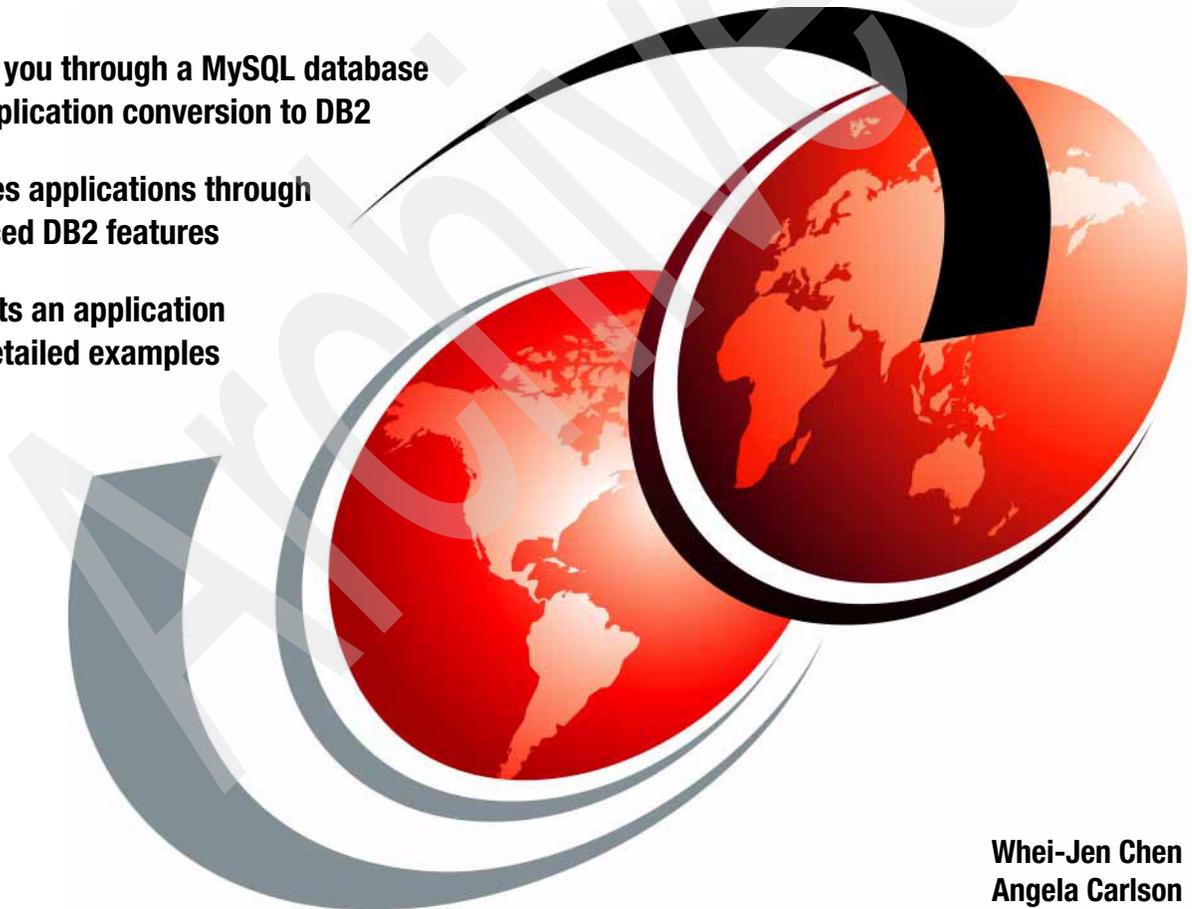


MySQL to DB2 Conversion Guide

Guides you through a MySQL database
and application conversion to DB2

Enriches applications through
advanced DB2 features

Converts an application
with detailed examples



Whei-Jen Chen
Angela Carlson



International Technical Support Organization

MySQL to DB2 Conversion Guide

December 2009

Archived

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

Second Edition (December 2009)

This edition applies to DB2 9.7 for Linux, UNIX, and Windows and MySQL 5.1.

© Copyright International Business Machines Corporation 2009. All rights reserved.

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

Contents

Notices	ix
Trademarks	x
Preface	xiii
The team who wrote this book	xiii
Become a published author	xiv
Comments welcome	xv
Summary of changes	xvii
December 2009, Second Edition	xvii
Executive summary	xix
Chapter 1. DB2 for Linux, UNIX, and Windows	1
1.1 Introduction	2
1.2 Product overview	3
1.2.1 DB2 Data Server Editions for the production environment	3
1.2.2 Products for accessing System z and System i host data	7
1.2.3 DB2 for pervasive platforms	8
1.2.4 Additional DB2 data server features	8
1.3 DB2 for Linux, UNIX, and Windows architecture	9
1.3.1 DB2 9.7 threaded architecture and process model	9
1.3.2 DB2 database objects	10
1.3.3 DB2 catalog	18
1.4 DB2 utilities	18
1.5 DB2 database access	21
1.5.1 DB2 clients and drivers	21
1.5.2 Application access	25
1.5.3 DB2 application programming interfaces	26
Chapter 2. MySQL database	35
2.1 MySQL licensing overview	36
2.2 MySQL architecture overview	36
2.2.1 Database client and non-client utilities	37
2.2.2 Database server	38
2.3 MySQL design and SQL compliance	41
2.3.1 MySQL directory structure	41
2.3.2 MySQL storage engines	45
2.3.3 MySQL standard SQL compliance	48

2.4 MySQL utilities	49
2.4.1 Overview of the MySQL server-side programs and utilities	50
2.4.2 Overview of the MySQL client-side programs and utilities	50
2.5 MySQL application programming interfaces	51
Chapter 3. Planning the conversion from MySQL to DB2	55
3.1 Conversion project planning overview	56
3.1.1 Benefits of converting to DB2	57
3.1.2 IBM conversion support	60
3.1.3 Education	61
3.2 Application assessment	62
3.3 System planning	63
3.3.1 Software	64
3.3.2 Hardware	64
3.3.3 Conversion tools	65
3.4 The conversion process	66
3.4.1 Preparing for the installation	67
3.4.2 Porting the database structure	67
3.4.3 Data porting	68
3.4.4 Application porting	70
3.4.5 Basic administration	72
3.4.6 Testing and tuning	72
Chapter 4. Conversion scenario	75
4.1 Application structure	76
4.1.1 Application flow	76
4.2 Database structure	84
4.3 System environment	85
Chapter 5. Installation	87
5.1 DB2 Express-C 9.7 on Linux	88
5.1.1 System requirements	88
5.1.2 Installation procedure	92
5.1.3 Instance creation	102
5.1.4 Client setup on Linux	103
5.2 Other software products	105
5.2.1 Apache2 installation with DB2 support	105
5.2.2 PHP installation with DB2 support	107
5.3 IBM Data Movement Tool installation and usage	112
5.3.1 IBM Data Movement Tool prerequisites	112
5.3.2 IBM Data Movement Tool installation	113
Chapter 6. Database conversion	115
6.1 Data type mapping	116

6.2	Data definition language differences	122
6.2.1	Database manipulation	123
6.2.2	Table manipulation	128
6.2.3	Index manipulation	136
6.2.4	Trigger manipulation	137
6.2.5	Procedures and function manipulation	138
6.3	Other considerations	138
6.4	Converting the database	142
6.4.1	Automatic conversion using porting tools	143
6.4.2	Manual conversion	144
6.4.3	Metadata transport	147
6.5	Sample database conversion	148
6.5.1	Converting database objects with the IBM Data Movement Tool	148
6.5.2	Manual database object conversion and enhancements	158
	Chapter 7. Data conversion	167
7.1	Data porting considerations	168
7.1.1	Data porting commands and tools	168
7.1.2	Differences in data formats	175
7.1.3	Differences in the user account management	177
7.2	Sample project: Data porting	192
7.2.1	Export user data from MySQL	192
7.2.2	Map MySQL user data to DB2 user data	193
7.2.3	Create DB2 user	194
7.2.4	Export MySQL application data	195
7.2.5	Convert MySQL application data to DB2 format	197
7.2.6	Import application data into DB2	197
7.2.7	Basic data checking	199
	Chapter 8. Application conversion	205
8.1	Data Manipulation Language differences and similarities	206
8.1.1	SELECT syntax	206
8.1.2	JOIN syntax	207
8.1.3	UNION syntax	208
8.1.4	Subquery syntax	209
8.1.5	Grouping, having, and ordering	209
8.1.6	Strings	210
8.1.7	Implicit casting of data types	213
8.1.8	String concatenation and NULL values	216
8.1.9	Record deletion	218
8.1.10	Built-in functions and operators	221
8.2	Application source conversion	222
8.2.1	Converting MySQL Perl applications to DB2	222

8.2.2	Converting MySQL PHP applications to DB2	225
8.2.3	Converting MySQL Ruby on Rails applications to DB2	237
8.2.4	Converting MySQL Java applications to DB2	240
8.2.5	Converting MySQL C/C++ applications to DB2	247
8.2.6	Converting Connector/ODBC applications to DB2	256
8.2.7	Condition handling in DB2	259
8.2.8	Special conversions	266
8.3	Additional application considerations	271
8.3.1	The purpose of locking	271
8.3.2	Concurrency control and transaction isolation	272
8.3.3	DB2 isolation levels	272
8.3.4	Locking	274
8.3.5	Specifying the isolation level in DB2	276
Chapter 9. Database administration		279
9.1	Database configuration	280
9.1.1	DB2 configuration	280
9.2	Database recovery	286
9.2.1	DB2 database recovery	286
9.3	Database replication	296
9.4	Data movement	298
9.4.1	DB2 data movement	298
9.5	High availability	304
9.6	Autonomics	306
9.7	Workload management	311
9.8	Database management tools	312
9.8.1	DB2 Control Center	312
9.8.2	IBM Optim and Data Studio tool suite overview	314
Chapter 10. Testing and tuning		321
10.1	Test planning	322
10.1.1	Principles of software tests	322
10.1.2	Test documentation	322
10.1.3	Test phases	322
10.1.4	Time planning and time exposure	323
10.2	Data checking techniques	324
10.2.1	IMPORT/LOAD messages	324
10.2.2	Data checking	328
10.3	Code and application testing	330
10.3.1	Checking the application code	331
10.3.2	Security testing	331
10.4	Troubleshooting	332
10.4.1	Interpreting DB2 informational messages	332

10.4.2	DB2 tools for troubleshooting	334
10.4.3	DB2 diagnostic logs	335
10.4.4	DB2 support information	339
10.4.5	Monitoring tools	342
10.4.6	Visual Explain	354
10.5	Initial tuning	356
10.5.1	Table space design	356
10.5.2	Physical placement of database objects	357
10.5.3	Buffer pools	360
10.5.4	Large transactions	363
10.5.5	SQL execution plan	366
10.5.6	Configuration Advisor	371
10.5.7	Design Advisor	375
Chapter 11. Advanced DB2 features		381
11.1	DB2 pureXML	382
11.2	Data compression	386
11.3	Partitioning features	388
11.3.1	Database partitioning feature	388
11.3.2	Table partitioning	389
11.3.3	Multidimensional clustering	392
11.4	Materialized query tables	394
11.5	User-defined data types	396
Appendix A. Mapping MySQL built-in functions and operators		399
A.1	Grouping related functions	400
A.2	String functions	402
A.3	Numeric functions	408
A.4	Date and time functions	409
A.5	Comparing operators and other functions	410
Appendix B. Sample code for user-defined functions		413
B.1	Sample code for BIT_AND	414
B.2	Sample code for FORMAT function	415
B.3	Sample code for RPAD and LPAD functions	416
B.4	Sample code for GREATEST function	422
B.5	Sample code for LEAST	427
B.6	Sample code for BIT_COUNT	431
B.7	Sample code for SUBSTRING_INDEX	432
B.8	Sample code for UNIX_TIMESTAMP	433
Related publications		435
	IBM Redbooks publications	435
	Other publications	435

Online resources	436
How to get IBM Redbooks publications	439
Help from IBM	439
Index	441

Archived

Notices

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

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

1-2-3®	i5/OS®	Rational®
AIX®	IBM®	Redbooks®
AS/400®	IMS™	Redbooks (logo)  ®
ClearCase®	Informix®	System i®
DB2 Connect™	InfoSphere™	System p®
DB2 Universal Database™	iSeries®	System z9®
DB2®	OpenPower®	System z®
developerWorks®	Optim™	Tivoli®
Distributed Relational Database Architecture™	OS/390®	UniData®
DRDA®	PartnerWorld®	WebSphere®
eServer™	POWER®	z/OS®
Everyplace®	pSeries®	z9®
HACMP™	pureXML®	zSeries®
	Rational Rose®	

The following terms are trademarks of other companies:

AMD, the AMD Arrow logo, and combinations thereof, are trademarks of Advanced Micro Devices, Inc.

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

SUSE, the Novell logo, and the N logo are registered trademarks of Novell, Inc. in the United States and other countries.

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

Red Hat, and the Shadowman logo are trademarks or registered trademarks of Red Hat, Inc. in the U.S. and other countries.

VMware, the VMware "boxes" logo and design are registered trademarks or trademarks of VMware, Inc. in the United States and/or other jurisdictions.

EJB, Enterprise JavaBeans, J2EE, Java, Java runtime environment, JavaBeans, JavaServer, JDBC, JSP, MySQL, Solaris, Sun, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Access, ActiveX, Expression, Microsoft, MS, SQL Server, Visual Basic, Visual Studio, Windows Mobile, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

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

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

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

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

Archived

Archived

Preface

Switching database vendors is often considered an exhausting challenge for database administrators and developers. Complexity, total cost, and the risk of downtime are often the reasons that restrain IT decision makers from starting the conversion project. The primary goal of this book is to show that, with the proper planning and guidance, converting from MySQL™ to IBM DB2® for Linux, UNIX, and Windows is not only feasible but straightforward.

If you picked up this book, you are most likely considering converting to DB2 and are probably aware of several of the advantages of converting to DB2 data server. In this IBM® Redbooks® publication, we discuss in detail how you can take advantage of this industry leading database server.

This book is an informative guide that describes how to convert the database system from MySQL 5.1 to DB2 9.7 on Linux®, and the steps involved in enabling the applications to use DB2 instead of MySQL.

This MySQL to DB2 migration guide also presents the best practices in conversion strategy and planning, conversion tools, porting steps, and practical conversion examples. It is intended for technical staff involved in a MySQL to DB2 conversion project.

The team who wrote this book

This book 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.



Angela Carlson is a Software Engineer with IBM Canada. She has four years of technical experience working in the IT industry, focusing on relational database technology and application development. She has experience in developing relational database applications with PHP, Perl, and Java™. In Angela's current position, she works closely with IBM Business Partners and their database needs to enable their solutions with DB2. She also researches and develops competitive material on DB2 and MySQL. Angela holds a Bachelor's degree of Software Engineering Science from the University of Western Ontario.

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

Boris Bialek
Program Director, Information Management Partner Technologies, IBM Canada

Irina Delidjakova
Information Management Emerging Partnerships and Technologies, IBM Canada

Vlad Barshai
Information Management Emerging Partnerships and Technologies, IBM Canada

Martin Schlegel
Information Management Partner Technologies, IBM Canada

Daniel Krook
Cloud Engineering and Experience, IBM U.S.

Emma Jacob
International Technical Support Organization, San Jose Center

Thanks to the authors of the previous edition of this book:

Authors of the first edition, *MySQL to DB2 UDB Conversion Guide*, SG24-7093, published in May 2004, were Whei-Jen Chen, Andreas Blank, Michael Hoeller, Rakesh Midha, and Klaus Subtil

Become a published author

Join us for a two- to six-week residency program. Help write a book dealing with specific products or solutions, while getting hands-on experience with

leading-edge technologies. You will have the opportunity to team with IBM technical professionals, IBM Business Partners, and Clients.

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us.

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

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

ibm.com/redbooks

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

redbooks@us.ibm.com

- ▶ Mail your comments to:

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

Archived

Summary of changes

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

Summary of Changes
for SG24-7093-01
for *MySQL to DB2 Conversion Guide*
as created or updated on December 1, 2009.

December 2009, Second Edition

This revision reflects the addition, deletion, or modification of new and changed information described below.

New information

- ▶ DB2 features and functions of DB2 for Linux, UNIX®, and Windows® Version 9, 9.5, and 9.7
- ▶ IBM Data Movement Tool
- ▶ MySQL 5.1 features

Changed information

- ▶ DB2 and MySQL features and functions
- ▶ Conversion scenarios and examples

Archived

Executive summary

This book describes how to migrate MySQL 5.1 to DB2 Version 9.7 on Linux and enable your applications on DB2. To further ease your migration, this informative guide will cover best practices in migration strategy and planning, as well as the step-by-step directions, tools, and practical conversion. After completing this book, it will be clear to the technical reader that a MySQL to DB2 migration is easy and straightforward.

Potential IBM clients seek migration information because DB2 offers performance and functional capabilities that the competition can't compare with. DB2 Express C, our lightweight community edition, is free to develop, deploy and redistribute, and is designed to give the IT community a powerful alternative to the open source or free databases currently available.

DB2 Express C offers the same high quality, reliable, scalable features that you would expect from an IBM enterprise database at no charge. Fixed Term License support is available as well, at a lower price than the competition. The decision to migrate becomes simple when you consider that DB2 can be easily deployed in the development stack, while offering many additional features and ease of use.

Enterprise class features aimed to lower the total cost of ownership can be found in every edition of DB2. DB2 has powerful autonomies which make installation, configuration, maintenance and administration virtually hands free. DB2 9.7's compression features help companies manage rising energy costs and reduce datacenter sprawl by reducing storage requirements and improving I/O efficiency.

IBM is committed to providing products to our clients that are powerful and affordable. DB2 provides industry leading features, such as pureXML, Workload Management, and Granular Security. Using DB2 pureXML® makes XML data processing even faster, more flexible, and more reliable. Manage workloads with new threshold, priority and OS integration features in DB2 9.7. Keep data secure from internal and external threats using the unparalleled security control in DB2 9.7.

Start taking advantage of these exciting new features and help your business manage costs and simplify application development. Migrate your database systems and applications today and discover why DB2 9.7 is a smarter product for a smarter planet.

Arvind Krishna

Arvind Krishna
General Manager
IBM Information Management

Archived



DB2 for Linux, UNIX, and Windows

The goal of this chapter is to give an overview of the DB2 database server, its architecture, and the tools and utilities that are available with the server and application programming interfaces.

In this chapter, we cover these topics:

- ▶ DB2 product overview
- ▶ DB2 for Linux, UNIX, and Windows architecture
- ▶ DB2 utilities
- ▶ DB2 database access

1.1 Introduction

IBM has an extremely strong history of database innovation and has developed a number of highly advanced data servers. It started in the 60s when IBM developed the Information Management System (IMS™), which is a hierarchical database management system. IMS was used to maintain inventory for the Saturn V moon rocket and the Apollo space vehicle. In the 70s, IBM invented the Relational Model and the Structured Query Language (SQL). In the 80s, IBM introduced DB2 for the mainframe (DB2 for z/OS®), which was the first database that used relational modeling and SQL. DB2 for distributed platforms (DB2 for Linux, UNIX, and Windows) was introduced in the 90s. Since then, IBM continues to develop on DB2 for both mainframe and distributed platforms. Although the relational data model has become more prevalent in the industry; IBM still realizes that the hierarchical data model is important. Therefore in July 2006, IBM launched the first hybrid (also known as multi-structured) data server.

The release of DB2 for Linux, UNIX, and Windows Version 9 (DB2 9) data server brought the most exciting and innovative database features to the market; these features were further enhanced with the release of DB2 9.5 and 9.7. DB2 9 introduced many important features for both database administrators and application developers. These features included pureXML®, autonomics, table partitioning, data compression, and label-based access control. DB2 9.5 enhanced the manageability of the DB2 data server by introducing the threaded engine, easier integration with high availability disaster recovery (HADR), workload management, enhancements to autonomics, and more. The focus of the DB2 9.7 release is to provide unparalleled reliability and scalability for the changing needs of your business. Therefore, DB2 9.7 introduces enhancements to Version 9 and Version 9.5 features, such as enhancements to data compression, performance, workload management, security, and application development.

When this book was written, DB2 9.7 had just been released on June 2009. DB2 9.7 is the database version that we use throughout the book. DB2 9.7 is a highly scalable and easy to install and manage hybrid data server. DB2 was developed to meet the demands of even the most critical database applications. This is managed through various autonomics capabilities, such as self-tuning memory management and automatic storage. DB2 provides a highly adaptable database environment while optimizing data storage through backups and deep data row compression. DB2 deep embedded capabilities allow for ubiquitous deployment in user directories and administrative installations for any size server. In a single database, DB2 provides native storage and processing of both transactional XML data in a pre-parsed tree format and relational data using pureXML technology.

1.2 Product overview

DB2 for Linux, UNIX, and Windows spans the spectrum from products on handheld devices to large clusters and mainframes (see Figure 1-1). You can obtain more detailed information from the DB2 Web site:

<http://www.ibm.com/db2/>

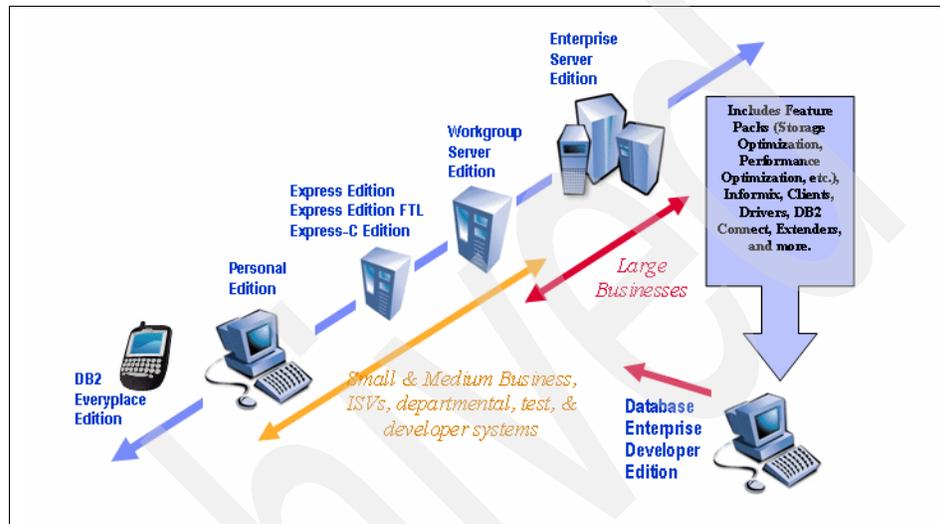


Figure 1-1 DB2 product overview

1.2.1 DB2 Data Server Editions for the production environment

DB2 provides packages for users based on their business needs. In DB2 9.5, IBM introduced the common client concept for all IBM data management products that provides equal engine functionality across system boundaries. This functionality means that DB2 scales up and down according to your needs. Be it a desktop or a heavy duty enterprise server, DB2 has consistent behavior across any platform, providing an easy move regardless of the size limit. DB2 widens the boundaries even further, because all IBM data servers, including Informix®, now share a common API for SQL. Therefore, whether using Java Database Connectivity (JDBC™) on DB2 for z/OS, Informix 11 on an embedded system, or DB2 for Linux, UNIX, and Windows, the driver code remains the same. In fact, all database connections to any IBM data server can be made with the same common client. In addition, IBM Optim™ Data Studio provides a new cross-family tool for both application development and database management, underlining commitment across and beyond DB2.

There are several DB2 package formats:

► **DB2 Personal Edition**

DB2 Personal Edition (PE) provides a single user database engine that is ideal for deployment to PC-based users. The PE includes the ability for remote management, the pureXML feature, and the SQL replication feature, making it the perfect choice for deployment in occasionally connected or remote office implementations that do not require multi-user capability, that is, point-of-sale systems.

PE does not accept remote database requests; however, it contains DB2 client components and serves as a remote client to a DB2 Server. The DB2 Personal Edition can also be used for connecting and managing other DB2 data servers in the network.

The Personal Edition includes most of the features included in DB2 Express Edition and runs in either 32-bit or 64-bit Intel® or AMD™ workstations for either Windows or Linux operating systems.

► **DB2 Express-C**

DB2 Express-C is the no-charge community version of the DB2 data server. It is targeted towards developers and Independent Software Vendors (ISVs) to allow the development and deployment of applications, including the no charge distribution of DB2 Express-C itself. All applications developed with this version of DB2 can be moved to a higher edition of DB2 for Linux, UNIX and Windows and even DB2 for z/OS without any application changes if using the common SQL API set of the DB2 family.

This version of DB2 is at no charge for download and is therefore perfectly suited for DB2 educators and students. DB2 Express-C does not restrict the database size and can be used in a 64-bit memory model. The code is optimized to use up to a maximum of 2 CPU cores and 2 GB of memory. No fix pack updates are available for this edition; however, new versions of Express-C are updated and freely available for download at any time.

While this version does not include all the features of higher editions of DB2, such as storage optimization, replication services, or high availability, it comes with the award-winning pureXML technology to leverage both relational and XML data by being able to natively store XML data in a single database. Several of these features can be activated by purchasing the *DB2 Express Fixed Term License (FTL)*. Obtaining the FTL provides 1 year of 24x7 support, plus the ability to use high availability and disaster recovery.

At any point, users of DB2 Express-C can receive advice on the IBM DB2 Express Forum, which is monitored by IBM DB2 developers, by accessing the following link:

<http://www.ibm.com/developerworks/forums/forum.jspa?forumID=805>

DB2 Express-C runs on Windows or Linux for both Intel and AMD on 32-bit or 64-bit architecture, as well as on Linux on Power (IBM System p® and System i®).

You can download DB2 Express-C from this link:

<http://www-01.ibm.com/software/data/db2/express/>

► **DB2 Express + Fixed Term License**

The plus fixed term license (FTL) is a 1-year license subscription with 24x7 IBM support to DB2 Express, providing service-level assurances for problem resolution and offering fix pack updates tied to the usual DB2 fix pack schedule.

FTL also raises the DB2 Express-C limitations and is optimized to run for 4 CPU cores and 4 GB of memory. Additional features with the DB2 Express FTL include the High Availability Disaster Recovery (HADR) feature and full SQL replication capability.

Licensing is quite easy with DB2 Express-C + FTL, because it requires only purchasing one FTL subscription per data server no matter how it is used or as explained before, its size. This licensing explicitly applies also to an HADR setup, where no distinction is made between cold, warm, or hot standby usage.

DB2 Express-C with FTL runs on Windows or Linux for both Intel or AMD on 32-bit or 64-bit architecture, as well as on Linux on Power (IBM System p and System i).

More information about DB2 Express + Fixed Term License is available at this Web site:

http://www-01.ibm.com/software/data/db2/express/support.html?S_TACT=105AGX28&S_CMP=DLMAIN

► **DB2 Express**

DB2 Express edition is specifically tailored for small and medium businesses (SMBs). It is designed for independent software vendors who need an easy-to-install 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 (LAN) environments.

DB2 Express utilizes up to 4 GB of memory and can be installed on a server with up to 200 processor value units. For more information about processor value units, visit this Web site:

<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0611zikopoulos2/>

Several key features and strengths of the Express Edition are simplified deployment, SQL replication, backup compression, autonomic management

features, such as the Self-Tuning Memory Manager, adaptive utility throttling, automatic storage management, configuration and design advisors, health and fault monitors, automated backup, pureXML, high availability, and more.

DB2 Express runs on Linux, Solaris™ x86, or Windows platforms.

► **DB2 Workgroup**

DB2 Workgroup Server Edition is used primarily in small to medium-sized business environments. It includes support for a per authorized user-based or a per processor-based licensing model that is designed to provide an attractive price point for smaller installations while still providing a fully functional data server on a wider range of platforms compared to lower function editions of DB2.

Workgroup Server now includes the High Availability Disaster Recovery (HADR) feature, Tivoli System Automation for MultiPlatforms (TSA MP), and Online Table Reorganization. The following feature packs are available:

- Query Optimization including Materialized Query Tables (cached tables)
- Multidimensional Clustering and Query Parallelism
- pureXML

This DB2 edition can be deployed on systems with up to 400 processor value units and 16 GB of memory.

While the DB2 Express edition only runs on Windows, Linux, or Solaris, the Workgroup Server edition adds support for AIX®, Hewlett-Packard UNIX (HP-UX) on Itanium64, and Solaris on x86 and Sparc.

► **DB2 Enterprise**

DB2 Enterprise Server Edition meets the database server needs for any size business. This product is the ideal foundation for building data warehouses, transaction processing, or Web-based solutions, as well as for a back-end for packaged solutions, such as enterprise resource planning (ERP), customer relationship management (CRM), and supply chain management (SCM). In addition, the DB2 Enterprise Server Edition offers connectivity and integration for other enterprise DB2 and Informix data sources.

The Enterprise Server Edition does not pose limits to the maximum memory or number of CPU cores. It can be licensed with either authorized user licenses or processor value unit licenses.

In addition to the features offered in the Workgroup Server Edition, the following features are also available: Query Parallelism, Multidimensional Clustering, Materialized Query Tables, Table Partitioning, and Connection Concentration. With the DB2 feature pack, you can add the following features: Performance Optimization Feature, Advanced Access® Control Feature, Storage Optimization Feature, and Geodetic Data Management Feature.

DB2 Enterprise Server Edition runs on Windows (32-bit and 64-bit), Linux (Intel/ AMD 64-bit, System i, System p, System z®), AIX, Solaris (Sparc and x64) and HP-UX (ia64).

► **InfoSphere™ Warehouse**

InfoSphere Warehouse (formerly known as DB2 Warehouse) is a powerful platform for building business intelligence (BI) solutions. InfoSphere Warehouse comes as single integrated software package, using DB2 Enterprise Server Edition as its base. It provides an ideal solution for companies that need to consolidate data marts, information silos, and business analytics.

InfoSphere Warehouse contains database management tools and embedded data movement and transformation tools. In addition to the base feature set of DB2 Enterprise Server Edition, the DB2 feature packs containing Data Partitioning Feature (DPF) are standard in all editions of InfoSphere Warehouse. Workload Management and Deep Data Row Compression are optional features for InfoSphere Warehouse Enterprise Base Edition.

The overall integrated software package for InfoSphere Warehouse has support for slice and dice analytics and provides direct support for optimized online analytical processing (OLAP) analytics against the dynamic warehouse. Embedded analytical features include capabilities for Data Mining, In-line Analytics, Cubing Services, and Unstructured Analytics.

► **Other DB2 editions**

There are also DB2 versions for System i and System z available. See the following Web site for details:

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

1.2.2 Products for accessing System z and System i host data

With the following DB2 products, you can extend enterprise systems to access the host data in a System z or System i system:

► **DB2 Connect™ Personal Edition**

The *DB2 Connect Personal Edition* provides the application programming interface (API) drivers and connectivity infrastructure to enable direct connectivity from desktop applications to System z and System i data servers. This product is specifically designed and is licensed for enabling two-tier, client/server applications running on individual workstations, and as such, is not appropriate for use on servers.

► **DB2 Connect Enterprise Edition**

The *DB2 Connect Enterprise Edition* addresses the needs of organizations that require robust connectivity from a variety of desktop systems to System i

and System z database servers. DB2 client software is deployed on desktop systems and provides drivers that connect client/server applications running on these desktop systems to a DB2 Connect server (gateway) that accesses host data. The licensing model for this product is user-based.

► **DB2 Connect Application Server Edition**

Technically, the *DB2 Connect Application Server Edition* product is identical to the DB2 Connect Enterprise Server; however, its licensing terms and conditions are meant to address the specific needs of multi-tier, client/server applications, as well as applications that utilize Web technologies. DB2 Connect Application Server Edition license charges are based on the size and number of processors that are available to the application servers where the application is running.

► **DB2 Connect Unlimited Edition**

The *DB2 Connect Unlimited Edition* product is ideal for organizations with extensive usage of DB2 Connect, especially where multiple applications are involved. This product provides program code of the DB2 Connect Personal Edition, as well as program code identical to the DB2 Connect Application Server Edition for unlimited deployment throughout an organization.

1.2.3 DB2 for pervasive platforms

DB2 Everyplace® Edition supports a wide variety of handheld devices, such as Palm OS, Windows Mobile® for Pocket PC, Windows desktops, Symbian, QNX Neutrino, and various Linux distributions. Its strength is to synchronize with data from other systems, using security features, including table-level encryption and communication encryption, such as Secure Sockets Layer (SSL).

DB2 Everyplace is also extremely flexible by being completely self-managed and by supporting many programmable API interfaces, such as Open Database Connectivity (ODBC), Java Database Connectivity (JDBC), .NET, DB2 call level interface (CLI), and so on. Its engine is highly flexible, which makes it easy to move DB2 Everyplace databases from one supported mobile device to another.

1.2.4 Additional DB2 data server features

In addition to the product offerings for Linux, UNIX, Windows, and accessing System z and System i host data, DB2 also offers a great variety of features, which are included in various DB2 packages.

Information about additional DB2 features and products can be found at this Web site:

http://www.ibm.com/software/data/db2/9/features.html?S_CMP=wspace

1.3 DB2 for Linux, UNIX, and Windows architecture

Data servers provide software services for the secure and efficient management of structured information. DB2 is a hybrid relational and XML data server. A *data server* refers to a computer where the DB2 database engine is installed. The DB2 engine is a full-function, robust database management system that includes optimized SQL support based on actual database usage and tools to help manage the data. In this section, we will discuss the DB2 engine architecture and the database objects that are used to maintain your database.

