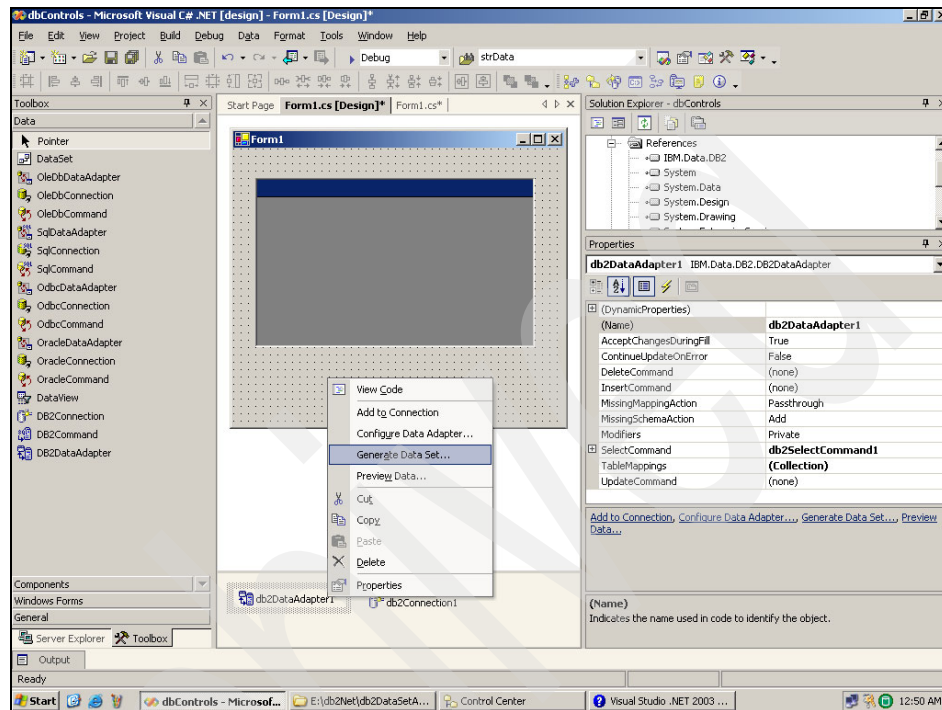


Figure 8-20 DB2DataAdapter and Db2Connection created

9. A window opens and asks for input tables (Figure 8-21). Select the tables that you want to add to the data set and click **OK**.

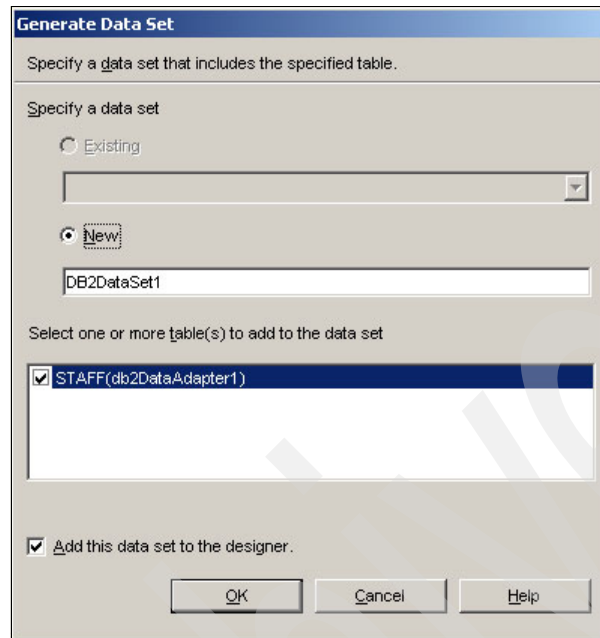


Figure 8-21 Generate Data Set

10. This creates the data set object db2DataSet11, as shown in Figure 8-22.

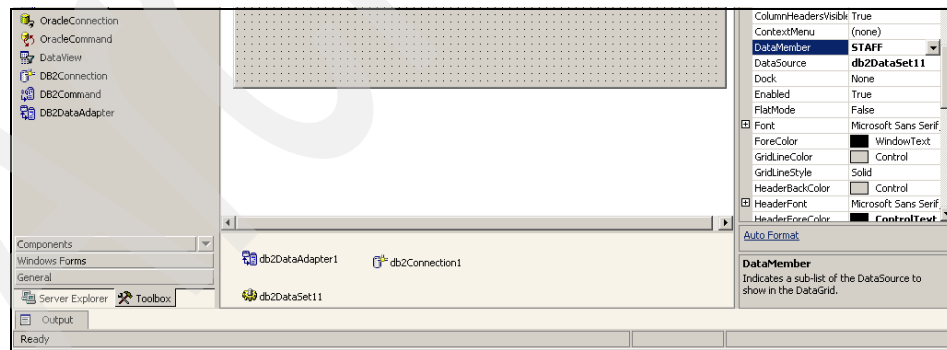


Figure 8-22 db2DataSet11 created

11. Drag and drop the DataGrid and text boxes onto the form for binding (Figure 8-23). The DataGrid control enables you to select, sort, and edit data. The DataMember and DataSource property are used to bind the control from the property window. To bind textbox controls, select **Text** in the **DataBindings** category and select the field name that you want to bind with text control.

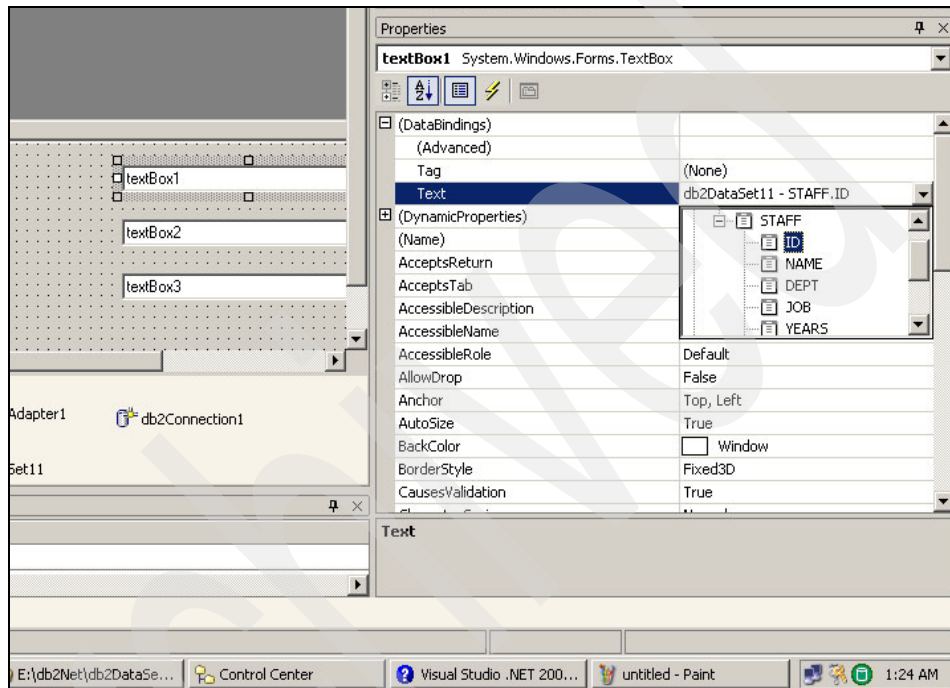


Figure 8-23 Drop DataGrid and text boxes on form

12. Finally, drop a button from the Toolbox window onto the form. Double-click this button and add the following code to clear and fill the data source:

```
db2DataSet11.Clear();  
db2DataAdapter1.Fill(db2DataSet11);
```

Note that you have to write only these two lines to populate all data in the controls.

13. Run the project and click **Fill** to populate Staff data in the controls (Figure 8-24).

ID	NAME	DEPT	JOB	YEARS	SALARY
10	Sanders	20	Mgr	7	18357.50
20	Pernal	20	Sales	8	20000.00
30	Marenghi	38	Mgr	5	17506.75
40	O'Brien	38	Sales	6	18006.00
50	Hanes	15	Mgr	10	20659.80
60	Quigley	38	Sales	(null)	16808.30
70	Rothman	15	Sales	7	16502.83

ID:

Name:

Department:

Figure 8-24 Staff Information

8.5.6 Accessing DB2 with Web Forms

In order to make database-interactive Web pages, follow the steps discussed previously, except open a new **Web Project** instead of a Windows Application project in step 1 on page 269.

8.6 Add-ins and stored procedures in CLR

DB2 UDB Version 8.2 introduced the IBM DB2 Development Add-In, which helps developers interact easily with DB2 tools from Visual Studio .NET.

DB2 UDB V8.2 also can write and run stored procedures that were written using Common Language Runtime (CLR). In the next section, we discuss these two features.

8.6.1 IBM DB2 Development Add-In overview

The IBM DB2 Development Add-In is a collection of features that integrate with the Microsoft Visual Studio .NET development environment. DB2 UDB V8.2 has these features, through which you can perform operations on DB2:

- ▶ Launch various DB2 development and administration tools.
- ▶ Create and manage DB2 projects in the Solution Explorer.
- ▶ Access and manage DB2 data connections in the IBM Explorer.
- ▶ Create and modify DB2 scripts, including scripts to create stored procedures and user-defined functions (UDFs).

8.6.2 DB2 Toolbar

With the DB2 Toolbar, you can launch the following DB2 tools (Figure 8-25):

- ▶ Development Center
- ▶ Control Center
- ▶ Replication Center
- ▶ Command Editor
- ▶ Task Center
- ▶ Health Center
- ▶ Journal
- ▶ Information Center

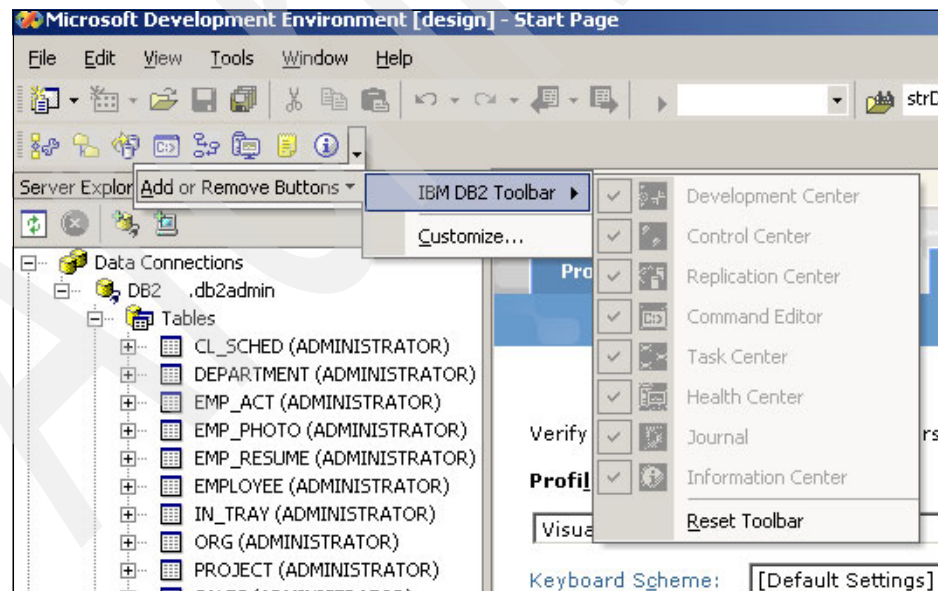


Figure 8-25 DB2 Toolbar

8.6.3 DB2 Database Project type

The IBM DB2 Development Add-In introduces a new IBM Projects folder (Figure 8-26), which includes a DB2 Database Project type for developing DB2 database server scripts (Figure 8-27 on page 281). With a DB2 Database Project, you can:

- ▶ Add new or existing SQL stored procedure scripts.
- ▶ Add new or existing SQL UDF scripts.
- ▶ Add new or existing scripts based on generic templates.
- ▶ Specify build configuration options, including script build order.
- ▶ Check script files into Microsoft Visual Source Safe source-control management system.

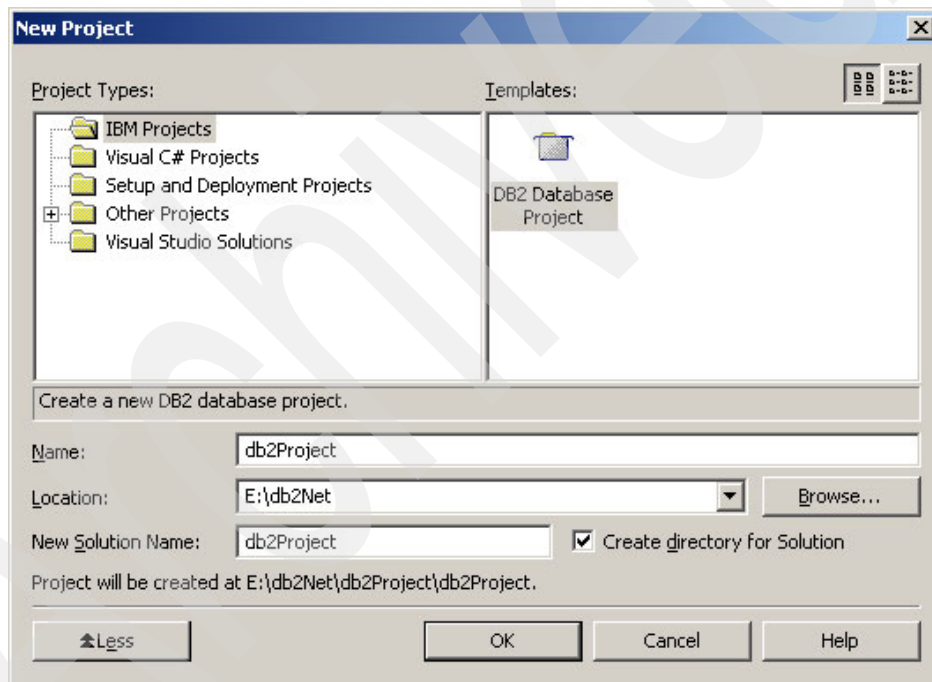


Figure 8-26 IBM Projects folder

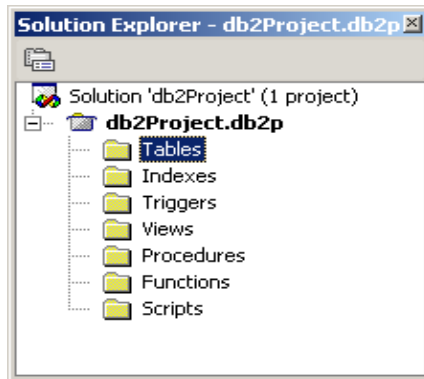


Figure 8-27 DB2 project type

8.6.4 Data Connections folder in the IBM Explorer

The IBM DB2 Development Add-In extends the Visual Studio .NET environment by adding a new tool called IBM Explorer (Figure 8-28), a dockable window that is similar to the Visual Studio .NET Server Explorer.

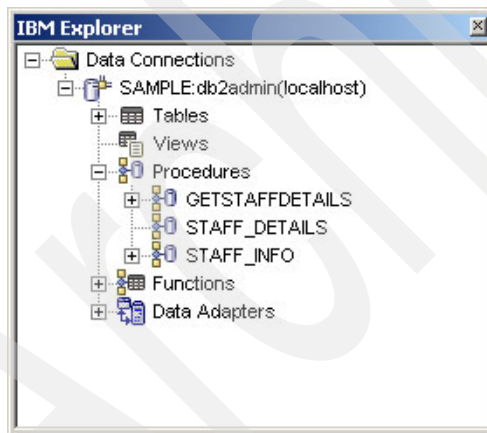


Figure 8-28 IBM Explorer

IBM Explorer provides Visual Studio .NET users with access to IBM database connections using the Data Connections folder, which is designed specifically for DB2 managed provider connections. From the Data Connections folder in the IBM Explorer, you can:

- Work with multiple named DB2 connections supporting connect of demand technology.