1.3.1 DB2 9.7 threaded architecture and process model

Figure 1-2 shows the DB2 9.7 architecture overview. Starting with DB2 9.5, a threaded model architecture is used. There are many advantages to using a threaded model architecture. Firstly, a new thread requires less memory and fewer operating system resources than a process, because certain operating system resources can be shared among all threads within the same process. Moreover, on specific platforms, the context switch time for threads is less than that for processes, which can improve performance.

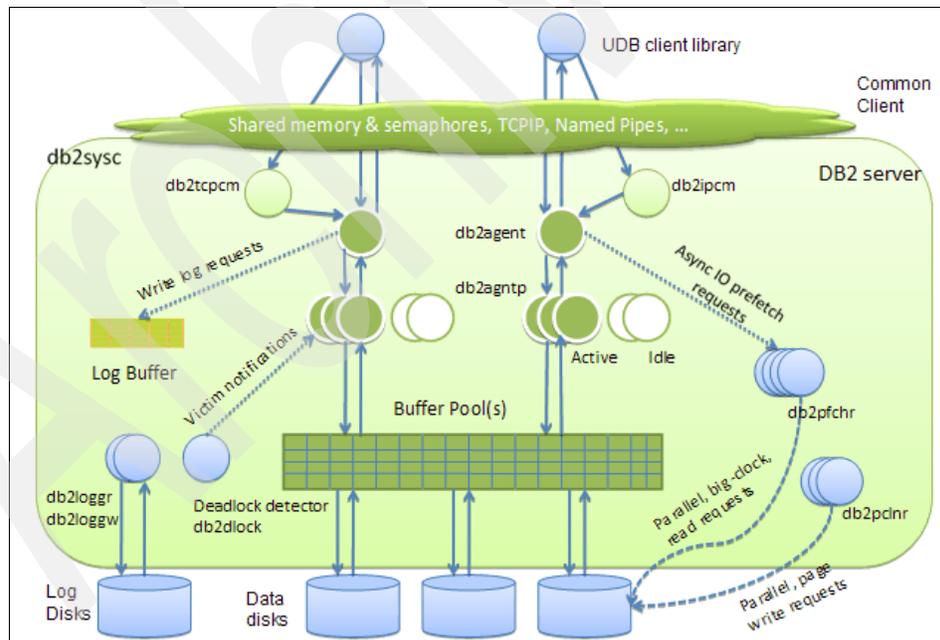


Figure 1-2 DB2 threaded architecture

From a client/server perspective, the client code and the server code are separated into separate address spaces. The application code runs in the client process, while the server code runs in a separate process. The client process can run on either the same machine as the data server or another machine, accessing the data server through a programming interface. The memory units are allocated for database managers, databases, and applications.

Because DB2 is running with a threaded architecture, all threads within the engine process share the same address space, meaning all threads can immediately see new memory allocations. This design creates a simplified memory model by allowing memory growth and shrinkage through control of a single memory parameter for an entire instance. This control is automatically performed by the *Self-Tuning Memory Manager*, which can also tune other memory parameters for best performance without DBA intervention. Its adaptive algorithm is able to react to unforeseen memory requirements in DB2 caused by workloads running against it.

To enable access to a specific database, the DB2 instance process responsible for the database must be running on the DB2 server. When an instance process is started, several processes are created, which interact with one another to maintain connected applications and the database. There are several background processes in DB2 that are pre-started; other processes start on a need-only basis. There are several important background processes:

- ▶ The main process is the *DB2 System Controller (db2sysc)*, which runs the entire DB2 engine infrastructure. The DB2 data server activities are performed by *Engine Dispatchable Units (EDU)*, which are defined as threads running within a single operating system process.
- ▶ A second DB2 background process is started together with the system controller and is called the *DB2 Watch Dog (db2wdog)*. Its responsibility is to watch and monitor the system controller and to react to error conditions.
- ▶ For autonomic tasks, another process is initialized when the database is activated, which can happen either manually or by being triggered by a client connection. This process is the *Autonomic Computing Daemon (db2acd)* and runs autonomic tasks, such as health-monitoring, auto-runstats, and the administration scheduler on the client side.
- ▶ There is also the *db2fmp* process, which, decoupled from the DB2 system controller process, serves thread-safe stored procedures and user-defined functions (UDFs).

1.3.2 DB2 database objects

In this section, we introduce the DB2 objects and their relationships to each other. Figure 1-3 on page 11 shows the fundamental DB2 database objects.

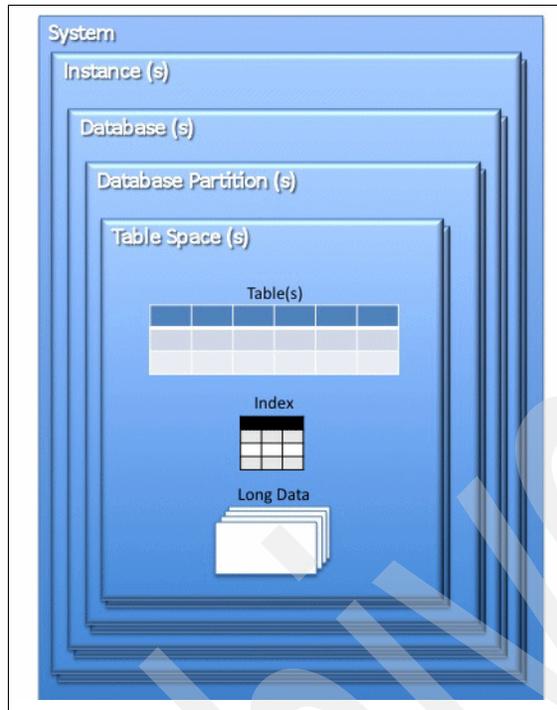


Figure 1-3 DB2 object relationships

These are a few major DB2 objects:

► **Instances**

A *DB2 instance* represents the database management system. It controls how data is manipulated and manages system resources assigned to it. Each instance is a complete, fairly independent environment, containing all the database partitions defined for a given parallel database system. An instance can have its own set of databases (which other instances cannot access directly), and all database partitions share the same system directories. Each instance has separate security from other instances on the same machine (system), allowing for situations where both production and development environments are run on the same machine without interference. In order to connect to a database, any database client must first establish a network connection to the instance.

► **Databases**

A *database* is a structured collection of data, which is stored within tables. Since DB2 9, data within tables can be stored as both relational data and XML documents natively in a pre-parsed tree format within a table column. Each database includes a set of system catalog tables that describes the

logical and physical structure of the object in the database, a configuration file containing the parameter values configured for the database, and a recovery log. Figure 1-4 shows the relationship between instances, databases, and tables.

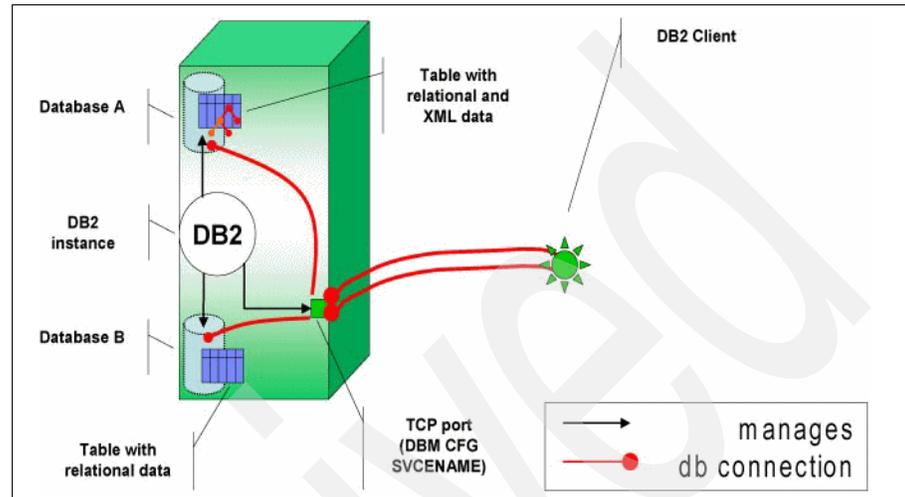


Figure 1-4 Relationship between instances, databases, and tables

► Database partition groups

A *database partition group* is a set of one or more database partitions (Figure 1-5 on page 13). A database partition group must be created prior to the creation of the tables in a database. This database partition group is where the table spaces will be stored. After the database partition group is there, a table space can be created where tables will be stored. If a partition group is not specified, there is a default group where table spaces are allocated. In a non-partitioned environment, all the data resides in a single partition; therefore, it is unnecessary to worry about partition groups in simple setups.

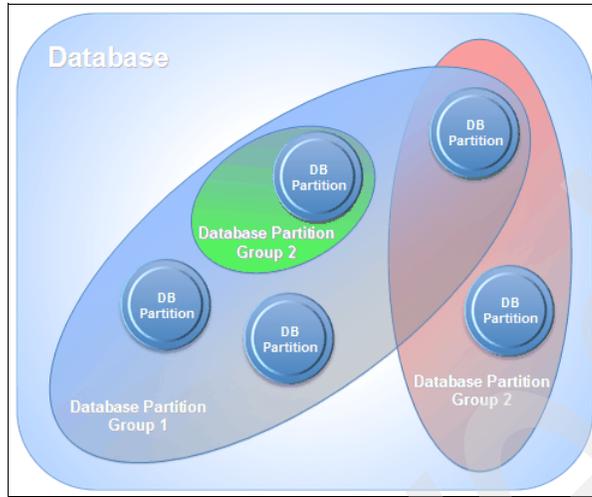


Figure 1-5 Database partition groups in a database

► **System catalog tables**

Each database includes a set of system catalog tables that describes the logical and physical structure of the data. DB2 creates and maintains an extensive set of system catalog tables for each database. These tables contain information regarding definitions of database objects, including user tables, views, indexes, and security information about the privileges that users have on these objects. Catalog tables are created when the database is created and are updated during normal operations. They cannot be explicitly created or dropped; however, they can be queried for their contents using the catalog views.

► **Table spaces**

A database is organized into subdivided *table spaces*, which store data. When creating a table, you can decide to have certain objects, such as indexes and large object (LOB) data, kept separately from the rest of the table data. A table space is equally spread over one or more physical storage devices, which is called *striping*. When using a database, if no additional table spaces are created by the user, the default user table space is used. DB2 allows you as much control as needed.

When table spaces are created, they reside in database partition groups. The table space definitions and attributes are maintained in the database system catalog. Each table space has at least one container assigned to it. A *container* is an allocation of physical storage, such as a file or a device.

Table spaces come in two types: system managed space (SMS) or database managed space (DMS), as shown in Figure 1-6 on page 14. In an SMS table

space, each container is a directory in the file system of the operating system. This type of table space allows the operating system's file manager to control the storage space. In a DMS table space, each container is either a re-sizable file or a pre-allocated physical device, such as a disk, which the database manager must control.

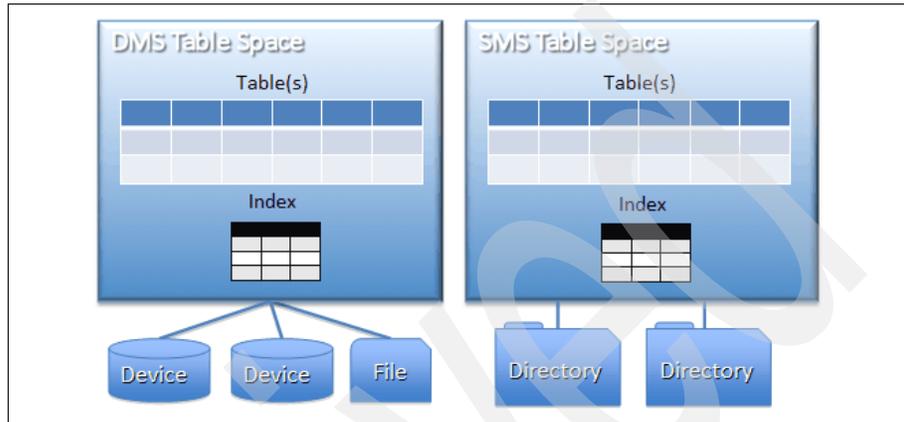


Figure 1-6 DMS/SMS table spaces and containers

When using SMS or DMS in combination with container files, you can choose how DB2 handles these files. For example, you can choose to enable various optimization features if supported by the operating systems, that is, *Direct I/O* (to bypass file system caching; always enabled with raw and block devices), *Vector I/O* (reading contiguous data pages from disk into contiguous portions of memory), and *Async I/O* (non-sequential processing of read and write requests across multiple disks to avoid delays from synchronous events).

When using the *Automatic Storage* feature in DB2, you can simply specify folders where the database can automatically create and manage DMS table spaces. When more space is required, the database manager automatically allocates more space. Table spaces can be automatically resized using this feature. This feature provides a convenient and worry-free operation scenario. You can perform manual operations without having to specify container files.

► Containers

A *container* is a physical storage device. It can be identified by a directory name, a device name, or a file name. A container is assigned to a table space. A single table space can span many containers, but each container can belong to only one table space.

► **Buffer pools**

A *buffer pool* is the amount of memory allocated to cache table and index data pages. The purpose of the buffer pool is to improve system performance. Think of it as a database-controlled file system cache. Data can be accessed much faster from memory than from disk. Therefore, the fewer times the database manager needs to read from or write to a disk (I/O) synchronously, the better the performance of the application. The size of the buffer pool is the single most important performance tuning area to help reduce the delay caused by synchronous I/O.

Buffer pool memory can be automatically tuned online in the same way as most other memory-related parameters. The feature responsible for automatic tuning is called the Self-Tuning Memory Manager, which allocates and releases new memory from the OS by shifting unused memory within DB2 to components.

► **Schemas**

A *schema* is an identifier, by default, the user ID, which qualifies tables and other database objects. A schema can be owned by an individual, and the owner can control access to the data and the objects within it. A schema name is used as the first part of a two-part object name. For example, a schema named Smith might qualify a table named SMITH.PAYROLL.

► **Tables**

A database presents data as a collection of *tables*. Data within a table is arranged in columns and rows. A table can contain XML documents that are natively stored as a parsed hierarchical format, as shown in Figure 1-7 on page 16. The data in the table is logically related, and relationships can be defined between tables. Table data is accessed by using *Structured Query Language (SQL)* or *XQuery with XPath* expressions. Both products are standardized query languages for defining and manipulating both relational and XML data in a database. A *query* is used in applications or by users to retrieve data from a database. A typical query for relational data uses SQL to create a statement in the form of:

```
SELECT <column_name> FROM <table_name>
```

A typical XQuery for the table that is shown in figure 1-7 looks like this query:

```
xquery
db2-fn:sqlquery("SELECT INFO FROM XMLEmployeeInfo
WHERE EmpID=1001")/customerinfo/name
```

This query allows us to iterate through all our table rows and return customer information from each XML document stored in each row for the example table that is outlined in Figure 1-7 on page 16. DB2 and the XQuery language also allow us to modify and update a subtree of an XML document in place without having to rewrite the whole document. This query is possible,

because we already store XML documents in a pre-parsed and hierarchical format:

```
db2 create table dept (EmpID int,..., XMLEmployeeInfo xml)
```

EmpID	...	XMLEmployeeInfo
1001	...	<pre><dept> <employee> <name>John Doe</name> <address><street>555 Bailey Ave</street> <city>...</city> <zip>...</zip> </address> </employee> </dept></pre>
1002	...	<dept><employee><name>Kathy Smith</name> ...
1003	...	<dept><employee><name>Jim Noodle</name> ...
...

Figure 1-7 Relational and XML data in a single table

When using the *Deep Data Row Compression* feature, DB2 is able to transparently compress and decompress table rows (for each table with compression turned on). This feature can effectively save 45-80% of the space on disk. Compressed rows in a table are compressed when pre-fetched to buffer pool memory and left in a compressed state until they are actually used. Although decompression of the data when it is fetched adds a slight overhead, I/O bound workloads will have a performance gain due to the reduced amount of data we actually need to read and write from or to disk, as well as saved memory.

► Views

A *view* provides another way of looking at data from one or more tables; it is a named specification of a result table. The specification is a SELECT statement that runs whenever the view is referenced in an SQL statement. A view has columns and rows just like a base table. All views can be used just like base tables for data retrieval. Figure 1-8 on page 17 shows the relationship between tables and views.

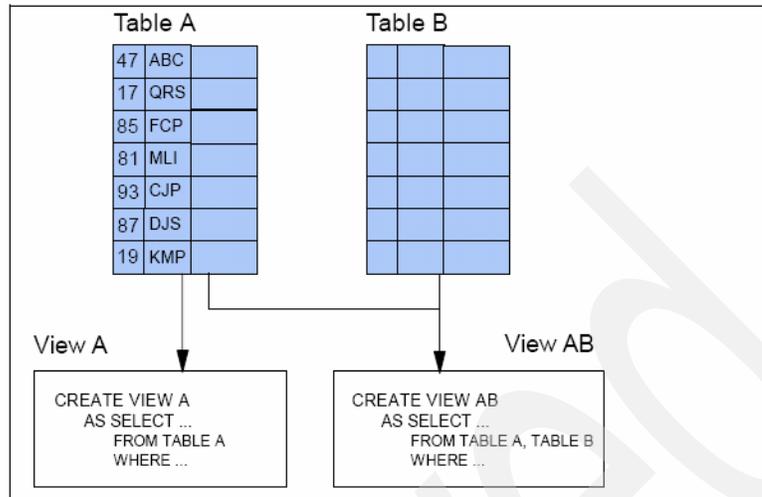


Figure 1-8 Relationship between tables and views

► Indexes

An *index* is a set of keys, each pointing to rows in a table. For example, table A has an index based on the first column in the table (Figure 1-9 on page 18). This key value provides a pointer to the rows in the table: value 19 points to record *KMP*. If searching for this particular record, a full table scan can be avoided, because we have an index defined. Except for changes in performance, users of this table are unaware that an index is being used. DB2 decides whether to use the index or not. DB2 also provides tools, such as the Design Advisor, that can help decide what indexes will be beneficial.

An index allows efficient access when selecting a subset of rows in a table by creating a direct path to the data through pointers. The DB2 SQL Optimizer chooses the most efficient way to access data in tables. The optimizer takes indexes into consideration when determining the fastest access path.

Indexes have both benefits and disadvantages. Be careful when defining indexes and take into consideration costs associated with update, delete, and insert operations and maintenance, such as reorganization and recovery.

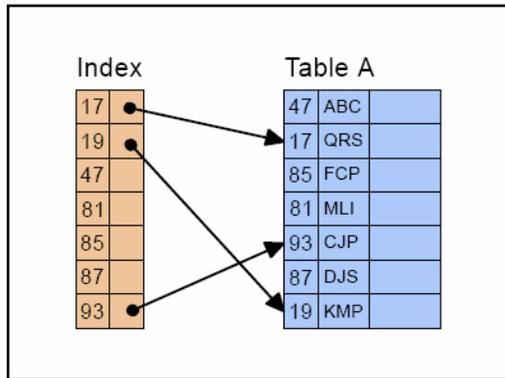


Figure 1-9 Relationship between indexes and tables

1.3.3 DB2 catalog

In DB2, the metadata is stored in a set of base tables and views called the *catalog*. The catalog contains information about the logical and physical structure of the database objects, object privileges, integrity information, and more.

The catalog is automatically created with the database. The base tables are owned by the *SYSIBM* schema and stored in the *SYSCATSPACE* table space. On top of the base tables, the *SYSCAT* and *SYSSTAT* views are created. *SYSCAT* views are the read-only views that contain the object information and are found in the *SYSCAT* schema. *SYSSTAT* views are views, which you can update, that contain statistical information that is found in the *SYSTAT* schema. You can obtain the complete DB2 catalog views in *DB2 SQL Reference Volume 1 and 2* available for download under the following link:

<http://www.ibm.com/support/docview.wss?rs=71&uid=swg27015148>

1.4 DB2 utilities

All DB2 system commands are installed in the `sql1lib/bin` directory during installation. Several of the most important commands in DB2 are listed in Table 1-1 on page 19, Table 1-2 on page 19, Table 1-3 on page 20, Table 1-4 on page 20, and Table 1-5 on page 20. You can obtain more information in the DB2 manuals, especially the *Command Reference* that is available at this link:

<http://www.ibm.com/support/docview.wss?rs=71&uid=swg27015148>

Table 1-1 DB2 instance commands

Command	Description	Example
db2start	Starts the default instance	db2start
db2stop	Stops the current instance	db2stop -f
db2icrt	Creates an instance	db2icrt -u db2fenc1 db2inst1
db2idrop	Drops an instance	db2idrop -f db2inst1
db2ilist	Lists all instance	db2ilist
db2imigr	Converts an instance after upgrading DB2	db2imigr -u db2fenc1 db2inst1
db2iupdt	Updates an instance after installation of a fix pack	db2iupdt -u db2fenc1 db2inst1

Table 1-2 Handling general database tasks

Description	Example
Deactivates a database	db2 deactivate db mydb
Views database manager settings	db2 get dbm cfg show detail
Changes a database manager setting	db2 update dbm cfg using health_mon off
Views database settings	db2 get db cfg show detail
Changes a database setting	db2 update db cfg using SELF_TUNING_MEM on
Views registry values	db2set
Changes registry parameters	db2set DB2AUTOSTART=yes
Views cataloged databases	db2 list db directory
Views cataloged nodes	db2 list node directory
Lists all connected applications	db2 list applications all
Forces applications off	db2 force application (41408, 55458)
Lists utilities	db2 list utilities
Gets a database snapshot	db2 get snapshot for database on mydb

Table 1-3 DB2 DAS instance commands

Command	Description	Example
db2admin	Starts and stops the DB2 Administration Server	db2admin start
dasauto	Autostarts DB2 Administration Server	dasauto -on
dasCRT	Creates a DB2 Administration Server	dasCRT -u dasusr1
dasdrop	Removes a DB2 Administration Server	dasdrop
dasmigr	Converts a DB2 Administration Server	dasmigr

Table 1-4 Informational commands

Command	Description	Example
db2level	Shows the current version and service level	db2level
db2look	Extracts DDL statements	db2look -d dep -a -e -o db2look.sql
db2dart	Database analysis and reporting tool	db2dart dbaddr
db2pd	Troubleshooting tool	db2pd -db sample -locks

Table 1-5 Graphical tools

Tool	Command	Purpose
IBM Data Studio		Integrated development environment package, which includes all administrative capabilities, as well as an integrated Eclipse development environment for Java, XML, and Web services
Optim Development Studio		A purchasable integrated development environment for advanced development of DB2 databases
Optim Database Administrator		A purchasable integrated development environment for advanced administration of DB2 databases
DB2 installer	db2setup	Installs DB2 and creates instances
DB2 instance installer	db2isetup	Creates instances
Control Center	db2cc	Administers instances, databases
Replication Center	db2rc	Administers replication between servers
Satellite Admin. Center	db2cc	Administers collections of DB2 servers

Command line processor (CLP)	db2 or db2cmd	Executes DB2 commands at the command line
Health Center	db2hc	Views/resolves health monitor alerts
Task Center	db2cc -tc	Schedules, runs tasks; notifies contacts
Journal	db2cc -j	Monitors jobs, recovery history, and so on
Configuration Assistant	db2ca	Configures instances and databases

Note: The Control Center and its associated components have been deprecated in Version 9.7 and might be removed in a future release. We recommend that you use the new suite of GUI tools for managing DB2 data and data-centric applications. These new tools include the IBM Data Studio, the Optim Development Studio, and the Optim Database Administrator.

1.5 DB2 database access

IBM introduced common application development and tooling support for DB2 for Linux, UNIX, and Windows (as of 9.5), DB2 for z/OS, and Informix Dynamic Server. This support generally lowers development costs through reuse and support of common components across IBM data servers.

More information to application development, tools, clients, and drivers can be obtained at this link:

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

and

<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0804zikopoulos>

1.5.1 DB2 clients and drivers

DB2 clients and drivers are used to access databases that reside on DB2 servers. A database cannot be created on a DB2 client.

IBM data server clients

DB2 Data Server 9.7 comes with the following clients:

- ▶ IBM Data Server Runtime Client
- ▶ IBM Data Server Client

The *IBM Data Server Runtime client* offers the basic client functionality and includes drivers for ODBC, CLI, ADO.NET, Object Linking and Embedding (OLE) DB, PHP, Ruby, Perl-DB2, JDBC, and SQLJ. This client already includes the drivers and the capabilities to define data sources. Furthermore, the Lightweight Directory Access Protocol (LDAP) is available, as well.

Additionally, the *IBM Data Server Client* provides vast amounts of sample code in various languages, header files for application development and graphical administration and development tools, such as the DB2 Control Center, the IBM Data Studio, the MS® Visual Studio® Tools, and more.

Figure 1-10 illustrates how to connect to a DB2 data server using the IBM data server clients.

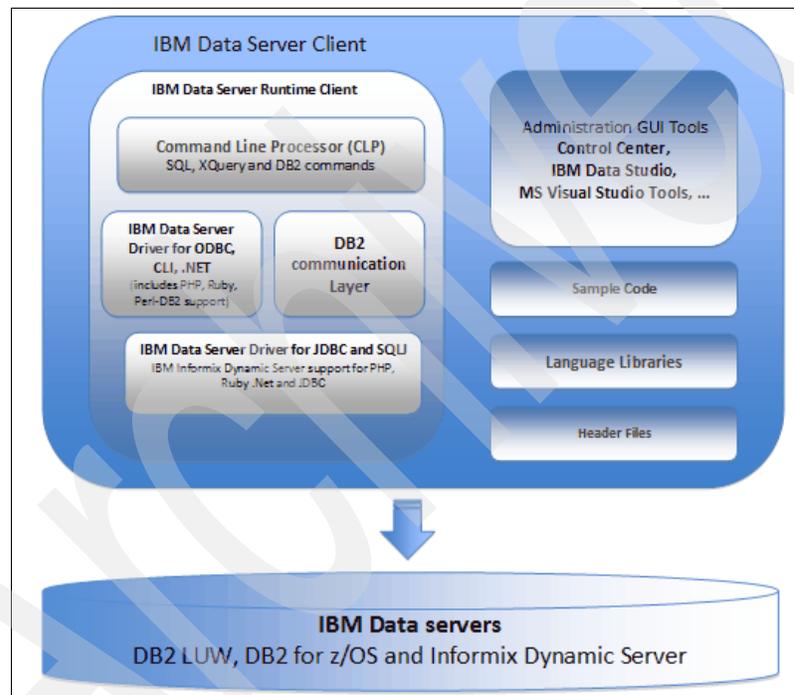


Figure 1-10 DB2 9.7 client and drivers

IBM Optim Data Studio is a comprehensive suite of integrated Eclipse tools geared toward both database developers and database administrators. It reduces the time to perform day-to-day administration tasks, create, deploy, and debug SQL and Java stored procedures, deploy data-centric Web services, and create queries for relational and XML data using SQL and XQuery. It supports multiple IBM data servers, including DB2 for Linux, UNIX, and Windows, DB2 for i5/OS® and z/OS, Apache Derby, and the Informix Dynamic Server. Because it is

built on the extensible Eclipse framework, this IDE includes a number of plug-ins to support programming languages, such as Java, C/C++, PHP, Ruby, Perl, and so on. Other plug-ins are available to maintain the written application sources in various source code repositories, for example, the Concurrent Versions System (CVS) or IBM Rational® ClearCase® from within IBM Optim Data Studio.

If running mixed versions of DB2 servers and clients, it is good to know that DB2 Clients from DB2 UDB Version 8 and DB2 9.1 or 9.5 for Linux, UNIX, and Windows are still supported and able to connect to a DB2 9.7 data server. In the reverse direction, the newer IBM data server clients from Version 9.7 can also connect to the earlier DB2 9.1 and DB2 UDB Version 8 servers using the IBM Data Server Driver for ODBC, CLI, and .Net. In this case, however, new DB2 Version 9.7 functionality is not available.

IBM data server drivers

The IBM data server drivers include the products:

- ▶ IBM Data Server Driver for JDBC and SQLJ
- ▶ IBM Data Server Driver for ODBC, CLI, and .NET

As of DB2 Version 9.5, both clients and drivers are decoupled from the server release schedule and can be downloaded separately. The *IBM Data Server Driver for JDBC and SQLJ* is already included in the IBM Data Server Runtime Client. It provides support for JDBC 3 and 4 compliant applications, as well as for Java applications using static SQL (SQLJ). Support is also provided for pureXML, SQL/XML, and XQuery. All of this support and other features, such as connection concentration, automatic client reroute, and more, are provided within in a single package called `db2jcc4.jar`. The *IBM Data Server Driver for ODBC, CLI and .Net* is a lightweight deployment solution for Windows applications to provide runtime support for applications without needing to install the Data Server client or the Data Server Runtime Client. On Windows, the driver comes as an installable image including merge modules to easily embed it in a Windows installer-based installation. On Linux and UNIX, there is another easy deployment solution called the IBM Data Server Driver for ODBC and CLI, which is available in tar format.

Communication protocols

DB2 primarily uses these protocols to communicate:

- ▶ TCP, IPv4, IPv6, and Named Pipes (Windows only) for remote connections
- ▶ Interprocess Communication (IPC) for local connections within a DB2 instance

For client/server communication, DB2 supports TCP/IP and Named Pipes for remote or local loopback connections and uses IPC for client connections, which are local to the DB2 server instance. Local and remote DB2 connections are illustrated in Figure 1-11.

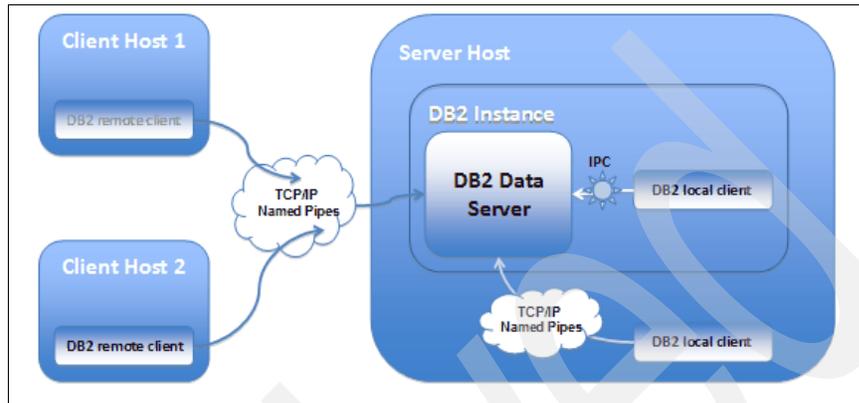


Figure 1-11 Client connection scenario

Protocols are automatically detected and configured during an instance creation. The `DB2COMM` registry variable identifies the protocol detected in a server. To enable a specific protocol on the server, the `db2set db2comm` command must be executed. For TCP/IP, a unique port address has to be specified in the database manager configuration. This port is registered in the services file (usually `/etc/services` on UNIX and Linux). For example, to reserve port 50000 with the service name `db2icdb2`, the entry in the services file is:

```
db2icdb2 50000/tcp
```

From the command line, this information can be then updated in the database manager with the following DB2 command:

```
db2 UPDATE DBM CFG USING SVCENAME db2icdb2
```

These tasks can also be performed using the DB2 Configuration Assistant utility.

Client/server configuration methods

The following methods are available for client/server configuration:

- ▶ Statically using command line processor (CLP) or Configuration Assistant (Discovery Service available)
- ▶ Dynamically using Lightweight Directory Access Protocol (LDAP)

At the client side, the database information is configured using either the `CATALOG` command or using the Configuration Assistant. The databases are

configured under a node, which describes host information, such as protocol use, port number, and so on. To configure a remote TCP/IP node, use the following command:

```
db2 CATALOG TCP/IP NODE node-name REMOTE host-name SERVER service-name
```

The service name registered in the server or the port number can be specified in the SERVER option. To catalog a database under this node, the command used is:

```
db2 CATALOG DATABASE database-name AS alias-name AT NODE node-name
```

When using the Configuration Assistant GUI tool to add a database connection, a database discovery can be started to find the desired database.

For information about how to enable and configure DB2 in an LDAP environment, have a look at the *Lightweight Directory Access Protocol (LDAP)* section within the “Database fundamentals” chapter in the DB2 Information Center. The DB2 Information Center contains searchable and structured Web pages and can be installed locally. Additionally, the Information Center is also publicly available online at this Web site:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp>

Note: DB2 Discovery method is enabled at the instance level using the DISCOVER_INST parameter, and at database level using the DISCOVER_DB parameter.

1.5.2 Application access

You can use various methods when deploying applications with DB2 Data Server 9.7:

- ▶ Single-tier

In this configuration, the application and the database reside on the same system. In enterprise environments, it can be rare to see such a configuration, because remote access to a database server is typically required. Nonetheless, this method is quite common for developing applications that can later be deployed transparently in a multi-tier DB2 environment without any changes or batch applications.

- ▶ Client/Server or 2-tier

The application and the database reside on separate systems. The machines where the application runs typically have a DB2 client installed, which communicates over the network to a database server. For the application, the physical location of the data is transparent. The application communicates with the DB2 client using a standard interface (for example, ODBC) and the

DB2 client takes over the task of accessing the data over the network. In certain cases, such as browser-based access or Java-based access, it is not necessary to have the DB2 client running on the same machine where the application executes.

DB2 provides exceptional flexibility for mixing and matching client and server platforms in a heterogeneous environment. DB2 client and server code is available for a wide variety of platforms. For example, the application can execute on a Windows-based machine with a DB2 client for Windows, which can then access a DB2 database on a Linux server. Likewise, the Linux machine can act as a client and access data from UNIX servers or mainframes.

► **Multi-tier**

In a multi-tier configuration, the application, DB2 client, and the data source typically reside on separate systems. Table 1-6 provides examples of these configuration scenarios.

Table 1-6 Multi-tier configuration examples

Client	Middle-tier	Server
Web-browser	Web server	DB2 database
Application client	Application server DB2 client	DB2 database server 1 DB2 database server 2
Application DB2 Client	DB2 Connect Gateway	System z®, System i®
Application DB2 Client	DB2 server	Secondary data sources (for example, Mainframe DB2, Non-DB2, and non-relational)

IBM recognizes that in many cases there might be a need for accessing data from a variety of distributed data sources rather than one centralized database. The data sources can be from IBM, such as DB2 or Informix, from non-IBM databases, such as Oracle®, or even from non-relational data, such as files or spreadsheets. As illustrated in the last scenario in Table 1-6, IBM offers the most comprehensive business integration solution by allowing federated access to a variety of distributed data sources.

1.5.3 DB2 application programming interfaces

In order to access or manage DB2 objects, several application programming interfaces (APIs) can be used, as shown in Figure 1-12 on page 27.

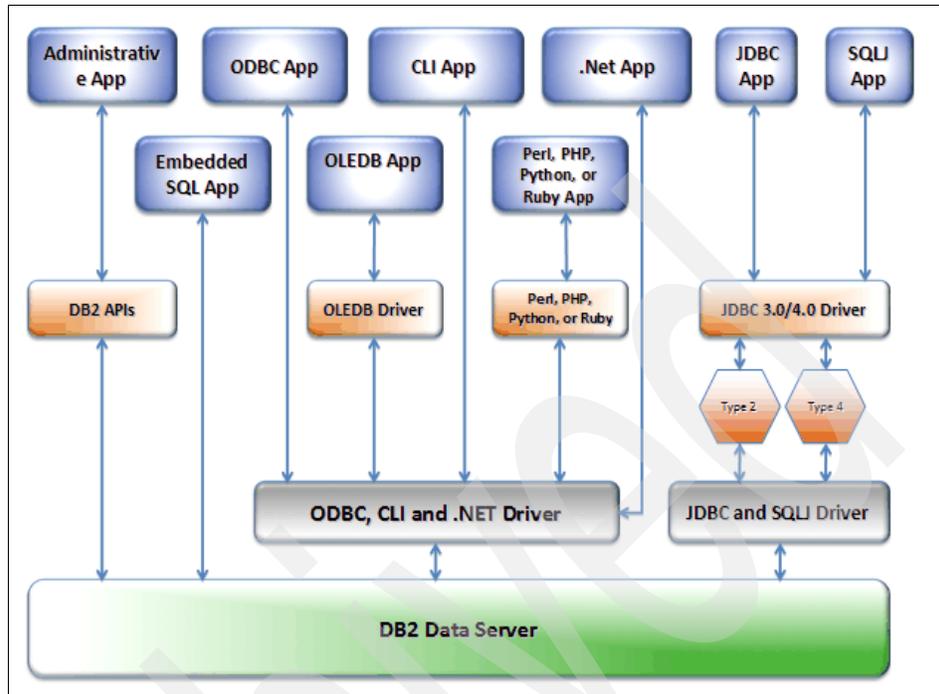


Figure 1-12 Application connections to DB2

DB2 administrative application programming interface

DB2 provides numerous administrative APIs, which allow applications to perform database administration tasks available in DB2 Control Center, such as importing and exporting data, or creating, activating, backing up, or restoring a database. These calls can be included within embedded SQL and DB2 CLI applications. Examples of API programs can be found in the DB2 home directory `sql11ib/sample/` for various programming languages. For additional information, refer to *DB2 Administrative API Reference*, SC23-5824.

Embedded SQL statements in applications

Two types of SQL statements have to be distinguished:

► Static SQL statements

With static SQL statements, the SQL statement type and the table and column names are known before compilation time. The specific data values for which the statement is searching are not known. These values can be represented in host language variables.

Before compiling and linking the program, pre-compiling and binding of the embedded SQL statements must be performed. Pre-compiling converts

embedded SQL statements into DB2 runtime API calls that a host compiler can process to create a bind file. The **bind** command creates a package in the database. This package then contains the SQL operation and the access plan that DB2 will use to perform the operations.

► **Dynamic SQL**

Dynamic SQL statements in an application are built and executed at run time. For a dynamically prepared SQL statement, the syntax has to be checked and an access plan has to be generated during the program execution.

Examples of embedded static and dynamic SQL can be found in the DB2 home directory: `sql11ib/samples/`.

DB2 call level interface

DB2 call level interface (DB2 CLI) is a programming interface that can be used from C and C++ applications to access DB2 data servers. DB2 CLI is based on the Microsoft® Open Database Connectivity (ODBC) specification and the International Organization for Standardization (ISO) CLI standard. The DB2 CLI library can be loaded as an ODBC driver by an ODBC driver manager. DB2 CLI includes support for many ODBC and ISO SQL/CLI functions, as well as DB2 specific functions. Figure 1-13 illustrates the ODBC driver manager environment and the DB2 CLI environment.

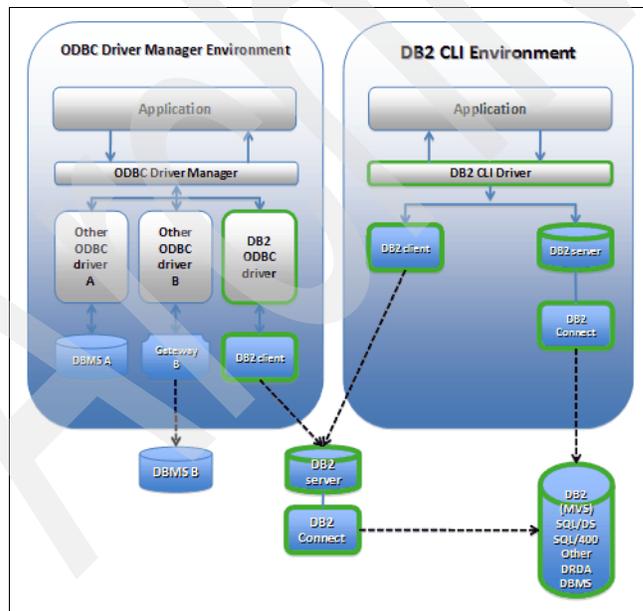


Figure 1-13 ODBC/CLI driver connectivity

When using DB2 CLI, the application passes dynamic SQL statements as function arguments to the database manager for processing. Because of this design, applications use the common access packages that are provided with DB2. DB2 CLI applications do not need to be pre-compiled or bound. Only compiling and linking the application are needed. Before DB2 CLI or ODBC applications can access DB2 databases, the DB2 CLI binds files, which come with the IBM Data Server Client, to each DB2 database that will be accessed. This binding occurs automatically with the execution of the first statement.

Typically, when building an ODBC application, an ODBC driver manager is needed, which is provided by platform vendors, such as Microsoft and others. An ODBC driver manager is available for Linux at this Web site:

<http://www.unixodbc.org/>

In environments without an ODBC driver manager, DB2 CLI is a self sufficient driver, which supports a subset of the functions provided by the ODBC driver. Examples of C programs using CLI calls can be found in the DB2 home directory: `sqllib/samples/cli`. For additional information regarding CLI, refer to *Call Level Interface Guide and Reference, Volume 1* and *Volume 2*, SC23-5844 and SC23-584.

Java database connectivity application

DB2 Java support includes Java database connectivity application (JDBC), a vendor-neutral dynamic SQL interface that provides data access to the application through standardized Java methods. For detailed information about the Java support, we strongly recommend the following articles and publications:

<http://www.ibm.com/support/docview.wss?uid=pub1sc23585301>

<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0804zikopoulos/>

Similar to DB2 CLI, pre-compiling or binding a JDBC program is not necessary. As a vendor-neutral standard, JDBC applications offer increased portability. The JDBC API, which is similar to the CLI/ODBC API, provides a standard way to access databases from Java code. The Java code passes SQL statements to the DB2 JDBC driver, which handles the JDBC API calls. Java's portability enables the delivery of DB2 access to clients on multiple platforms, requiring only a Java-enabled Web browser or a Java Runtime Environment (JRE)[™].

DB2 Data Server 9 offers multiple ways of creating Java applications, by either using a type 2 or type 4 JDBC driver:

- ▶ **Type 2 driver**

With a type 2 driver, calls to the JDBC application driver are translated to Java native methods. The Java applications that use this driver must run on a DB2 client, through which JDBC requests flow to the DB2 server.

Tip: If prototyping CLI calls before placing them in a program, use the `db2cli.exe` (Windows) or `db2cli` (Linux) file in the `sqllib/samples/cli` directory.

Note: The type 2 driver is still included, but it was deprecated. We recommend that you convert any existing applications to use the type 4 driver.

Figure 1-14 illustrates the Java type 2 driver connectivity.

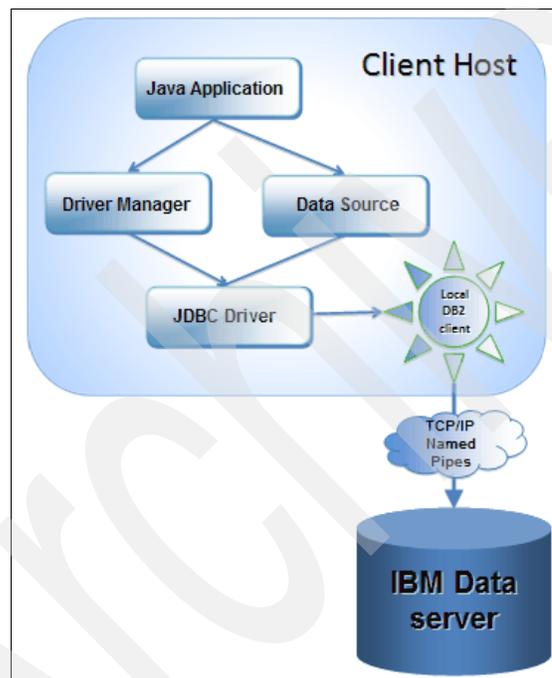


Figure 1-14 Java type 2 driver connectivity

► Type 4 driver

The *JDBC type 4 driver* can be used to create both Java applications and applets. To run an applet that is based on the type 4 driver, a Java-enabled browser is required, which downloads the applet and the JDBC driver (`db2jcc4.jar`). To run a DB2 application with a type 4 driver, an entry for the JDBC driver in the class path is required and no DB2 client is required.

The JDBC driver is included in the IBM Data Server Client Driver for JDBC and SQLJ and is architected as an abstract JDBC processor that is independent of driver-type connectivity or target platform. Examples of JDBC calls can be found in `sql11ib/samples/java/jdbc`.

Figure 1-15 illustrates the java type 4 driver connectivity environment.

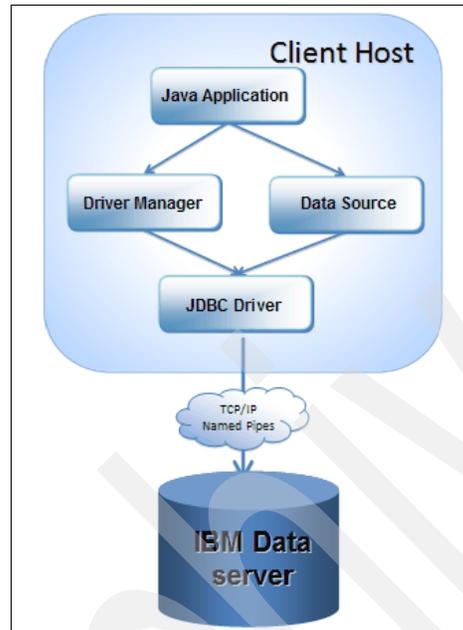


Figure 1-15 Java type 4 driver connectivity

Embedded SQL for Java

DB2 Java embedded SQL (SQLJ) support is provided by the IBM Data Server Runtime Client. With DB2 SQLJ support, in addition to DB2 JDBC support, SQLJ applets, applications, and stored procedures can be built to contain static SQL and use embedded SQL statements that are bound to a DB2 database.

SQLJ applications use JDBC as a foundation for tasks, such as connecting to databases and handling SQL errors, but also contain embedded static SQL statements in separate SQLJ source files. Unlike any other languages that can contain embedded SQL (COBOL, C, and C++), the Java code is not precompiled, instead the SQLJ translator converts SQLJ clauses into JDBC statements. Because SQLJ shares its underlying connection with that of JDBC applications, it can connect to DB2 using either a type 2 or type 4 driver. Examples of SQLJ calls can be found in `sql11ib/samples/java/sqlj`.

ActiveX Data Objects and Object Linking and Embedding (Windows only)

DB2 supports ActiveX® Data Object (ADO) applications that use the Microsoft Object Linking and Embedding (OLE) DB to ODBC bridge. ADOs provide the ability to write applications to access and manipulate data in a database server through an OLE DB provider. Therefore, applications will have uniform access to data that is stored in diverse information sources.

Remote Data Objects (RDO) provide an information model for accessing remote data sources through ODBC. RDO offers a set of objects that makes it easy to connect to a database, execute queries and stored procedures, manipulate results, and commit changes to the server. Because RDO implements a thin code layer over the ODBC API, it requires an ODBC data source to be created for the DB2 database to which you connect.

ADO.NET

DB2 supports the Microsoft ADO.NET programming interface through a native managed provider. These applications can use the DB2 .Net, the OLE DB .Net, or the ODBC .NET data provider. High performing Windows Forms, Web Forms, and Web Services can be developed using the ADO.NET API. DB2 supports a collection of features that integrate seamlessly into Visual Studio 2003, 2005, and 2008 to make it easier to work with DB2 servers and to develop DB2 procedures, functions, and objects.

The IBM Data Server Provider for .Net extends data server support for the ADO.NET interface and delivers high performing, secure access to IBM data servers:

- ▶ DB2 Version 9 (or later) for Linux, UNIX, and Windows
- ▶ DB2 Universal Database™ Version 8 for Windows, UNIX, and Linux
- ▶ DB2 for z/OS and OS/390® Version 6 (or later), through DB2 Connect
- ▶ DB2 for i5/OS Version 5 (or later), through DB2 Connect
- ▶ DB2 Universal Database Version 7.3 (or later) for VSE and VM, through DB2 Connect
- ▶ IBM Informix Dynamic Server, Version 11.10 or later
- ▶ IBM UniData®, Version 7.1.11 or later
- ▶ IBM UniVerse, Version 10.2 or later

When used in conjunction with stored procedures and the federated database capabilities of DB2 data servers and DB2 Connect servers, this data access can be extended to include a wide variety of other data sources, including non-DB2 mainframe data and Informix Dynamic Server (IDS), Microsoft SQL Server®,

Sybase, and Oracle databases, as well as any data source that has an OLE DB Provider available.

For more information about developing ADO.NET and OLE DB, refer to *DB2 for Linux, UNIX, and Windows Developing ADO.NET and OLE DB Applications*, SC23-5851-01, which is available at this Web site:

<http://www.ibm.com/support/docview.wss?uid=pub1sc23585101>

Perl DBI

DB2 supports the Perl Database Interface (DBI) specification for data access through the DBD::DB2 driver. The Perl DBI module uses an interface that is similar to the CLI and JDBC interfaces, which makes it easy to port Perl prototypes to CLI and JDBC. As of DB2 9.5, Perl DBI comes with support for DB2 pureXML technology, which allows you to insert XML documents without the need to parse or validate XML. The Perl driver also supports multi-byte character sets, which means your application does not have to deal with the conversion itself when interacting with the database.

You can obtain more information about Perl DBI at this Web site:

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

PHP

The *PHP Hypertext Preprocessor* is a modular and interpreted programming language intended for the development of Web applications. Its functionality can be customized through the use of extensions. DB2 supports PHP through an extension called `pdo_ibm`, which allows DB2 access through the standard PHP Data Objects (PDO) interface. In addition, the `ibm_db2` extension offers a procedural API that, in addition to the normal create, read, update, and write database operations, also offers extensive access to the database metadata. The most up-to-date versions of `ibm_db2` and `pdo_ibm` are available from the PHP Extension Community Library (PECL):

<http://pecl.php.net/>

For more information about the PHP application development support that DB2 data server for Linux, UNIX, and Windows offers, refer to this Web site:

<http://www.ibm.com/software/data/db2/ad/php.html>

Ruby on Rails

DB2 9.5 has drivers for Ruby, which is an object-oriented programming language. Combined with the open source Ruby framework called Rails, the development of Web-based and database-driven applications can be extremely quick. Included in DB2 is the `IBM_DB` Ruby adapter, which allows any database-backed Ruby application to interface with IBM data servers.

For more information about IBM Ruby projects and the RubyForge open source community, refer to the following Web site:

<http://rubyforge.org/projects/rubyibm/>

Archived

MySQL database

When planning a conversion project, it is critical to understand how the source data server operates to successfully convert all functionality to the destination data server. The goal of this chapter is to discuss the MySQL database architecture and design, while keeping in mind how the features will be converted to DB2.

This chapter discusses the following MySQL details:

- ▶ License and origin
- ▶ Architecture overview
- ▶ Data handling
- ▶ Utilities

2.1 MySQL licensing overview

MySQL is a dual-licensed relational database management system developed and distributed by MySQL AB (<http://www.mysql.com>), which is a company that builds its business by providing services around MySQL. In late 2007, MySQL AB was acquired by Sun™ Microsystems, which was later acquired by Oracle in April of 2009. At the time of writing this book, the acquisition of Sun Microsystems by Oracle is still ongoing and the future of the MySQL licensing model is still uncertain. Currently, MySQL implements a dual licensing model. The first is a General Public License (GPL), the “open source” license, which is mostly linked to open source projects. That is, if you implement this licensing model and distribute MySQL with applications, the application source code must be open source as well. The second licensing model is for commercial use covering all other environments outside of the purely open source and non-commercial environments. The commercial license has a significant price tag equal to the price of other commercial database products.

DB2 offers a no-charge community edition (DB2 Express-C) of the DB2 data server. This edition of DB2 is completely free to develop, to deploy, and to distribute. DB2 Express-C is a full-function relational and XML data server and has the same reliability, flexibility, and power of the higher editions of DB2. DB2 also offers the DB2 Express + Fix Term license option, which is priced comparably with the MySQL Enterprise Gold pricing. For more details about each of the available DB2 editions, refer to 1.2.1, “DB2 Data Server Editions for the production environment” on page 3.

MySQL was initially developed for UNIX and Linux applications. It became popular when Internet Service Providers (ISPs) discovered that MySQL can be offered at no charge to their Internet customers, providing all of the storage and retrieval functionality that a dynamic Web application needs. It was also advantageous, because ISPs primarily use Linux or UNIX as their base operating system, in combination with APACHE as their favorite Web server environment. Today, MySQL is also used as an integrated or embedded database for various applications running on almost all platform architecture.

2.2 MySQL architecture overview

MySQL is built on a client/server architecture model that can be installed on Windows, UNIX, Macintosh, and Linux platforms. The server is the host that manages the database objects and data, whereas the client requests and works with the actual data. MySQL client programs connect to the MySQL server either through a local or remote connection. If the client is running remotely, MySQL

always connects using TCP/IP. If the client is running on the local server, any of the supported connection protocols can be used.

Figure 2-1 illustrates the conceptual architecture of the MySQL database. The next several sections cover the functionality of the integrated components in more detail.

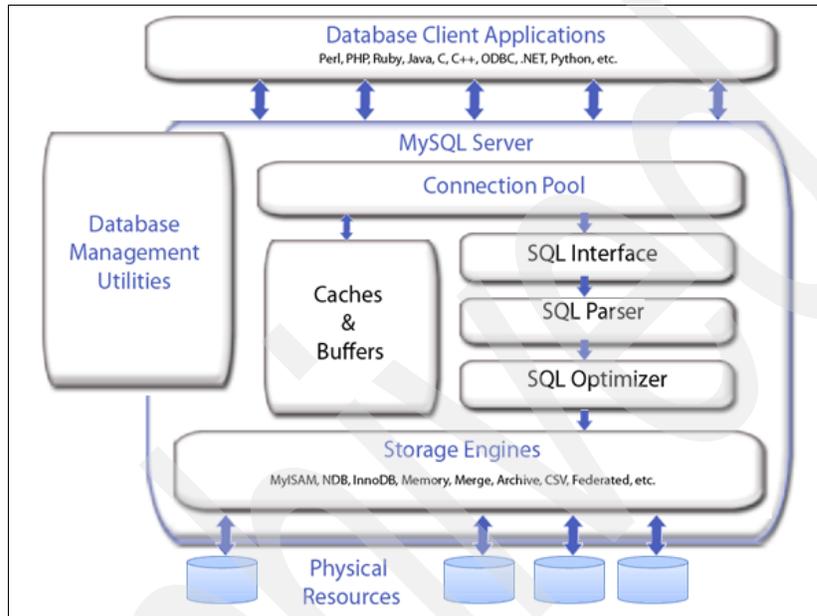


Figure 2-1 Conceptual MySQL architecture

2.2.1 Database client and non-client utilities

The client layer represents the interface between the user and the database. Most components are provided by MySQL AB in the server installation bundle. The graphical tools need to be downloaded and installed separately. MySQL clients include these tools:

- ▶ Query interface
- ▶ Administrative interface and utilities
- ▶ Application interface and utilities

The client layer is the front-end component with which users will interact. This component presents three types of users that interact with the server: query users, administrators, and applications.

Query users

Query users can interact with the database server through a query interface called *mysql*, which allows users to issue SQL statements and view the results returned from the server using the command line. There is also a graphical tool called MySQL Query Browser that provides a graphical interface to create and execute database queries.

In DB2, you can use the command line process for the same functionality. Use the **db2** or **db2cmd** command to start the command line processor.

Administrators

Administrators use the administrative interface and utilities, such as **mysqladmin** and *MySQL Administrator*. These tools can be used for creating or dropping databases and users, as well as managing the MySQL server. These tools connect to the database server using the native C client library. There are also utilities that can be used for administrative purposes, but they do not connect to the MySQL server. Instead, they work directly with the database files. These tools are **myisamchk** for table analysis, optimization, and crash recovery and **myisampack** for creating read-only compressed versions of MyISAM tables.

DB2 offers a rich set of database management GUI tools, such as the DB2 Control Center, the Optim Database Administrator, and the IBM Optim Data studio. These tools simplify database administration by providing one single tool to completely manage your entire database environment. Also, you can use these tools to query the database. The GUI tools are discussed in detail in 9.8, “Database management tools” on page 312.

Applications

Applications communicate with the database server through MySQL APIs that are available for various programming languages, such as C++, PHP, Java, Perl, .NET, and so on. We discuss these APIs in more detail later in the chapter.

2.2.2 Database server

The database server represents the core functionality of the database architecture. It is responsible for storing, accessing, and managing the databases and database objects. MySQL server is multi-threaded, allowing multiple users to access multiple databases. Figure 2-1 on page 37 shows the components of the MySQL server. As shown in Table 2-1 on page 42, you can find the following major components in MySQL:

- ▶ Connection pool
- ▶ SQL interface
- ▶ Parser

- ▶ Optimizer
- ▶ Caches and buffers
- ▶ Database management utilities
- ▶ Storage engines
- ▶ Physical resources

Connection pool

The *connection pool* assigns user connections to the database and corresponding memory caches. The utilities and programs that are included with MySQL connect using the Native C API. Other applications can connect using the standard drivers that MySQL offers, such as C++, PHP, Java, Perl, .NET, and so on. MySQL supports TCP/IP, UNIX socket file, named pipe, and shared memory networking protocols, depending on the type of operating system that is used. For more details about application programming interfaces, see 2.5, “MySQL application programming interfaces” on page 51.

SQL interface

The *SQL interface* accepts and conveys the SQL statements from the connecting user or MySQL application. This layer is independent of the storage ending layer, and, therefore, SQL support statements are not dependent on the type of storage engine being used. The SQL statement is then passed to the SQL parser for further processing.

SQL parser

The *parser* analyzes the SQL statements and validates the SQL query syntax. The parser breaks up the statement and creates a parse tree structure to validate the SQL query syntax and prepare the statement for the optimizer.

SQL optimizer

The *SQL optimizer* verifies that the tables exist and that the records can be accessed by the requesting user. After security checking, the query is analyzed and optimized to improve the performance of the query process.

Cache and buffers

Caches and buffers are used to increase the speed of the server by decreasing the amount of data that the MySQL server needs to read from disk. The memory caches are used for both global and engine specific caches. MySQL uses a query cache to increase the performance of data queries. The query cache holds both the query and the result of the query. There is also a cache to store the table descriptors and a cache to hold the host name. MySQL stores the user account information in the grant table buffers and the MyISAM index blocks in the key buffer.

Database management utilities

Database management utilities assist users and DBAs with managing and administering database objects. The majority of these tools are independent of the storage system that is being used. The following major components make up the database management utilities:

► Administration

Administrative utilities are used to help manage the database server, database objects, and data. Several of the utilities that are available with MySQL can assist with compressing MyISAM tables, managing MyISAM log files, managing and repairing tables, checking user privileges, and converting tables' storage engine.

► Backup and recovery

Backups can be used for restoring a system crash, for updating hardware, or for replicating a test or development environment. MySQL has multiple techniques to back up a database. If an SQL-level backup is needed, the `SELECT INTO` command can be used to output the table data to a file. The `mysqlhotcopy` script can be used to make a fast copy of a database or an individual table. The `mysqldump` utility can be used to back up a single database or multiple databases.

► Replication

Replication allows for a secondary or subordinate server, called the slave, to maintain a copy of the master server's database without having to perform a full backup and recovery. The slave database is updated with only the recent changes that occurred on the master database. There can be multiple slave systems.

Database management utilities also include the following components:

- Security
- Configuration
- Conversion and metadata

Pluggable storage engine

The MySQL storage engine is responsible for accessing and modifying the stored data. MySQL supports multiple storage engines, each with its own characteristics. The DBA can choose a specific storage engine depending on how the data is going to be accessed and used. Storage engines are allocated on a table basis. It is possible to mix and match various storage engines within a database. If no storage engine is specified, the default MyISAM storage engine will be used. We discuss each storage engine in 2.3.2, "MySQL storage engines" on page 45.

Physical resource

This is the bottom layer of the MySQL architecture and represents the secondary storage or physical disk. This layer is accessed through the storage engines to store or retrieve data. These types of data are stored in this layer:

- ▶ Data files (user data)
- ▶ Data dictionary (metadata)
- ▶ Indexes
- ▶ Log information
- ▶ Statistical data

In the next section, we describe how the database objects and data are physically stored on the server.

2.3 MySQL design and SQL compliance

In this section, we discuss the following topics:

- ▶ MySQL directory structure
- ▶ MySQL storage engines
- ▶ MySQL standard SQL compliance

2.3.1 MySQL directory structure

MySQL can be installed as one or more databases within one server. For each database, a directory is created with the same name as the database. This directory holds all of the database data and indexes. On Linux distributions, when installing using Red Hat Package Manager (RPM) packages, the default directory for the database and log files is `/var/lib/mysql`. Installation of MySQL can also be performed using source distribution, where the default directory and location of all MySQL files can be chosen. For UNIX distributions, the databases are stored under `/usr/local/mysql/data`. For Windows, the databases are stored under `C:\Program Files\MySQL\MySQL Server 5.1\data`. Table 2-1 on page 42 shows the default installation directories for MySQL Windows, Linux, and UNIX installations.

Table 2-1 MySQL default directories

Contents	Windows default directory	Linux default directory	UNIX default directory
Default installation directory	C:\Program Files\MySQL\MySQL Server 5.1		/usr/local/mysql
Databases and log files	<Default Dir>\data	/var/lib/mysql	<Default Dir>/data
Client programs and scripts	<Default Dir>\bin	/usr/bin	<Default Dir>/bin
mysqld server	<Default Dir>\bin	/usr/sbin	<Default Dir>/bin
Manual	<Default Dir>\Docs	/usr/share/info	<Default Dir>/docs
Example scripts	<Default Dir>\examples		
Include files	<Default Dir>\include	/usr/include/mysq l	<Default Dir>/include
Libraries	<Default Dir>\lib	/usr/lib/mysql	<Default Dir>/lib
Utilities and scripts	<Default Dir>\scripts		<Default Dir>/scripts
Error message files	<Default Dir>\share	/usr/share/mysql	<Default Dir>/share/mysql

In the example in Figure 2-2 on page 43, there are two databases on this MySQL server. The first database is the *mysql* database, which, by default, holds the security information. The second database is the sample database *inventory*, which is discussed in more detail in Chapter 4, “Conversion scenario” on page 75.

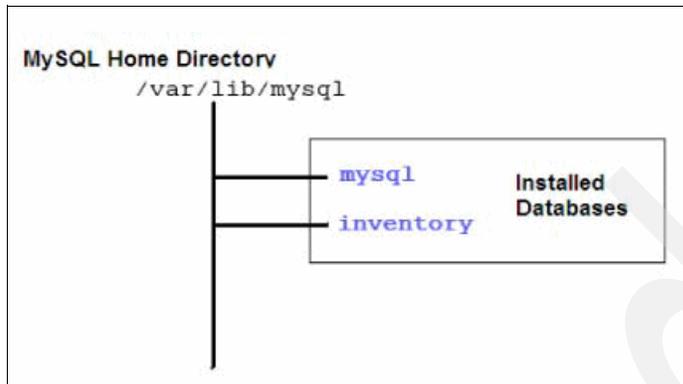


Figure 2-2 MySQL example directory structure

The following files are created for each database directory in the MySQL home directory:

- ▶ Files with the frm extension contain the structural definition of the table and the view, which is known as the *schema*.
- ▶ Files with the MYD extension contain the table data.
- ▶ Files with the MYI extension contain the table indexes.
- ▶ If there are triggers, there also are files with the TRN and the TRG extensions.

Example 2-1 shows the files that are created for each table in our sample database.

Example 2-1 MySQL example table files

```

mysqlServer:/var/lib/mysql/inventory # ls -lrt
total 428
-rw-rw---- 1 mysql users 8670 Jul 21 23:11 locations.frm
-rw-rw---- 1 mysql users 65 Jul 21 23:11 db.opt
-rw-rw---- 1 mysql users 4976 Jul 21 23:15 locations.MYD
-rw-rw---- 1 mysql users 4096 Jul 21 23:18 locations.MYI
-rw-rw---- 1 mysql users 8918 Aug 7 13:21 owners.frm
-rw-rw---- 1 mysql users 8804 Aug 7 13:29 inventory.frm
-rw-rw---- 1 mysql users 1584 Aug 7 14:21 empGroup.frm
-rw-rw---- 1 mysql users 1585 Aug 7 14:22 techGroup.frm
-rw-rw---- 1 mysql users 1587 Aug 7 14:23 bossGroup.frm
-rw-rw---- 1 mysql users 8746 Aug 7 17:23 groups.frm
-rw-rw---- 1 mysql users 1595 Aug 10 01:42 managerGroup.frm
-rw-rw---- 1 mysql users 1596 Aug 10 01:43 generalGroup.frm
-rw-rw---- 1 mysql users 1587 Aug 10 02:58 testGroup.frm
-rw-rw---- 1 mysql users 128 Aug 10 02:58 groups.MYD
-rw-rw---- 1 mysql users 8886 Aug 10 03:29 services.frm
  
```

```

-rw-rw---- 1 mysql users 8620 Aug 10 03:54 status.frm
-rw-rw---- 1 mysql users 2048 Aug 10 03:54 status.MYI
-rw-rw---- 1 mysql users 160 Aug 10 03:54 status.MYD
-rw-rw---- 1 mysql users 8692 Aug 10 03:56 severity.frm
-rw-rw---- 1 mysql users 2048 Aug 10 15:31 groups.MYI
-rw-rw---- 1 mysql users 37876 Aug 11 15:45 inventory.MYD
-rw-rw---- 1 mysql users 27648 Aug 11 16:39 inventory.MYI
-rw-rw---- 1 mysql users 212 Aug 24 16:07 severity.MYD
-rw-rw---- 1 mysql users 40 Aug 24 16:08 updateDate.TRN
-rw-rw---- 1 mysql users 378 Aug 24 16:08 services.TRG
-rw-rw---- 1 mysql users 2048 Aug 27 20:21 severity.MYI
-rw-rw---- 1 mysql users 45576 Sep 9 03:08 services.MYD
-rw-rw---- 1 mysql users 37888 Sep 11 00:46 services.MYI
-rw-rw---- 1 mysql users 30720 Sep 11 13:41 owners.MYI
-rw-rw---- 1 mysql users 61144 Sep 11 13:41 owners.MYD

```

Log files, by default, are created in the `mysql` home directory. The security data tables in the `mysql` database are in the `/<mysql home directory>/mysql` directory. Table 2-2 lists the files of the security data tables.

Table 2-2 Permission information stored in tables

File	Description
columns_priv.MYD, columns_priv.MYI, columns_priv.frm	Permission for individual columns within a table
db.MYD, db.MY, db.frm	Permission for individual database
func.MYD, func.MYI, func.frm	Permission for user-defined functions
host.MYD, host.MYI, host.frm	General permission by host
procs_priv.MYD, procs_priv.MYI, procs_priv.frm	Permission for stored procedures
tables_priv.MYD, tables_priv.MYI, tables_priv.frm	Permission for individual tables within a database

user.MYD, user.MYI, user.frm	General Permission by user
------------------------------------	----------------------------

By default, DB2 uses a better approach for the logical and physical distribution of database objects. DB2 differs from MySQL in that DB2 stores all database objects in table spaces. The benefits of table spaces are increased performance and simplified management. To take advantage of this advanced database distribution, refer to 6.2.1, “Database manipulation” on page 123, where we discuss in detail the conversion of a MySQL database structure to DB2.