- ▶ Specify database catalog filters and local caching for higher performance and scalability.
- ▶ View properties of server objects including tables, views, stored procedures, and functions.
- ▶ Retrieve data from tables and views.
- ▶ Execute test runs for stored procedures and UDFs.
- ▶ View source code for stored procedures and functions.
- ▶ Generate ADO .NET code using drag and drop.

8.6.5 DB2 SQL Editor

The IBM DB2 Development Add-In also provides a DB2 SQL Editor. With the editor, you can change and view the code in your DB2 routines and scripts. The DB2 SQL Editor integrates with the Microsoft Visual Studio .NET IntelliSense feature, which enables intelligent auto-completion while typing DB2 scripts.

8.7 Developing DB2 stored procedures in .NET

A stored procedure is a set of SQL statements and optional control-of-flow statements stored under a unique name in a database. The optional control-of-flow can be written in PL/SQL, Java. In DB2 UDB V8.2, you can develop stored procedures using a code based on Common Language Runtime (CLR).

To develop stored procedures in .NET, you need Microsoft .NET Visual Studio 2003 and DB2 UDB V8.2 on Windows. We used the Sample database while demonstrating stored procedure in .NET.

Before starting development, make sure IBM Explorer has a connection to the Sample database. If not, launch IBM Explorer by clicking **View → IBM Explorer** and add a connection for Sample by right-clicking the **Data Connection** folder.

Cataloging DB2 CLR procedures and deploy the CLR assemblies to the local or remote DB2 server requires a DB2 database project. The DB2 project contains the set of script files that defines your CLR procedures, and the list of assemblies that are required by these CLR procedures.

1. Create a DB2 database project by clicking **File → New Project**.

2. In the Project Types pane, select **IBM Projects**, and then select **DB2 Database Project** in the Templates pane. Name and situate the project and click **OK** (Figure 8-29).

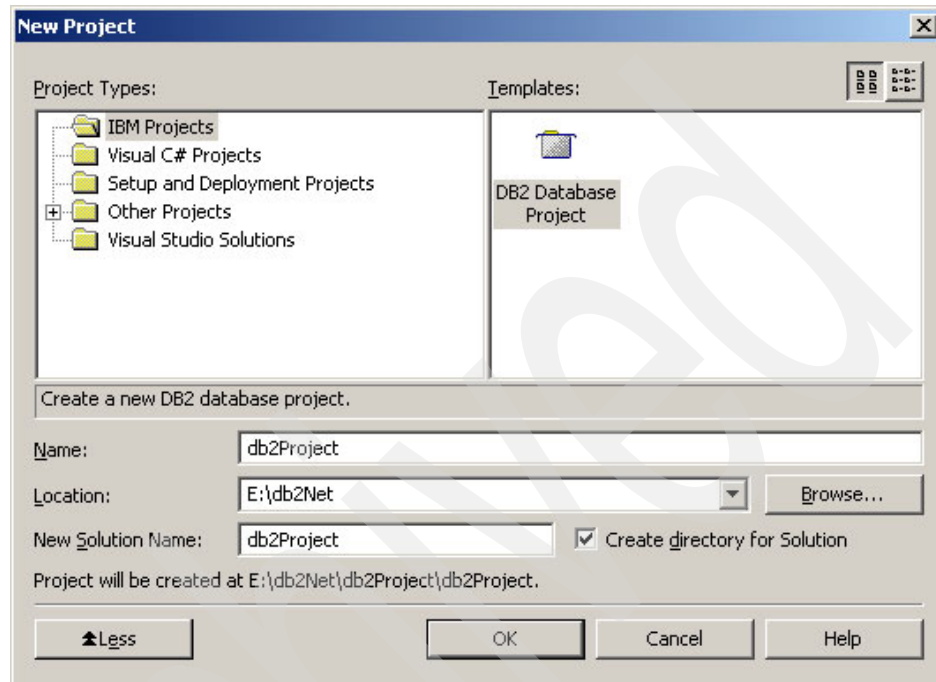


Figure 8-29 Open a DB2 project

3. A pop-up window, DB2 Data Connection, appears to get data connection related information (Figure 8-30). Select the **SAMPLE** database under Connection name and click **OK**.

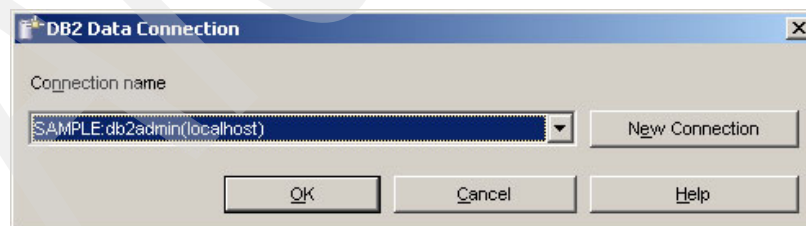


Figure 8-30 DB2 Data Connection window

This creates a new DB2 Project in Solution Explorer (Figure 8-31). Here you can add and define various DB2-related objects.

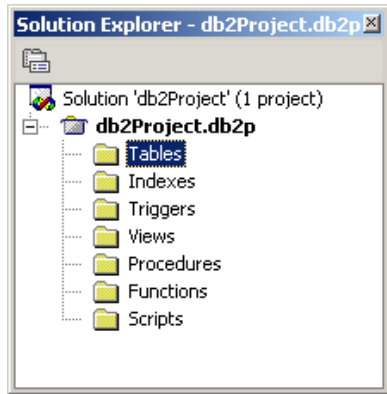


Figure 8-31 DB2 project created in Solution Explorer

To write stored procedure in .NET, you must create a project for code and logic of stored procedure. When you install DB2 UDB V8.2, the Visual C# Projects folder is extended to include a new project template, DB2 Class Library, that serves the purpose of writing code and logic for the stored procedure.

To write stored procedure code, select **db2Project** in the Solution Explorer and right-click to select **Add** → **New Project**. In the Add New Project window, expand the **Visual C# Projects** folder and choose the **DB2 Class Library** template. Name the project and click **OK**. A default class, **DB2Class1.cs**, is automatically created with default stored procedure code. Note that the project automatically references to the DB2 managed provider assembly, **IBM.Data.DB2**.

The Solution Explorer with the new project looks similar to Figure 8-32.

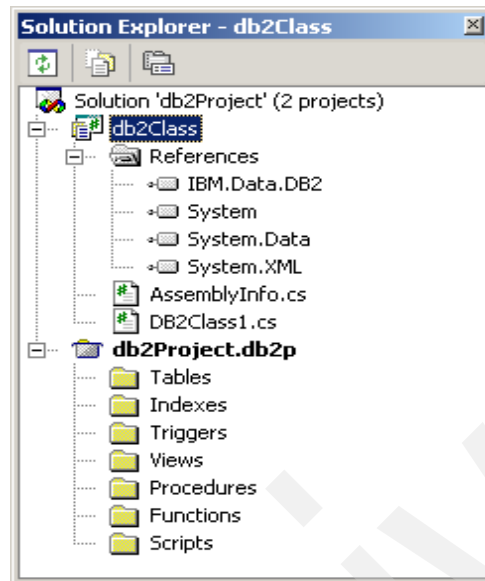


Figure 8-32 Solution Explorer: db2Class

The CLR methods and assemblies must be built before they can be deployed and cataloged on the DB2 server. To set up the build order of these projects and to ensure correct application build and deployment, you must set Project Dependencies.

To set the project dependencies and build order, right-click the **db2Project** and select **Project Dependencies**. A Project Dependencies window opens. Select **db2Class** and click **OK** (Figure 8-33).

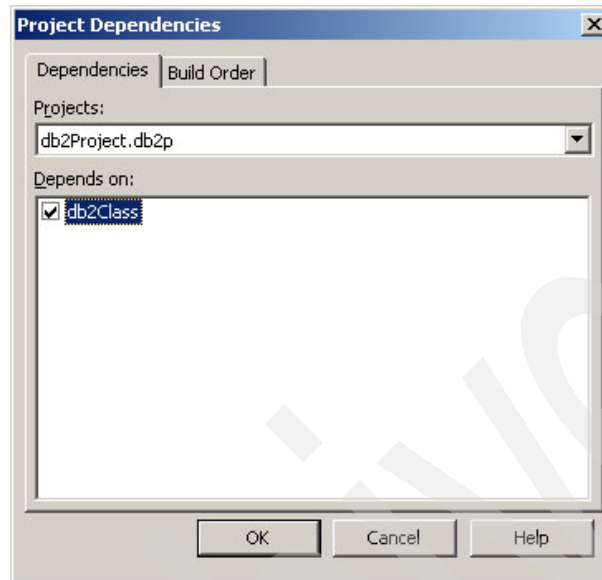


Figure 8-33 Project Dependencies: db2Class

To demonstrate CLR-based stored procedure, we create the `getStaffDetails` method, which takes Staff ID as an input parameter and returns a name as an output parameter.

In a code window of `DB2Class1.cs`, modify or add the code in Example 8-14 for the `getStaffDetails` procedure. Note that this code is the same as discussed in the DB2 .NET Provider section.

Example 8-14 getStaffDetails code section

```
using System;
using IBM.Data.DB2;

namespace db2Class
{
    public class DB2Class1
    {
        public static void getStaffDetails(
            String inStaffID,
            out String outStaffName)
        {
            // Create new command object from connection context
```

```

DB2Command myCommand = DB2Context.GetCommand();

outStaffName = "";
try
{
    myCommand.CommandText =
        "SELECT Name FROM STAFF " +
        "WHERE ID = '" + inStaffID + "'";
    DB2DataReader reader = myCommand.ExecuteReader();
    if (reader.Read())
    {
        outStaffName = reader.GetString(0);
    }
    reader.Close();
}
catch( DB2Exception eR )
{
    outStaffName = "DB2Exception: " + eR.Message;
}
catch( Exception eS )
{
    outStaffName = "Exception: " + eS.Message;
}

}
}

```

8.7.1 Save and build the solution

You can add more than one stored procedure in the same project by right-clicking the **db2Class** project and selecting **Add → Add New Item**. The Add New Item window pops up, from which you select the **DB2 Procedure Class** project template to add a new procedure class in the project.

After you complete and build the solution, the stored procedure must be cataloged on DB2 Server. The **db2Project** serves this purpose by generating a script file. To catalog the stored procedure:

1. Right-click the **db2Project** project and select **Add → Add New Item**.

2. In the Add New Item window (Figure 8-34), select the DB2 **CLR Procedure Wizard** and name it DB2CLRProc.

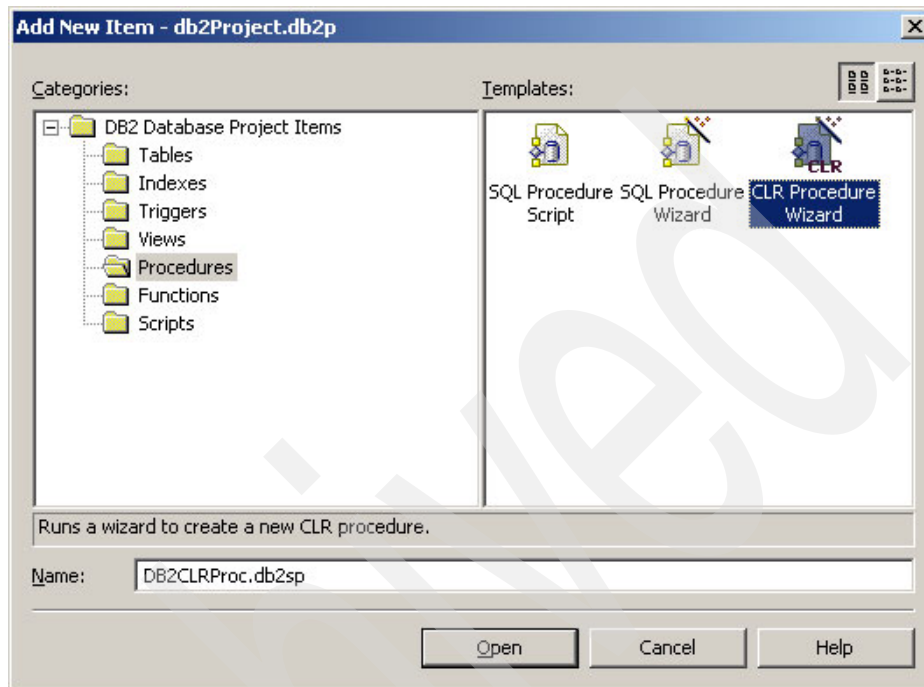


Figure 8-34 Add New Item: db2Project.db2p

3. Click **Open**, which pops up the DB2 CLR Procedure Wizard (Figure 8-35). Click **Next**.

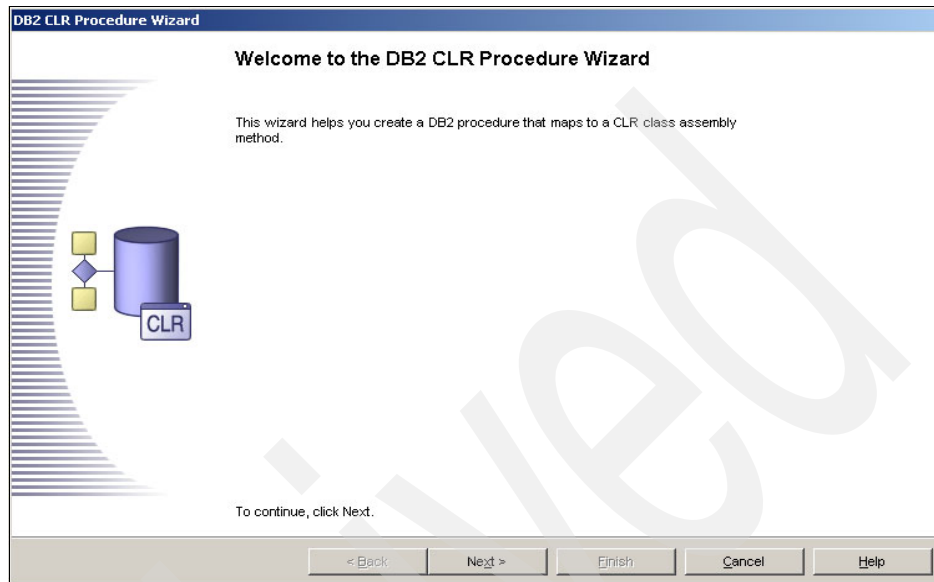


Figure 8-35 DB2 CLR Procedure Wizard

4. In the Available CLR methods pane, select the methods for script generation (Figure 8-36). click **Next**.

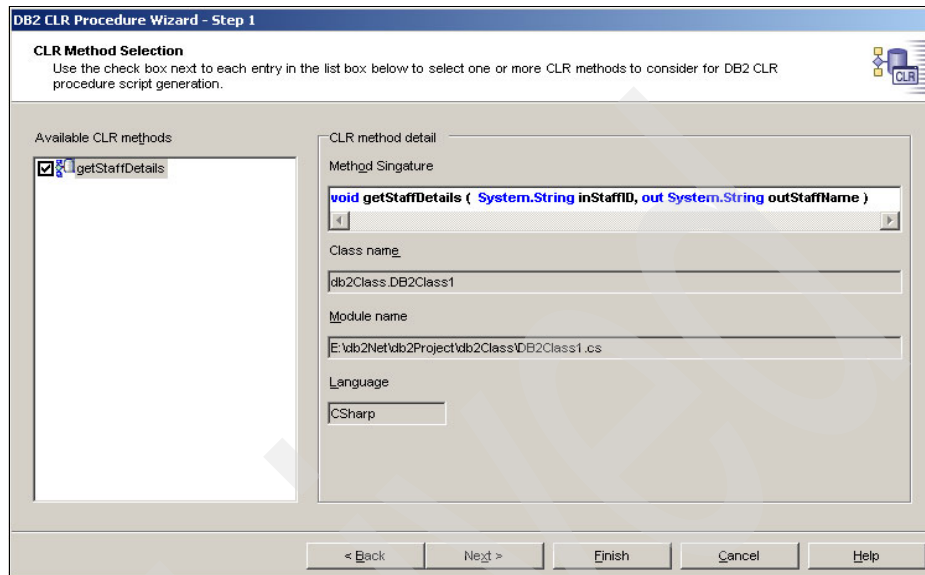


Figure 8-36 CLR Method Selection

5. Optionally, you can specify values for the DB2 CLR procedure (Figure 8-37).

DB2 CLR Procedure Wizard - Step 2

Specifications
You can optionally specify values for various properties of a DB2 CLR procedure including name, specific name, schema, and a brief comment.

Selected CLR methods

- getStaffDetails
 - inStaffID
 - outStaffName

CLR procedure detail

Module name: E:\vb2Net\vb2Project\vb2Class\DB2Class1.cs

Class name: db2Class.DB2Class1

DB2 procedure detail

Procedure name: GETSTAFFDETAILS

Schema name: ADMINISTRATOR

Specific name: GETSTAFFDETAILS

SQL usage: Modifies SQL data

Number of result sets(s): 0

Comment:

< Back Next > Finish Cancel Help

Figure 8-37 CLR Procedure wizard: Specifications

6. The last screen of the wizard shows a summary of the CLR procedure (Figure 8-38).

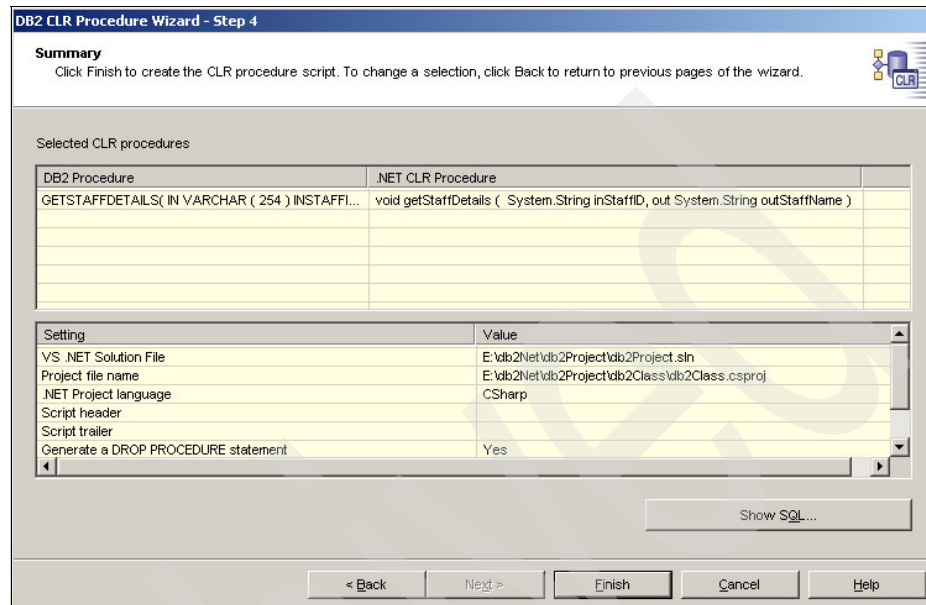


Figure 8-38 DB2 CLR Procedure wizard: Summary

7. To deploy assemblies on the local or remote DB2 server right-click the **db2Project** project and select **Assemblies**.

8. This opens the CLR Assemblies window (Figure 8-39). From the window, select the path for db2Class.dll and click **OK** to build the solution.

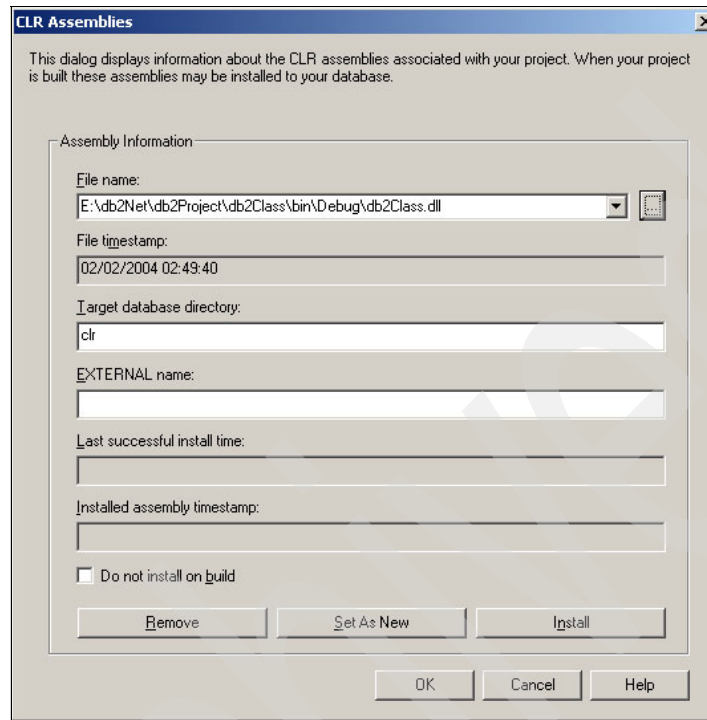


Figure 8-39 CLR Assemblies

You can see the newly created stored procedure by expanding the **Procedures** node in IBM Explorer (Figure 8-40).

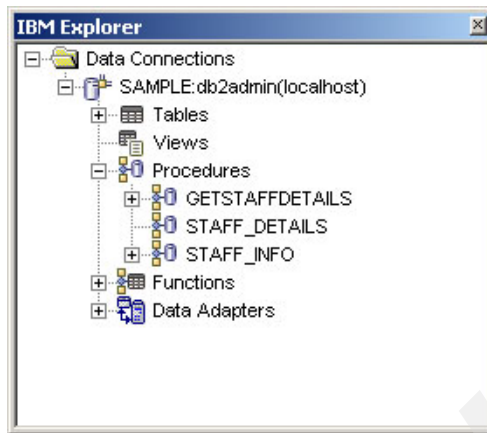


Figure 8-40 Created stored procedure

To run the stored procedure, right-click the **GETSTAFFDETAILS** procedure and select **Run Procedure**. A Parameter Window (Figure 8-41) opens. Enter Staff Id and click **OK** to see the result.

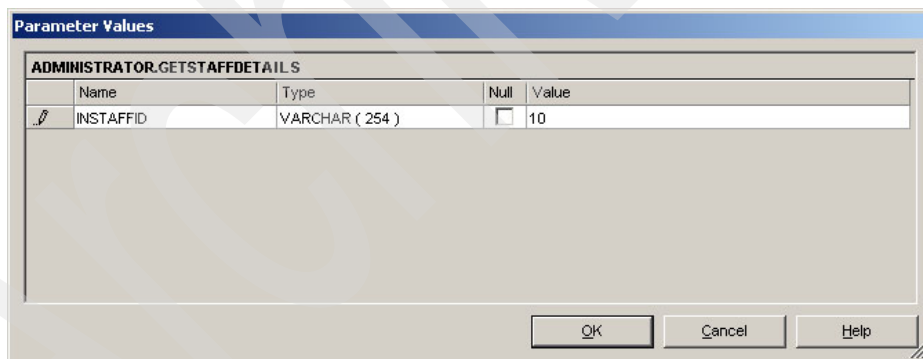


Figure 8-41 Parameter Values window

Consuming DB2 Web services in .NET

DB2 is at the cutting edge of Web services technology for databases. A Web service is an application component (or service) that runs on a Web server and enables client programs to call methods over HTTP. Web services use standard Internet protocols such as HTTP, XML, and SOAP to provide connectivity and interoperability between different platforms or across companies.

Key features of Web services include:

- ▶ Promote interoperability
- ▶ Enable just-in-time integration
- ▶ Reduce complexity through encapsulation

9.1 Web services and DB2

Web services provide a simple interface between the provider and consumer of application resources using a Web Service Description Language (WSDL).

Web Service provider is a service that provides application component interfaces to other applications. *Web service consumer* is an application that uses a Web service.

Figure 9-1 summarizes this and focuses on the role of consumer and provider.

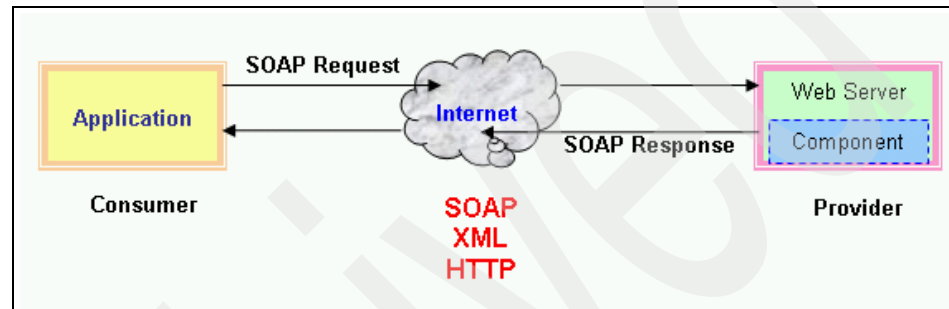


Figure 9-1 Web services: consumer and provider

DB2 UDB V8.2 can act as both a Web service provider and consumer. As a Web service provider, you can create a WSDL interface to DB2 UDB data by using the Web Services Object Runtime Framework (WORF), also known as Document Access Definition Extension (DADX) files. DB2 provides SQL statements or stored procedure data as a Web service using DADX files.

As a Web Service consumer, you can invoke Web services from within Structured Query Language (SQL) statements by invoking a set of user-defined functions (UDFs)

In this chapter, we explain the Web Services Object Runtime Framework and how to create WORF service on Apache Tomcat Server, then how to consume that service in .NET client.

For more information about DB2 as Web Services consumer and WORF, refer to *XML for DB2 Information Integration*, SG24-6994.

9.2 Web Services Object Runtime Framework (WORF)

Web Services Object Runtime Framework (WORF) provides an environment to easily create simple XML-based Web services that access DB2 UDB. Web

services are sets of business functions that applications or other Web services can invoke programmatically over the Internet.

To provide an environment, WORF uses an Apache Simple Object Access Protocol (SOAP) (Version 2.3 or later) engine, or the Apache Axis engine (Version 1.2) and the Document Access Definition Extension (DADX). A DADX document specifies a Web Service using a set of operations that are defined by XML Extender DAD documents or SQL statements. Web services that are specified in a DADX file are called DADX Web services. The WORF runtime environment provides a simple mapping of XML schema to SQL data types.

The WORF framework provides the following features:

- ▶ Resource-based deployment and invocation
- ▶ Automatic service redeployment, when defining resource changes
- ▶ Automatic documentation and test-page generation
- ▶ HTTP GET and POST bindings
- ▶ Automatic WSDL and XSD generation
- ▶ Support for UDDI

9.2.1 Document access definition extension (DADX)

A DADX file specifies how to create a Web service by using a set of operations that are defined by SQL statements or XML Extender DAD documents. The DADX file can contain standard SQL statements, such as SELECT, INSERT, UPDATE, DELETE, and CALL statements, to query and update a database and call stored procedures.

Web services, or functions invoked over the Internet, specified in a DADX file are called DADX Web services, also referred as DB2 Web services or DB2 Information Integrator Web services.

Even with minimal knowledge of XML or SQL, you can create DADX documents by using a simple text editor and tools provided in WebSphere Studio.

DADX files support two kinds of Web service operations:

- ▶ SQL operations

SQL-based querying is the ability to send SQL statements, including stored procedure calls, to DB2 and to return results with a default tagging.

SQL operations query the database by executing SELECT, UPDATE SQL statements. The DADX SQL operation also includes calling stored procedure.
- ▶ XML collection operations (requires DB2 XML Extender)

These storage and retrieval operations help you to map XML document structures to DB2 tables so that you can either compose XML documents

from existing DB2 data, or decompose (store untagged elements or attribute content) XML documents into DB2 data. This method is useful for data interchange applications, particularly when the contents of XML documents are updated frequently.

Table 9-1 summarizes various DADX operations and corresponding elements.

Table 9-1 DADX operations and elements

Operations	Element	Description
SQL operations	<query>	Queries the database
	<update>	Updates the database
	<call>	Calls stored procedures
XML collection operations	<retrieveXML>	Generates XML documents
	<storeXML>	Stores XML documents

Note that you can perform these operations using either the static or dynamic approach. In the static (pre-defined) approach, the query is in the DADX file and only parameters are sent; in the dynamic query approach, the whole SQL statement is sent as a parameter. To process SQL statements at run time, you must enable the dynamic query service (DQS) that is provided by WORF by using a DADX file. In such a case, the DADX file includes only a <DQS/> tag.

With the static approach, you can perform the following operations:

- ▶ SQL Select, Insert, Update, Delete
- ▶ Stored procedure call
- ▶ XML collection operations (requires XML Extender functionality)

With the dynamic approach, you can perform these operations:

- ▶ executeQuery, executeUpdate, execute, executeCall, getTables, getColumns

Figure 9-2 on page 299 shows a simple DADX file that performs an SQL operation explaining the meaning of elements.

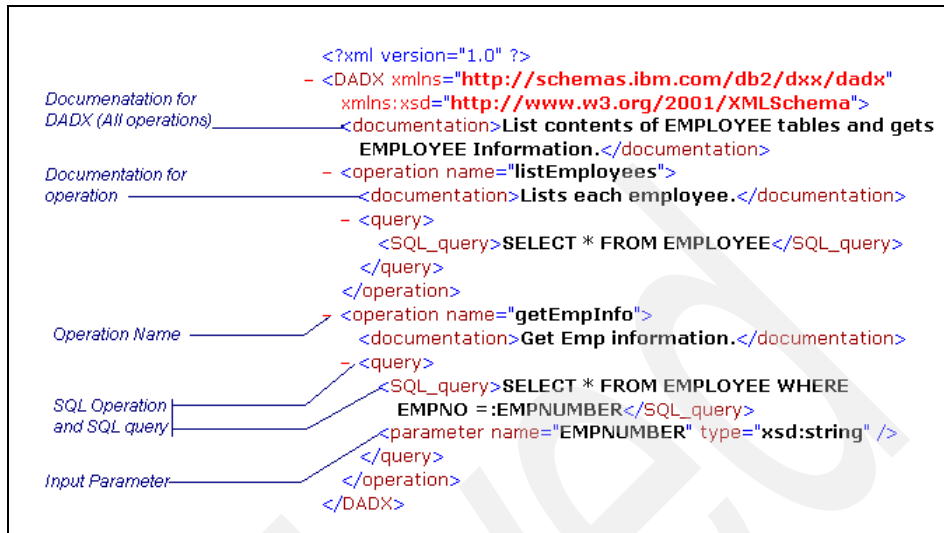


Figure 9-2 DADX file and elements

When you browse this DADX in a browser, the Documentation operations appear as shown in Figure 9-3.