2.3.2 MySQL storage engines

Although MySQL supports multiple storage engines per database, each table must be managed by a specific storage engine. All storage engines will create table definitions in a file with the `.frm` extension upon creation. MySQL storage engines can be split into two categories:

- ▶ Non-transaction-safe
- ▶ Transaction-safe

Transaction-safe storage engines have commit and rollback capabilities and can be recovered from a crash. Therefore, these storage engines guarantee the Atomicity, Consistency, Isolation, and Durability (ACID properties) of a database. Non-transaction-safe storage engines are faster and require less memory and disk space. However, non-transaction-safe storage engines do not guarantee the database is left in a consistent state, because they do not support ACID properties.

We explain these storage engines in this section:

- ▶ Non-transaction-safe tables can be managed by the following storage engines:
 - MyISAM
 - Memory
 - Merge
 - Archive
 - Comma separated value (CSV)
 - Federated
 - Blackhole
- ▶ Transaction-safe tables can be managed by the following storage engines:
 - InnoDB

MySQL supports transactions with the InnoDB and NDB transactional storage engines. Although both storage engines provide full ACID compliance, the performance and throughput are often enough of a concern to justify converting to DB2.

In DB2, all tables support transactions. Therefore, tables that are managed by MySQL InnoDB, MyISAM, ARCHIVE, and CSV storage engines can all be converted to a DB2 regular table. We discuss the details of converting MySQL tables to a DB2 table in 6.2.2, “Table manipulation” on page 128.

MyISAM storage engine

The MyISAM storage engine, based on ISAM code, was enhanced to overcome the disadvantages of the ISAM storage engine and to provide more useful extensions. MyISAM has been the default storage engine of MySQL since Version 3.23. This storage engine stores the table definition in a file with the .frm extension. The index is stored in a file with the .MYI extension. And, the data is stored in a file with the .MYD extension. The MyISAM storage engine uses table-level locking.

MyISAM supports three storage formats:

- ▶ Static tables
These tables have a fixed length and are the default format that MyISAM uses, if no VARCHAR, VARBINARY, BLOB, or TEXT columns are used.
- ▶ Dynamic tables
The tables are used as the default if VARCHAR, VARBINARY, BLOB, or TEXT columns are defined in the table.
- ▶ Compressed tables
These tables are read-only static and dynamic tables that have been compressed. To write data to a compressed table, the table has to be uncompressed first.

Memory storage engine

The memory storage engine, formerly known as the *heap table*, uses hashed and B-tree indexes and can only be held in memory. Therefore, they are primarily used as temporary tables. This storage engine only creates the .frm file for the table definition. The memory storage engine uses table-level locking.

Merge storage engine

The merge storage engine allows the grouping of multiple MyISAM tables across the same disk or separate disks into one table. The merge storage engine can only be used with MyISAM tables that have the same columns and indexes. The

.MRG file contains the list of MyISAM tables that make up the new Merge table. If a table created with this storage engine is dropped, the Merge .frm and .MRG files are removed and the individual MyISAM tables still exist. Tables created with the merge storage engine support SELECT, UPDATE, INSERT, and DELETE operations if the user has the privileges on the underlying MyISAM tables.

Archive storage engine

The archive storage engine is typically used for storing large amounts of data that will not be modified. The archive engine only supports SELECT and INSERT operations. Data stored in the archive engine is stored in compressed format without indexing on the data. This storage engine will store the table definition in a file with the .frm extension. The data is stored in a file with the .ARZ extension. And, the metadata is stored in a file with the .ARM extension. The archive storage engine uses row-level locking.

CSV storage engine

The CSV (comma separated value) storage engine stores data in comma separated text files. The table definition is stored on the server in a file with the .frm extension, and the data is stored in a file with the .CSV extension.

Federated storage engine

The federated storage engine allows tables on a remote server to be accessed as though they were stored locally. The table definition, the .frm file, is stored locally, but no data is actually stored on the local server.

Blackhole storage engine

The blackhole storage engine creates a table where no data is stored and no data can be retrieved. This storage engine only creates the .frm file for the table definition.

InnoDB storage engine

MySQL 4.0 InnoDB is enabled by default. The InnoDB engine is a completely separate database back end that is produced by a company called Innobase Oy (<http://www.innodb.com>) and placed under MySQL. InnoDB tables are transaction safe and support foreign key constraints. This storage engine stores the table definition in a file with the .frm extension. Table data and indexes are stored in a table space consisting of several files or even raw disk partitions. InnoDB uses B-tree indexes to locate data in the tables. This storage engine uses row-level locking and does not lock rows for read statements. DB2 9.7 has introduced a similar approach to locking, which is discussed further in 8.3.3, “DB2 isolation levels” on page 272. This change in DB2 locking behavior makes porting an application much easier.

InnoDB was acquired by Oracle in 2006, which resulted in a development effort by MySQL AB to build their own transactional safe storage engine. This development is ongoing at the time of writing this book.

NDB cluster storage engine

The NDBCLUSTER, which is also known as NDB, is a transaction safe storage engine. This storage engine was introduced in MySQL 4.1. This storage engine allows for clustering of a database in a shared-nothing architecture. This type of architecture allows a table to be split apart across multiple servers, where each server has its own set of data. In order for the table to be a part of the cluster, the table must be using NDBCLUSTER storage engine. As of MySQL 5.1.24, the support for the NDBCLUSTER storage engine has been removed from the standard MySQL release and is now available in its own release, which is known as MySQL Cluster NDB.

2.3.3 MySQL standard SQL compliance

Up to Version 5.1, MySQL complies most closely to SQL-92 and meets Open Database Connectivity (ODBC) levels 0-3.51 standards. MySQL aims to fully comply with the ANSI/ISO SQL standard. The most important missing compatibilities of the MySQL default table MyISAM toward the ANSI-SQL-92 standard refers to these areas:

► **Transactions**

The MySQL default storage engine MyISAM does not support transactions. For this storage engine, the developers of MySQL followed the “atomic operations” data integrity model. Auto-commit is always enabled, by default, for MyISAM. Therefore, every time that a statement is executed, the changes are committed to the database, as shown in Example 2-2.

Example 2-2 Atomic operation

```
INSERT INTO table1 ...  
COMMIT  
DELETE FROM table2 ...  
COMMIT
```

MySQL supports transactions with the InnoDB and NDB transactional storage engines. Both engines provide full ACID compliance. In contrast, all tables in DB2 support transactions and provide full ACID compliance.

► **Referential integrity**

Referential integrity is the state in which all values of all foreign keys are valid. The relationship between certain rows of the DEPT and EMP tables, as shown in Figure 2-3 on page 49, illustrates referential integrity concepts and

terminology. For example, referential integrity ensures that every foreign key value in the DEPT column of the EMP table matches a primary key value in the DEPTNO column of the DEPT table.

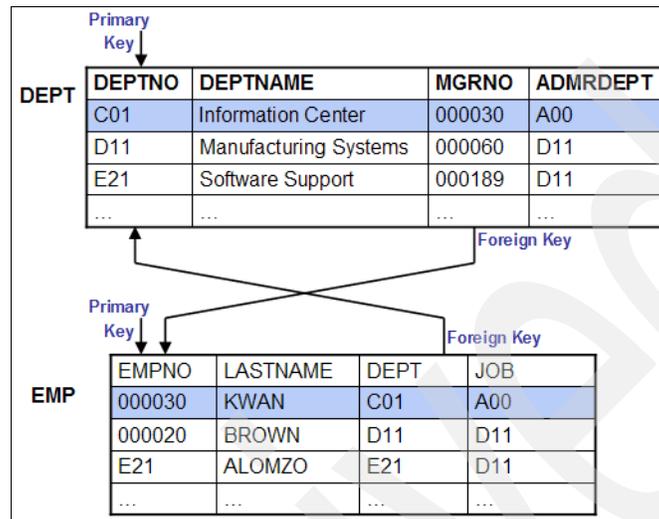


Figure 2-3 Referential integrity

In MySQL Server, MyISAM tables do not support foreign key constraints. MySQL only parses the FOREIGN KEY syntax in CREATE TABLE commands, but MySQL does not use or store this information. Only InnoDB tables support checking foreign key constraints, including ON DELETE CASCADE and ON UPDATE CASCADE.

DB2 supports referential integrity on all of the tables that are created. In 6.5, "Sample database conversion" on page 148, we discuss how to take advantage of foreign keys in DB2.

2.4 MySQL utilities

MySQL is distributed with a set of support utilities. The Internet, however, provides a larger set of third-party tools to manage MySQL databases. In this section, we attempt to give a brief overview of the MySQL distributed set of supported tools. For more information, visit this Web site:

<http://dev.mysql.com/doc/refman/5.1/en/programs.html>

DB2 has a full set of utilities available for working with and managing the database environment. We describe DB2 utilities in 1.4, “DB2 utilities” on page 18.

2.4.1 Overview of the MySQL server-side programs and utilities

This section introduces the distributed set of server-side tools, which is included in the MySQL package.

MySQL server programs

These programs are used to start, stop, and manage the MySQL server for various purposes:

- ▶ `mysqld`
- ▶ `mysqld-nt`
- ▶ `mysqld_safe`
- ▶ `mysqld-debug`
- ▶ `mysqld_multi`
- ▶ `mysql.server`
- ▶ `mysqld-max`
- ▶ `mysqlmanager`

Setup programs

The rest of the programs are used for setting up operations during the installation or upgrade of the MySQL server:

- ▶ `mysql_install_db`
- ▶ `mysql_fix_privilege_tables`
- ▶ `make_binary_distribution`
- ▶ `mysqlbug`
- ▶ `comp_err`
- ▶ `make_win_bin_dist`
- ▶ `mysql_secure_installation`
- ▶ `mysql_tzinfo_to_sql`
- ▶ `mysql_upgrade`

2.4.2 Overview of the MySQL client-side programs and utilities

The distributed set of client-side tools, which is provided by MySQL AB, includes these tools:

- ▶ `mysql`
- ▶ `mysqladmin`
- ▶ `mysqlcheck`

- ▶ mysqldump
- ▶ mysqlimport
- ▶ mysqlshow
- ▶ mysqlaccess
- ▶ mysqlbinlog
- ▶ mysqlhotcopy
- ▶ myisamchk
- ▶ myisampack
- ▶ innochecksum
- ▶ my_print_defaults
- ▶ myisamlog
- ▶ mysql_convert_table_format
- ▶ mysql_fix_extensions
- ▶ mysql_tableinfo
- ▶ mysql_setpermissions
- ▶ mysql_waitpid
- ▶ mysql_zap
- ▶ replace
- ▶ perror
- ▶ msql2mysql
- ▶ mysql_config
- ▶ MySQL Administrator
- ▶ MySQL Query Browser

2.5 MySQL application programming interfaces

The MySQL application programming interfaces (APIs) are interfaces by which an application program communicates with the MySQL database. This section gives an overview of which APIs are available for MySQL.

There are four major approaches to connect to a MySQL database, as shown in Figure 2-4 on page 52:

- ▶ JDBC with Connector/J
- ▶ .NET with Connector/NET
- ▶ ODBC with Connector/ODBC
- ▶ Other APIs with C Library

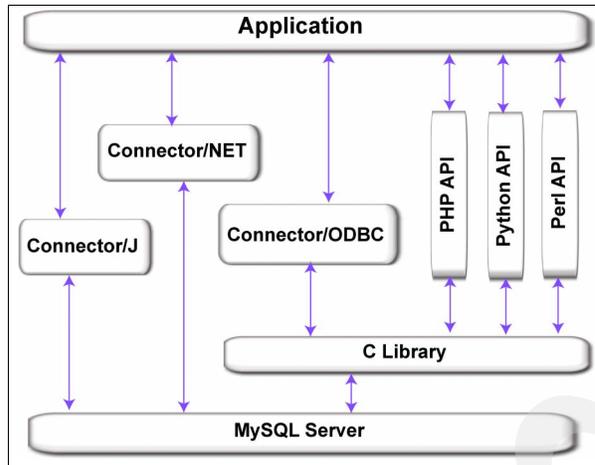


Figure 2-4 MySQL APIs

The first approach is to connect the Java application using Java Database Connectivity (JDBC) and the Connector/J, which is officially supported by MySQL. This connector is written in Java and does not use the C client library to implement the client/server communication protocol.

The second approach is to connect the .NET application using Connector/NET, which is written in C# and does not use the C client library to implement the client/server communication protocol. This approach is officially supported by MySQL AB.

The third approach is to connect using Connector/ODBC for applications that use ODBC standards. This connector uses the embedded C client libraries to implement the client/server communication protocol. This approach is officially supported by MySQL.

The fourth approach is to use the third-party APIs that are provided by programming languages, such as PHP, Perl, or Python. These APIs will use the embedded C client libraries to implement the client/server communication protocol. The third-party APIs are not officially supported by MySQL. The following list shows several of the APIs that are available for MySQL:

- ▶ C API

The API to connect from C programs to a MySQL database. You can obtain more details at this Web site:

<http://dev.mysql.com/doc/refman/5.1/en/c.html>

▶ C++ API

The API to connect to a MySQL database from C++. You can obtain information at this Web site:

http://forge.mysql.com/wiki/Connector_C%2B%2B_Binary_Builds

▶ PHP API

PHP contains support for accessing several databases, including MySQL. You can obtain information about MySQL access in the PHP documentation, which can be downloaded from this Web site:

<http://www.php.net/download-docs.php>

▶ PERL API

The Perl API consists of a generic Perl interface and a special database driver. The generic interface in Perl is called Database Interface (DBI). For MySQL, the driver is called DBD::mysql. You can obtain information about the DBI at this Web site:

<http://dbi.perl.org/>

▶ Python API

The API to connect to MySQL for Python is called MySQLdb. You can obtain the API at this Web site:

<http://sourceforge.net/projects/mysql-python/>

▶ Ruby API

You can obtain the API to access MySQL servers from Ruby programs at this Web site:

<http://tmtm.org/en/mysql/ruby/>

DB2 supports the most frequently used MySQL programming languages. These languages include PHP, Java, Perl, Python, Ruby, C#, C/C++, and Visual Basic®. With proper planning and knowledge, you can convert these applications to DB2 with minimal effort. In Chapter 8, “Application conversion” on page 205, we discuss and provide examples for converting applications from MySQL to DB2.

Archived



Planning the conversion from MySQL to DB2

Proper planning influences the success of a project significantly. This chapter discusses the overall conversion planning stages. This chapter includes the considerations required prior to porting, various assessments that must be considered, and finally, the porting steps. In this chapter, we discuss the following conversion topics in detail:

- ▶ Project planning
- ▶ Application assessment and system planning
- ▶ The conversion process:
 - Porting preparation and installation
 - Database structure porting
 - Data porting
 - Application porting
 - Basic administration
 - Testing and tuning

We also provide information describing how IBM conversion specialists can support you in your conversion project in any of the steps involved, and we discuss the available conversion tools, such as the IBM Data Movement Tool.

3.1 Conversion project planning overview

In order to understand what needs to be done and how long it will take, each conversion or conversion project begins with a conversion assessment. A systematic and organized analysis provides a detailed picture of the full project. The assessment requires in-depth knowledge of the application to be converted and the products that will be used.

To assess an application for conversion, you must create an application profile that describes the application architecture, technologies used, application functions, application interface, application environment characteristics, database detail, application size, and so on.

Based on the application profile, you can plan the software and hardware needs for the target system. The planning stage is also a good time to consider many of the rich functions and features of the DB2 product family, which can increase productivity and reduce maintenance costs. While the most common reason for the one-to-one conversion from MySQL to DB2 is to use the more advanced features of DB2 that MySQL does not provide, reducing the run time of transactions dramatically also helps. The advanced optimizer in DB2 showed improvements of approximately 20x for the real production load in various projects.

Knowledge regarding the skills required and the resources that are available for the conversion project is required. IBM provides a variety of DB2 courses to help IBM clients learn DB2 quickly. For more information, go to these Web sites:

<http://www.ibm.com/developerworks/>
<http://www.ibm.com/industries/education/index.jsp>
<http://www.ibm.com/developerworks/data/bootcamps>

A conversion assessment provides you with the overall picture of the conversion tasks and efforts needed. From a conversion assessment, a conversion project plan can be created to manage each conversion step.

To execute the typical conversion steps, various tools are available to help you save time in your conversion project. IBM offers the no charge conversion tool, the IBM Data Movement Tool, for converting from various relational database systems to DB2.

The process of database and application conversion consists of the following major steps, which are discussed in detail later:

- ▶ Porting preparation and installation database structure porting
- ▶ Data porting
- ▶ Application porting

- ▶ Basic administration
- ▶ Testing and tuning
- ▶ User education

Experienced IBM specialists can support you during any phase of the conversion project with special conversion offerings that are provided worldwide by IBM.

3.1.1 Benefits of converting to DB2

DB2 offers open, industrial-strength database management for e-business, business intelligence, transaction processing, and a broad range of applications. At the same time, great focus and investment have been put into simplifying administration within DB2 with autonomic computing.

Several primary motivators for conversion exist:

- ▶ **Multiplatform support**

DB2 is a true cross-platform database management system (DBMS), running on a wide variety of systems, including Linux, Windows 98/NT/2000/XP, Solaris, Hewlett-Packard UNIX (HP-UX), and AIX. For more information about platform support, see this Web site:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp>

- ▶ **Performance optimization**

DB2 has dominated key performance benchmarks many times over the years, including both Online Transaction Processing (OLTP) and Online Analytical Processing (OLAP) benchmarks. This world-class performance means that for businesses running DB, they can do more work with their existing hardware, thereby avoiding or delaying costly server upgrades:

http://www.ibm.com/software/data/db2/9/editions_features_perf_ent.html

- ▶ **Reliability**

For many businesses, database downtime causes both money and opportunity loss. DB2 can minimize the downtime that is associated with many planned activities, such as altering a table, and many unplanned events, such a power outage, by either eliminating the downtime completely or reducing its duration to a few seconds. The DB2 High Availability Feature provides 24x7 availability for your DB2 data server through replicated failover support and data recovery modules. For more details, visit this Web site:

http://www.ibm.com/software/data/db2/9/editions_features_ha.html

► **Industrial scalability**

DB2 has the best scalability by being the only DBMS to have transaction processing and business intelligence scaling up to 1,000 nodes. For more information about scalability, see this Web site:

<http://www.ibm.com>

► **Integrated support for native environments**

DB2 conforms to many standards, including operating system support. It maps closely onto internal resources for performance and scalability, making it more reliable and tightly integrated.

► **Deep compression**

Businesses with large volumes of data know how expensive storage can be. DB2 can dramatically reduce this cost with industry-leading data compression technologies that compress rows, indexes, temporary tables, LOBs, XML, and backup data with compression rates that can reach over 80%. This deep compression allows DB2 to keep more data in memory, thereby avoiding performance-robbing disk I/O, which in turn, causes database performance to increase considerably. For more details about deep compression, see this Web site:

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

► **Security**

Unauthorized data access is an ever present threat that can cost businesses considerable sums of money, their reputation, or both. DB2 offers a comprehensive suite of security features that effectively and decisively minimizes this threat. DB2 provides additional peace of mind with lightweight security audit mechanisms that check the validity of any unauthorized data access:

http://www.ibm.com/software/data/db2/9/editions_features_advaccess.html

► **pureXML**

DB2 pureXML revolutionizes the management of XML data with breakthrough native XML support provided only by DB2. It eliminates much of the work typically involved in the management of XML data, and it serves data at unmatched speeds. If you work with XML data, you need to know about DB2 pureXML. Now, DB2 9.7 for Linux, UNIX, and Windows opens new opportunities to efficiently analyze XML data in data warehouses:

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

► **Integrated system management tools**

DB2 has a number of tools for managing the database system. IBM Optim Data studio is a rich and extensible solution to help you develop DB2 applications and to manage databases. The Health Monitor and the Health

Center help to easily capture and monitor the overall health of the database. The Replication Center is a tool that is used to set up and administer a replication environment. The Configuration Assistant and the Control Center can be used to assist in configuring and maintaining database objects. These tools are just a few of the management tools that are available for DB2. For a list of all of the tools and for more information about the tools, see this Web site:

<http://www.ibm.com/software/data/db2imstools/products/db2-luw-tools.html>

► **Self-managing and resource tuning capability**

DB2 also contains self-managing and resource tuning database technology, such as the Self-Tuning Memory Manager and automatic storage, which provides enhanced automation to configure, tune, and manage databases. This innovative database manageability allows database administrators to spend less time managing routine tasks and to focus on tasks that help enterprises gain and maintain a sustainable competitive advantage. For more information about autonomies, visit this Web site:

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

► **Multi-vendor SQL and application development support**

Businesses cannot afford to have their application development teams struggling with unfamiliar SQL syntax and foreign database APIs. With DB2, non-DB2 application developers can use the full cross-vendor support for many APIs, such as Java Database Connectivity (JDBC), Open Database Connectivity (ODBC), and .NET, as well as rich support of other database vendors' SQL, functions, and data types:

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

► **Web services applications**

DB2 can be accessed as a Web service provider and is usually teamed with the IBM WebSphere® family products to provide a complete Web services framework. For more information, visit this Web site:

<http://www.ibm.com/software/websphere/>

► **Data warehousing functionality**

IBM offers the InfoSphere Warehouse Enterprise Edition, which provides a powerful range of capabilities that goes beyond traditional data warehouses. InfoSphere Warehouse is a comprehensive platform that includes tooling and infrastructure to help data warehouse architects and administrators efficiently design, deploy, and maintain an enterprise data warehouse. For more information, visit this Web site:

<http://www.ibm.com/software/data/infosphere/warehouse/>

3.1.2 IBM conversion support

With assistance from IBM DB2 experts, you can greatly reduce the cost, time, and errors that are associated with the conversion process. DB2 experts are available to assist your organization regarding any phase of the conversion process, including these phases:

- ▶ Assessment of the database and application conversion efforts
- ▶ Project planning
- ▶ System planning
- ▶ Database design
- ▶ Porting preparation and DB2 installation
- ▶ Database structure and data conversion
- ▶ Application conversion
- ▶ Basic administration
- ▶ Testing and tuning of the DB2 data server

IBM Business Partner conversion projects

IBM PartnerWorld® is designed to help IBM Business Partners succeed in the marketplace and strengthen their relationship with IBM. When enrolled, you can utilize PartnerWorld tools and resource to grow and profit in the market. You can work with IBM experts to provide you with leading-edge support. Visiting the PartnerWorld Web site, you can find information about sales and marketing, technical resources, and products and technologies. You can also learn about upcoming events, educational opportunities, certifications, products, and a variety of promotions.

There is *no cost* to join PartnerWorld; you can find more information and register by visiting the following link:

<http://www.ibm.com/partnerworld>

If you are a partner and have a conversion project in mind, contact us at askdata@ca.ibm.com and title your e-mail “MySQL to DB2 migration.”

Client conversion projects

In more than 3,500 cases, IBM conversion specialists around the world have advised clients about conversion projects to DB2. Before starting the assessment phase, contact the *Software Migration Project Office (SMPO)* for *no charge* conversion estimates, as well as access to a team of conversion experts.

If you are a client and have a conversion project in mind, contact one of the following contacts according to your geography:

- ▶ In North America and Latin America, contact db2mig@us.ibm.com
- ▶ In the U.K., Europe, Middle East, and Africa, contact emeadbct@uk.ibm.com
- ▶ In Japan, India, and Asia Pacific, contact dungi@hkl.ibm.com

You can obtain more information about the DB2 conversion team at the Software Migration Project Office (SMPO) Web site:

<http://www.ibm.com/software/solutions/softwaremigration/dbmigteam.html>

You can obtain the most up-to-date details about current offerings, success stories, literature, and other information about DB2 Migrate Now! at this Web site:

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

3.1.3 Education

DB2 provides an easy-to-use, feature-rich environment. Therefore, it is important that those individuals involved in the conversion process are appropriately trained to take full advantage of its offerings.

There is extensive training material available, such as courses, self-study guides, and IBM Redbooks publications. IBM also offers a variety of DB2 courses; one extremely useful course is the DB2 9.7 Bootcamp. For course details and scheduled courses near you, visit this Web site:

<http://www.ibm.com/developerworks/wikis/display/im/DB2+9.7+Bootcamp>

For further information regarding DB2 training, visit the DB2 Web site:

<http://www.ibm.com/software/data/education/>

This DB2 for Linux, UNIX, and Windows conversion Web site can help you find the information that you need to port an application and its data from other database management systems to DB2. The porting and conversion steps, which are described in this chapter, appear in the order that they are commonly performed. In addition to the technical information that is available at this site, IBM clients and IBM Business Partners need to check out the Information for IBM clients and Information for IBM partners links:

<http://www.ibm.com/developerworks/db2/zones/porting/partners.html>

<http://www.ibm.com/developerworks/db2/zones/porting/customers.html>

Here, you will find additional links and information regarding assistance or available resources for your porting project:

<http://www.ibm.com/developerworks/db2/zones/porting/index.html>

3.2 Application assessment

An *application assessment* is the first step in conversion planning. The assessment helps you to understand the scope of the conversion project and to prepare a detailed conversion project plan to design the target system.

You need to understand how your application works and what resources are needed. There are probably many characteristics within your application that will influence system planning and the scope of the conversion effort.

The application assessment requires the following information:

- ▶ Application architecture:
 - Stand-alone
 - Client/server
 - Tier architecture
- ▶ Database architecture:
 - Number of databases
 - Number of tables
 - Number of indexes
 - Users, access rights, and privileges
- ▶ Size of the data that is stored in the database:
 - Bytes stored in tables
 - Bytes stored in indexes
 - Log file size
- ▶ Source code language:
 - PHP
 - Perl
 - Java
 - C/C++
 - Any other programming language
- ▶ Database interface:
 - Direct database access through an API
 - Database access layer, such as ODBC or JDBC
- ▶ Operating system:
 - Linux
 - AIX or UNIX
 - Windows
 - Any other operating system

- ▶ Hardware used:
 - CPU
 - Memory
 - Hard disk

In a client/server environment, be sure to describe the application in both the client environment and the server environment.

3.3 System planning

Based on the application profile that was created during the application assessment, you should be able to plan your target system properly.

With DB2 support for various platforms and multiple operating systems, such as Linux, Windows, and AIX and so on, platform limitation should not be an issue. You can select on which system the converted application will run based on the application nature and future enhancement requirements.

As shown in Figure 3-1, the target system can be the same system as the source system, or another system with a separate operating system and hardware. You might even want to make your database server a separate machine from the machine on which your application runs on (creating a *two-tier architecture*).

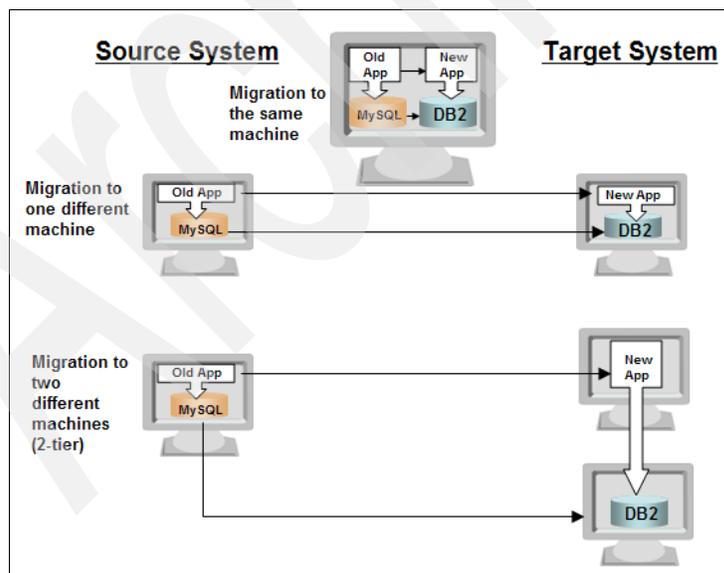


Figure 3-1 Sample conversion scenarios

If you decide to use a new machine for the conversion program, you need to plan what kind of hardware you want to use and which operating system you want to install on the new machine.

In either case, check if the hardware of your target system meets the minimum requirements, paying particular attention to the following areas:

- ▶ Operating system
- ▶ DB2
- ▶ Application
- ▶ Data
- ▶ Conversion tools (if used)

We discuss system requirements in more detail in 5.1.1, “System requirements” on page 88.

3.3.1 Software

You must determine which software must be installed on your target system, including:

- ▶ Operating system (Linux, AIX, UNIX, Windows, or others)
- ▶ DB2 version
- ▶ Application to be converted
- ▶ Conversion tools (if used and installed on the target system)
- ▶ Any software on your source system that is required by your application to run properly, including but not limited to:
 - HTTP server
 - Web application server
 - Development environment
 - Additional software (such as Lightweight Directory Access Protocol (LDAP) or others)

Be sure to have the latest versions and fix packs of the planned products. Ensure that the chosen operating system supports the chosen software.

3.3.2 Hardware

When starting the conversion process, it is important to have a target platform that meets the minimum requirements of all the software that will be installed on it. Check the supported hardware platforms, depending on the chosen software.

Your application also requires hardware resources. Be sure to have enough disk space for your application and the transformed data.

Virtual images are a great option for operating enablement environments. Virtual images reduce the hardware impact and allow simple test environments for the enablement effort without incurring additional costs. Keep in mind that the disk space is still needed.

3.3.3 Conversion tools

There are no charge commercial tools available to assist you in converting your application from MySQL to DB2. The tools offer a variety of functions and are supported on various operating systems. IBM offers the no charge IBM Data Movement Tool. In addition to this conversion tool, IBM also offers an assortment of no charge IBM tools, such as the new IBM Optim Data Studio, which can ease the learning curve by making database administration and debugging queries and functions effortless.

When deciding to use a tool, be sure that it fulfills the requirements that are appropriate for your platform.

IBM Data Movement Tool

The IBM Data Movement Tool offers schema and data movement support from MySQL, Oracle, Microsoft SQL Server, Microsoft Access, Sybase Adaptive Server Enterprise, PostgreSQL, and Ingres to DB2 9.7. The IBM Data Movement Tool replaces the Migration Tool Kit with a greatly simplified workflow and high speed data movement.

The tool can be used to extract the data from the source database into flat files, to generate scripts to create the database objects, and to import the data using the DB2 LOAD utility. At the time that this book was written, the Data Movement Tool supported the following database objects for a MySQL to DB2 conversion: tables, constraints, indexes, primary keys, and foreign keys (with InnoDB).

At the time that this book was written, the Data Movement Tool did not support the following database objects for a MySQL database conversion: views, procedures, functions, triggers, and packages.

The IBM Data Movement tool is available in both graphical user interface (GUI) and command-line interface:

- ▶ Graphical user interface

The GUI interface offers the IBM Data Movement Tool conversion functionality by using a Java interface. It provides an easy to use interface for beginners.

► Command-line interface

The command-line interface offers a way to operate the IBM Data Movement Tool from the command line. The command-line interface is intended for experienced users, who want to run end-to-end conversions without user interaction.

For our conversion scenario, we use the new IBM Data Movement Tool to convert database objects and data from MySQL to DB2. If you want to download the IBM Data Movement Tool or receive more information about it, refer to:

<http://www.ibm.com/developerworks/data/library/techarticle/dm-0906datamovement/index.html>

We discuss the installation of the IBM Data Movement Tool in Chapter 5, “Installation” on page 87.

Other available conversion tools

Various other tools are available online, such as SQLWays. Note that IBM does not service these conversion tools and does not guarantee the usage of these tools. SQLWays is a commercial tool by Ispirer Systems for the Windows, UNIX, and Linux operating systems. For more details, visit this Web site:

<http://www.ispirer.com/>

3.4 The conversion process

For accurate planning, it is important to understand the steps to complete the conversion project. To estimate the project effort correctly and to convert the database and application successfully, you must plan each step. Figure 3-2 shows the basic porting process steps.

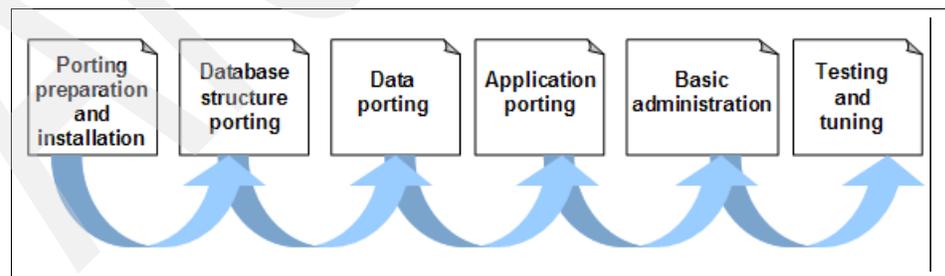


Figure 3-2 Steps of the conversion process

3.4.1 Preparing for the installation

After deciding which target system you want to use, you can set up the installation of the hardware components, the operating system with users and access rights, the network connections, the required software, and finally, the DB2 version of your choice.

For more information about the installation of DB2 and the IBM Data Movement Tool, refer to Chapter 5, “Installation” on page 87.

3.4.2 Porting the database structure

After you install DB2 on the target system, you can port the database structure from your source MySQL database to DB2.

The database structure is usually described through Data Definition Language (DDL) statements. DDL scripts include the creation of tables, keys, and indexes. Because the DDL syntax for MySQL and DB2 differs, a conversion process is required. Using a tool to convert the database structure can save time and effort. Figure 3-3 shows the conversion database structure steps.