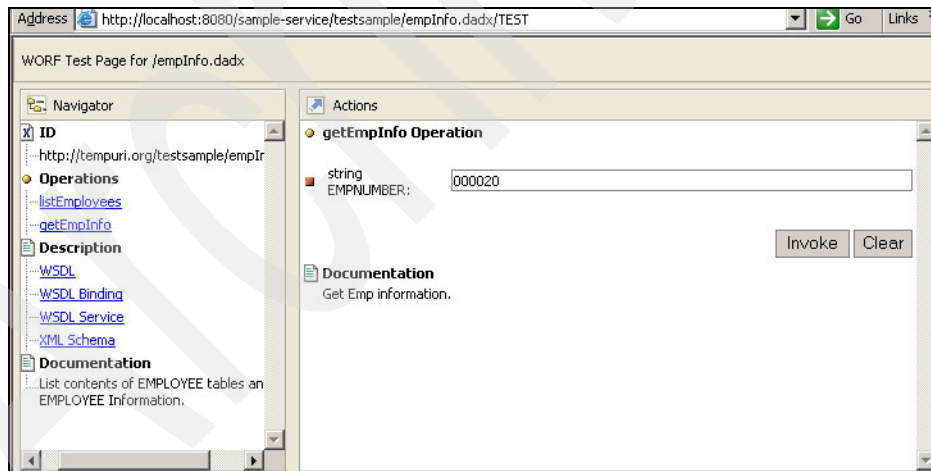


Figure 9-3 browsing the DADX file in a browser

9.2.2 DB2 XML Extender

The DB2 XML Extender is an end-to-end solution for storing and retrieving XML documents. DB2 XML Extender enables XML documents to be stored intact (and, optionally, indexed in side tables) using the XML column access method, or as a collection of relational tables using the XML collection access method.

DB2 XML Extender uses the XML document format called Document Access Definition (DAD) to define the mapping between XML and relational data.

To use XML Extender, ensure that the database administrator has set up any databases that are required for the application, and enable them for use by XML Extender.

For more information about DB2 XML extender, refer to *Integrating XML with DB2 XML Extender and DB2 Text Extender*, SG24-6130.

9.2.3 How WORF processes a Web service request

Figure 9-4 shows how WORF processes a Web service request.

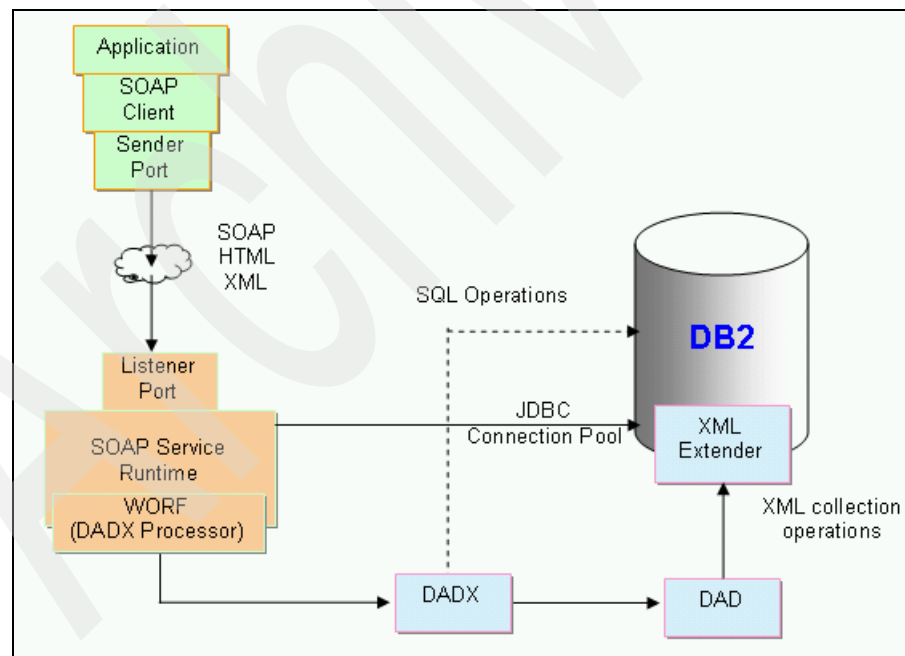


Figure 9-4 WORF processes a Web service request

The application, which can be a Java or .NET client, sends a service request using the SOAP client. The Worf processes the received URL request and checks for a DADX or DTD file and the requested action. The action can be TEST, XSD, or WSDL.

When Worf receives the request, it performs the following steps in response to a Web service request:

1. Load the DADX file specified in the URL or request.
2. Check whether the request is for operation (SQL or XML Collection operation) or command (TEST, XSD, or WSDL).
3. If the request is for operation, check whether it is for SQL or XML Collection operation:
 - a. If the DADX is for SQL operation:
 - i. Replace parameters in the DADX file with requested values.
 - ii. Connect to DB2 and execute any SQL statements or SQL calls specified in DADX file.
 - iii. Format the result into XML.
 - iv. Return the response to the service requestor.
 - b. If the DADX is for XML Collection operation:
 - i. Load a DAD file, if requested.
 - ii. Replace parameters with requested values.
 - iii. Perform an XML collection operation on DB2 using XML Extender.
 - iv. Format the result into XML.
 - v. Return the response to the service requestor.

If the request is for a command, generate the required files, test pages, or other responses and return the response to the service requestor.

9.2.4 Setting up a Worf environment for Windows

Worf Version 8.2 can be run on Apache application server or WebSphere Application Server. In this chapter we show how to run Worf Version 8.2 with Apache application server Version 5.0.

Installing Worf on an Apache Tomcat server

To run Worf service on Apache, the first step is to make sure that all necessary JAR files are present on the machine and added in the class path of the server. Table 9-2 on page 302 shows the JAR files that are required to run Worf, with description and source URL.

Note: Not all listed JAR files are required for running a simple WORF service (mail.jar, qname.jar). Those that are required are:

- ▶ soap.jar
- ▶ xerces.jar (or the JARs of your Java XML parser)
- ▶ mail.jar
- ▶ activation.jar
- ▶ worf.jar
- ▶ db2java.zip (or the JDBC implementation JAR of your database server)

Table 9-2 JAR files

JAR	Description	Source URL
soap.jar for Apache SOAP engine, or axis.jar for Apache axis engine.	Implementation of the Simple Object Access Protocol	http://ws.apache.org/soap/ http://ws.apache.org/axis/
xerces.jar	Java XML parser	http://xml.apache.org/xerces-j/
mail.jar	Java Mail	http://java.sun.com/products/javamail/
activation.jar	JavaBeans Activation Framework	http://java.sun.com/products/javabeans/glasgow/jaf.html http://java.sun.com/products
worf.jar	WORF service	Copy worf.jar from the DB2 installation media.
j2ee.jar, version 1.3 or later	Java 2 Enterprise Edition	http://java.sun.com/j2ee/download.html http://java.sun.com/products
qname.jar	Java API for QName	http://java.sun.com/products
wsdl4j.jar	The Web Services Description Language for Java Toolkit enables creation, representation, and manipulation of WSDL documents describing services.	http://www-124.ibm.com/developerworks/projects/wsdl4j/
jaxrpc.jar	Java API for XML-based RPC	http://ws.apache.org/ or http://java.sun.com/products

JAR	Description	Source URL
log4j-1.2.8.jar	Log API	http://logging.apache.org/log4j/docs/download.html
commons-logging.jar	Common logging API	http://jakarta.apache.org/commons/logging/
commons-logging-api.jar	Simple wrapper API around multiple logging APIs	http://jakarta.apache.org/commons/logging/
commons-discovery.jar	Discovering, or finding, implementations for pluggable interfaces.	http://jakarta.apache.org/commons/discovery/
db2java.zip, or jcc.jar The name of the driver class depends on the driver package that you use. You can modify the driver package that you use in the group.properties file.	JDBC implementation JAR of your database server	The DB2 installation has db2java.zip or jcc.jar files.

After downloading these JAR files, add a line for each file in the *setclasspath.bat* file, which resides in the \bin directory of Apache Tomcat server. For example:

```
set CLASSPATH = %CLASSPATH%;C:\worf\wsdl4j.jar
```

To take the effect of added class paths in *setclasspath.bat*, make sure to start and stop the server with the *startup.bat* or *shutdown.bat*.

9.2.5 Creating and deploying DB2 WORF service

In this section we show you how to create a simple DADX Web service: deployment and testing of a Web service. In particular, we explain:

- How to deploy and test a sample Web application that comes with worf.zip
- How to create a new DADX Web service

Deploying and testing a sample Web application

As discussed earlier, to run a WORF service you must install worf.jar with the proper class path added in the *setclasspath.bat* file.

The dxxworf.jar file comes with the worf.zip file, which also contains a sample Web application with Web services that use WOF. The sample application also has files for exposing and testing the DADX Web services.

To deploy the sample application:

1. Stop the Apache server if it is running.
2. Copy the sample WAR file (services.war or apache-services.war) into the /webapps directory.
3. Restart the server to deploy the sample war file.

Test the installation in a browser with the URL:

`http://localhost:8080/apache-services/LIST`

Note that this URL depends on the WAR file name that you copied in step 2.

The Web application Web Services Listing Page appears as shown in Figure 9-5.

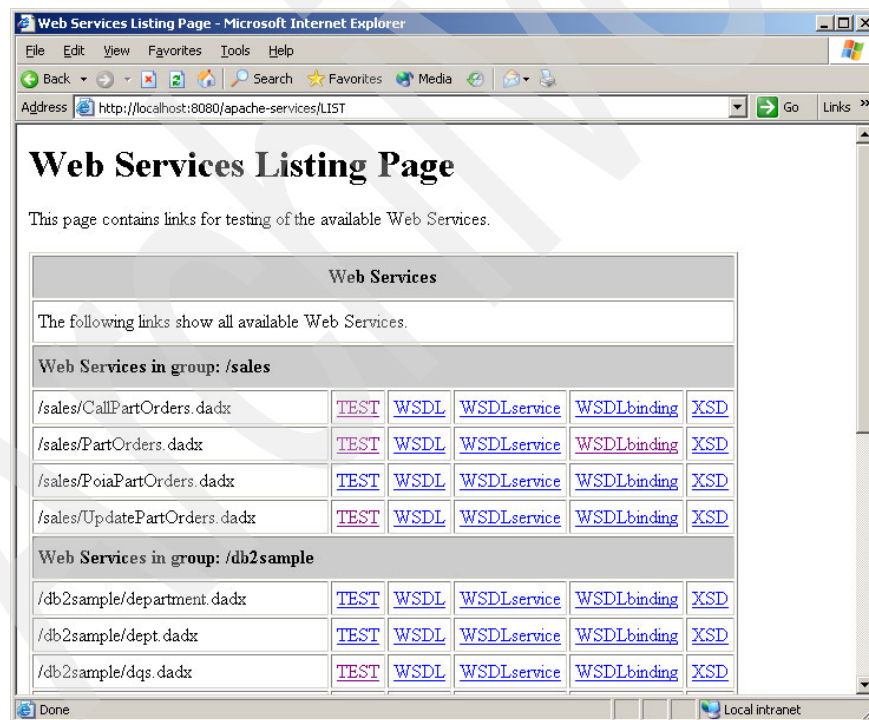


Figure 9-5 Web Services Listing

Note that to work the listed services you must run setup scripts. The scripts for setting up the databases are part of the samples. For more information, see the related notes in the DB2 documentation.

Creating a new DADX Web service

In this section, we discuss how to create a new DADX Web service on Apache server. To test and expose the Web service, we copy the necessary files from the sample application discussed in previous section.

Creating a directory structure and defining a group of Web services

The DADX Web service requires a group. The groups are stored in the directory WEB-INF/classes/groups; /WEB-INF is the directory used by J2EE Web applications to store resources that are not directly available to HTTP requests. This means that users cannot see the contents of your DADX files.

To create a directory structure and define a group of Web services:

1. Open the **webapps** folder under Apache Tomcat Web server.
2. Create the \sample-service\WEB-INF\classes\groups\test_sample directory, as shown in Figure 9-6 on page 306. This groups folder stores all Web services. In our example, we created a test_sample group directory as shown.
3. Create a lib folder to store custom JAR files.
4. Copy the **worf** folder from the \sample-service folder. The worf folder has the necessary interfaces to expose and test the DADX Web service.

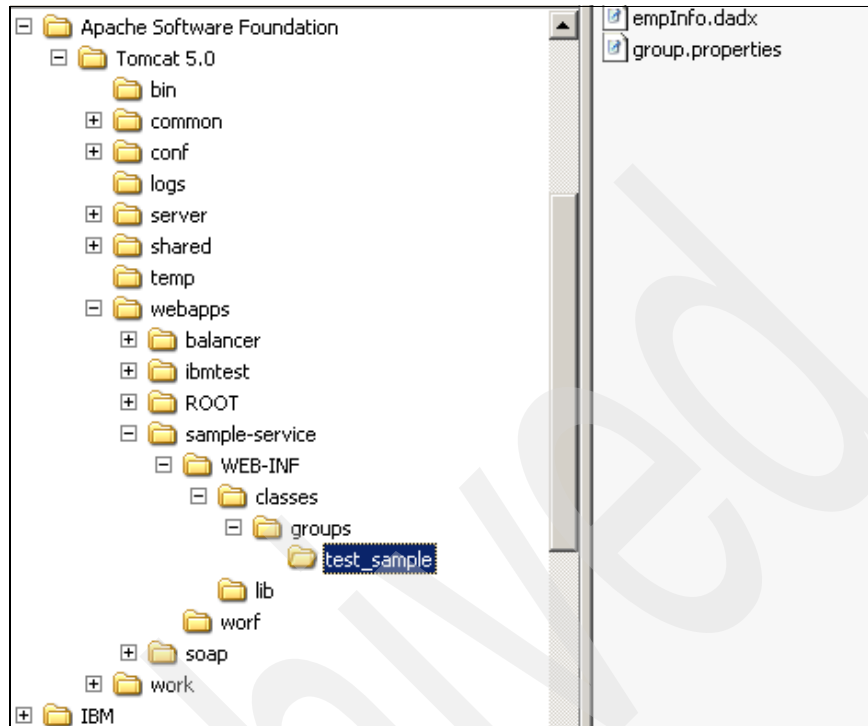


Figure 9-6 Web directory

Defining web.xml

Create a web.xml file to define group names, and store it in the WEB-INF folder. You can have multiple group names in the same web.xml file. The code in Example 9-1 shows the web.xml file created for test_sample.

Example 9-1 web.xml sample

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app
  PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
  "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <servlet>
    <servlet-name>list</servlet-name>
    <display-name>list</display-name>

    <servlet-class>com.ibm.etools.webservice.rt.list.servlet.ListInvoker</servlet-
    class>

    <load-on-startup>-1</load-on-startup>
  </servlet>
```



```

    <servlet>
      <servlet-name>wsil</servlet-name>
      <display-name>wsil</display-name>

<servlet-class>com.ibm.etools.webservice.rt.wsil.servlet.WSILInvoker</servlet-c
lass>
      <load-on-startup>-1</load-on-startup>
    </servlet>
    <servlet>
      <servlet-name>test_sample</servlet-name>

<servlet-class>com.ibm.etools.webservice.rt.dxx.servlet.DxxInvoker</servlet-cla
ss>
      <init-param>
        <param-name>faultListener</param-name>
        <param-value>org.apache.soap.server.DOMFaultListener</param-value>
      </init-param>
      <load-on-startup>-1</load-on-startup>
    </servlet>

    <servlet-mapping>
      <servlet-name>test_sample</servlet-name>
      <url-pattern>/testsample/*</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
      <servlet-name>list</servlet-name>
      <url-pattern>/LIST</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
      <servlet-name>wsil</servlet-name>
      <url-pattern>/inspection.wsil</url-pattern>
    </servlet-mapping>
    <welcome-file-list>
      <welcome-file>index.html</welcome-file>
    </welcome-file-list>
  </web-app>

```

Creating a group.properties file

The group.properties file is a standard Java properties file that contains database connection information for each group of DADX Web services.

The group.properties file for our test_sample example consists of:

```

dbDriver=COM.ibm.db2.jdbc.app.DB2Driver
dbURL=jdbc:db2:sample
userID=administrator
password=pass2jit
namespaceTable=namespacetable.nst

```

```

autoReload=true
reloadIntervalSeconds=5
groupNamespaceUri=http://schemas.ibm.com/sample
enableXmlClob=true
useDocumentStyle=true

```

Table 9-3 describes the properties in the group.properties file.

Table 9-3 Properties in group.properties file

Property	Required	Description
initialContextFactory	Optional	Defines a DataSource for the database connection and enables connection pooling and distributed transactions. Used with data source JNDI property.
datasourceJNDI	Optional	Specifies the JNDI name of the DataSource for the database. Used with initialContextFactory. Not used with Jakarta Tomcat.
dbDriver	Optional	Specifies the Java class name of the JDBC driver for connecting to the database.
dbURL	Optional	Specifies the JDBC URL of the database.
userID	Required	User ID for the database.
password	Required	Password for the database.
namespaceTable	Optional	Specifies the resource name of the namespace table. References a Namespace Table (NST) resource that defines the mapping from DB2 XML Extender DTDIDs to XML Schema (XSD) name spaces and locations.
autoReload	Optional	Used with reloadIntervalSeconds. Specifies whether to reload a resource.
reloadIntervalSeconds	Optional	Used with autoReload. Controls resource loading and caching. Specifies the integer automatic reloading time interval in seconds.
useDocumentStyle	Optional	The default value is false, which means that the Web services at run time use RPC encoding. If you set this value to true, then the Web services at run time use document style and literal encoding.

Note:

- ▶ When using group.properties for connection pooling:
 - Define group.properties using the parameter initialContextFactory with datasourceJNDI.
- ▶ When using group.properties for regular JDBC database connections:
 - Define group.properties using the parameter dbURL with dbDriver.
- ▶ If you define both types of connections, the application tries the DataSource first. If WOF cannot obtain the DataSource, then it tries the JDBC. The complete set of properties is listed below.
- ▶ If you are using the Microsoft .NET client for accessing the DADX Web service, set useDocumentStyle = true.

Create and store this group.properties file in the Apache Tomcat server directory WEB-INF/classes/groups/test_sample.

Creating a DADX file

A document access definition extension (DADX) file specifies how to create a Web service by using a set of operations that are defined by SQL statements or XML Extender Document Access Definition (DAD) documents.

In our example, we perform two SQL operations to get EMPLOYEE information. The first operation, listEmployees, lists information about all employees. The second operation, getEmpInfo, takes the employee number as its input parameter and returns employee information for that number.

Declaring and referencing parameters in the DADX file

To pass SQL parameters, use a colon prefix as shown:

```
<SQL_query>SELECT * FROM EMPLOYEE WHERE EMPNO =:EMPNUMBER</SQL_query>
```

To define a parameter, use the <parameter> element as shown:

```
<parameter name="EMPNUMBER" type="xsd:string"/>
```

Table 9-4 shows supported SQL and XML schema types that are required for declaring and referencing parameters in the DADX file.

Table 9-4 SQL and XML schema type

SQL type	XML schema simple type
CHAR, VARCHAR	string
DECIMAL, NUMERIC	decimal
INTEGER, SMALLINT	int

SQL type	XML schema simple type
FLOAT	float
REAL, DOUBLE PRECISION	double
DATE	date
TIME	time
TIMESTAMP	string

Example 9-2 shows the empInfo.dadx file. Create and copy this file into the Apache Tomcat server directory WEB-INF/classes/groups/test_sample.

Example 9-2 empInfo.dadx

```

<?xml version="1.0"?>
<DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <documentation>List contents of EMPLOYEE tables and gets EMPLOYEE
Information.</documentation>
  <operation name="listEmployees">
    <documentation>Lists each employee.</documentation>
    <query>
      <SQL_query>SELECT * FROM EMPLOYEE</SQL_query>
    </query>
  </operation>
  <operation name="getEmpInfo">
    <documentation>Get Emp information.</documentation>
    <query>
      <SQL_query>SELECT * FROM EMPLOYEE WHERE EMPNO =:EMPNUMBER</SQL_query>
    </query>
    <parameter name="EMPNUMBER" type="xsd:string"/>
  </operation>
</DADX>

```

Testing the empInfo.dadx file

The worf folder that was copied from the DB2 sample application has files that are necessary for testing DADX Web service.

After performing all of the steps described so far in this section, restart the Web server and browse to the following URL to test the created DADX Web service:

<http://localhost:8080/sample-service/testsample/empInfo.dadx/TEST>

Figure 9-7 on page 311 shows empInfo.dadx with the TEST command.

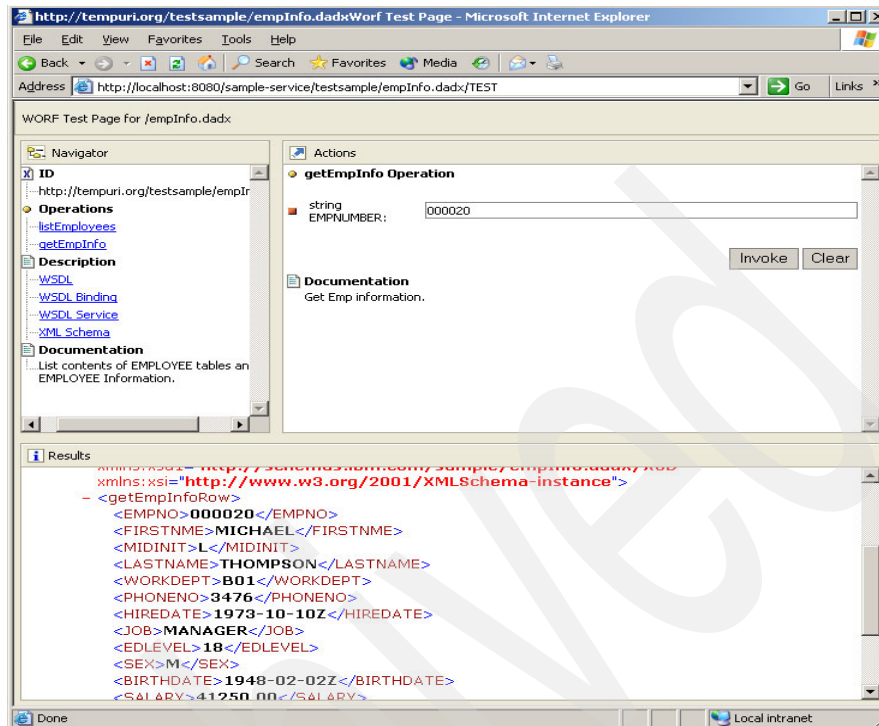


Figure 9-7 *empInfo.dadx* with TEST command

Generating WSDL from the DADX file

The DADX document contains the information that is required to implement the Web Service as well as the information required to generate the WSDL document that describes the Web service.

WSDL is used to describe the interface of business services. It is used when publishing services to a UDDI registry.

To generate the WSDL, submit the following URL:

`http://localhost:8080/sample-service/testsample/empInfo.dadx/WSDL`

WORF dynamically generates the WSDL document. You can publish this in UDDI or some other Web service directory.

Figure 9-8 on page 312 shows a snippet of WSDL that was generated for the *empInfo.dadx* Web service.

```

<?xml version="1.0" encoding="UTF-8" ?>
- <definitions targetNamespace="http://tempuri.org/sample-
  service/testsample/empInfo.dadx/WSDL" xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://tempuri.org/sample-service/testsample/empInfo.dadx/WSDL"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsd1="http://tempuri.org/testsample/empInfo.dadx">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">List contents of
    EMPLOYEE tables and gets EMPLOYEE Information.</wsdl:documentation>
- <types>
- <schema elementFormDefault="qualified"
  targetNamespace="http://tempuri.org/testsample/empInfo.dadx"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://tempuri.org/testsample/empInfo.dadx">
- <element name="listEmployeesResult">
- <complexType>
- <sequence>
- <element maxOccurs="unbounded" minOccurs="0"
  name="listEmployeesRow">
- <complexType>
- <sequence>
  <element name="EMPNO" type="string" />
  <element name="FIRSTNAME" type="string" />
  <element name="MIDINIT" type="string" />
  <element name="LASTNAME" type="string" />
  <element name="WORKDEPT" nillable="true" type="string" />
  <element name="PHONENO" nillable="true" type="string" />
  <element name="HIREDATE" nillable="true" type="date" />
  <element name="JOB" nillable="true" type="string" />
  <element name="EDLEVEL" type="short" />
  <element name="SEX" nillable="true" type="string" />

```

Figure 9-8 WSDL sample

Performing XML collection operations

You can generate or store XML documents with the <retrieveXML> or <storeXML> operations. These operations call XML Extender stored procedures and require a DAD file or an XML collection reference. The stored procedures generate or store XML documents using the mapping in a DAD file, or by referring to an enabled XML collection.

Example 9-3 shows a more complex DADX file that generates an XML document. It references a stored procedure using the <retrieveXML> element. The <DAD_ref> element specifies the name of a DAD file.

Example 9-3 DADX

```

<?xml version="1.0"?>
<DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns="http://www.w3.org/1999/xhtml">
    Provides queries for part order information at myco.com.
    See

```

```

    <a href="../../documentation/PartOrders.html"
tar-get="_top">PartOrders.html</a>
    for more information.
</wsdl:documentation>
<operation name="findAll">
    <wsdl:documentation xmlns="http://schemas.xmlsoap.org/wsdl/">
        Returns all the orders with their complete details.
    </wsdl:documentation>
    <retrieveXML>
        <DAD_ref>getstart_xcollection.dad</DAD_ref>
        <no_override>
    </retrieveXML>
    </operation>
</DADX>

```

The stored procedure refers to the `getstart_xcollection.dad` file to determine which tables to use when generating the XML documents, and the XML document structure.

For more information about XML collection operations, refer to *XML for DB2 Information Integration*, SG24-6994.

9.3 Consuming the DADX Web service using .NET client

In the last section, we saw how to create a simple DADX file for performing an SQL operation. In this section, we show how to consume that Web service in .NET client.

For demonstration purposes, we create a Console application and then access the `getEmpInfo` Web method.

1. Open a new Console application in C# project and name it `worfcClient`. Rename the default `Class1` to `wClient` in the Project Explorer window.
2. To add a Web reference, right-click **Project name** in Project Explorer and select **Add Web Reference**. An Add Web Reference window appears.

3. Type the URL to get WSDL for empInfo DADX Web service and click **Go**. The Add Web Reference window retrieves information for all methods, as shown in Figure 9-9.

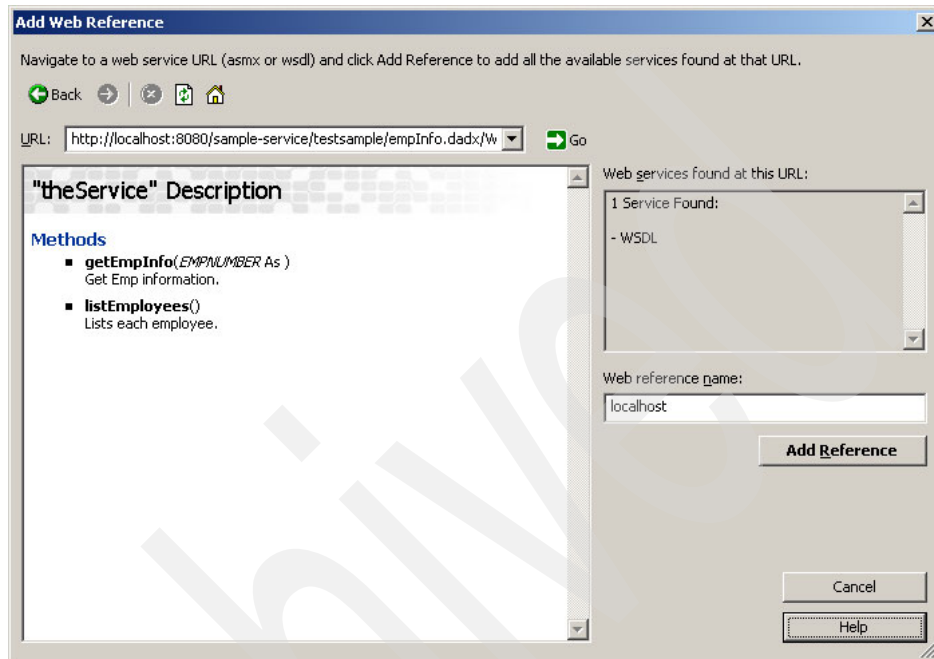


Figure 9-9 Add Web Reference window

4. Click **Add Reference** to add a Web reference in Solution Explorer (Figure 9-10).

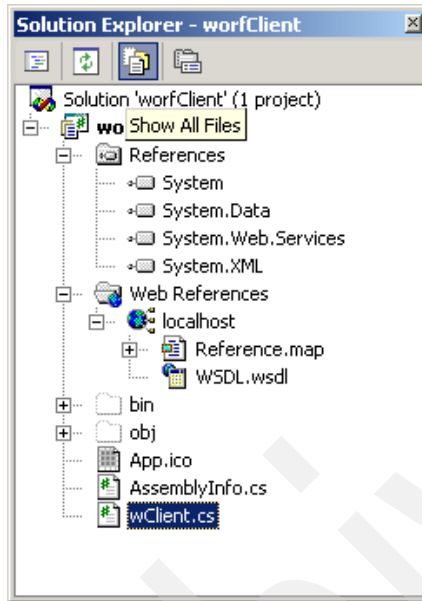


Figure 9-10 Solution Explorer

5. Add the following code in the Main method of the wClient class.