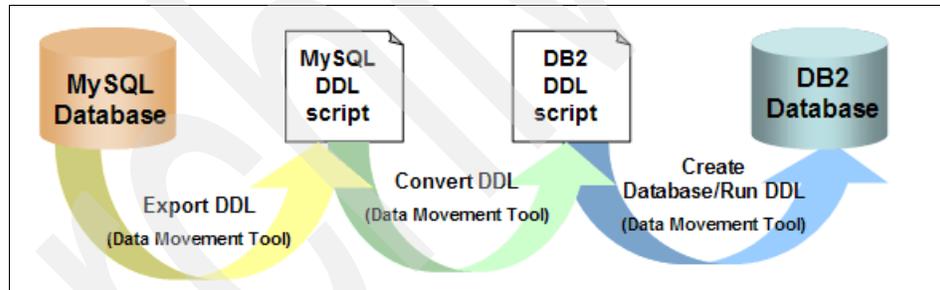


Figure 3-3 Database structure porting process

Exporting DDL from MySQL databases

Information for the tables, keys, indexes, and functions within your source MySQL database must be retrieved.

Tools, such as IBM Data Movement Tool, can perform the majority of this task automatically. Regardless of the tool used, verify the output results. This step can also be performed manually; however, you need to make sure to cover all of the database objects.

Converting DDL to DB2 syntax

Because the DDL used for MySQL slightly differs from the DDL used for DB2, the DDL used to create MySQL databases and objects must be converted to DB2 syntax. The syntax differences between MySQL and DB2 are described in more detail in Chapter 6, “Database conversion” on page 115.

This conversion of DDL syntax is supported by tools, such as IBM Data Movement Tool, or, it can be performed manually, as well. In this case, watch closely for the different data types that are used by MySQL and DB2.

If you plan to change the logical model of your database structure to enhance your application and take advantage of DB2 functions and features, the DDL needs to be modified in this step.

Creating DB2 database structure

After all DDL scripts are converted to DB2 syntax, you can create your DB2 database and execute the DDL scripts to create the database structure.

These actions are supported by tools, such as the IBM Data Movement Tool. Make sure to check that all database objects, such as tables, keys, indexes, and functions, are created successfully.

3.4.3 Data porting

With the database installed on your new system and the structure of your database created, the data extracted must be loaded from the source to the target system.

Databases usually provide mechanisms for exporting (dumping) and importing (loading) data. We categorized the MySQL data into two types:

- ▶ User data: Data that contains information about users, access rights, and privileges
- ▶ Application data: Data created and used by an application

Figure 3-4 on page 69 shows the steps of data movement.

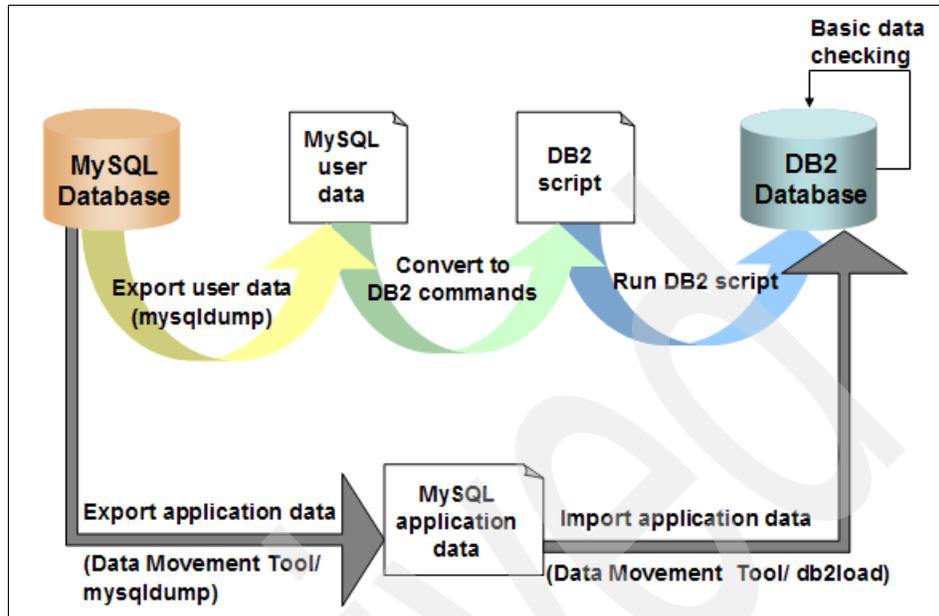


Figure 3-4 Data porting process

Exporting MySQL user data

MySQL and DB2 use different database security mechanisms. The database object access privileges are stored in the MySQL database. You need to understand the MySQL access rights of your application and how you can map them to DB2. Depending on your application, you probably have to export user data from the MySQL database.

Mapping MySQL user data to DB2 user data

Because DB2 users, access rights, and privileges are maintained in a separate way, you have to translate the MySQL user data into DB2 commands that grant privileges and use operation system user ID management functions to create the users.

In this step, you create scripts to create the required users and grant them access privileges to the DB2 database and objects based on the MySQL user data.

Creating DB2 user data

After you have mapped your MySQL user data to DB2 user data, your users must be created in the DB2 system, and the necessary privileges must be granted to them.

The scripts with the DB2 commands for creating users and granting privileges must be run.

Exporting MySQL application data

You have to dump the application data out of the MySQL database. This step is supported by the IBM Data Movement Tool, or you can perform it manually. Be aware of the differences in the format of DATE, TIME, and other data types.

Converting MySQL application data to DB2 format

You must convert the MySQL application data to a format that can be stored within DB2. The IBM Data Movement Tool handles the conversion of all MySQL data types for you.

Importing application data into DB2

The exported (and maybe converted) data must finally be loaded into the DB2 tables. The IBM Data Movement Tool also supports this step, and of course, data can be loaded into DB2 manually.

Basic data checking

After you have loaded all data into your database, you need to perform basic data checking. For example, you must verify the correct number of rows per table and the correct representation of the field values. Also, be sure that you have all the users created in your DB2 system.

All of the steps that are described in this section are performed on the MySQL sample database and are explained in great detail in Chapter 7, “Data conversion” on page 167.

3.4.4 Application porting

Various tools can greatly help the conversion process of a database structure and its data. The conversion of the application can include manual conversions for cases where nonstandard SQL functions are utilized. DB2 9.7 introduced a number of features that simplify this task from other relational database vendors to DB2. Tools exist that support tasks, such as syntax highlighting, but in most cases, the main conversion must be done manually. A good way is by simply performing a search for typical SELECT, INSERT, UPDATE, and DELETE key words and mark these key words with specific comments. If the application supports SQL92-compliant database systems, the work is significantly reduced.

The extent to which you have to change the application code depends on the database interface that is used in the source application. When a database

access layer is used, the adoption is not that complicated; otherwise, the effort to enable the application will likely be higher.

Application source code changes

Because the *Data Manipulation Language (DML)* of MySQL and DB2 differs, you might have to change the SQL statements in your application code directly.

You might find MySQL behaviors that are not natively supported in DB2, so a concept must be established to allow the application to behave in the same way as it did prior to conversion.

Database interface

Regardless of the interface used between an application and a database application, the access to the database must be changed, because the database server has been modified.

If standardized interfaces, such as ODBC or JDBC are used, the changes will be less significant than if the application uses the native API of the database product.

Handling conditions

Depending on the implementation of your application, there might be changes in the condition handling part of the application.

Additional considerations

DB2 offers rich, robust functions, which you can take advantage of in your applications. The following list shows several of these features that you might want to consider using in your application and which differ in MySQL:

- ▶ Concurrency
- ▶ Locking
- ▶ Isolation-level transactions
- ▶ Logging
- ▶ National language support
- ▶ XML support

The steps listed in this section are performed with various sample application code and explained in great detail in Chapter 8, “Application conversion” on page 205.

3.4.5 Basic administration

Regardless of the conversion, regular maintenance work performed by the database administrator must still be performed. Any administrative issues must also be taken into account during the conversion planning process.

Every database has its own method for backup and recovery, because these tasks are common, vital tasks in database administration. The database must be backed up regularly, and the data retention period must be defined based on the business requirements.

If you have backup and recovery tasks defined on the source system, you probably want to convert these tasks, as well. Be sure to port any existing scripts for backup tasks to support DB2.

Both the database backup and recovery functions must be tested to ensure a safe environment for your application.

Log files

DB2 logs differently than MySQL, so database administrators must be aware of the logging level that can be set, where log information is stored, and how to read these logs.

We describe DB2 database administration in detail in Chapter 9, “Database administration” on page 279.

3.4.6 Testing and tuning

After your new system is up and running, you have to verify that the data and application functionality have been ported completely, and that system behavior has not changed in a negative way.

If you succeed with testing, you can then proceed to tuning your database and application in order to speed up your application.

Checking data

Aside from the basic data checks that must be performed when exporting and importing data, checking that your application handles your data correctly and manipulates the expected fields on inserts, updates, or deletes is vital.

Performance checking can be done manually, or a script can be used to have the data checked.

Code and application testing

It is extremely important that the behavior of your application has not changed. Interactions between components, as well as each module of your application, must be tested. We recommend a code review of all changed code, as well.

Troubleshooting

Whenever the conversion leads to a problem, such as incorrect data or unexpected application behavior, you have to determine the problem in order to fix it.

You must understand error messages from the application, as well as DB2 error messages. The troubleshooting process includes studying the DB2 log files.

See the DB2 technical support Web site for help with specific problems:

<http://www.ibm.com/software/data/support/>

Basic tuning

When your new system is working perfectly, you might want to tune it for even better performance. With the correct database configurations, and hints from DB2 tuning tools, you can speed up your queries quite easily.

DB2 provides tools, such as *Design Adviser*, *Performance Monitor*, or *Index Advisor*, to support you in speeding up your DB2 system. DB2 also provides the Self-Tuning Memory Manager, which keeps your database at an optimal state at all times without any DBA intervention.

We discuss testing and initial tuning of a DB2 database in detail in Chapter 10, “Testing and tuning” on page 321.

Archived

Conversion scenario

To illustrate the process of converting a database from MySQL to DB2, we have developed a small sample Web application written in PHP. We created the back-end database using MySQL 5.1 MyISAM tables. This application scenario is a hardware inventory and services tracking application. Although, the application is written in PHP, it is possible to convert other popular programming languages, such as Perl, Java, and C/C++. Chapter 8, “Application conversion” on page 205 also provides samples of these popular programming languages.

We discuss the following topics in detail:

- ▶ Application description
- ▶ Database structure
- ▶ System environment

4.1 Application structure

The application that is used for this scenario is a Web-based inventory management tracking system. With this application, the user can organize and monitor employee inventory, service tickets, users, and groups.

4.1.1 Application flow

The flow diagram (Figure 4-1) describes the application flow and functions at a high level.

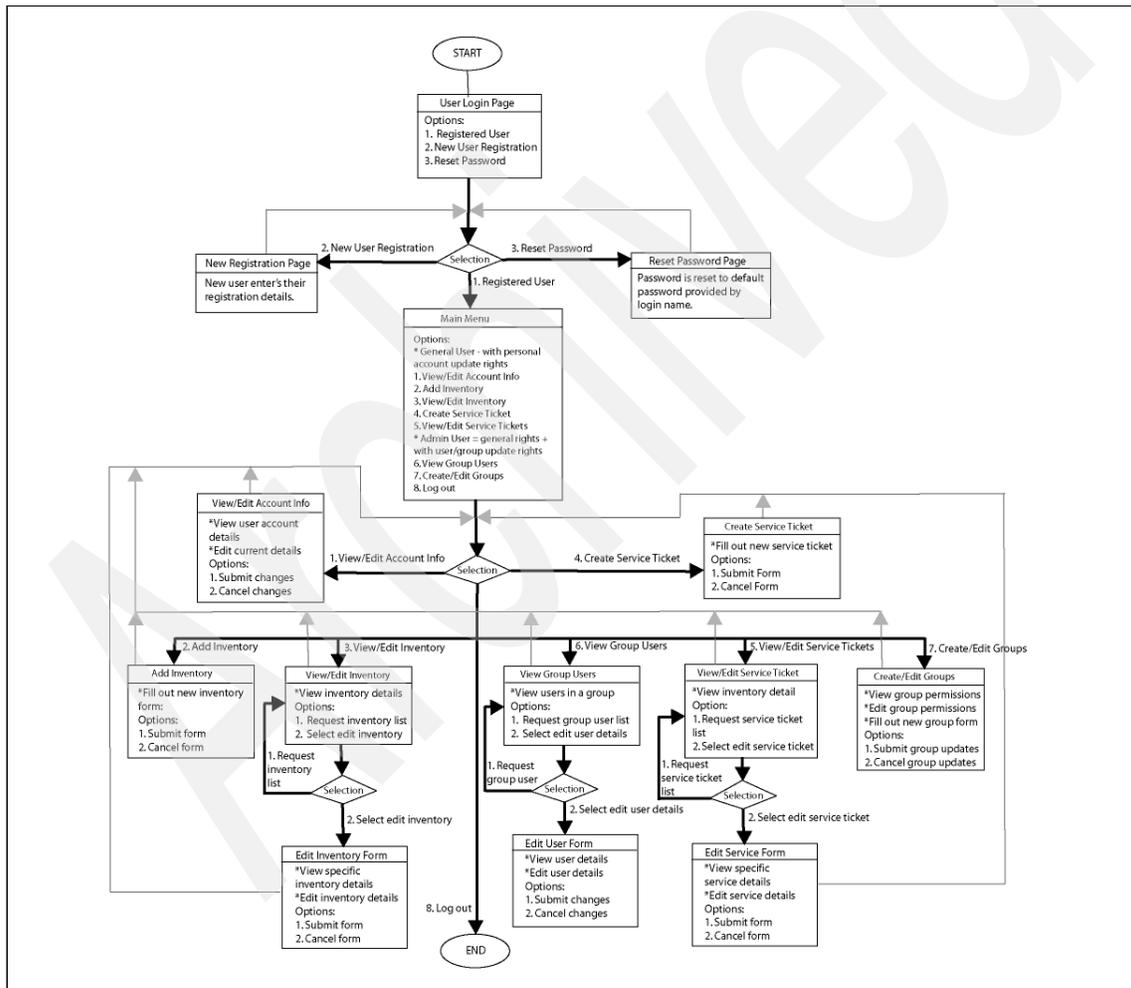
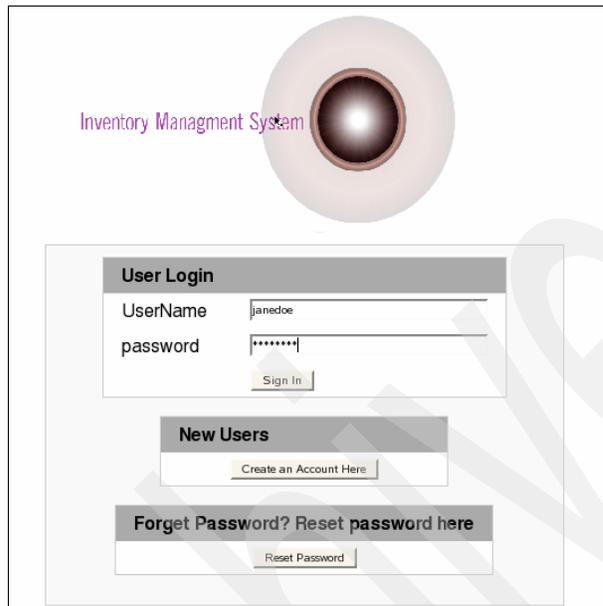


Figure 4-1 Flow diagram for the sample application

When a user enters the Web site, the login page (Figure 4-2) provides three functions:

- ▶ User Login: To log in as an already registered user
- ▶ New Users: To create a new user account
- ▶ Reset Password: To request resetting the password to a default password



The screenshot shows a web application login page. At the top center is a large, stylized eye icon. Below the eye, the text "Inventory Management System" is displayed in a purple font. The main content area is enclosed in a light gray border and contains three distinct sections. The first section, titled "User Login", has a dark gray header. Below the header are two input fields: "UserName" with the text "janedoe" and "password" with asterisks. A "Sign In" button is positioned below the password field. The second section, titled "New Users", also has a dark gray header and a "Create an Account Here" button. The third section, titled "Forget Password? Reset password here", has a dark gray header and a "Reset Password" button.

Figure 4-2 Login page of the sample Web application

With User Login, a registered user uses their user ID and password to log in. The application verifies the user name, password, and user permissions against the registered users in the database. The lowest level of permissions allows the user to view, edit, and create inventory and service requests. If the user has permissions to view, create, and edit groups and users, the user is given extra functionality to do so.

From New Users, a new user can create an account. Completing the registration form (Figure 4-3 on page 78) creates a new user account in the application. The application verifies that the user name provided by the user is unique. By default, new users have the lowest level of permissions. A user who is allowed to edit groups can add a user to a group with more than default-level permissions.

Welcome new user! [Sign in](#)

Inventory Management System - New User Sign Up

[HelpNow](#) | [Feedback](#)

Updated on 5 August 2009

Inventory Management System

Inventory Management Sign up.

Please fill out the following.

First Name	Jane
Last Name	Doe
Email Address	jdoe@email.com
Location	FLOOR: 4 -- OfficeRoomC
Desk Number	146
Phone Number	(555)-555-5555
Fax Number	(555)-555-5555
Please create a login name	janedoe
Please create a password	*****
Please verify password	*****

Figure 4-3 Registration form of the sample application

Using Forget Password, a user can reset the user's account password by entering the user's first, last, and user name, as shown in Figure 4-4 on page 79. The new password will be displayed to the user.

Inventory Management System

Please fill out the following and your new password will be emailed to you.

Reset Password Form

First Name

Last Name

Login Name

Password has been reset to 'temp4now'. Please logging and change it as soon as possible. [Sign in](#)

Figure 4-4 Reset the password window of the sample application

When successfully logged in, the management options are presented to the user. A set of management options is displayed and can vary, depending on the type of permissions held by the logged in user. Figure 4-5 shows the welcome page options for a user with the highest level of permissions.

Welcome Jane Doe [Edit settings](#) | [Sign out](#)

Inventory Management System HelpNow | Feedback

Home Page	Updated on 5 August 2009
View/Edit Account Info	WELCOME Jane Doe
Add Inventory	Please choose from the following:
View/Edit Inventory List	View/Edit Account Info
Create Service Ticket	Add Inventory
View/Edit Service Tickets	View/Edit Inventory List
Administration	Create Service Ticket
View Group Users	View/Edit Service Tickets
Create/Edit Group	View Group Users
	Create/Edit Group

Figure 4-5 Welcome menu for the administration login of the sample application

Using the View/Edit Account Info option, users can view account details, as shown in Figure 4-6 on page 80. Users can update their details by editing the fields and submitting the form.

Welcome Jane Doe [Edit settings](#) | [Sign out](#)

Inventory Management System

HelpNow | Feedback

Updated on 5 August 2009

User Details

To change user details modify the current fields in the form and select submit.

Field	Current Values
User Name	janedoe
First Name	Jane
Last Name	Doe
email Address	jdoe@email.com
Location	FLOOR: 4 --- OfficeRoomC
Desk Number	146
Phone Number	(555)-555-5555
Fax Number	(555)-555-5555
Password	*****
Password	*****

Figure 4-6 View/Edit account information of the sample application

With Add Inventory, the user can associate new inventory with their user account. Figure 4-7 shows a typical completed Add New Inventory form.

Welcome Jane Doe [Edit settings](#) | [Sign out](#)

Inventory Management System

HelpNow | Feedback

Updated on 5 August 2009

Add New Inventory

To create a new inventory record fill out the following form and select submit.

Field	Current Values
User Name	janedoe
Item Name	C printer
Manufacturer	companyD
Model	X75-3TA
year	2008
Serial Number	276908
Location	FLOOR: 1 --- OfficeRoomA

Figure 4-7 Add new inventory window of the sample application

Using View/Edit Inventory List, users have the ability to view their assigned inventory and other users' inventory using owner, location, inventory type, or service created against the inventory (Figure 4-8 on page 81). From this page,

users can select to update inventory records by selecting Edit. To see the Edit button, the user must have permissions to update the inventory record.

Welcome Jane Doe [Edit settings](#) | [Sign out](#)

Inventory Management System

Updated on 5 August 2009

Inventory Details

Inventory Details for Jane Doe

Inventory ID	Item Name	Manufacturer	Model	Year	Serial	Floor	Room	Owner ID	Edit
703	C printer	companyD	X75-3TA	2008	276908	1	OfficeRoomA	Jane Doe	<input type="button" value="Submit"/>

View inventory list for a specific username

Users:

View inventory list for a specific location

Locations:

View inventory list for a specific inventory attribute

Item Name	<input type="text" value="desktop"/>	<input type="button" value="Submit"/>
Manufacturer	<input type="text" value="companyD"/>	<input type="button" value="Submit"/>
Year	<input type="text" value="2000"/>	<input type="button" value="Submit"/>

View inventory list for a specific service ticket attribute

Severity	<input type="text" value="5"/>	<input type="button" value="Submit"/>
Open Date	<input type="text" value="2009-01-01"/>	<input type="button" value="Submit"/>
Close Date	<input type="text" value="2009-01-01"/>	<input type="button" value="Submit"/>

Figure 4-8 View/Edit inventory list window of the sample application

Using Create Service Ticket (Figure 4-9), users can open service tickets against their assigned inventory.

Welcome Jane Doe [Edit settings](#) | [Sign out](#)

Inventory Management System

Updated on 5 August 2009

Create Service Ticket

To create a service ticket please fill out the following fields in the form and select submit.

Field	Current Values
User Name	504
Inventory	<input type="text" value="companyD -- C printer(X75-3TA)"/>
Severity	<input type="text" value="low"/>
Notes	<input type="text"/>
Service Owner	<input type="text" value="Sarah King"/>

Figure 4-9 Create Service Ticket window of the sample application

By clicking the View/Edit Service Tickets option, users have the ability to view their created and assigned service tickets, as shown in Figure 4-10. A user can also view all service tickets based on the user, inventory location, inventory type, and service type. From this page, the user can select to update the service ticket by selecting Edit. To see the Edit button, the user must have the permissions to update the inventory record.

Welcome Jane Doe. [Edit Account](#) | [Sign out](#)

Inventory Management System

[Help Now](#) | [Feedback](#)

Home Page Updated on 5 August 2009

View/Edit Account Info **Services Details**

Add Inventory **Service Details for Service Tickets Assigned to Jane Doe**

View/Edit Inventory List

Service ID	Service Owner	Item Name	Manufacturer	Model	Severity	Open Date	Target Close Date	Close Close Date	Description	Inventory Owner Name	Room Name	Floor Num	Edit
807	Sarah King	C printer	companyD	X75-3TA	5	2009-08-11	2009-08-25		Needs maintenance	Jane Doe	OfficeRoomA	1	Submit

Create Service Ticket

View/Edit Service Tickets

Administration

View Group Users

Create/Edit Group

View Service Tickets for a specific username

Users: [Submit](#)

View Service Tickets for a specific location

Locations: [Submit](#)

View Service Tickets for a specific inventory attribute

Item Name: [Submit](#)

Manufacturer: [Submit](#)

Year: [Submit](#)

View Service Tickets for a specific service ticket attribute

Severity: [Submit](#)

Open Date: [Submit](#)

Close Date: [Submit](#)

Figure 4-10 View/Edit service tickets window of the sample application

With the View Group Users option, users with administration permissions can view all users within a specific group. From this page, the user can update any given user account by selecting Edit. Figure 4-11 on page 83 shows a user viewing the manager group user details.

Welcome Jane Doe [Edit settings](#) | [Sign out](#)

Inventory Management System

[HelpNow](#) | [Feedback](#)

Updated on 5 August 2009

Group Details

Inventory Details for managerGroup

ID	First Name	Last Name	Login Name	Room Number	Floor Number	Phone Number	Number of inventory currently assigned	Edit
16	Kim	Lopez	LopezKim	OfficeRoomB	5	(555)-960-4190	2	<input type="button" value="Submit"/>
26	Kim	Perez	PerezKim	OfficeRoomE	5	(555)-805-3527	2	<input type="button" value="Submit"/>
37	Michael	Hayes	HayesMichael	OfficeRoomF	1	(555)-489-1646	1	<input type="button" value="Submit"/>
88	Tasha	Baker	BakerTasha	OfficeRoomD	5	(555)-246-1890	3	<input type="button" value="Submit"/>
171	Cathy	Adams	AdamsCathy	OfficeRoomB	2	(555)-856-0620	1	<input type="button" value="Submit"/>
181	Robert	Lewis	LewisRobert	OfficeRoomD	1	(555)-057-3375	1	<input type="button" value="Submit"/>
497	Michelle	Evans	EvansMichelle	OfficeRoomG	3	(555)-118-6966	1	<input type="button" value="Submit"/>
502	Tyler	Hutchison	TylerHutchison	ConfRoomE	1	(555)-543-4567	0	<input type="button" value="Submit"/>

View Group Details

Select a group for details.

Groups:

Figure 4-11 View group users window of the sample application

Using the Create/Edit Group option (Figure 4-12), a user with administration permissions can create and edit groups.

Welcome Jane Doe [Edit settings](#) | [Sign out](#)

Inventory Management System

[HelpNow](#) | [Feedback](#)

Updated on 5 August 2009

[Home Page](#)

[View/Edit Account Info](#)

[Add Inventory](#)

[View/Edit Inventory List](#)

[Create Service Ticket](#)

[View/Edit Service Tickets](#)

[Administration](#)

[View Group Users](#)

[Create/Edit Group](#)

Group Permissions

To change group details modify the current fields in the form and select submit.

Group	Edit Users Details	Edit and grant groups to users	Edit Other Users Inventory Details	Edit Other Users Service Details
general	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
tech	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
manager	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
boss	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
emp	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Create New Group

To create a new group fill out the fields in the form and select submit.

Group	Edit Users Details	Edit and grant groups to users	Edit Other Users Inventory Details	Edit Other Users Service Details
<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure 4-12 Create/Edit Group window of the sample application

4.2 Database structure

The database that is used for this scenario consists of seven tables, four views, one trigger, and one stored procedure:

- ▶ The *LOCATIONS* table contains the building layout information.
- ▶ The *OWNERS* table contains the user account information for all of the users that have access to the application. Each owner is associated with a location from the *LOCATIONS* table.
- ▶ The *INVENTORY* table contains the hardware inventory information. Each piece of inventory is associated with an owner from the *OWNER* table and a location from the *LOCATIONS* table.
- ▶ The *SERVICES* table contains all service tickets that have been opened against a specific piece of hardware inventory from the *INVENTORY* table. Each service has one service owner from the *SERVICE* table associated with the service. Each service is also associated with a severity level from the *SEVERITY* table and a status from the *STATUS* table.
- ▶ The *STATUS* table contains the details of the various stages of a service ticket throughout its life cycle.
- ▶ The *GROUPS* table contains the group name and permissions.
- ▶ The *SEVERITY* table contains the degree of importance of getting the service ticket resolved and the estimated time for each degree of importance.
- ▶ The *BOSSGROUP*, *EMPGROUP*, *GENERALGROUP*, *MANAGERGROUP*, and *TECHGROUP* views contain a summary of the user details and user inventory in each of the groups.
- ▶ The *UPDATEDATE* trigger can be used to update the closing date field in the *SERVICES* table to the current date when the ticket is marked as closed.
- ▶ The *UPDATE* stored procedure can be used to update the average number of days that it takes to resolve a service ticket in the *SEVERITY* table. The stored procedure accepts a specific severity level as an input.

For detailed table information, see Figure 4-13 on page 85. We discuss the data type conversion between MySQL and DB2 in detail in Chapter 6, “Database conversion” on page 115.

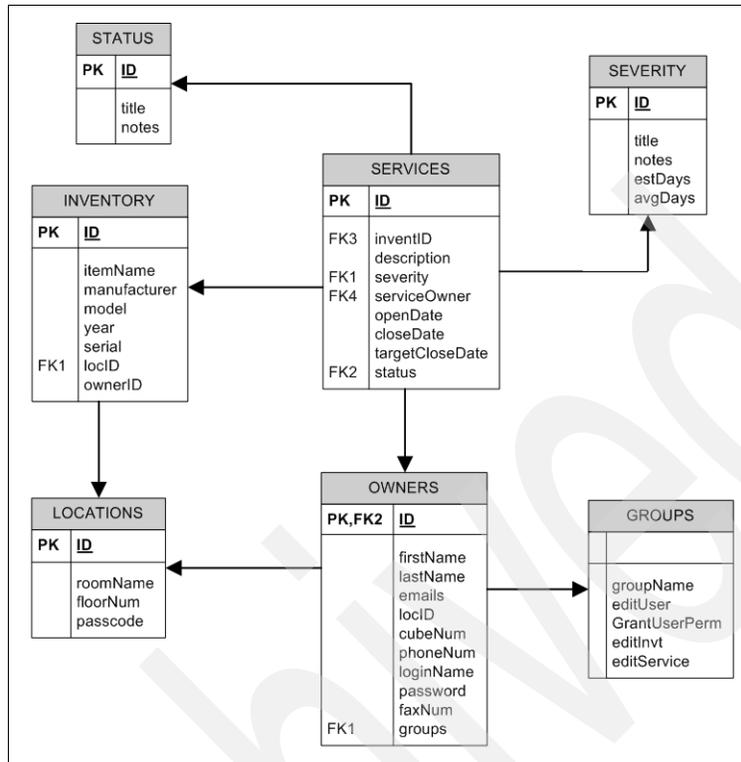


Figure 4-13 Inventory database structure diagram for the sample application

4.3 System environment

In this section, we discuss the hardware and software environment that is used for this conversion scenario.

This database and application were developed in a virtual machine using a VMware® workstation with the following device settings allocated to the image:

- ▶ Memory: 1020 MB
- ▶ Hard Disk: 40 GB
- ▶ Processors: 2

In our conversion scenario, we set up two servers. Our original server had the following software installed on the VMware image:

- ▶ SUSE® Linux 10 SP2
- ▶ MySQL 5.1.36 Community (MySQL AB)
- ▶ Apache 2.0
- ▶ PHP 5.3.0

The second VMware image, the destination server, has the following software installed on the VMware image:

- ▶ SUSE Linux 10 SP2
- ▶ DB2 9.7 Express-C
- ▶ Apache 2.0
- ▶ PHP 5.3.0
- ▶ IBM Data Movement Tool

For more information about the VMware workstation and working with VMware images, go to this Web site:

<http://vmware.com/>

DB2 9.7 Express-C for Linux, UNIX and Windows can be downloaded from this Web site:

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

The IBM Data Movement Tool is used to simplify and greatly decrease the time that it takes to convert from MySQL to DB2. This tool is available at no charge from IBM at the following URL:

<http://www.ibm.com/developerworks/data/library/techarticle/dm-0906datamovement/index.html>

With the IBM Data Movement Tool, the conversion of database objects, such as tables and data types, and the conversion of data can be done automatically into equivalent DB2 database objects.

We discuss the installation and configuration of DB2 and the IBM Data Movement Tool in the next chapter.

Installation

In this chapter, we discuss the target system environment setup. For the database server, we guide you through the installation process of DB2 9.7 for Linux, including the hardware and software prerequisites. The application server has to be examined to ensure that the existing software has the proper DB2 support. If this is a completely new system setup, make sure that all the required software is included in the installation list. Furthermore, we describe the download and steps required to set up the IBM Data Movement Tool.

5.1 DB2 Express-C 9.7 on Linux

Before you start your conversion project, it is important to ensure that the system you choose meets the necessary operating system, hardware, software, and communication requirements.

5.1.1 System requirements

This section provides information about supported Linux distributions, hardware, software, and communication requirements for DB2 on Linux.

Linux distributions supported by DB2

Table 5-1 on page 89 lists IBM DB2 validated and recommended Linux distributions at the time of writing this book. For the most recent list of all of the supported Linux distributions that have successfully completed the IBM DB2 for Linux validation program, check this Web site:

<http://www.ibm.com/software/data/db2/linux/validate/>

Table 5-1 Currently supported Linux distributions, kernels, and libraries

Platform	Distribution	Kernel	Library	Comment
x86	Red Hat® Enterprise Linux (RHEL) 5	2.6.18-92	libstdc++.so.5	
	SUSE Linux Enterprise Server (SLES) 11	2.6.27.19-5	glibc-2.9-13.2	
	SUSE Linux Enterprise Server (SLES) 10 SP2	2.6.16	glibc-2.4-31	
	Ubuntu 8.0.4	2.6.24-19	glibc-2.7.so	
x86_64 AMD64/ EM64T	Red Hat Enterprise Linux (RHEL)	5.2.6.18-92l	libstdc++.so	All 32-bit compatibility libraries are required for 32-bit applications.
	SUSE Linux Enterprise Server (SLES) 11	2.6.27.19-5	glibc-2.9-13.2	All 32-bit compatibility libraries are required for 32-bit applications.
	SUSE Linux Enterprise Server (SLES) 10 SP2	2.6.16	glibc-2.4-31	All 32-bit compatibility libraries are required for 32-bit applications.
	Ubuntu 8.0.4	2.6.24-19	glibc-2.7.so	

Hardware requirements

DB2 products are supported on the following hardware:

- ▶ HP-UX
 - Itanium®-based HP Integrity Series Systems
- ▶ Linux
 - x86 (Intel Pentium®, Intel Xeon®, and AMD) 32-bit Intel and AMD processors
 - x64 (64-bit AMD64 and Intel EM64T processors)

- POWER® (IBM eServer™ OpenPower®, iSeries, pSeries®, System i, System p, and POWER Systems that support Linux)
- System z or System z9®
- ▶ Solaris
 - UltraSPARC or SPARC64 processors
 - Solaris x64 (Intel 64 or AMD64)
- ▶ Windows
 - All Intel and AMD processors capable of running the supported Windows operating systems (32-bit and 64-bit base systems)

For more information about DB2 9.7 system requirements and other DB2 release system requirement, check this Web site:

<http://www.ibm.com/software/data/db2/9/sysreqs.html>

Disk requirements for DB2 servers

The disk space that is required for your product depends on the type of installation that you choose and the type of file system that you have. The DB2 setup wizard provides dynamic size estimates based on the components selected during a typical, compact, or custom installation. The following sizes are the sizes for the DB2 Express-C installation:

- ▶ **Typical: Requires 500 - 610 MB**
 With the Typical installation type, DB2 is installed with most of the features and functionality, including graphical tools, such as the Control Center and DB2 Instance Setup wizard.
- ▶ **Compact: Requires 450 - 550 MB**
 With the Compact installation type, only basic DB2 features and functions are installed. A minimal configuration will be performed, and graphical tools are not included.
- ▶ **Custom: Requires 450 - 1080 MB**
 With the Custom installation type, you can select the features that you want to install. The disk space needed varies based on the selected features.

When you install DB2 Enterprise Server Edition or Workgroup Server Edition using the DB2 setup wizard, size estimates are dynamically provided by the installation program based on the installation type and component selection.

If the space required for the installation type and components exceeds the space found in the path specified, the setup program issues a warning about insufficient space. The installation is allowed to continue. If the space for the files being

installed is in fact insufficient, installation will stop, and the setup program will need to be aborted if additional space cannot be provided.

Remember to include disk space for required databases, software, and communication products.

On the Linux and UNIX operating systems, 2 GB of free space in the /tmp directory is recommended.

Memory requirements for servers

At minimum, a DB2 database system requires 256 MB of RAM. For a system running a DB2 product and the DB2 GUI tools, a minimum of 512 MB of RAM is required. For improved performance, 1 GB of RAM is recommended. These requirements do not include additional memory requirements for other software components running on your system.

When determining memory requirements, ensure to remember the following considerations:

- ▶ DB2 products that run on HP-UX Version 11i for Itanium-based™ systems require 512 MB of RAM at a minimum.
- ▶ For IBM Data Server Client support, these memory requirements are for a base of five concurrent client connections. You will need an additional 16 MB of RAM for every five client connections.
- ▶ Memory requirements are affected by the size and complexity of your database system, as well as by the extent of database activity and the number of clients accessing your system.
- ▶ For DB2 server products, the self-tuning memory feature simplifies the task of memory configuration by automatically setting values for several memory configuration parameters. When enabled, the memory tuner dynamically distributes available memory resources among several memory consumers, including sort memory, the package cache, lock list memory, and buffer pools.
- ▶ Additional memory can be required for non-DB2 software that might be running on your system.
- ▶ Specific performance requirements can determine the amount of memory needed.
- ▶ On the Linux operating system, we recommend a SWAP space at least twice as large as RAM.

Communication requirements

When using TCP/IP as the communication protocol, no additional software is needed for connectivity. For more supported communication protocols, refer to the DB2 manual *Quick Beginnings for DB2 Servers 9.5*, GC10-4246, at this Web site:

<http://publibfp.boulder.ibm.com/epubs/pdf/c2358642.pdf>

Or, you can visit the IBM DB2 Database for Linux, UNIX, and Windows Information Center at this Web site:

<https://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp>

5.1.2 Installation procedure

Table 5-2 shows the four installation methods in which DB2 can be installed. For completeness, we have provided the information for supported Linux, UNIX, and Windows operating systems. Each of the methods listed has its own advantages and disadvantages and varies upon the environment. For a discussion about the preferred method, refer to *Up and Running with DB2 on Linux*, SG24-6899.

Table 5-2 DB2 installation methods

Installation method	Linux	UNIX	Windows
DB2 setup wizard	Yes	Yes	Yes
db2_install	Yes	Yes	No
Response file installation	Yes	Yes	Yes
Payload file deployment (Manual installation)	Yes	Yes	No

For this particular project or any other conversion project in general, the *DB2 Data Server Client* is required. It provides libraries for application development. If the application server and the database server are to be placed on the same system, you can install both the DB2 server and Data Server Client in one step by selecting the Custom installation type.

For this project, we perform the following steps to install DB2 9.7 on Linux:

1. Log on to Linux as a root user.
2. Download DB2 9.7 Express-C from this Web site:
<http://www.ibm.com/software/data/db2/express/download.html>
3. Save the tar file to the `/usr/local/src` directory.

4. Change directories to the `/usr/local/src` directory:

```
cd /usr/local/src/expc
```

5. Extract the tar file:

```
tar -xzf db2exc_970_LNX_x86.tar.gz
```

6. Change to the directory:

```
cd /usr/local/src/expc
```

7. Launch the DB2 setup wizard, which opens the panel that is shown in Figure 5-1:

```
./db2setup
```

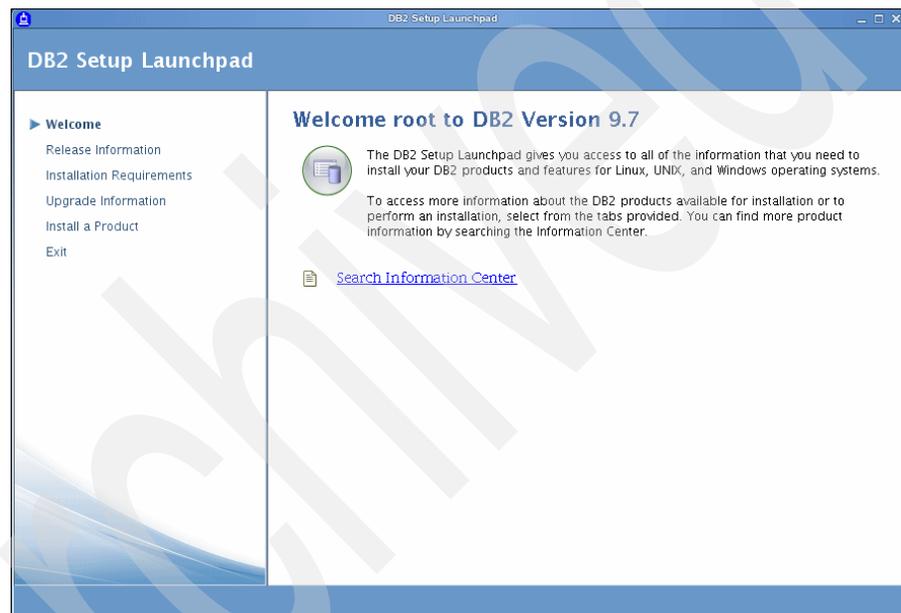


Figure 5-1 DB2 Setup Launchpad

Note: Starting in DB2 9.5, you can also perform a non-root installation of DB2.

8. When the DB2 Launchpad opens, choose **Install a Product**, as shown in Figure 5-2 on page 94.

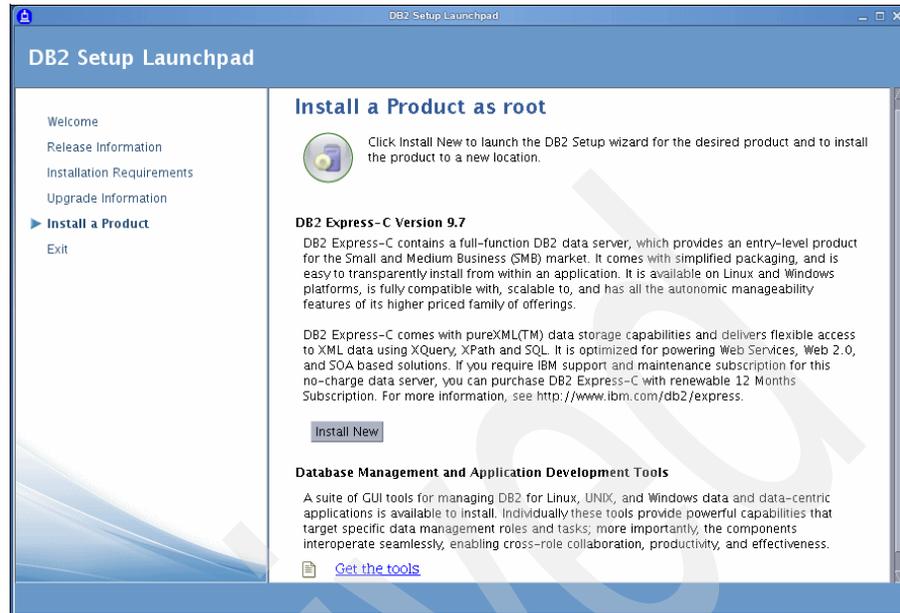


Figure 5-2 DB2 setup launchpad

9. Select **Install New** under the option to install the server to launch the DB2 setup wizard.
10. Go to the Software License Agreement panel, and read the Software License Agreement, as shown in Figure 5-3 on page 95. If you agree with the agreement, select **Accept**, and click **Next**.

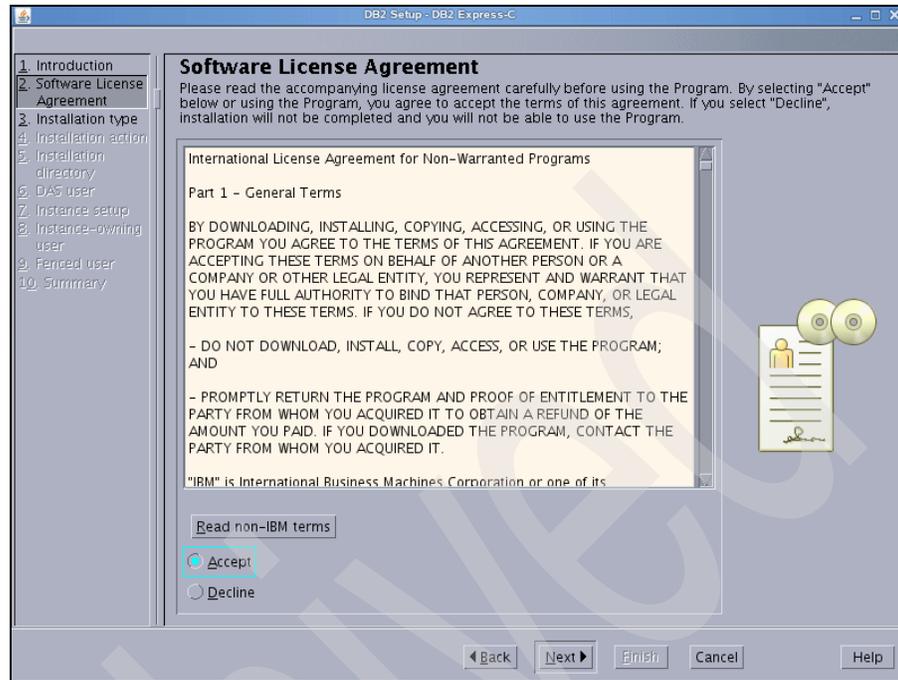


Figure 5-3 DB2 Software License Agreement panel

11. In the Installation Type panel, click **Custom**, as shown in Figure 5-4 on page 96, and click **Next**.

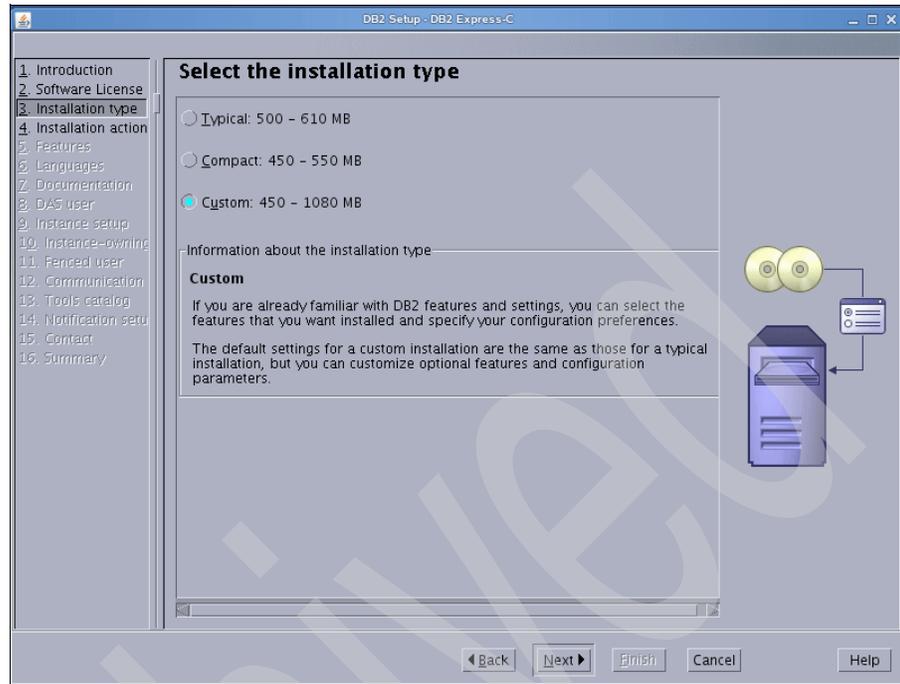


Figure 5-4 DB2 installation type panel

12. Click **Next** again to get the Features panel. Select the **Application Development tools** option, as shown in Figure 5-5 on page 97, and click **Next**.

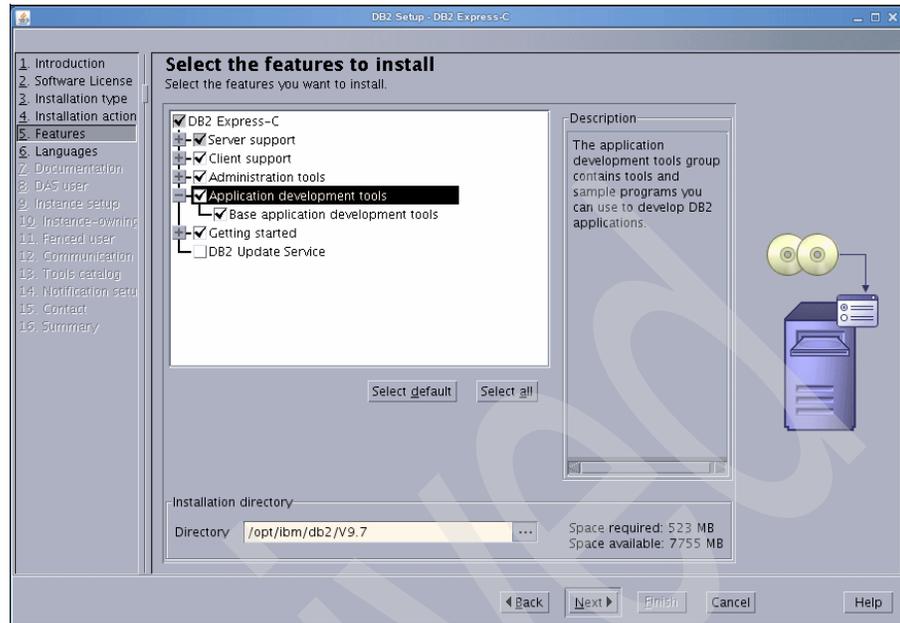


Figure 5-5 DB2 custom installation with application development tools selected

13. In the Languages panel, choose the type of languages to install, and click **Next**.
14. In the Documentation panel, choose where to access the DB2 Information Center. You can choose to install it as part of this process, or you can access the online DB2 Information Center at any time. Click **Next**.
15. In the Database Administration Server (DAS) panel, enter the DAS user information. Linux group and user accounts do not have to be created prior to this step; DB2 will create the required Linux system group and user automatically. For the example installation, we use the default name dasusr1 and choose a password for this user, as shown in Figure 5-6 on page 98, and click **Next**.

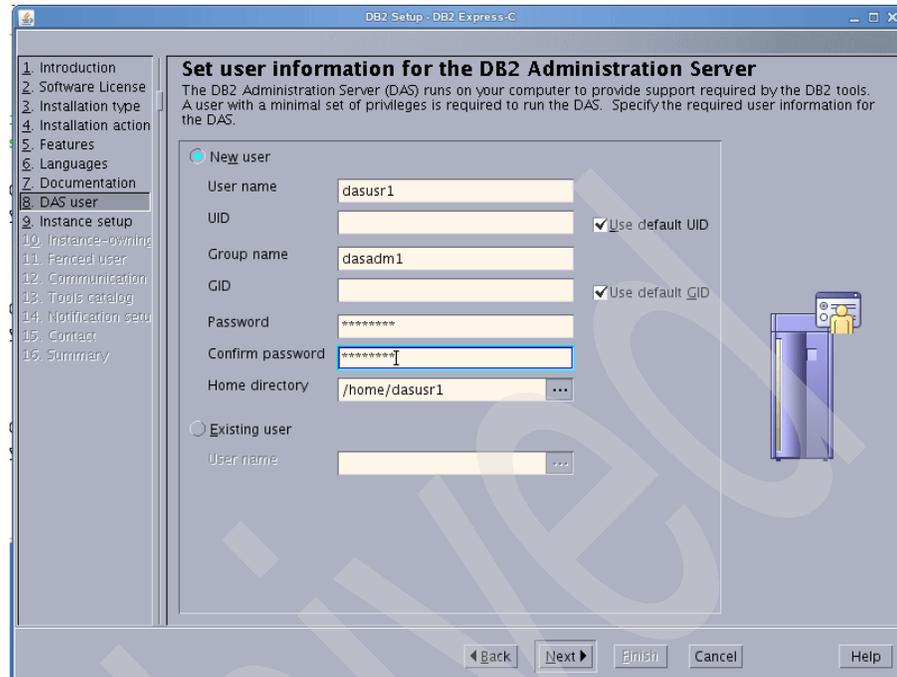


Figure 5-6 DB2 setup administration server panel

16. In the Instance setup panel, you can choose whether you want to set up an instance during the DB2 installation. By selecting **Create a DB2 instance** and clicking **Next**, we let DB2 create the instance for us.
17. Enter the instance owner information in the Instance Owner panel. Linux group and user accounts do not have to be created prior to this step; DB2 will create the Linux group and user. For the example installation, we use the default db2inst1 settings and create a password for this user, as shown in Figure 5-7 on page 99, and click **Next**.

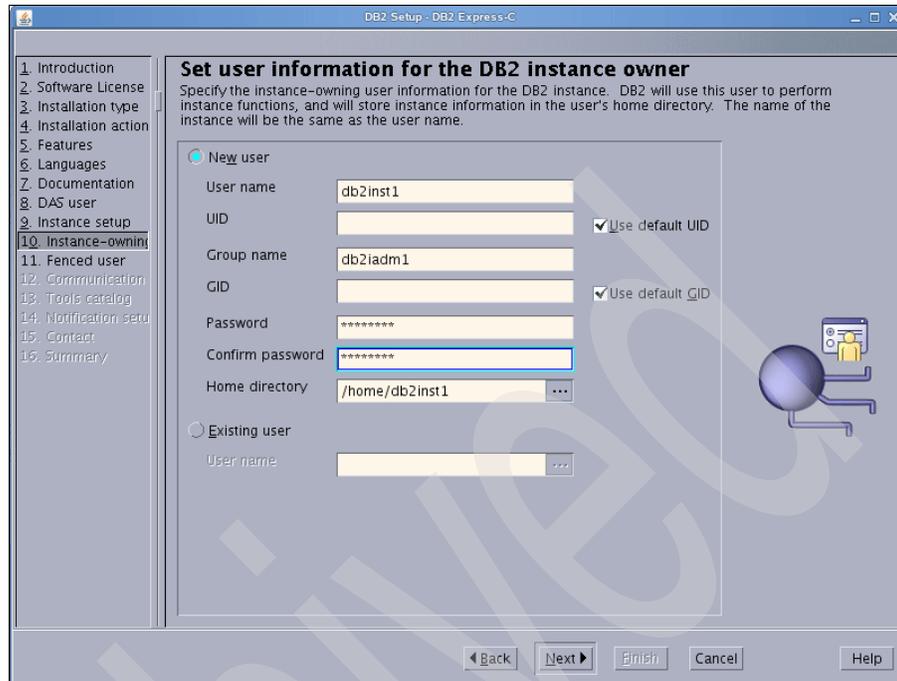


Figure 5-7 DB2 set up DB2 instance owner

18. In the Fenced user panel, we allow DB2 create the ID for us, as shown in Figure 5-8 on page 100, and click **Next**.

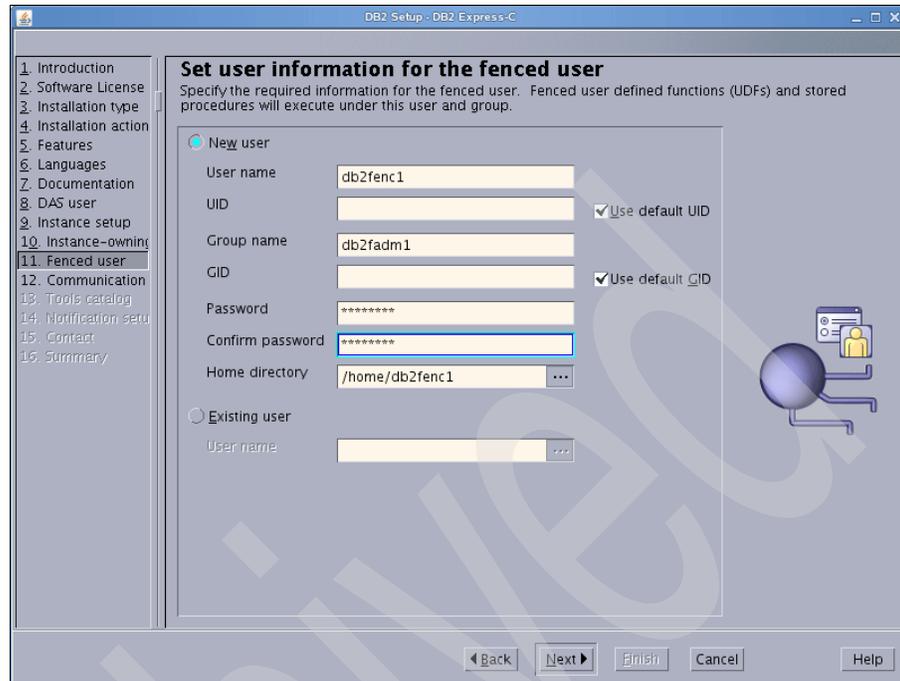


Figure 5-8 DB2 set up DB2 fenced user

19. In the Instance Communication panel, configure the DB2 instance TCP/IP communication. For the example installation, we use the default settings and click **Next**.
20. In the DB2 Tools Catalog Configuration panel, we create a local tools catalog by selecting the db2inst1 instance. We also use the default local database TOOLSDB and default schema SYSTOOLS and then, click **Next**. Figure 5-9 on page 101 shows the DB2 Tools Catalog Configuration panel.

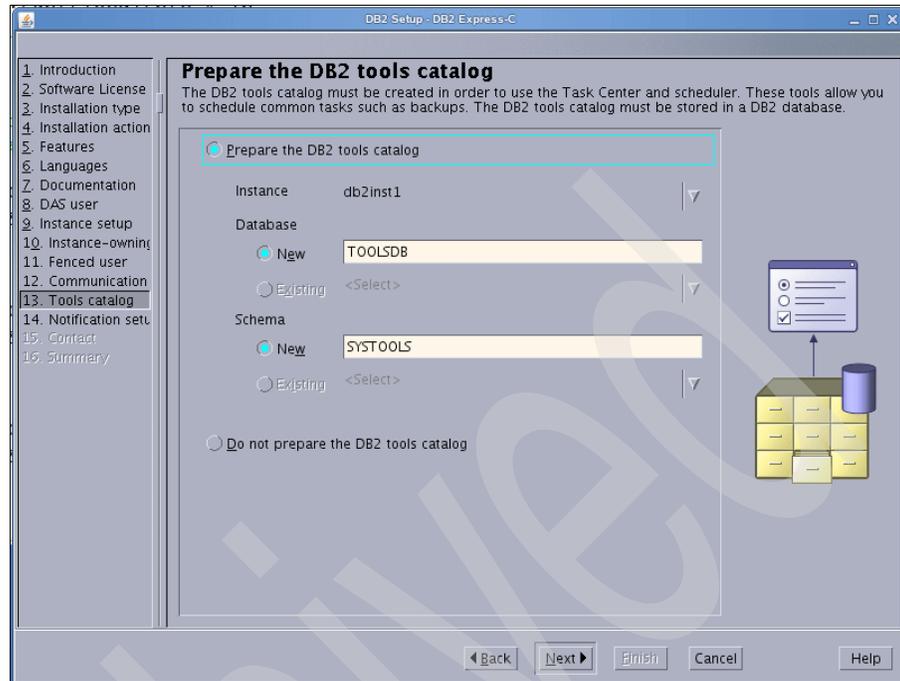


Figure 5-9 Prepare DB2 tools catalog

21. At the end, the setup wizard provides a summary of the installation options selected. Review it and click **Finish** to start the installation.

Fix pack installation

We recommend that you install the latest DB2 fix pack:

1. Download the fix pack from this Web site:
http://www.ibm.com/software/data/db2/support/db2_9/
2. Change to the directory in which the installation image is located.
3. Enter the **installFixPak** command to launch the installation.
4. Update instances to run against the new code with the **db2iupd** command.
5. Update DB2 Administration Server (DAS) using the **dasupdt** command.
6. Restart applications.

The db2setup command options

The **db2setup** command provides options for specifying the locations of the log files, trace file, or response file created during installation (see Figure 5-10 on page 102). Log files are useful for verifying the installation status and tracing the

problem if the installation failed. By default, **db2setup** starts the graphical user interface (GUI) and picks up the language flag from the operating systems settings.

```
>>-db2setup----->
      '--i--language-' '--l--log_file-'
>-----<
      '--t--trace_file-' '--r--response_file-' +--?--+
                                          '--h-'
```

Figure 5-10 The db2setup command options

If the log file option is not specified, the db2setup.log file and db2setup.err file are stored in the /tmp directory on a Linux operating system. Example 5-1 shows an example of the db2setup.log file.

Example 5-1 DB2setup log file

```
DB2 Setup log file started at: Fri Jul 24 15:20:15 2009 EDT
=====

Operating system information: Linux 2.6.16.60-0.21-smp.#1 SMP Tue May 6
12:41:02 UTC 2008 i686

Product to install:                DB2 Express-C
Installation type:                  Custom
...

```

5.1.3 Instance creation

For our project, we allow DB2 to create the DB2 instance automatically during installation (Figure 5-8 on page 100). Although this option is the default, you can turn automatic instance creation off during installation and create instances and databases manually after the installation has completed.

If you have chosen not to create instances during the DB2 installation or need to add another instance after installation, there are two options to create instances manually:

- ▶ The **db2i setup** command starts a graphical tool for creating and configuring instances, as shown in Figure 5-11 on page 103. It allows you to specify all the required configuration parameters, such as the instance owner and communication protocol, in an easy, guided fashion. The command can be found in the /opt/ibm/db2/V9.7/instance directory on a Linux operating system.

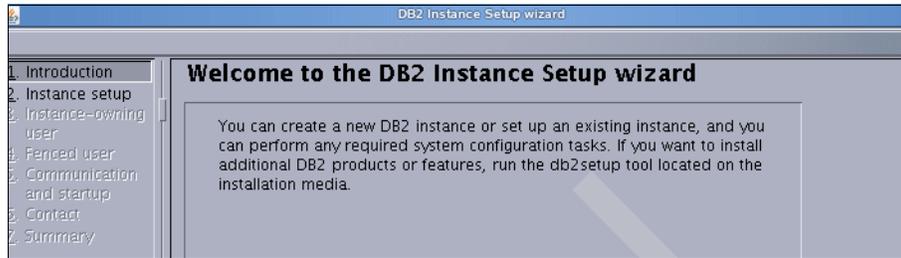


Figure 5-11 Graphical user interface for `db2isetup`

- ▶ The second option is the `db2icrt` command. It is a command-line alternative to create the instances, for example:

```
db2icrt -u db2fenc1 db2inst1
```

The command to create the DAS user is the `dasCRT` command. Use it in the following way:

```
dasCRT -u dasadm1
```

As part of the GUI instance creation, the installer suggests three users identified as `db2inst1`, `db2fenc1`, and `dasadm1`. These are default names for the instance users. If you do not want to use the default names, you can choose your own names by creating the system user IDs and groups ahead of time and inputting these parameters in the wizard when prompted. The installer will also add the following entry to the `/etc/services` file in order to allow communication from DB2 clients:

```
db2c_db2inst1 50000
```

In this entry, `db2c_db2inst1` indicates the service name, and `50000` indicates the port number. DB2 allows for multiple instances on one server installation to allow for various environments, that is, test, production, development, and so on. Subsequent instances can be created on the same server simply by using one of the methods introduced here.

5.1.4 Client setup on Linux

This section discusses installation and configuration of DB2 clients to access a remote DB2 server. DB2 provides two types of clients at no charge:

- ▶ DB2 Runtime Client: This client is best suited for enabling applications to access DB2 servers.
- ▶ DB2 Client: This client includes all the functionality found in the DB2 Runtime Client, plus functionality for the client/server configuration, database administration, and application development.

All clients are supported on Linux, AIX, HP-UX, Solaris, and Windows operating systems.

Note: Typically, in a production environment, DB2 clients are installed on separate physical machines from the DB2 server. However, for an application development environment, it can be useful to have everything, such as the DB2 database server plus the clients, on the same machine.

To access a remote DB2 database, you can either run the easy to use graphical tool Configuration Assistant or use the catalog commands to provide entries for the following three directories:

- ▶ **NODE** directory: A list of remote DB2 instances
- ▶ **ADMIN NODE** directory: A list of remote DB2 Administration servers
- ▶ **DATABASE** directory: A list of databases

To use the command-line tools, first catalog the DB2 node. The *DB2 node* is the server where the database resides. Then, catalog the database. See Example 5-2.

Example 5-2 Cataloging the DB2 node and the database

```
--
-- catalog database node
--
CATALOG TCPIP NODE db2node REMOTE server1 SERVER 50001

--
-- catalog the DAS on the remote node
--
CATALOG ADMIN TCPIP NODE db2das remote SERVER1
-
-- catalog database
--
CATALOG DATABASE invent AS inventdb AT NODE db2node
```

After installing your DB2 Client, configure it to access a remote DB2 server using the Configuration Assistant. The graphical interface can be launched through the DB2 Control Center or run on its own by using the command **db2ca**. For more details, refer to the IBM DB2 manual *Quick Beginnings for DB2 Clients*, GC10-4242.

5.2 Other software products

All software requirements for the target environment must be identified in the conversion planning and preparation stage. Prior to conversion of any data objects, all software must be installed and configured.

The sample application in this book is written in PHP with Apache2. Therefore, the next two sections discuss how to prepare the target system for Apache2 and PHP.

5.2.1 Apache2 installation with DB2 support

When converting an application from one server to another server, you must ensure that all software is properly installed on the new server. In our conversion scenario, we use Apache.

Installation steps

The following steps explain how we install Apache2 on SUSE 10 SP2:

1. Download the Apache package.

The source code for Apache package is available at this Web site:

<http://httpd.apache.org/download.cgi>

In our conversion scenario, we use Version 2.2.11 of Apache, and the package that we download is `httpd-2.2.11.tar.gz`.

2. Change the working directory.

Use the `cd` command to make your working directory the directory to which you download the tar file:

```
db2server: # cd /usr/local/src/
```

3. Uncompress the source package.

The following command decompresses the contents of the source package into a directory called `httpd-2.2.11`:

```
db2server:/usr/local/src # tar -xzf httpd-2.2.11.tar.gz
```

4. Change the working directory.

Use the `cd` command to make the newly created directory your working directory:

```
db2server:/usr/local/src # cd httpd-2.2.11/
```

5. Specify the configuration options for the PHP source.

Possible configuration options can be listed by issuing the following command:

```
db2server:/usr/local/src/httpd-2.2.11# configure -help
```

6. Run the configure script.

The configure script builds the Makefile. The following script is the configuration command that we use to set up our server:

```
db2server:/usr/local/src/httpd-2.2.11# ./configure
--prefix=/usr/local/apache2
--enable-so
--enable-cgi
--enable-info
--enable-rewrite
--enable-speling
--enable-usertrack
--enable-deflate
--enable-ssl
--enable-mime-magic
```

Where option:

--prefix	Specifies the Apache install directory
--enable-so	Dynamically Shared Object (DSO) capability, which allows you to load modules into Apache at run time
--enable-cgi	Enables support for CGI scripts
--enable-info	Enables server information
--enable-rewrite	Enables rule-based URL manipulation
--enable-speling	When enabled, Apache corrects common URL misspellings
--enable-usertrack	Enables user-session tracking
--enable-deflate	Deflates transfer encoding support
--enable-ssl	Enables SSL/TLS support (mod_ssl)
--enable-mime-magic	Automatically determines Multipurpose Internet Mail Extensions (MIME) type

You can run the **configure** command with the **--help** option to get a full list of options for configuring Apache.