```
static void Main(string[] args){
    localhost.getEmpInfoResponse EmpInfoResponse;
    localhost.getEmpInfoResultGetEmpInfoRow[] EmpInfoRow;
    localhost.theService proxy = new localhost.theService();
    localhost.getEmpInfo empInfo= new localhost.getEmpInfo();
    empInfo.EMPNUMBER="000020";
    EmpInfoResponse = proxy.getEmpInfo(empInfo);
    EmpInfoRow = EmpInfoResponse.@return.getEmpInfoResult;
    for(int i=0; i < EmpInfoRow.Length; i ++){
        Console.WriteLine (EmpInfoRow[i].FIRSTNAME + " " +
        EmpInfoRow[i].LASTNAME + " " + EmpInfoRow[i].SALARY);
    }
}
```

This code sends the employee number 000020 to the getEmplInfo Web service and displays the employee's First Name, Last Name, and Salary on the console. An output of the code appears on the console, as shown in Figure 9-11.

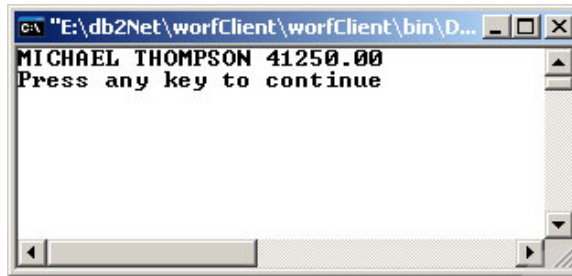


Figure 9-11 Net Console output

Problem resolution

In this chapter, we introduce the most frequently used IBM acronyms in the literature when referring to problem resolution:

RAS

Reliability, availability, and serviceability

PD/PSI

Problem determination / problem source identification

We also introduce the enhanced DB2 diagnostic log and new tools for examining this log.

10.1 RAS

Reliability, availability, and serviceability are attributes of application system design, program design, and coding that assure that the final product will be available when needed; that it will reliably perform the functions it was designed for; and that if problems do occur, they will be as non-disruptive as possible, they can be repaired as quickly as possible, and the user operation can and will be resumed with a minimum of down time.

It is expected that these attributes will stay the same over the entire product life irrespective of what changes, updates, and enhancements are added.

Customers expect that the software product they acquired will be capable of performing its intended functions under stated conditions for a stated period of time.

A software vendor expects product that is coded in such way that it is possible to quickly determine the cause and the solution to a problem or error that affects its operation.

This noble goal is achieved by using techniques for preventing fault occurrence (such as defensive programming), minimizing the effects of faults when they occur, reducing the time for repair, and enabling the customer to resolve problems without extensive assistance from the software vendor.

10.2 PD/PSI

Problem determination (PD) is a task performed by either the customer or software vendor to determine whether a fault is caused by hardware error, software error, or human (operator or user) error.

Problem source identification (PSI) is the set of activities that are performed after problem determination to determine the specific failure of the hardware or software component.

PSI helps determine appropriate corrective action to be taken to alleviate or resolve the problem.

10.3 Information to collect

A detailed and precise problem description is the key to resolving a problem. For example, which sentence describes problem better:

- ▶ I have some problems with my application.
- ▶ When my application fetches rows from a table, it gets an error code of -180.

The first step in problem analysis is to completely describe the problem. Without a problem description, you will not know where to start investigating the cause of the problem. This step includes asking such basic questions as:

- ▶ 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?

Collecting information in an orderly, organized, and comprehensive fashion will significantly speed up the time it takes to find the source of the problem and its resolution. You need to:

- ▶ Describe your environment: exact hardware and software level of all participating servers (both DB2 and non-DB2), such as application servers.
- ▶ Describe the history of events leading to this problem: is it re-creatable or intermittent?
- ▶ Describe what sequence of events led to this occurrence of the problem.

You should be prepared to answer a question that appears to be very simple, but may be difficult to answer: What has changed (in your environment, both hardware and software) since the last known good execution of the same application in similar conditions operating on similar data?

If you are facing an intermittent problem that you cannot re-create, the following list may be helpful. Collect all of this data from the re-creatable case that demonstrates the problem:

- ▶ Data Base Manager configuration
- ▶ Database configuration
- ▶ SQL Code / State (*always* include a reason code)
- ▶ Precise and concise problem description
- ▶ DB2DIAG.LOG

Note: DB2DIAG.LOG is the most important piece of data and provides excellent diagnostic information. Almost 90% of known problems can be resolved by examining contents of this log.

- Any dump files and trap files in the DB2 instance home directory (C:\Program Files\IBM\SQLLIB\DB2 if you accepted the default location during installation).

10.4 db2support utility

The db2support utility is designed to automatically collect all DB2 and system diagnostic data. This program generates information about a DB2 server, including information about its configuration and system environment.

The output of this program is stored in one compressed file named db2support.zip, located in the directory specified as part of the command invoked at the command line.

In one simple step, the tool can gather database manager snapshots, configuration files, and operating system parameters, which should make problem determination quicker. A sample call of the utility:

```
db2support . -d db2_emp -c
```

The dot represents the current directory where the output file is stored. The rest of the command is optional. -d and -c instructs the utility to connect to the db2_emp database, and to gather information about database objects such as table spaces, tables, or packages.

10.5 Understanding DB2DIAG.LOG

The DB2 diagnostic log (DB2DIAG.LOG) is a primary tool to communicate state and actions taken between the DB2 management system (engine and other components) and a human being: end user, application developer, or database administrator.

The level of detail of information that is contained in the DB2 diagnostic log is controlled by the DIAGLEVEL instance configuration parameter. DIAGLEVEL may have values from 0 (zero) to 4, with 3 being the default.

Values of DIAGLEVEL denote the increasing level of details that are logged in the diagnostic log file:

- 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 location of the collected diagnostic information is controlled by yet another instance-level configuration parameter, namely DIAGPATH.

This configuration parameter specifies the directory that will contain the error file, event log file (on Windows NT® only), alert log file, and any dump files that might be generated, based on the value of the DIAGLEVEL parameter.

You will probably use most of the time for new implementation, database migration, development, and QA systems diagnostic information collected at diagnostic level 3 (the default value).

At times you may be advised by DB2 Support Group to change the value of the DIAGLEVEL parameter to 4 and restart the instance with a new DB2DIAG.LOG file.

Important information to be collected is the description of the environment: the level of DB2 code installed, operating system under which the DB2 UDB system is executing, amount of memory installed/available, and so forth.

Previously, you would have had to run the `db2diaglevel` command as well as OS-specific commands to collect needed information piece by piece. With DB2 UDB V8.2, this task was made much easier. Each time DB2 creates a diagnostic log, the first entry inserted is the environment description (Example 10-1). We will use this entry to describe key fields of a diagnostic log entry.

Tip: The diagnostic log file size grows with time, so it is common practice among DB2 users and DBAs to archive diagnostic logs periodically. One way to do so is to have a scheduled task run every day at midnight to rename the DB2DIAG.LOG file to DB2DIAG.LOG.yyyy-mm-dd (where yyyy, mm and dd denote year, month and day respectively).

Example 10-1 Environment description entry in DB2DIAG.LOG

```
2004-05-04-12.47.44.343000-420 I1H849          LEVEL: Event
PID      : 3752                      TID   : 3760          PROC  : db2syscs.exe
INSTANCE: DB2                      NODE   : 000
FUNCTION: DB2 UDB, RAS/PD component, _pdlogInt, probe:120
START    : New db2diag.log file
DATA #1 : Build Level, 124 bytes
Instance "DB2" uses "32" bits and DB2 code release "SQL08020"
with level identifier "03010106".
Informational tokens are "DB2 v8.1.7.342", "s040429", "WR21326", FixPak "7".
DATA #2 : System Info, 1304 bytes
System: WIN32_NT POLONIUM 5.2 x86 Family 6, model 11, stepping 1
CPU: total:2 online:2
Physical Memory: total:3776 free:3446 available:1983
Virtual Memory: total:5667 free:5470
```

Swap Memory: total:1891 free:2024
Information in this record is only valid at the time when this file was
created (see this record's time stamp)

To explain each of the fields in this diagnostic log entry:

The first field highlighted in the example is a timestamp of when this entry was taken.

```
2004-05-04-12.47.44.343000-420 I1H849          LEVEL: Event
PID      : 3752                      TID   : 3760          PROC  : db2syscs.exe
INSTANCE: DB2                      NODE  : 000
FUNCTION: DB2 UDB, RAS/PD component, _pdlogInt, probe:120
START    : New db2diag.log file
DATA #1  : Build Level, 124 bytes
```

The next field is a time zone, denoted as the difference in minutes from Universal Time Coordinated (UTC). We are in Pacific Standard Time, hence seven hours behind (-420 minutes) UTC.

```
2004-05-04-12.47.44.343000-420 I1H849          LEVEL: Event
PID      : 3752                      TID   : 3760          PROC  : db2syscs.exe
INSTANCE: DB2                      NODE  : 000
FUNCTION: DB2 UDB, RAS/PD component, _pdlogInt, probe:120
START    : New db2diag.log file
DATA #1  : Build Level, 124 bytes
```

The next field is a record ID that identifies this particular diagnostic log entry.

```
2004-05-04-12.47.44.343000-420 I1H849          LEVEL: Event
PID      : 3752                      TID   : 3760          PROC  : db2syscs.exe
INSTANCE: DB2                      NODE  : 000
FUNCTION: DB2 UDB, RAS/PD component, _pdlogInt, probe:120
START    : New db2diag.log file
DATA #1  : Build Level, 124 bytes
```

The next field is a severity level of the logged event. It may have following values:

- ▶ Info
- ▶ Event
- ▶ Warning
- ▶ Error
- ▶ Severe

```
2004-05-04-12.47.44.343000-420 I1H849          LEVEL: Event
PID      : 3752                      TID   : 3760          PROC  : db2syscs.exe
INSTANCE: DB2                      NODE  : 000
FUNCTION: DB2 UDB, RAS/PD component, _pdlogInt, probe:120
START    : New db2diag.log file
DATA #1  : Build Level, 124 bytes
```


The next three fields describe the Process ID (PID), Thread ID (TID), and Process Name (PROC) of the process that triggered this diagnostic log entry.

```
2004-05-04-12.47.44.343000-420 I1H849          LEVEL: Event
PID      : 3752                TID   : 3760        PROC  : db2syscs.exe
INSTANCE: DB2                NODE : 000
FUNCTION: DB2 UDB, RAS/PD component, _pdlogInt, probe:120
START    : New db2diag.log file
DATA #1 : Build Level, 124 bytes
```

The next two fields describe the Instance Name (INSTANCE) and Node number (NODE) where this diagnostic log resides.

```
2004-05-04-12.47.44.343000-420 I1H849          LEVEL: Event
PID      : 3752                TID   : 3760        PROC  : db2syscs.exe
INSTANCE: DB2                NODE   : 000
FUNCTION: DB2 UDB, RAS/PD component, _pdlogInt, probe:120
START    : New db2diag.log file
DATA #1 : Build Level, 124 bytes
```

The next lines are regard a specific diagnostic log entry. They describe function logging (FUNCTION), message being logged (START), actual text of message being logged (in this case, confirmation that we are starting with a new DB2DIAG.LOG file). There may be a variable number of description lines depending on the nature and severity of the event that is being logged.

```
2004-05-04-12.47.44.343000-420 I1H849          LEVEL: Event
PID      : 3752                TID   : 3760        PROC  : db2syscs.exe
INSTANCE: DB2                NODE   : 000
FUNCTION: DB2 UDB, RAS/PD component, _pdlogInt, probe:120
START    : New db2diag.log file
DATA #1 : Build Level, 124 bytes
```

Some of the diagnostic log entries have arguments and data associated with the logged function. This example has only one set of data, namely enhanced in DB2 UDB V8.2 build level information.

```
2004-05-04-12.47.44.343000-420 I1H849          LEVEL: Event
PID      : 3752                TID   : 3760        PROC  : db2syscs.exe
INSTANCE: DB2                NODE   : 000
FUNCTION: DB2 UDB, RAS/PD component, _pdlogInt, probe:120
START    : New db2diag.log file
DATA #1 : Build Level, 124 bytes
Instance "DB2" uses "32" bits and DB2 code release "SQL08020"
with level identifier "03010106".
Informational tokens are "DB2 v8.1.7.342", "s040429", "WR21326", FixPak
"7".
DATA #2 : System Info, 1304 bytes
System: WIN32_NT POLONIUM 5.2 x86 Family 6, model 11, stepping 1
CPU: total:2 online:2
```

```
Physical Memory: total:3776 free:3446 available:1983
Virtual Memory: total:5667 free:5470
Swap Memory: total:1891 free:2024
Information in this record is only valid at the time when this file was
created (see this record's time stamp)
```

One of the additional enhancements in DB2 UDB Version 8.2 is logging external events that may influence DB2 UDB behavior. Example 10-2 shows an entry in DB2 Global Registry that was added with the DB2 Registry variable DB2COMM set to value TCPIP.

Example 10-2 Example of logging environment changes

2004-05-04-12.48.38.625000-420	I1472H296	LEVEL: Event
PID : 3868	TID : 3872	PROC : db2set.exe
INSTANCE: DB2	NODE : 000	
FUNCTION: DB2 UDB, oper system services, db2set_main, probe:40		
CHANGE : CFG DB2SET: DB2COMM:[g] From: "" To: "TCPIP"		

10.6 DB2DIAG tool

DB2 UDB V8.2 includes a new tool called *DB2DIAG.EXE*, which can be used for filtering and formatting entries in the DB2 UDB diagnostic log.

Issuing the **db2diag -h** command outputs a short description of the command (Example 10-3).

Example 10-3 DB2DIAG log analysis tool

db2diag - The db2diag.log Analysis Tool	
db2diag is a tool for filtering and formatting the db2diag.log file	
Command syntax:	
<pre> .----- .----- v v >>--db2diag--+-+-----+-----+-----+--< --option-- --filename-- </pre>	
Command parameters:	
filename	- one or more space-separated path names of diagnostic logs
-help , -h , ?	- help information. To get help on help, try "db2diag -h h"
-filter , -g	- case-sensitive search for a list of field-pattern pairs
-gi	- case-insensitive search for a list of field-pattern pairs
-gv	- case-sensitive invert matching

-gvi	, -giv	- case-insensitive invert matching
-invert	, -v	- invert the sense of matching for all filtering options
-exist		- record field must exist in order to be processed
-pid		- find all records for a list of process IDs
-tid		- find all records for a list of thread IDs
-node	, -n	- find all records for a list of nodes
-error	, -e	- find all records for a list of errors
-level	, -l	- find all records for a list of severity levels
-history	, -H	- display the history of logged records for a time interval
-time	, -t	- display all the records within a particular time interval
-count	, -c	- display a count of matching records
-verbose	, -V	- display all record fields whether they contain data or not
-strict		- display records using one "field: value" pair per line
-cbe		- display records in the Common Base Event (CBE) format
-fmt		- format tool's output using a format string
-output	, -o	- save output into a file
-follow	, -f	- continuously display appended records as the file grows
-archive	, -A	- archive a diagnostic log file
-rc		- display descriptions of DB2 error return codes, ZRC or ECF

You may use additional options to further enhance your filtering and formatting of the DBb2 UDB V8.2 diagnostic log:

- ▶ **db2diag -h <option1[,option2[,option3...]]>**
Displays additional help and usage examples for one or more options specified in the options list
- ▶ **db2diag -h brief**
Displays help for all options without examples
- ▶ **db2diag -h examples**
Displays a few typical examples to get started
- ▶ **db2diag -h tutorial**
Displays more advanced examples covering all features
- ▶ **db2diag -h notes**
Displays usage notes and restrictions that apply
- ▶ **db2diag -h all**
Displays help in the most complete form with detailed information about all options and usage examples

If you are not using the Data Partitioning Feature (DPF) of DB2 UDB V8.2, you should set DB2DIAGLEVEL to 4 whenever you are trying to re-create a problem. Make sure that for each re-creation scenario, you start with fresh copy of the

diagnostic log. Simply rename the existing DB2DIAG.LOG and DB2 will create a new copy of the log with environmental information as the first entry in the log.

We recommend that you rename the older log using the date and time of the last entry as the part of the diagnostic log name, for example:

```
ren db2diag.log db2diag.log.040516.2150
```

When using the DPF feature (you are using multiple nodes of DB2), set DIAGLEVEL to 3 because diagnostic information collected at level 4 in a multi-node environment may quickly become too big to effectively use it.

Remember that diagnostic information collected at level 4 has much more information than the preceding levels, so DB2 will run slower, but this slowdown mainly occurs at DB2 start processing, during initial connection to the database, and during error processing.

10.7 Prevention versus resolution

Preventing an occurrence of a problem may be a more effective way to resolve the problem than actual resolution.

When starting Control Center, pay attention to whether any alerts are present. Figure 10-1 on page 327 shows two Health Center alerts.

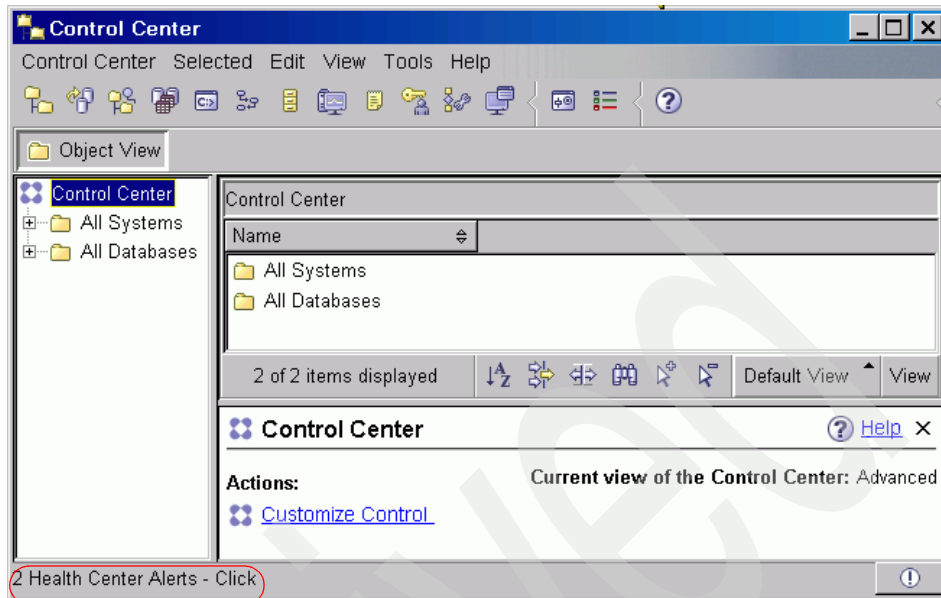



Figure 10-1 Control Center with two Health Center Alerts

Open the Health Center for details about the alerts. You can configure the Health Center to notify you by e-mail or page or to take specific actions by running a script when a health indicator enters an alert state.

By clicking the health beacon , you launch the Health Center and you can investigate (and resolve) the alert. If you wish to be notified by a pop-up window, as shown in Figure 10-2 on page 328, you have to enable this feature on the Health Center status beacon page in the Control Center tools settings.

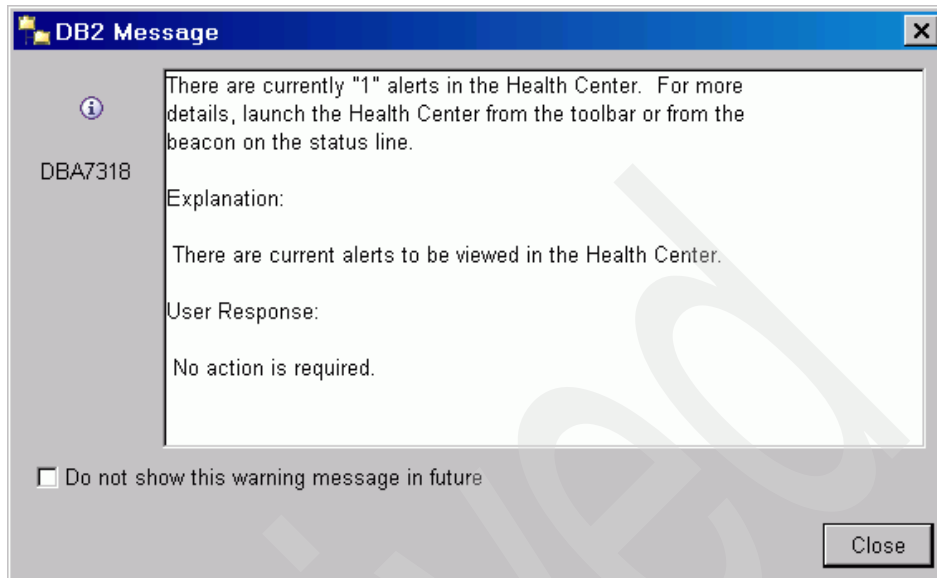


Figure 10-2 DB2 pop-up message informing that there are alerts in Health Center

DB2 UDB is fully integrated with Windows, so whenever an entry is made in the DB2 diagnostic log, an entry is also made in the Event log. This can be viewed with the Windows Event Log Viewer tool, accessed by **Start → Program Files → Administrative Tools → Event Viewer**.

Event Viewer entries are made in the Application Section for application-related events, in the Security section for operating system Security events, and in the System section for OS services themselves.

Keep in mind that application-related events will have as a source the short name of the service under which the DB2 instance is run (for example, DB2-0), as shown in Figure 10-3 on page 329.

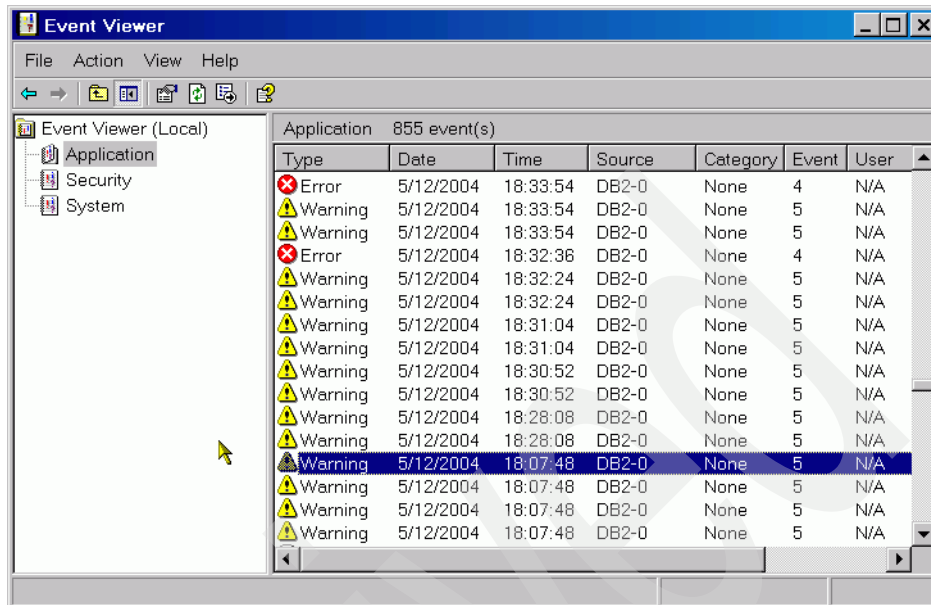


Figure 10-3 Application section of Event Viewer showing DB2-0 service related events

In our example, there were several events logged for DB2-0 service. This is the service under which the DB2 instance is running.

Entries are logged for several events. The same entries are also logged in the DB2 diagnostic log (Figure 10-4 on page 330). Some of the entries represent alerts issued by Health Monitor. They also could trigger external actions if defined to in the Health Center; for example, execute the task or external script, which in turn could send a pager alert, send an e-mail message, or attempt to perform corrective action.

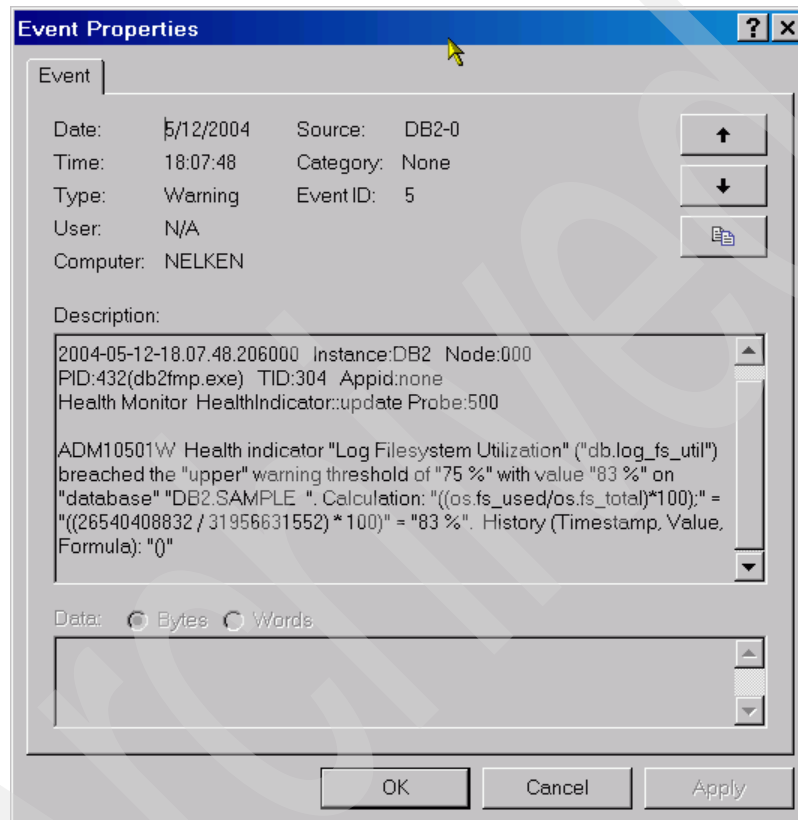


Figure 10-4 Health Center Alert logged in the Event Log.

This figure illustrates a situation in which log space utilization exceeded the warning threshold and triggered the Health Monitor (Figure 10-4). It is possible that taking action now could prevent the error from happening.

However, if you ignore warnings, you may experience a disruption of the services. In our example, we are experiencing a log-full error (Figure 10-5 on page 331).

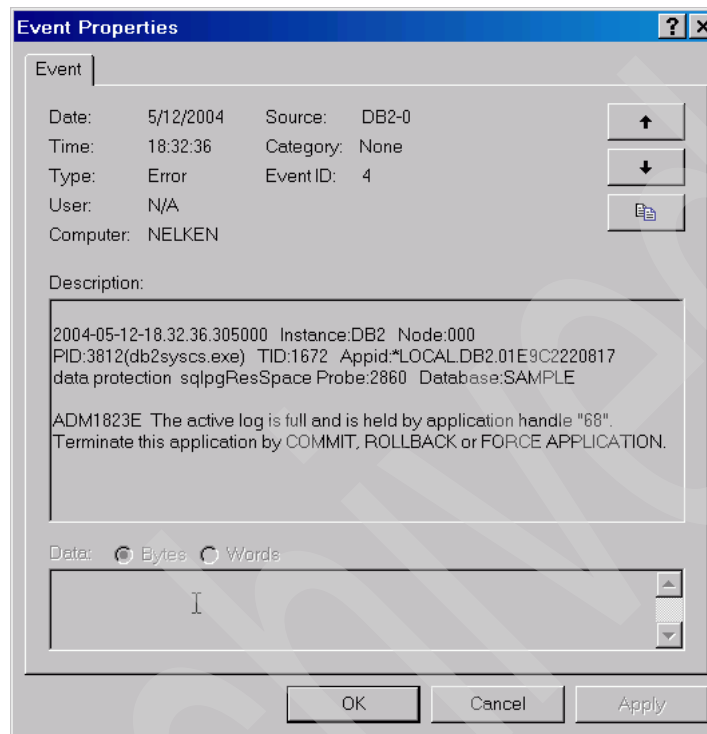


Figure 10-5 Log-full error logged in the Event Viewer

Note that DB2 identified the application holding the uncommitted transactions and suggested corrective action.

Event entries are written to the event log only on the Windows system. You should always check the DB2 diagnostic log for error/warning messages logged by DB2. The disk-full entry was logged in the diagnostic log as shown in Example 10-4.

Example 10-4 Log-full error logged in the diagnostic log

```

2004-05-12-18.32.36.415000-240 E4102H504          LEVEL: Error
PID       : 3812                                TID  : 1672          PROC  : db2syscs.exe
INSTANCE: DB2                                NODE : 000          DB   : SAMPLE
APPHDL   : 0-72                                APPID: *LOCAL.DB2.01E9C2220817
FUNCTION: DB2 UDB, data protection, sqlpgResSpace, probe:2860
MESSAGE  : ADM1823E The active log is full and is held by application handle
           "68". Terminate this application by COMMIT, ROLLBACK or FORCE

```

APPLICATION.