7. Compile Apache.

After configuring the source files, the compile process is started using the **make** command. We pipe the output to a log file in order to check for failure afterward:

```
db2server:/usr/local/src/httpd-2.2.11 # make > apacheMake.out
```

8. Install Apache.

After Apache compiles successfully, it can be installed as the root user:

```
db2server:/usr/local/src/httpd-2.2.11 # make install
```

9. Add the apachectl script to the following directories.

Use the **ln** command to create a link to the apachectl file in the `/usr/bin` and `/etc/init.d` directories:

```
ln -s /usr/local/apache2/bin/apachectl /usr/bin/apachectl
ln -s /usr/local/apache2/bin/apachectl /etc/init.d/
```

10. Start the Apache httpd server.

Use the following command to start the apache httpd server:

```
apachectl start
```

5.2.2 PHP installation with DB2 support

There are four major interfaces that are available for connecting to a DB2 database in PHP:

- ▶ The *ibm_db2* extension interface provides an API to read/write from/to the database. This extension makes it easy to convert Unified Open Database Connectivity (ODBC) applications to use *ibm_db2*.
- ▶ *PDO_ODBC* and *PDO_IBM* are object-oriented extension interfaces that are used to connect to a database. Both extension interfaces are based on PDO PHP Data Object (PDO) standards. *PDO_IBM* includes the IBM database driver for PDO.
- ▶ *Unified ODBC* is the traditional procedural interface and supports multiple database servers. The Unified ODBC extension is not optimized for DB2 and therefore not recommended for new applications.

When converting a PHP application on an existing server, preparation and verification must be performed to have DB2 connection support. If a precompiled package of PHP is used, by default, only a predefined set of functions and database support is integrated. To see which configuration options are included in the previous installed PHP version, the PHP function *phpinfo()* can be invoked out of a Web page file. The included configuration commands are listed in the third section of the PHP information, as shown in Figure 5-12 on page 108.

System	Linux dmsrvr 2.6.16.60-0.21-smp #1 SMP Tue May 6 12:41:02 UTC 2008 i686
Build Date	Jul 27 2009 01:17:36
Configure Command	'./configure' '--prefix=/usr/local/apache2/php' '--with-ibm-db2=/home/db2inst1/sqllib' '--with-apxs2=/usr/local/apache2/bin/apxs' '--with-config-file-path=/usr/local/apache2/php'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/apache2/php
Loaded Configuration File	(none)
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20090626
PHP Extension	20090626
Zend Extension	220090626
Zend Extension Build	API220090626,NTS
PHP Extension Build	API20090626,NTS
Debug Build	no
Thread Safety	disabled
Zend Memory	enabled

Figure 5-12 PHP configuration options

Note: All commands and procedure descriptions that are provided in this section refer to SUSE Linux Enterprise Server 10 SP2. The commands and procedures can vary for other versions or Linux distributions.

Installation steps

In order to use the IBM DB2 libraries, you must recompile PHP. These installation steps explain how to update your PHP install and how to install PHP from the beginning:

1. Back up the `httpd.conf` and `php.ini` files.

To ensure the configuration files for Apache and PHP are not lost when installing the new PHP version, we recommend backing up the `/etc/httpd/httpd.conf` and if you have a previous version of PHP installed, back up the `/etc/php.ini` files.

2. Download the PHP package.

The source code for PHP is available at this Web site:

<http://www.php.net/downloads.php>

In our conversion scenario, we use Version 5.3.0 of PHP, and the package that we downloaded was php-5.3.0.tar.gz.

Download the ibm_db2 PECL extension at this Web site:

http://pecl.php.net/package/ibm_db2

In our conversion scenario, we use Version 1.2.8, and the package that we downloaded was ibm_db2-1.8.2.tgz.

Download the PDO_IBM PECL extension at this Web site:

http://www.pecl.php.net/package/PDO_IBM

In our conversion scenario, we use Version 1.3.0, and the package that we downloaded was PDO_IBM-1.3.0.tgz.

3. Uncompress the source package.

The following command decompresses the contents of the source package to a directory called php-5.3.0:

```
db2server:/usr/local/src # tar xzf php-5.3.0
```

4. Add the PECL extensions to the php install directory.

Use the `mv` command to move the compressed files to the extension directory in the install directory:

```
db2server:/usr/local/src # mv ibm_db2-1.8.2.tgz php-5.3.0/ext/.
db2server:/usr/local/src # mv PDO_IBM-1.3.0.tgz php-5.3.0/ext/.
```

5. Change the working directory.

Use the `cd` command to make the `ext` directory your working directory:

```
db2server:/usr/local/src # cd php-5.3.0/ext/
```

6. Uncompress the PECL extension packages.

The following command decompresses the contents of the extension packages into directories called `ibm_db2-1.8.2` and `PDO_IBM-1.3.0`:

```
db2server:/usr/local/src/php-5.3.0/ext/ # gzip -d < ibm_db2-1.8.2.tgz |
tar -xvf -
db2server:/usr/local/src/php-5.3.0/ext # gzip -d < PDO_IBM-1.3.0.tgz |
tar -xvf -
```

7. Rename the extension directories.

Use the `mv` command to rename the extension directories:

```
db2server:/usr/local/src/php-5.3.0/ext # mv ibm_db2-1.8.2 ibm_db2
db2server:/usr/local/src/php-5.3.0/ext # mv PDO_IBM-1.3.0 pdo_ibm
```

8. Change the working directory.

Use the **cd** command to make the PHP install directory your working directory:

```
db2server:/usr/local/src # cd /usr/local/src/php-5.3.0/
```

9. Remove the configure file.

Use the **rm** command to remove the PHP configure script:

```
db2server:/usr/local/src/php-5.3.0 # rm configure
```

10. Rebuild the configure file.

The **buildconf** command rebuilds the configure file to include the new extensions:

```
db2server:/usr/local/src/php-5.3.0 # ./buildconf --force
```

11. Verify that the extension is now within the configure file.

Use the following command to verify that the **buildconf** command worked successfully:

```
db2server:/usr/local/src/php-5.3.0 # ./configure --help | grep  
with-ibm-db2  
db2server:/usr/local/src/php-5.3.0 # ./configure --help | grep pdo-ibm
```

12. Specify the configuration options for the PHP source.

A list of the possible configuration options can be seen by issuing the following command:

```
db2server:/usr/local/src/php-5.3.0 # configure -help
```

13. Run the configure script.

The configure script builds the Makefile. For our purposes, we specify the **configure** command:

```
db2server:/usr/local/src/php-5.3.0 # ./configure  
--prefix=/usr/local/apache2/php  
--with-IBM_DB2=/opt/ibm/db2/V9.7  
--with-pdo-ibm=/opt/ibm/db2/V9.7  
--with-pdo-odbc=ibm-db2,/home/db2inst1/sql1lib  
--with-ibm-db2=/opt/ibm/db2/V9.7  
--with-apxs2=/usr/local/apache2/bin/apxs  
--with-config-file-path=/usr/local/apache2/php
```

Where option:

--prefix	Specifies the PHP install directory
--with-IBM_DB2	Is for the ibm_db2 extension
--with-pdo-ibm	Is for the PDO_IBM extension
--with-pdo-odbc	Is for the PDO_ODBC extension

--with-ibm-db2 Is for the Unified ODBC extension

--with-apxs2 Is for the Apache apxs tool, which allows you to build extension modules to add to Apache's functionality

--with-config-file-path Allows you to specify the path to the `php.ini` file

If a particular extension is not required, it can be removed from the **configure** command and the install.

14. Compile PHP.

After configuring the source files, start the compile process by using the **make** command. We pipe the output to a log file in order to check for failures afterward:

```
db2server:/usr/local/src/php-5.3.0 # make > phpMake.out
```

15. Install PHP.

After PHP has compiled successfully, install it as the root user:

```
db2server:/usr/local/src/php-5.3.0 # make install
```

16. Configure Apache.

If Apache was already configured for PHP on your server, you only have to make certain changes. As part of the installation process, PHP automatically modifies the Apache configuration file `/usr/local/apache2/conf/httpd.conf`. Confirm that the `httpd.conf` has the following lines. If the lines are missing, update the file:

- `LoadModule php5_module modules/libphp5.so`
- `directoryIndex index.php index.html index.htm index.html.var`
- `AddType text/html php`
- `AddType application/x-httpd-php.php`

17. Check the `php.ini` file.

Either copy `php.ini-dist` in the Configuration File Path (see Figure 5-12 on page 108) of PHP and rename it to `php.ini`, or if your PHP version is the same as before, copy the version that was saved in step 1 back in this directory.

18. Start the Apache `httpd` server.

Apache has to be restarted to use the configuration changes:

```
db2server:/usr/local/src/php-5.3.0 # apachectl restart
```

5.3 IBM Data Movement Tool installation and usage

The IBM Data Movement Tool is an ingenious tool that offers schema and data movement from various database providers to DB2. The tool accepts information about the source database and target database, extracts the database objects and data from source database, and can input this data into the target DB2 database.

For the IBM Data Movement user guide and a no charge download of the IBM Data Movement Tool, visit this Web site:

<http://www.ibm.com/developerworks/data/library/techarticle/dm-0906datamovement/>

5.3.1 IBM Data Movement Tool prerequisites

The following prerequisites are required to successfully deploy the IBM Data Movement Tool for MySQL.

Software requirements

We describe the software that is required to use the IBM Data Movement Tool in this topic.

General requirements

In general, you need the following software:

- ▶ Latest version of the IBM Data Movement Tool.
- ▶ MySQL: Ensuring that MySQL is running (usually the daemon can be started with the command **safe_mysqld &** from an account with root permissions).
- ▶ DB2 V9.7 needs to be installed on the target server. Use the command **db2start** to ensure that the DB2 Server is up and running.
- ▶ Java Version 1.5 or higher must be installed on your target server. To verify your current Java version, run the **java -version** command. By default, Java is installed as part of DB2 for Linux, UNIX, and Windows in `<install_dir>\SQLLIB\java\jdk` (Windows) or `/opt/ibm/db2/V9.7/java/jdk` (Linux).
- ▶ You must have the Java Database Connectivity (JDBC) drivers for the MySQL source database (`mysql-connector-java-5.1.8-bin.jar` or the latest driver) and the DB2 target database (`db2jcc.jar`, `db2jcc_license_cu.jar`, or `db2jcc4.jar`, `db2jcc4_license_cu.jar`) installed on the server with the IBM Data Movement Tool.

Operating system

IBM Data Movement Tool supports the following operating systems:

- ▶ Windows
- ▶ z/OS
- ▶ AIX
- ▶ LINUX
- ▶ Solaris
- ▶ HP-UX
- ▶ MacIntosh

For the purpose of this document, we have used the IBM Data Movement Tool with DB2 Version 9.7 and MySQL Version 5.1.36. We recommend that you install the IBM Data Movement Tool on the DB2 server side to achieve the best data movement performance.

5.3.2 IBM Data Movement Tool installation

Installing the IBM Data Movement Tool is simple and straightforward:

1. Download the latest version of the IBM Data Movement Tool from this Web site:

<http://www.ibm.com/developerworks/data/library/techarticle/dm-0906datamovement/>

2. Create a directory for the IBM Data Movement Tool, and copy `IBMDDataMovementTool.zip` to this directory:

```
db2server:/opt/ibm # mkdir IBMDDataMovementTool
db2server:/usr/local/src # cp IBMDDataMovementTool.zip
/opt/ibm/IBMDDataMovementTool/
```

3. Expand the compressed package using this command:

```
db2server:/opt/ibm/IBMDDataMovementTool/ # unzip IBMDDataMovementTool.zip
```

To start the IBM Data Movement Tool, execute `IBMDDataMovementTool.sh` for Linux and UNIX and the `IBMDDataMovementTool.cmd` command for Windows. Figure 5-13 on page 114 shows the IBM Data Movement Tool window.

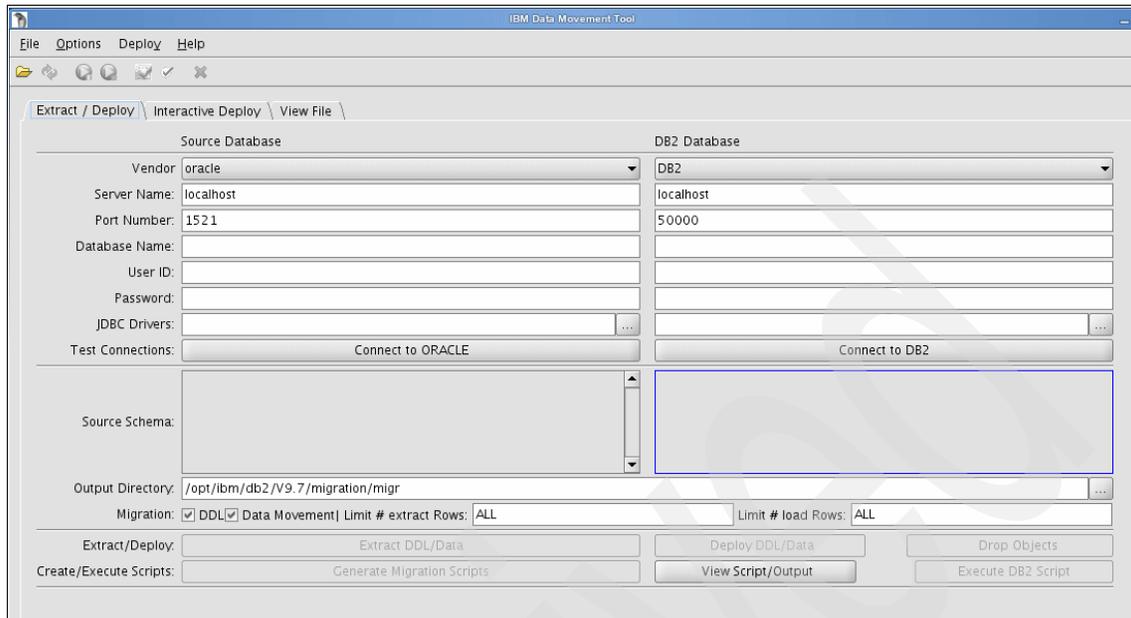


Figure 5-13 IBM Data Movement Tool

Database conversion

After you create the conversion plan and install and set up all of the required software, it is time to translate the source MySQL database structure into DB2.

In this chapter, we discuss the process of converting the database structure from the MySQL 5.1 server to the DB2 9.7 server. Before this discussion, we must evaluate the differences between the MySQL database structure and the DB2 database structure.

In the first section, we discuss data type mapping, taking a closer look at MySQL and DB2 data types and the differences between them. Following this section, we provide Data Definition Language (DDL) differences, providing a syntax comparison between MySQL and DB2.

In the succeeding section, we provide additional considerations for users while porting the database schema from MySQL to DB2.

In the last section, we provide detailed information regarding the database schema porting steps using the following approaches:

- ▶ Porting using the IBM Data Movement Tool
- ▶ Manual porting
- ▶ Metadata transform

6.1 Data type mapping

In this section, we compare the differences between MySQL and DB2 data types. In general, all MySQL data types can be mapped to DB2 data types. With the assistance of the IBM Data Movement Tool, this process can be effortless.

Every column in the database table has an associated data type, which determines the values that this column can contain. DB2 supports both built-in data types and user-defined data types (UDT) whereas MySQL only supports built-in data types. Figure 6-1 shows the built-in data types of MySQL.

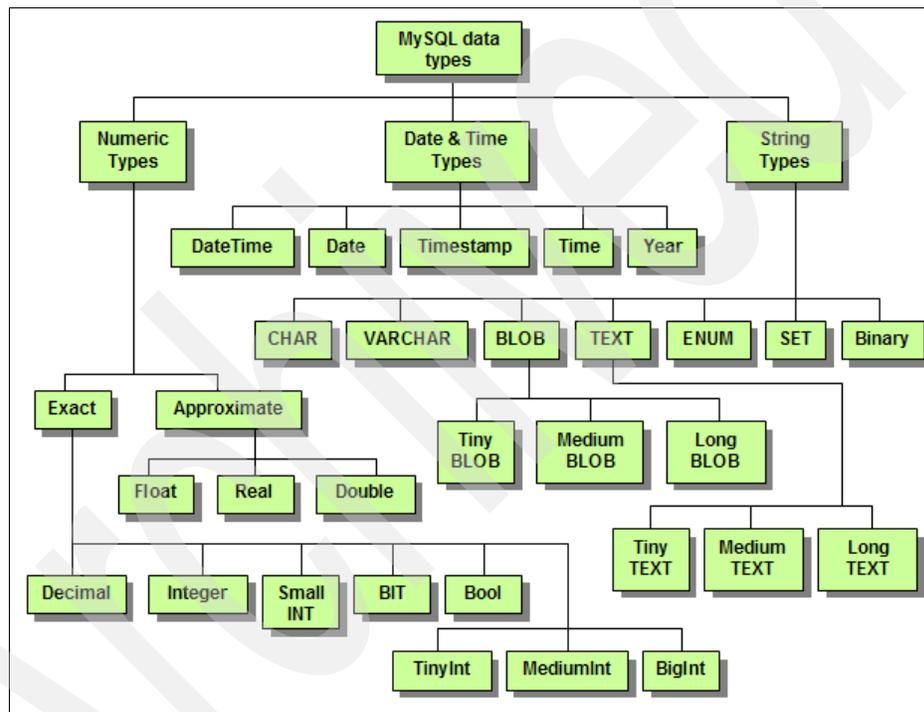


Figure 6-1 MySQL data types

Figure 6-2 on page 117 shows the built-in data types that are supported by DB2.

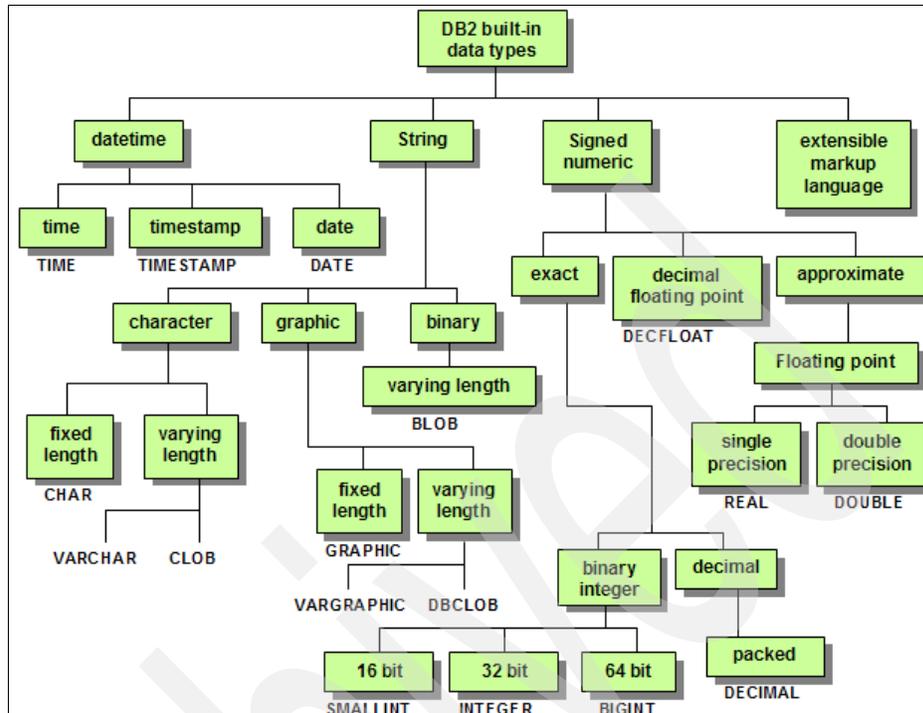


Figure 6-2 DB2 data types

MySQL data types are grouped into three categories and can be converted to DB2 data types following these suggested rules:

► Numeric type:

– TINYINT

A tiny integer is a single-byte integer in MySQL that can be mapped to a DB2 SMALLINT for similar functionality.

– SMALLINT

A small integer is a two-byte integer with a precision of five digits. With MySQL, the range of signed small integers is -32768 to 32767, making it replaceable by DB2 SMALLINT. For unsigned MySQL small integers, the range is 0 to 65535, making it replaceable by DB2 INTEGER.

– BIT, BOOL, and BOOLEAN

These types are synonyms for TINYINT(1). Instead of BIT, BOOL, and BOOLEAN, DB2 uses SMALLINT with check constraint.

- MEDIUMINT

This type is a medium-sized integer with a signed range of -8388608 to 8388607 or 0 to 16777215 for the unsigned range. DB2 uses an INTEGER for this type, which has a range of -2147483648 to 2147483647.

- INTEGER and INT

An integer is 4-byte integer for both MySQL and DB2. The range of a signed MySQL INTEGER and DB2 INTEGER is -2147483648 to 2147483647; therefore, the DB2 INTEGER can be used. For the unsigned MySQL INTEGER, the range is 0 to 4294967295. This type can be replaced by DB2 BIGINT.

- BIGINT

A big integer is an 8-byte integer for both MySQL and DB2. A DB2 BIGINT can be used for a signed MySQL BIGINT, because the range of a signed MySQL BIGINT and a DB2 BIGINT is -9223372036854775808 to 9223372036854775807. For unsigned MySQL BIGINT, the range is 0 to 18446744073709551615; DB2 DECIMAL will cover the range.

- FLOAT

A FLOAT in MySQL is a single precision floating-point number ranging from -3.402823466E+38 to -1.175494351E-38, 0, and 1.175494351E-38 to 3.402823466E+38. Whereas in DB2, it is a double precision floating-point number ranging from -1.79769E+308 to -2.225E-307, or from 2.225E-307 to 1.79769E+308. Hence, a FLOAT value in MySQL can be directly mapped to a DOUBLE in DB2.

- DOUBLE

A DOUBLE is a double precision floating-point number for both DB2 and MySQL. A signed MySQL DOUBLE can be directly mapped to DOUBLE in DB2. An unsigned MySQL DOUBLE can be mapped to DECIMAL in DB2.

- REAL

This type is a synonym for DOUBLE in MySQL and, therefore, can be mapped to the same DB2 data types as the MySQL DOUBLE data type.

- DECIMAL, NUMERIC, and FIXED

A DECIMAL in MySQL is mapped to a DECIMAL in DB2. Although MySQL and DB2 implement decimals differently, externally, they behave the same way.

- ▶ Date and time type:

- DATE

A DATE in MySQL and DB2 DATE both use four bytes (first two bytes for the year, the third byte for the month, and the last byte for the day). The

range of the MySQL date for the year is 1000-9999, whereas DB2 supports a date range from 0001-9999, allowing DB2 to map to this MySQL data type.

- DATETIME

A date and time combination in MySQL is displayed as *YYYY-MM-DD HH:MM:SS*, ranging from year 1000 - 9999. In DB2, *TIMESTAMP* is used for a similar purpose, which is a seven part value (year, month, day, hour, minute, second, and microsecond).

- TIMESTAMP

A time stamp in MySQL is a date/time combination with a range of 1970-01-01 00:00:00 to the year 2037. It is automatically set to the date and time of the most recent operation if you do not give it a valid value. This data type can be mapped to the DB2 *TIMESTAMP*, or an optional choice is the DB2 *TIME* data type.

- TIME

MySQL time represents a clock ranging from -838:59:59 to 838:59:59. It is mapped to DB2 *TIME*, which is a 24-hour clock.

- YEAR

The MySQL year can be in either a two-digit or four-digit format, representing the year from 1901 - 2155 mapped to *SMALLINT* or *CHAR(4)* in DB2.

- ▶ String and character types:

- CHAR

A fixed length string in MySQL is represented with the same name in DB2, mapping to a *CHAR* in DB2.

- VARCHAR

A variable-length string in MySQL has a maximum length of 65,535, and a DB2 variable-length string has a maximum length of 32,672. A MySQL *VARCHAR* can be mapped to a DB2 *VARCHAR* when the MySQL *VARCHAR* variables are shorter than 32,672; otherwise, *VARCHAR* can be mapped to a character large object (*CLOB*).

- BINARY

MySQL *BINARY* stores binary byte strings, which can be mapped to DB2 *CHAR(I)*.

- VARBINARY

MySQL *VARBINARY* stores binary byte strings, which can be mapped to DB2 *VARCHAR(I)*.

- TINYBLOB
MySQL TINYBLOB is a binary large object column with a maximum length of 255, which can be mapped to DB2 BLOB(255).
- TINYTEXT
MySQL TINYTEXT is a character stream of maximum length 255, which can be mapped to DB2 CLOB(255).
- BLOB
MySQL BLOB is a binary data column with a maximum length of 65,535, which can be mapped to DB2 BLOB(65 KB).
- TEXT
TEXT is a TEXT column with a maximum length of 65,535 and is mapped to DB2 CLOB(65 KB).
- MEDIUMBLOB
MySQL MEDIUMBLOB is a blob column with a maximum length of 16,777,215, which can be mapped to DB2 BLOB(16 MB).
- MEDIUMTEXT
MySQL MEDIUMTEXT is a text column with a maximum length of 16,777,215, which can be mapped to DB2 CLOB(16 MB).
- LONGBLOB
MySQL LONGBLOB is an extremely large BLOB column with a maximum length of 4,294,967,295, which can be mapped to DB2 BLOB(2 GB).
- LONGTEXT
MySQL LONGTEXT is a text column with a maximum length of 4,294,967,295, which can be mapped to DB2 CLOB(2 GB).
- ENUM
MySQL has a special ENUM type, which is a string object that can have only one value chosen from a list of values, 'value1', 'value2', ..., NULL, which can be mapped to DB2 VARCHAR() with check constraints.
- SET
MySQL has another special SET type, which is a string object that can have zero or more values, which must be chosen from the list of values, 'value1', 'value2',..., which can be mapped to DB2 VARCHAR() with check constraints.

Table 6-1 shows the default data type mappings between the two databases that are used by the IBM Data Movement Tool. We use this mapping for our sample conversion.

Table 6-1 MySQL to DB2 data type mapping

MySQL 5.1	DB2 9.7
TINYINT	SMALLINT
TINYINT UNSIGNED	SMALLINT
SMALLINT	SMALLINT
SMALLINT UNSIGNED	INTEGER Optional: SMALLINT
BIT	SMALLINT
BOOLEAN	SMALLINT
MEDIUMINT	INTEGER
MEDIUMINT UNSIGNED	INTEGER
INTEGER/INT	INTEGER
INTEGER/INT UNSIGNED	BIGINT Optional: INTEGER
BIGINT	BIGINT
BIGINT UNSIGNED	DECIMAL Optional: BIGINT
FLOAT	DOUBLE
FLOAT UNSIGNED	DOUBLE
DOUBLE	DOUBLE
DOUBLE UNSIGNED	DECIMAL Optional: DOUBLE
REAL	DOUBLE
REAL UNSIGNED	DOUBLE
NUMERIC DECIMAL DEC	DECIMAL(31,0)
NUMERIC(P) NUMERIC(P,0) DECIMAL(P) DECIMAL(P,0) DEC(P) DEC(P,0)	DECIMAL(min(P,31),0)
DECIMAL UNSIGNED	DECIMAL

NUMERIC UNSIGNED	DECIMAL
NUMERIC(p,s) DECIMAL(p,s) DEC(p,s) where: s > 0 && p >= s s > 0 && p < s s < 0	DECIMAL(min(p,32), min(s,32)) DECIMAL(min(s,32), min(s,32)) DECIMAL(min(p,32),0)
DATE	DATE
DATETIME	TIMESTAMP Optional: TIME
TIMESTAMP	TIMESTAMP
TIME	TIME
YEAR	CHAR(4) Optional: SMALLINT
CHAR	CHAR
VARCHAR	VARCHAR
BINARY	CHAR(I) FOR BIT DATA
VARBINARY	VARCHAR(I) FOR BIT DATA
TINYBLOB	BLOB(255) Optional: VARCHAR(255)
TINYTEXT	CLOB(255)
BLOB	BLOB(65535)
TEXT	CLOB(65535)
MEDIUMBLOB	BLOB(16777215)
MEDIUMTEXT	CLOB(16777215)
LONGBLOB	BLOB(2GB)
LONGTEXT	CLOB(2GB)
ENUM	VARCHAR() with check constraints
SET	VARCHAR() with check constraints

6.2 Data definition language differences

In this section, we address the Data Definition Language (DDL) syntax differences between MySQL and DB2 DDL statements. These differences can be syntactical, semantic, or functional.

Both MySQL and DB2 follow the structured query language (SQL), which is a standardized language that is used to access databases and their objects, as defined by the American National Standards Institute (ANSI)/International Organization for Standardization (ISO).

The data definition language is a set of SQL statements. You can use these statements for a variety of tasks, including the creation or deletion of databases and database objects (tables, views, and indexes), definitions of column types, and definitions of referential integrity rules.

6.2.1 Database manipulation

By default, MySQL database objects are stored in a single directory. Although this design is simple, it poses a big performance bottleneck due to slow disk seek, slow search, and non-indexing of data. MySQL depends on the operating system's capabilities for distributing its data across disks. It uses symbolic links to link different disks for different databases, as well as for database and table distribution.

On Linux machines, MySQL can use the file system mounting options. However, in most cases, MySQL uses symbolic links, which can be done by creating a directory where you have extra space:

```
bash> cd <file system with space>
bash> mkdir mysqldata
```

Then, create a symbolic link to the newly created directory using these statements:

```
bash> cd /var/lib/mysql
bash>ln -sf <file system with space>/mysqldata data
```

Then, create a database from the mysql prompt using this statement:

```
mysql>CREATE DATABASE inventory
```

MySQL users can distribute tables using symbolic linking or the data and index directory options of the CREATE TABLE statement.

MySQL stores data in single files, multiple files, or table spaces, depending on the table type being used. Figure 6-3 shows an example of storage engines that fall under one of these three types.

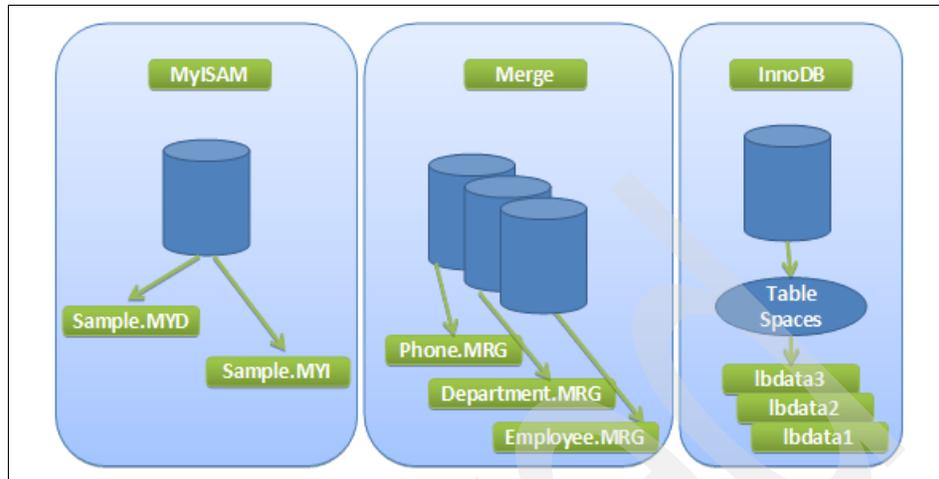


Figure 6-3 MySQL physical storage levels

The tables on the left side of the diagram are managed by the MyISAM storage engine. For MyISAM tables, MySQL creates a .MYD file for data and a .MYI file to store indexes, with only one file for all data and indexes. The tables in the middle of the diagram are managed by the Merge storage engine. With Merge tables, the .MRG file contains the names of the tables to be used and a .FRM file for table definition. In a Merge table, various tables are used, each of these tables having its own data file. However, as a whole, a Merge table uses multiple data files. The tables on the right side of the diagram are managed by the InnoDB storage engine. For InnoDB tables, MySQL stores data in a table space identified by the path parameter, `innodb_data_file_path`. Multiple data files can be used for InnoDB.

MySQL also has a feature called *user-defined partitioning*, which allows for table data to be horizontally split across file systems depending on a specific set of data defined by the user. For each partition that is created, there is a corresponding .MYD file for data and .MYI file for the index.

In contrast to MySQL, DB2 stores everything in table spaces. *Table spaces* are logical representations of physical containers on the file system. DB2 uses a better approach for the logical and physical distribution of the database and the database elements in different sectors, as shown in Figure 6-4 on page 125. After completing the conversion of your database from MySQL to DB2, you can use these features to enhance the performance of your application.

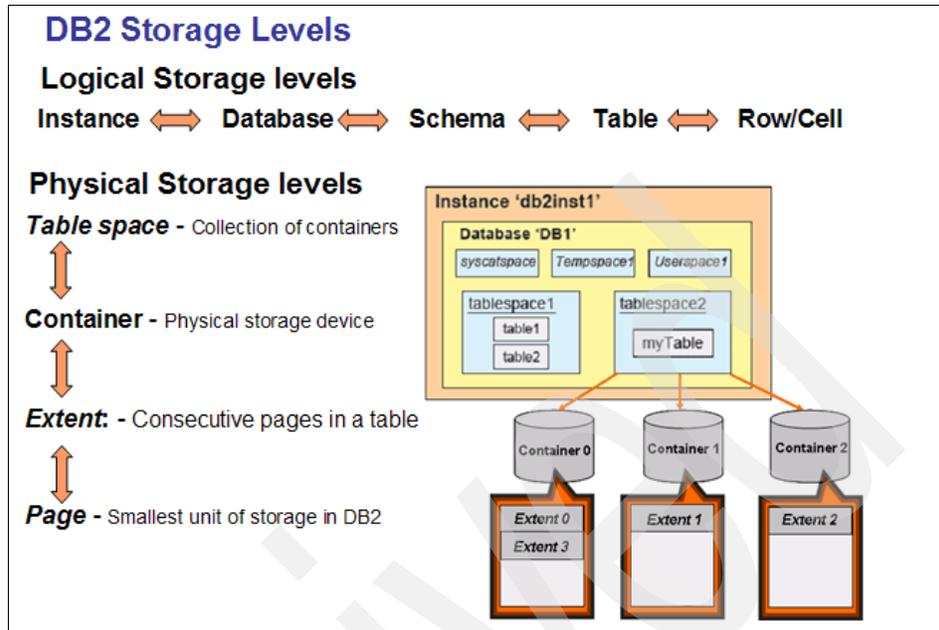


Figure 6-4 DB2 storage levels

Instance

A DB2 server can have more than one instance. One instance can have multiple databases. One instance per application database has the advantage that the application support and the database support do not have to coordinate with one another to take the database or the instance offline. For conversion purposes, a single instance can be created for your database application environment using the `db2icrt` command:

```
bash> db2icrt -u db2fenc1 db2inst1
```

Database

A *database* represents your data as a collection of data in a structured fashion. It includes a set of system catalog tables that describes the logical and physical structure of the data, a configuration file containing the environment parameter values that are used by the database, and a recovery log with ongoing transactions and transactions that can be archived.