```
2004-05-12-18.32.36.536000-240 I4608H469          LEVEL: Error
PID      : 3812                      TID   : 1672          PROC  : db2syscs.exe
INSTANCE: DB2                      NODE  : 000          DB   : SAMPLE
APPHDL   : 0-72                    APPID: *LOCAL.DB2.01E9C2220817
FUNCTION: DB2 UDB, data protection, sqlpWriteLR, probe:6680
RETCODE  : ZRC=0x85100009=-2062548983=SQLP_NOSPACE
          "Log File has reached its saturation point"
          DIA8309C Log file was full.
```

Information contained in the DB2 diagnostic log is sufficient to identify the culprit application and to take corrective action.

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 335. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *DB2 UDB Exploitation of the Windows Environment*, SG24-6893
- ▶ *DB2 UDB Performance Expert for Multiplatforms: A Usage Guide*, SG24-6436
- ▶ *Integrating XML with DB2 XML Extender and DB2 Text Extender*, SG24-6130
- ▶ *Scaling DB2 UDB on Windows Server 2003*, SG24-7019
- ▶ *XML for DB2 Information Integration*, SG24-6994

Other publications

These publications are also relevant as further information sources (DB2 UDB Version 8.2 books use the -01 suffix):

- ▶ *IBM DB2 UDB Administration Guide: Implementation V8*, SC09-4820-01
- ▶ *IBM DB2 UDB Administration Guide: Performance V8*, SC09-4821-01
- ▶ *IBM DB2 UDB Administration Guide: Planning V8*, SC09-4822-01
- ▶ *IBM DB2 UDB Command Reference V8*, SC09-4828-01
- ▶ *IBM DB2 UDB Data Movement Utilities Guide and Reference V8*, SC09-4830-01
- ▶ *IBM DB2 UDB Data Recovery and High Availability Guide and Reference V8*, SC09-4831-01
- ▶ *IBM DB2 UDB Guide to GUI Tools for Administration and Development*, SC09-4851-01
- ▶ *IBM DB2 UDB SQL Reference, Volume 1, V8*, SC09-4844-01
- ▶ *IBM DB2 UDB SQL Reference, Volume 2, V8*, SC09-4845-01
- ▶ *IBM DB2 UDB System Monitor Guide and Reference V8*, SC09-4847-01

- ▶ *IBM DB2 UDB Application Development Guide: Building and Running Applications V8*, SC09-4825-01
- ▶ *IBM DB2 UDB Application Development Guide: Programming Client Applications V8*, SC09-4826-01
- ▶ *IBM DB2 UDB Application Development Guide: Programming Server Applications V8*, SC09-4827-01
- ▶ *IBM DB2 UDB Call Level Interface Guide and Reference, Volume 1, V8*, SC09-4849-01
- ▶ *IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 2, V8*, SC09-4850-01
- ▶ *DB2 XML Extender Administration and Programming Guide Version 8 Release 1*, SC27-1234-01
- ▶ *IBM DB2 UDB Quick Beginnings for DB2 Clients Version 8.2*, GC09-4832-01
- ▶ *IBM DB2 UDB Quick Beginnings for DB2 Servers Version 8.2*, GC09-4836-01
- ▶ *IBM DB2 UDB What's New V8*, SC09-4848-01
- ▶ *IBM DB2 UDB Installation and Configuration Supplement Version 8.2*, GC09-4837-01
- ▶ *Data Warehouse Center Application Integration Guide Version 8.2*, SC27-1124-01
- ▶ *Federated Systems Guide Version 8 Release 1*, GC27-1224-01

Online resources

These Web sites and URLs are also relevant as further DB2 information sources:

- ▶ DB2 UDB V8.2 Manuals
<http://www.ibm.com/software/data/db2/udb/support/manualsv8.html>
- ▶ Database and Data Management
<http://www.ibm.com/software/data/>
<http://www.ibm.com/software/data/highlights/db2tco.html>
- ▶ DB2 Developer Domain
<http://www7b.software.ibm.com/dmdd/>
- ▶ DB2 Universal Database
<http://www.ibm.com/software/data/db2/udb/>
<http://ibm.com/db2/v8>

- ▶ DB2 Universal Database V8 Application Development
<http://www.ibm.com/software/data/db2/udb/ad>
- ▶ DB2 Technical Support
<http://www.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/index.d2w/report>
- ▶ DB2 Extenders
<http://www.ibm.com/software/data/db2/extenders/>
- ▶ IBM Manuals for Data Management Products
<http://www.ibm.com/software/data/db2/library/>
- ▶ DB2 NOW!
<http://www.ibm.com/db2/migration>

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

Symbols

.NET 2, 23, 25, 237, 254, 268
.NET Data Provider 25
.NET Framework 256

Numerics

32-bit 28
64 KB limit on statement size 12
64-bit 50

A

AcceptChanges 244
Active Directory 24, 121
ActiveX Data Object 238
activity analyzer 71
activity monitor 175
AD 121
add-ins 24–25, 278–279
ADO.NET 25, 238–240, 268
alter table 13
Apache 301
Apache Axis 297
applets 221
ARCHRETRYDELAY 69
AS/400 254
AST 15
attribute 239
authentication 14, 23, 138, 141
authentication plugins 144
authorization 141
authorization ID 141
AUTO_MAINT 75
AUTO_RUNSTATS 75
AUTO_TBL_MAINT 75
automated summary table (AST) 6
automatic client reroute 21, 215–218
automatic database backup 54
automatic reorganization policy 80–86
automatic RUNSTATS 72
automatic statistics collection 70–72
automatic statistics profiling 70, 76, 78–80
autonomic computing 2, 7

AVG 163

B

background process 70
backup 16
Backup and Restore function 41
backup images 67
backup mode 62
BeginTransaction 240
Bernoulli sampling 163
BI 5–6
binary large object 265
BLOB 265
block-level sampling 163
business intelligence 5–6
business intelligence enhancements 15

C

C 12
C# 26, 237, 246
C++ 12
CALL statement support 12
CallableStatement 228
change monitor 71
character large object 265
classes 239
classloader 224
CLASSPATH 223
Clear (DataSet method) 244
client
 Administration Client 11, 36
 Application Development Client 11, 36
 Run-Time Client 10, 36
 Run-Time Client Lite 11, 36
client installation 36
CLOB 265
CLP 22
CLR 24, 237, 261, 278
CM 71
command
 activate database 174
 ALTER TABLE 218
 BACKUP DATABASE 64

- CATALOG DATABASE 140
- db2advis 170
- db2diaglevel 321
- db2expln 165
- db2set 119
- db2start 143
- db2stop 28, 49
- db2trc 143
- force 47
- LIST DATABASE CONFIGURATION 85
- LOAD 71
- PRUNE HISTORY 69
- quiesce 47
- RECOVER 69
- RESTORE DATABASE 64
- TAKEOVER 207
- UPDATE ALTERNATE SERVER FOR DATA-BASE 216
- CommandText 241
- CommandType 241
- Common Language Runtime 24, 237, 261, 278
- compact installation 37
- compatibility 14
- compiler 12
- complex expression 233
- compression 66
- configFileName 44
- Configure Automatic Maintenance wizard 72
- ConnectionString 240
- Control Center 7, 20, 54
- counters
 - database-related 174
- CPU 16
- CreateCommand 240
- CreateParameter 241
- custom installation 37

D

- DADX 296–299
 - Documentation operations 299
 - dynamic approach 298
 - SQL operations 297
 - static approach 298
 - XML collection operations 297
- DADX Web services 297
- Data Connections folder 281
- data consumer 245
- data encryption 138
- data manipulation language 71
- Data Provider 254
- DATA_ENCRYPT_CMP 140
- database administration 7
- database configuration parameter
 - AUTO_DB_BACKUP 63
 - AUTO_MAINT 64
 - AUTO_PROF_UPD 78
 - AUTO_REORG 85
 - AUTO_STATS_PROF 78
 - DFT_PREFETCH_SZ 167
 - DIAGLEVEL 321
 - locktimeout 13
 - LOGINDEXBUILD 218
 - UTIL_HEAP_SZ 171
- database disconnection method
 - close 235
- database manager configuration parameter
 - BACKBUFSZ 64
 - CLNT_PW_PLUGIN 157–158
 - GROUP_PLUGIN 158
 - RESTDUFBSZ 64
 - SRVCON_ATUH 159
 - SRVCON_AUTH 157
 - SRVCONS_PW_PLUGIN 157
- database migration 48
- Database Partitioning Feature 9
- database-related counters 174
- data-centric 71
- data-manipulation-driven feedback loop 70
- DataReader 242
- DataSetName 244
- DataSource 225–226
- DB2
 - Backup and Restore function 41
 - data encryption 138
 - database administration tools 7
 - Database Partitioning Feature 9
 - Database Project type 280
 - database software family 3–4
 - federated system 8
 - technology strategy priorities 2–3
- DB2 .NET Provider 265
- DB2 administration add-in 25
- DB2 Development Add-In 278–279
- DB2 development add-in 24
- DB2 Everyplace 4
- DB2 Information Integrator Web services 297
- DB2 registry variable

- DB2_GRP_LOOKUP 119
- DB2 Server 3
- DB2 Setup wizard 28, 35
- DB2 SQL Editor 282
- DB2 SQL optimizer 6
- DB2 stored procedures 282–284
- DB2 Toolbar 279
- DB2 UDB 3–4
 - capabilities and benefits 4–9
 - for Windows 9
 - multimedia extensibility 4
 - partner solutions 5
 - scalability 4
 - Web enablement 5
- DB2 Universal JDBC Driver 222
- DB2 Web services 297
- db2advis 15, 87
- db2ckmig 48
 - command syntax 49
- DB2Command 257
- DB2Connection 240, 257
- DB2DataAdapter 257
- DB2DataReader 257
- DB2DIAG.EXE 324
- DB2DIAG.LOG 320
- db2licd 49
- db2logmgr 68
- db2relocatedb 44–46
 - execution example 46
 - steps 45
- db2rspgn 31
- db2support.zip 320
- DB2Transaction 257, 263
- db2wi.log 35
- DBCLOB 265
- DBMS optimizer 71
- DDL backup 62
- DeleteCommand 243
- Design Advisor 15, 86
- development center 11
- diagnostic level 48
- diagnostic log 320
- disaster recovery 17, 189
- disconnection from database 235–236
- disks assignments 42, 44
- distributed data 8
- DML 71
- Document Access Definition Extension 297
- domain handling 23

- double-byte character large object 265
- DPF 9
- DRDA 8
- DriverManager 224–225
- dxxworf.jar 304

E

- encryption 14, 138
- enterprise content management 2
- enterprise resource planning 5
- Enterprise Server Edition 9
- ERP 5
- error handling 160
- ESE 9
- ExecuteNonQuery 242
- ExecuteReader 242
- ExecuteScaler 242
- explicit connection to database 234
- Explicit rebind 165
- Express Edition 10

F

- FAILARCHPATH 68
- failover 21, 200, 207–213
- feedback loops 70
- feedback warehouse table 78
- FieldCount 242
- file system 62
- force application 47

G

- generated column 13
- generated column property
 - changing 13
- getConnection method 234
- GetXML 244
- group lookup 23
- group name 129
- group plug-in 150
- GSS-API 145
 - plugins 145
- GSSPLUGIN_SERVER_ENCRYPT 153

H

- HADR 21, 190
 - automatic client reroute 215–218
 - backup and restore restrictions 215

- failover 207–213
- index logging 218–220
- managing 205–207
- monitoring 206–207
- performance restrictions 214
- primary active system 190
- restrictions 213–214
- setup 190–205
- standby active system 190
- HADR action windows 22
- HADR wizard 22
- HasRows 242
- Health Center 54
- health indicator 19
- Health Monitor 19, 71, 81
- high availability 189–220
- High Availability Disaster Recovery 21, 190

I

- IBM.Data.DB2 257
- identity attribute 13
- implicit connection to database 234–235
- INCLUDE LOGS 17, 67
- index advisor 86
- index logging 218–220
- InsertComman 243
- installation type 28
 - compact 37
 - custom 37
 - typical 37
- interoperability 295
- iSeries 3, 24
- iterator Java class 231

J

- JAR 301
- JAR file 303
- Java 2
- Java identifier 233
- Java package 223
- JDBC 222
- JDBC DataSource interface 234
- JDBC driver type 222
- JDBC drivers 222
- JDBC method 223
- JNDI 226

K

- Kerberos 145

L

- LANMAN 120
- large object 265
- large object data streaming 138
- LDAP 143
- Lempel-Ziv (LZ) compression 66
- Linux 4
- LOB
 - BLOB 265
 - CLOB 265
 - DBCLOB 265
- LOB data streaming 138
- local system account 24, 132
- LocalSystem 132
- lock wait mode 13
- locking 12
- log mirroring 215
- LOGRETAIN 68
- LSA 24, 132

M

- materialized query table 6, 15
- Materialized Query Table advisor 86
- MDC 15, 166
- memory configuration 18
- method 239
 - BeginTransaction 263–264
 - Class.forName 224
 - Commit 263
 - ExecuteNonQuery 263
 - executeQuery 227
 - ExecuteReader 267
 - executeUpdate 227
 - GetBytes 267
 - getCnn 266
 - getConnection 224
 - GetXml 260
 - MemoryStream 267
 - open 240
 - PreparedStatement.executeQuery 228
 - RollBack 263
- Microsoft .NET 25
- migrate command 49
 - syntax 49
- migration 40–51

- 32-bit to 64-bit 50
- between versions of DB2 UDB 47
- between versions of Windows 41
- database 48–49
- illegal database states 48
- moving disks 42
- restrictions 47
- steps 48
- Windows and DB2 simultaneously 50
- moving disks 42
- MQT 6, 15
- MSCS 216
- multidimensional analysis (MDA) 7
- multidimensional clustering 15, 166
- Multidimensional Clustering advisor 86
- multimedia extensibility 4

N

- namespace 247
- nested group 23, 120
- Net.Data 5
- NUMARCHRETRY 68

O

- object
 - command 241
 - connection objec 240
 - DB2DataReader 242
 - Db2Parameter 262
- object-relational technology 4
- ODBC 249
- ODBCCommand 257
- ODBCConnection 257
- ODBCDataAdapter 257
- ODBCDataReader 257
- ODBCTransaction 257
- offline database backup 54
- OLAP 5, 7
- OLE DB 246
- OleDbCommand 257
- OleDbConnection 240
- OleDbDataAdapter 257
- OleDbDataReader 257
- OleDbTransaction 257
- OLTP 168
- online analytical process 5
- online database backup 54
- online transaction processing 168

- Open DataBase Connectivity 249
- OS/390 3, 24

P

- partitioning advisor 15, 86
- PDE 10
- PE 10
- peer state 200
- Persistent Storage Module 12
- Personal Developer's Edition 10
- Personal Edition 10
- plug & play 43
- plugin 14
- plug-ins 148–149
- plugins 142, 145–146, 149–160
- PreparedStatement 228
- primary database 21
- problem determination 318
- problem source identification 318
- products path 29
- PSI 318
- PSM 12

Q

- Q Apply 22
- Q Capture 22
- Q replication 22
- query 163
- query feedback warehouse 77
- query optimization 76
- query-centric 72
- query-driven 76
- query-driven feedback loop 70
- quiesce 47
 - command syntax 47

R

- RAS 318
- raw disk 43
- RDBMS 3
- ReadXml 245
- Recommendation Advisor 19
- registry variable 29
- reliability 9
- relocation 45
- REOPT 165
- REOPT bind option 12

- REORG 61
- Reorg with policy 19
- REPEATABLE 164
- response file 30, 35
 - installation 35
- ResultSet 228
- row-level Bernoulli sampling 163
- RUNSTATS 16
- RUNSTATS 19, 61, 70
- runstats option
 - SET PROFILE 79
 - SET PROFILE NONE 79
 - SET PROFILE ONLY 79
 - UPDATE PROFILE 79
 - UPDATE PROFILE ONLY 80
 - USE PROFILE 79
- RUNSTATS options 79–80
- Run-Time Client 10
- Run-Time Client Lite 11

S

- sampling
 - block level 163
 - row-level Bernoulli 163
- savepoint 14
- scalability 4, 282
- security 14, 129
 - enhancements 14
- security check 141
- security exit 14
- security identifier 118
- security plug-in 142–144
- SelectCommand 243
- SERVER_ENCRYPT 140
- server-side plug-in 149
- serviceability 318
- SET LOCK WAIT / NO WAIT 12
- setclasspath.bat 303
- shutdown.bat 303
- SID 118
- Simple Object Access Protocol 297
- SMP 4
- SOAP 297
- spatial information 4
- special characters in user IDs 127–128
- SQL 11, 94
- SQL Java 231
- SQL parser 24

- SQL statement 227
- SQLCODE 216
- SQLDA 138
- SQLJ 231
- sqlj.runtime 233
- standby database 21
- Star Joins 7
- startup.bat 303
- statement (object) 227
- statement size 162
 - limit 12
- statistics collection 70
- steps of using db2relocatedb 45
- stored procedures 26, 282–284
- Structured Query Language 296
- SUM 163
- System Performance Monitor 174
- system security 23–24
- System.Data.ODBC 257
- System.Data.OleDb 257

T

- table maintenance objectives 72
- table reorganization 80–86
- TABLESAMPLE 16
- tape backup 62
- terminate command 49
- throttle utilities 20
- time-series data 4
- transaction 23
- transaction log 67
- trigger 12
- TSM backup 62
- typical installation 37

U

- UDE 10
- UDF 14, 296
- Universal Developer's Edition 10
- UNIX 4
- update dbm cfg 48
- UpdateCommand 244
- user defined function 296
- user IDs
 - special characters in 127–128
- user-defined function 14
- USEREXIT 68
- userid/password authentication 144

V

Visual Studio 24
VM 3, 25, 254
VSE 3, 25, 254

W

Web enablement 5
Web service consumer 296
Web Service Description Language 296
Web Service provider 296
Web Services Object Runtime Framework 26
web.xml 306
WebService 26
WebSphere Application Server 301
Win Form 268
Windows Task Manager 173
WOF 26, 296, 301
WOF framework 297
Workgroup Server Edition 9
Workgroup Server Unlimited Edition 10
workload 72, 86, 94
WriteXml 245
WSDL 296, 311
WSE 9
WSUE 10

X

XBSA backup 62
XML 23, 260, 296

Z

z/OS 3, 24



DB2 UDB V8.2 on the Windows Environment

(0.5" spine)
0.475" <-> 0.875"
250 <-> 459 pages



Redbooks

DB2 UDB V8.2 on the Windows Environment

Experience the power of V8.2 on the Windows environment

Learn about new autonomic and management enhancements

Develop applications using DB2 with .NET and Java

IBM DB2 Universal Database Version 8.2 is another significant jump in DBRM technology. It delivers new features to address the ever-increasing demands and requirements of information management customers. This redbook is an update of *DB2 UDB Exploitation of the Windows Environment* (SG24-6893), with a focus on DB2 UDB Version 8.2 functions and features.

We illustrate, step-by-step, the installation and migration processes. The new autonomic computing technology that makes the DBA's job easier is described in detail. The new security features and their integration with the Windows environment are discussed intensively.

This book covers the new performance and monitoring enhancements of Version 8.2. New, exciting high availability functions are described in detail.

Further, we discuss aspects of DB2 application development with .NET 2003 and Java and provide programming details using ADO.NET, CLR, and Java.

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