Database partition

A *database partition* is part of a database, containing its own data, indexes, configuration files, and transaction logs. A database partition is sometimes called a *node* or a *database node*. A *partitioned database environment* is a database installation that supports the distribution of data across database partitions. This

can be used if you want to spread your DB2 database across multiple servers in a cluster or along multiple nodes. There are no database partition group design considerations when using a non-partitioned database. The database partition group can be created within a database by using the following command:

```
db2> CREATE DATABASE PARTITION GROUP MaxGroup ON ALL DBPARTITIONNUMS
```

Creating a database

The database in DB2 can be created simply by issuing the following command:

```
db2>CREATE DATABASE invent
```

This command generates a new database with a default path, and it generates table spaces. It creates three initial table spaces and the system tables, and it creates the recovery log.

You can use the CREATE DATABASE statement with options to personalize the database and take advantage of DB2 advanced features, such as automatic storage, which simplifies the storage management for table spaces, as shown in Example 6-1. When using automatic storage, it is possible to specify a group of storage devices for DB2 to use for your database. This specification allows DB2 to allocate and grow this specified space as table spaces are created and populated. Automatic storage is turned on by default when creating a database.

Example 6-1 The CREATE DATABASE statement

```
db2> CREATE DATABASE invent AUTOMATIC STORAGE YES on '/db2fs/invent'
```

Dropping a database

In MySQL, you can drop the database using this statement:

```
mysql> DROP DATABASE [if exists] inventory
```

This statement removes all of the database files (.BAK, .DAT, .HSH, .ISD, .ISM, .ISM, .MRG, .MYD, .MYI, .db, and .frm) from your file system.

DB2 has a similar command:

```
db2> DROP DATABASE invent [at DBPARTITIONNUM]
```

This command deletes the database contents and all log files for the database, uncatalogs the database, and deletes the database subdirectory.

Alter database

The MySQL alter database command allows you to change the overall characteristics of a database. For example, the character set clause changes the

database character set, and the collation clause changes the database collation. Use this basic syntax for altering the database:

```
mysql>ALTER DATABASE inventory CHARACTER SET charset_name COLLATE
collation_name
```

In DB2, you can use the UPDATE DATABASE CONFIGURATION and UPDATE DATABASE MANAGER CONFIGURATION commands to set the database and database manager configuration parameters. These commands allow modification of various configuration parameters, such as log file size, log file path, heap size, cache size, and many others. You can take advantage of the DB2 autonomic features by enabling the automatic maintenance and Self-Tuning Memory Manager features. *Automatic maintenance* allows for scheduling database backups, keeping statistics current, and reorganizing tables and indexes. The *Self-Tuning Memory Manager* provides constant tuning of your database without the need for DBA intervention. The following examples show how these commands can be set in DB2:

```
db2> UPDATE DATABASE MANAGER CONFIGURATION using diaglevel 3
db2> UPDATE DATABASE ONFIGURATION for invent
using auto_maint on
    auto_tbl_maint on
    auto_runstats on
    auto_reorg on
    self_tuning_mem on
```

This example does not show all of the parameters that are available in DB2. For more information about how to set up automatic maintenance and Self-Tuning Memory Manager, visit this Web site:

<http://publib.boulder.ibm.com/infocenter/db21uw/v9r7/index.jsp>

In addition, you can use these commands to change the physical and logical partitioning of a database and to allocate table spaces and paging configurations.

Table space

A *table space* is a storage structure containing tables, indexes, large objects, and long data. Table spaces reside in database partition groups and allow the assignment of database location and table data directly onto containers. DB2 allows for two types of table spaces: *System Managed Space (SMS)*, where the operating system allocates and manages the space where the tables are stored, and *Data Managed Space (DMS)*, where the database administrator has the ability to decide which devices or files to use and allows DB2 to manage this space. Another option is to enable automatic storage for the table spaces. No container definitions are needed in the latter case, because the DB2 database manager assigns and manages the container automatically.

Any DB2 database must have at least the following three table spaces:

- ▶ One catalog table space, which contains system catalog tables
- ▶ One or more user table spaces, which contain user-defined tables
- ▶ One or more temporary table spaces, which contain temporary tables

You can create more table spaces by using the following commands:

```
db2> CREATE REGULAR TABLESPACE tblsp1 PAGESIZE 8192 MANAGED BY SYSTEM using
('/home/db2inst1/database/user8K') extentsize 8 prefetchsize 8 bufferpool
bp8k
db2> CREATE TABLESPACE tblsp2 MANAGED BY DATABASE using (device
'/dev/rhdisk6' 10000, device '/dev/rhdisk7' 10000, device '/dev/rhdisk8'
10000) overhead 12.67 transferrate 0.18
db2> CREATE TABLESPACE tblsp3 MANAGED BY AUTOMATIC STORAGE
```

Schema

A *schema* is an identifier, such as a user ID, that helps group tables and other database objects. A schema can be owned by an individual, and the owner can control access to the data and the objects within it. A schema is also an object in the database. It can be created automatically when the first object in a schema is created. We can create a schema:

```
db2>CREATE SCHEMA inventschema AUTHORIZATION inventUser
```

6.2.2 Table manipulation

Tables are logical structures that are made up of columns and rows, which are maintained by the database manager.

MySQL tables

As shown in Figure 6-5 on page 129, MySQL supports two types of tables: transaction-safe tables and non-transaction-safe tables. Transaction-safe tables (managed by InnoDB or NDB storage engines) are crash safe and can take part in transactions providing concurrency features that allow commit and rollback. Alternatively, non-transaction-safe tables (managed by MyISAM, MEMORY, MERGE, ARCHIVE, or CSV storage engines) are less safe but are much faster and consume less space and memory.

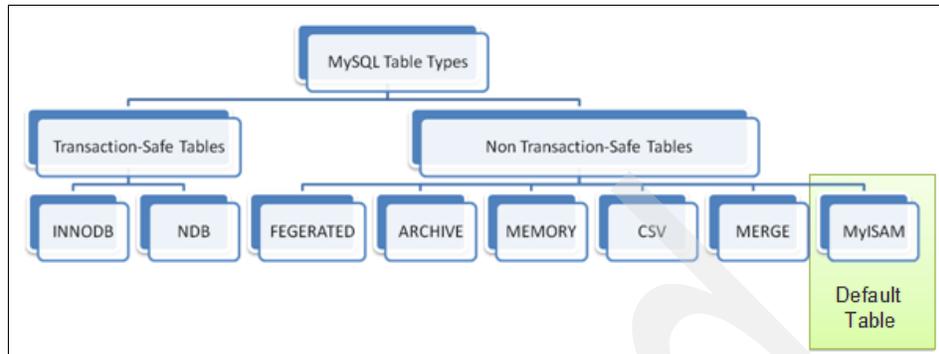


Figure 6-5 MySQL table types

The following tables are the basic storage elements in MySQL:

► InnoDB table

InnoDB is a transaction-safe storage engine with commit, rollback, and crash recovery capabilities. It provides locking on the row level and consistent non-locking read-in select statements.

InnoDB is a complete database service placed under MySQL, having its own buffer pool for caching data and indexes in main memory and supporting multiple table spaces. An InnoDB table can be created using the following command:

```
mysql>CREATE TABLE tblinnodb (i int ,f float ) type=innodb;
```

► NDB

NDB is a transaction-safe storage engine. This storage engine allows for clustering of a database in a shared-nothing architecture. This type of architecture allows tables to be split across multiple servers, with each server having its own set of data. In order for the table to be a part of the cluster, the table must use the NDB storage engine.

► MyISAM table

MyISAM is the default table type in MySQL. It is based on the ISAM code and supports concurrent insert, big files on the operating system and file system that support big files, better indexing, index compression, and table compression.

MyISAM is a default MySQL table, so it can be created either by specifying `type=myisam` or by not specifying any type:

```
mysql>CREATE TABLE tblmyisam (i int ,f float );
```

▶ **MERGE**

The *MERGE* storage engine is a group of MyISAM tables in one table, grouped together across the same disk or multiple disks. The tables must have exact column and index structures.

▶ **MEMORY (HEAP)**

MEMORY was previously called the HEAP storage engine. Tables that are created in the MEMORY storage engine are stored in memory. The table definition is stored on disk, but the rows are stored in memory. Therefore, the table still exists after a reboot; however, all rows within this table are lost.

▶ **ARCHIVE**

ARCHIVE stores the data in compressed format on disk and is typically used to store large amounts of data. There are no indexes in the ARCHIVE storage engine.

▶ **CSV**

The *CSV* storage engine, which stands for comma separated value, allows data to be stored in text files. This data can then be accessed via SQL calls.

▶ **FEDERATED**

The *FEDERATED* storage engine works with non-transactional tables and allows tables to be accessed on a remote server as though they were stored locally. The table definition is stored locally; however, no data is actually stored on the local server.

DB2 conversion of MySQL table

In DB2, all the tables take part in the transaction. Therefore, tables that are managed by MySQL InnoDB, MyISAM, ARCHIVE, and CSV storage engines can all be converted to a DB2 regular table. A *DB2 regular table* is a general purpose table, which is created with the CREATE TABLE statement and is used to hold persistent user data.

Create table syntax

In this section, we give a high-level overview of the difference in the CREATE TABLE syntax of MySQL and DB2. MySQL CREATE TABLE statements are quite simple with few exceptions. Example 6-2 shows the creation of MySQL InnoDB. Example 6-3 on page 131 shows the DB2 conversion. You can notice that no major change is required.

Example 6-2 Creating MySQL InnoDB managed table

```
mysql>CREATE TABLE tblinno(col1 int, col2 char(10)) engine=InnoDB ;
```

Example 6-3 DB2 conversion of creating MySQL InnoDB managed table

```
db2>CREATE TABLE tblinno(col1 int, col2 char(10))
```

Example 6-4 shows how to create a table using the MyISAM storage engine. Example 6-5 is the DB2 conversion. Note the changes:

- ▶ Changes in the data type according to data type mapping
- ▶ Instead of auto_increment, generated by default because identity is used

Example 6-4 Creating MySQL MyISAM table

```
mysql>CREATE TABLE tblmysiam (  
    wk_id int(11) unsigned NOT NULL auto_increment,  
    user_id int(11) unsigned default NULL,  
    cnt int(10) unsigned default NULL,  
    cat_id int(12) unsigned default NULL,  
    status varchar(10) default NULL,  
    PRIMARY KEY (wk_id)) type=MyISAM;
```

Example 6-5 Conversion of MySQL MyISAM table creation

```
db2>CREATE TABLE tblmysiam (  
    wk_id INT NOT NULL GENERATED BY DEFAULT AS IDENTITY,  
    user_id INT default NULL,  
    cnt INT default NULL,  
    cat_id INT default NULL,  
    status VARCHAR(10) default NULL,  
    PRIMARY KEY (wk_id));
```

Example 6-6 is a MySQL table creation example using the ARCHIVE engine. Example 6-7 is the DB2 conversion using row compression.

Example 6-6 Creating MySQL ARCHIVE table

```
mysql>CREATE TABLE tblarchive(col1 int, col2 char(10)) type=ARCHIVE;
```

Example 6-7 DB2 conversion of creating MySQL ARCHIVE table

```
db2>CREATE TABLE tblarchive(col1 int, col2 char(10)) COMPRESS YES
```

Example 6-8 on page 132 shows how to create a MySQL table with partitioning. Example 6-9 on page 132 is the DB2 conversion; again, no major change is required.

Example 6-8 Creating MySQL table using partitioning with the default MyISAM storage engine

```
mysql>CREATE TABLE partsales (id INT, item VARCHAR (20) )
      PARTITION BY RANGE (id)(
      PARTITION p1 VALUES LESS THAN (10),
      PARTITION p2 VALUES LESS THAN (20)
      );
```

Example 6-9 DB2 conversion of MySQL table creation with partitioning

```
db2>CREATE TABLE partsales (id INT, item VARCHAR (20)
      PARTITION BY RANGE (id)(
      PARTITION p1 STARTING FROM (MINVALUE) ENDING (10),
      PARTITION p2 ENDING (20)
      );
```

Alter table

ALTER TABLE is a statement that is used to modify one or more properties of a table. The syntax of the ALTER TABLE statement for MySQL and DB2 is quite similar and is shown in Example 6-10.

Example 6-10 MySQL and DB2 ALTER TABLE example

```
mysql>ALTER TABLE partsales modify status varchar(20);
```

```
db2>ALTER TABLE partsales alter column status set data type varchar(20)
```

ALTER TABLE in DB2 now supports dropping columns. DB2 drops columns using a temporary table. The syntax of ALTER TABLE for MySQL and DB2 is similar, as shown in Example 6-11.

Example 6-11 MySQL and DB2 dropping column example

```
mysql> ALTER TABLE sales drop column c1, drop column c2;
```

```
db2> ALTER TABLE sales drop c1 drop c2
```

Drop table

Tables can easily be deleted from the database by issuing the DROP TABLE statement as shown.

For MySQL:

```
DROP [TEMPORARY] TABLE [IF EXISTS] tbl_name [, tbl_name,...] [RESTRICT
|CASCADE]
```

For DB2:

```
DROP table tblname
```

Non-persistent storage tables

The following descriptions explain the differences between MySQL and DB2 temporary table types:

► **MERGE table**

A *merge table* groups multiple MyISAM tables into one table. Dropping the merge table leads to dropping the merge specifications, but tables that made up the merge table continue to exist.

Tables can be merged only if they have identical columns and key information. Example 6-12 shows the statements for merge table in MySQL.

Example 6-12 Usage of merge table in MySQL

```
mysql> CREATE TABLE table1
      (col1 int not null auto_increment primary key, col2 char(20));
mysql> CREATE TABLE table2
      (col1 int not null auto_increment primary key,col2 char(20));
mysql> CREATE TABLE tblmerge
      (col1 int not null auto_increment, col2 char(20), key(col1));
      TYPE=MERGE UNION=(table1,table2) INSERT_METHOD=LAST;
mysql> INSERT INTO table1 (col2) VALUES ("value1"),("value2");
mysql> INSERT INTOtable2 (col2) VALUES ("value3"),("value4");
```

DB2 uses the update-able UNION ALL view to achieve the merge table feature. UNION ALL views are commonly used for logically combining different but semantically related tables. The UNION ALL view is also used for unification of like tables for better performance, manageability, and integrating federated data sources.

The following example shows using the UNION ALL command for views:

```
db2>CREAT VIEW UNIONVIEW as SELECT * FROM table1 UNION ALL SELECT * FROM
table2
```

► **MEMORY table**

A *MEMORY table* is a hashed index that is always stored in memory. Memory tables are fast but are not crash safe. When MySQL crashes or has a scheduled reboot, the MEMORY table will still exist on the reboot but the data is lost.

A MySQL Memory table can be created using the following command:

```
mysql> CREATE TABLE memtable type=MEMORY SELECT * FROM table1;
```

MySQL MEMORY tables can be converted to DB2 as temporary tables, materialized query tables, or indexes depending upon your requirements.

– Temporary table

DB2 temporary tables are tables that are used for storing data in non-persistent, in-memory, session-specific tables. When a session is over, the table definition for the table is lost. When your application uses the MEMORY table in this fashion, temporary tables can be declared in your application by calling these statements:

- Create user temporary table space if it does not exist by using this statement:

```
db2>CREATE USER TEMPORARY TABLESPACE discompose MANAGED BY SYSTEM  
using ('usertemp1')
```

- Declare the temporary table in the application:

```
db2>DECLARE GLOBAL TEMPORARY TABLE distemper LIKE table1 ON COMMIT  
db2>DELETE ROWS NOT LOGGED IN discompose
```

– Materialized query table

Materialized query tables (MQT), which are also known as summary tables, can also be used to improve the query performance. An MQT is a table whose definition is based on the results of a query and whose data is in the form of precomputed results. If the SQL compiler determines that a query will run more efficiently against an MQT than a base table or tables, the query executes against the MQT:

```
db2>CREATE TABLE sales AS (SELECT * FROM table1) DATA INITIALLY  
DEFERRED REFRESH DEFERRED
```

We discuss MQT in more detail in 11.4, “Materialized query tables” on page 394.

In addition, DB2 also supports tables for clustering and query performance enhancement. These tables can also be used according to various requirements.

▶ Multidimensional clustering (MDC) tables

Multidimensional clustering (MDC) tables have a physical cluster on more than one key or dimension at the same time. An MDC table maintains clustering over all dimensions automatically and continuously, thus eliminating the need to reorganize the table in order to restore the physical order of the data.

When creating MDC tables, the performance of many queries might improve, because the optimizer can apply additional optimization strategies. The advantages of MDC tables include quicker and less frequent scanning, because of dimension block, faster lookups, the use of block-level index “AND” and “OR”, and faster retrieval.

Issue the following command to create the MDC table:

```
db2>CREATE TABLE tblmdc
      (col1 int, col2 int, col3 int, col4 char(10))
      ORGANIZE BY DIMENSIONS(col1,col2,col3)
```

We discuss MQT in more detail in 11.3.3, “Multidimensional clustering” on page 392.

► Range-clustered tables (RCT)

Range-clustered tables (RCT) are implemented as sequential clusters of data that provide fast, direct access. RCT is a table layout scheme where each record in the table has a predetermined offset from the logical start of the table. The advantages associated with RCT are direct and include quicker access times, less maintenance, less logging, less locking, smaller buffer pool sizes, and less indexing. Create RCT by using the following command:

```
db2> CREATE TABLE tblrct
      (col1 int not null, col2 int not null, col3 char(10),
      col4 float)
      ORGANIZE BY KEY SEQUENCE
      (col1 starting from 1 ending at 10000)
      allow overflow
```

► Typed table/hierarchy table

Typed tables are tables that are defined with a user-defined structured type. With typed tables, you can establish a hierarchical structure with a defined relationship between tables called a *table hierarchy*. The table hierarchy is made up of a single root table, supertables, and subtables. Example 6-13 shows the creation of a typed table.

Example 6-13 Usage of typed/hierarchy table

```
//Here is the SQL to create the BusinessUnit typed table
db2>CREATE TABLE BusinessUnit OF BusinessUnit_t (REF IS Oid USER GENERATED)
//Here is the SQL to create the tables in the Person table hierarchy
db2>CREATE TABLE Person OF Person_t (REF IS Oid USER GENERATED)
db2>CREATE TABLE Employee OF Employee_t UNDER Person INHERIT SELECT PRIVILEGES
(SerialNum WITH OPTIONS NOT NULL, Dept WITH OPTIONS SCOPE BusinessUnit )
db2> CREATE TABLE Student OF Student_t UNDER Person INHERIT SELECT PRIVILEGES
db2> CREATE TABLE Manager OF Manager_t UNDER Employee INHERIT SELECT PRIVILEGES
db2> CREATE TABLE Architect OF Architect_t UNDER Employee INHERIT SELECT
PRIVILEGES
```

► Views

Views are the named specification of a result table. This specification is a select statement that is run whenever the view is referenced in an SQL statement. It can be used just like a base table.

You can create a simple view by issuing a create statement, as shown in Example 6-14.

Example 6-14 Create view example

```
db2>CREATE TABLE table1(col1 INTINT, col2 INT, col3 CHAR(20),col4 FLOAT,
col5 CHAR(30))
db2>CREATE TABLE itable2(col6 INT, col7 INT, col8 CHAR(20),col9 FLOAT,
col10 CHAR(30))
db2>CREATE VIEW myview(col1,sum1,col4,col10) AS
    SELECT col1,col1+col6,col4,col10
    FROM table1, table2
    WHERE col1<10 AND col6>10
```

6.2.3 Index manipulation

An *index* is an object in the database system, which uses techniques for faster retrieval of the data from tables. It is an ordered set of pointers to rows of a base table that is managed by the database manager directly.

MySQL supports both single column indexes and multi-column indexes. MySQL has five types of indexes:

- Primary key
- Unique
- Non-unique
- Fulltext
- Spatial

DB2 supports all of the index types that are supported by MySQL with the same terminology, allowing them to map directly during conversion.

Create index

The following example shows the MySQL CREATE INDEX syntax:

```
CREATE [ONLINE|OFFLINE] [UNIQUE|FULLTEXT|SPATIAL] INDEX index_name
[index_type]
ON tbl_name (index_col_name,...)
index_col_name:
col_name [(length)] [ASC | DESC]
```

Creating an index in DB2 is quite similar:

```
CREATE [unique] INDEX index-name on tablename (columnnames ASC|DESC)
SPECIFICATION ONLY INCLUDE(column-name)
CLUSTER/EXTEND USING index-extension-name (constant-expression)
PCTFREE 10/PCTFREE integer LEVEL PCTFREE integer MINPCTUSED integer
ALLOW/DISALLOW REVERSE SCANS
PAGE SPLIT SYMMETRIC/PAGE SPLIT HIGH/LOW
COLLECT STATISTICS DETAILED SAMPLED
```

Drop index

This example shows the DROP INDEX statement for MySQL and DB2:

```
mysql> DROP INDEX index_name ON tbl_name;
db2> DROP INDEX index_name
```

6.2.4 Trigger manipulation

Triggers are a predefined set of actions to be executed automatically in response to a set of events. Triggers are commonly defined to respond to the following events: INSERT, UPDATE, or DELETE.

There are two types of triggers: statement triggers and row triggers. Statement triggers are executed in response to a single INSERT, UPDATE, or DELETE statement. Row triggers are executed for each row that is affected by the INSERT, UPDATE, or DELETE statement. There are many benefits to having triggers. They allow you to log information about changes to a table, can be used to validate an insert, can restrict access to specific data, or can make data modifications and comparisons as a change occurs.

DB2 supports both row-based triggers and statement-based triggers; MySQL supports only row-based triggers. Both MySQL and DB2 support INSERT, UPDATE, and DELETE-triggered events and can be defined to fire using BEFORE and AFTER. DB2 has an additional firing classification called INSTEAD OF, which allows the triggers to perform INSERT, UPDATE, or DELETE on views. MySQL does not support a trigger on a view.

The following example shows the CREATE TRIGGER syntax for MySQL:

```
CREATE [DEFINER = { user | CURRENT_USER}]
TRIGGER trigger_name trigger_time trigger_event
ON tbl_name FOR EACH ROW trigger_stmt
```

Creating a trigger in DB2 has similar syntax:

```
CREATE TRIGGER trigger-name NO CASCADE BEFORE/AFTER/INSTEAD OF
    INSERT/DELETE/UPDATE OF column-name ON table-name/view-name
    REFERENCE OLD AS correlation-name/NEW AS correlation-name/OLD TABLE
    AS identifier/NEW TABLE AS identifier
    FOR EACH ROW/FOR EACH STATEMENT triggered-action
```

6.2.5 Procedures and function manipulation

Stored procedures and functions are routines that are stored and run on the database server. Stored procedures can be developed to compute or manipulate results from a query on the database server prior to sending data back to the application. This capability can result in increased performance, because unnecessary data is not sent across the network to the application. This capability also decreases the dependency between the application and the database, because the application no longer needs to manipulate the results. Both DB2 and MySQL follow the standard syntax that is specified by SQL:2003 for stored procedures and functions, therefore, creating a procedure or function in DB2 is similar to MySQL.

6.3 Other considerations

Up until now, we have discussed approaches for converting database elements, which use similar approaches in the two products. In subsequent sections, we discuss the conversion of database objects that do not map directly from MySQL to DB2. We also discuss server and database placement architecture.

Multiple servers

In certain cases, multiple MySQL servers are placed on the same machine. Possible reasons include user management, testing, or potentially differentiating applications. MySQL provides an option to run multiple servers on same machine using several operating parameters.

There are several ways to configure a new server; we used this method:

```
bash> /path/to/mysqld --socket=file_name --port=port_number
```

Or, you can use this method:

```
bash> MYSQL_UNIX_PORT=/tmp/mysqld-new.sock
bash> MYSQL_TCP_PORT=3307
bash> export MYSQL_UNIX_PORT MYSQL_TCP_PORT
ash> mysql
```

DB2 supports similar functionality using multiple database manager instances on the same machine. Each database manager instance has its own configuration files, directories, and database. Figure 6-6 shows the typical scenario.

DB2 also supports the creation of separate instances on the same machine with a different DB2 version.

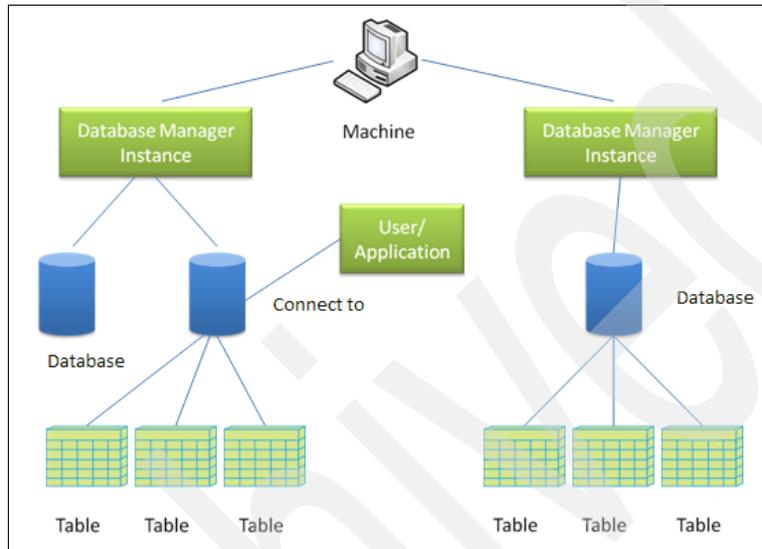


Figure 6-6 Typical DB2 setup

Multiple databases and schema conversion

Within MySQL, tables in a database cannot be logically grouped by a schema or qualified by a name based on the application. To fulfill this requirement, MySQL tables are placed in separate databases.

MySQL supports the access of tables in multiple databases using the same connection. So, an application connected to MySQL can use two tables in separate databases in a single statement. The queries in Example 6-15 show how MySQL can use two tables in separate databases in a single statement.

Example 6-15 Multiple database example

```
mysql>CREATE DATABASE mydb1;
mysql>CREATE DATABASE mydb2;
mysql>CONNECT mydb1;
mysql>CREATE TABLE table1(col1 INT, col2 CHAR(10));
mysql>INSERT INTO table1 VALUES(1,"new");
mysql>CONNECT mydb2;
mysql>CREATE TABLE table2(col1 INT, col2 CHAR(10));
mysql>INSERT INTO table2 VALUES(1,"val 1");
```

```
mysql>SELECT * FROM table2, mydb1.table1;
```

This works in MySQL and gives the following result

```
+-----+-----+-----+-----+
| col1 | col2 | col1 | col2 |
+-----+-----+-----+-----+
|    1 | va1 1 |    1 | new  |
|    1 | va1 1 |    2 | new  |
+-----+-----+-----+-----+
```

Figure 6-7 shows the architecture of multiple MySQL databases accessed by a single application using the same connection.

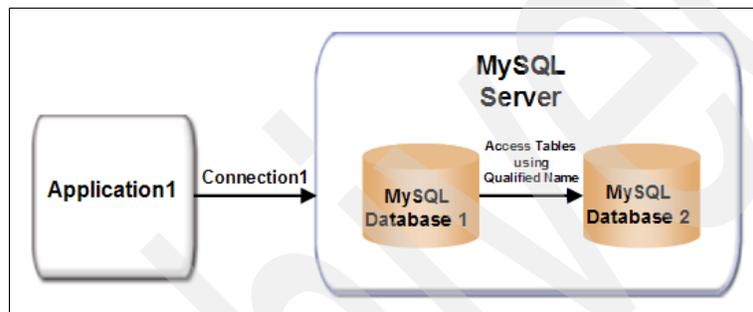


Figure 6-7 MySQL application using multiple DBs instead of multiple schema

DB2 supports access to multiple databases using database links in a federated system. For a non-federated system, DB2 applications use a more logical technique by using multiple schemas to replace the use of multiple databases within the same application. Each database can have multiple schemas and each table belongs to a particular schema.

When converting the MySQL applications using multiple databases, all databases used in the applications can be placed under a single DB2 database.

Each MySQL database becomes represented by a DB2 schema to hold the tables under that database. All tables can be accessed by the application using a single connection. The DBA only needs to manage one database.

You can create DB2 schema by using the following command:

```
db2>CREATE SCHEMA schema1 AUTHORIZATION schema1
```

The tables created in the particular schema can be accessed using the full table qualifier table schema.table name.

Table placement

MySQL does not support table spaces for managing physical location and page size or distributing tables onto the different table spaces, except with the optional InnoDB storage engine, which supports multiple table spaces by distributing them into separate files.

DB2 supports table spaces to establish the relationship between the physical storage devices that are used by your database system and the logical containers that are used to store data. Table spaces reside in database partition groups. They allow you to assign the location of table data directly onto containers.

Prior to the conversion of your database structure, you must create proper table spaces of various sizes in DB2. Individual or multiple tables can then be assigned to the tables spaces. If the table space design is done effectively, it can greatly increase performance. We discuss this topic in more detail in Chapter 10, “Testing and tuning” on page 321.

List information

MySQL provides a show command to list the information about databases, tables, columns, or status information about the server.

DB2 provide commands for getting information about instances, databases, table spaces, and other objects. The DB2 system catalogs contain all of the necessary information about tables, columns, indexes, and other objects. You can use the **describe** and **List** commands to display database and table structure or use the select statement to get the details of the table definition.

Table 6-2 shows examples of MySQL and corresponding DB2 statements or commands to list database or table-related information.

Table 6-2 MySQL to DB2 conversion of list information statement

MySQL	DB2
Show databases	List database directory
Show tables from <dbname>	List tables
Show columns from <tablename>	Describe table <tablename>
Show index from <tablename>	SELECT indname FROM syscat.indexes WHERE tabname=tablename

Referential integrity refers to the constraints that are defined on the table and its columns, which help you to control the relationship of data in various tables. Essentially, referential integrity involves primary keys, foreign keys, and unique keys.

Primary keys and unique keys are treated similarly in MySQL and DB2. However, MySQL currently only parses foreign key syntax in the CREATE TABLE statements. MySQL does not use or store the information about foreign keys, except in InnoDB tables, which support checking foreign key constraints, including CASCADE, ON DELETE, and ON UPDATE.

DB2 provides full support for foreign keys. With the full referential integrity functionality from DB2, your application can be released from the job of taking care of the data integrity issues. Example 6-16 shows the creation and usage of foreign key constraints in DB2.

Example 6-16 Foreign key constraint usage

```
db2> ALTER TABLE table1 ADD CONSTRAINT foreign1 FOREIGN KEY (id)
REFERENCES table2 ON DELETE SET NULL
```

We discuss foreign key creation in more detail in 6.5.2, “Manual database object conversion and enhancements” on page 158.

6.4 Converting the database

After understanding the MySQL database structure/schema relative to various types of DB2 objects, such as the database, tables, views, indexes, and referential integrity, this section provides details of the database structure/schema conversion from an existing MySQL database to DB2.

You can convert database schema in various ways; however, there are three most common approaches:

- ▶ Automatic conversion using porting tools
- ▶ Manual porting
- ▶ Metadata transport

In general, all of these approaches use existing MySQL databases as input and pass them through the following functional engine:

- ▶ Capture database schema information from MySQL
- ▶ Modify schema information for DB2
- ▶ Create the database in DB2 with structure

6.4.1 Automatic conversion using porting tools

Using porting tools is the easiest and most common approach. The IBM Data Movement Tool can simplify your move to DB2. There are a number of third-party upgrade tools available on the market, which can be used, but IBM does not guarantee the correct functionality of these tools.

IBM Data Movement Tool

The IBM Data Movement Tool is a simple conversion tool that uses the MySQL database to be converted to create the database objects and data records for the new DB2 database. With the IBM Data Movement Tool, you can automatically convert data types, tables, columns, and indexes into equivalent DB2 database elements. The IBM Data Movement Tool provides database administrators (DBAs) and application programmers the functionality needed to automate the conversion task. The strength of this tool is shown in large-scale data movement projects. This tool has been used to move up to 4 TB of data in just three days with good planning and procedures.

You can reduce the downtime, eliminate human error, and cut back on person hours and other resources associated with the database conversion by using the following features found in the IBM Data Movement Tool:

- ▶ Extract DDL statements from the MySQL source database.
- ▶ Extract data from the MySQL source database.
- ▶ Generate and run DB2 DDL conversion scripts.
- ▶ Automate the conversion of database object definitions.
- ▶ View and refine conversion scripts.
- ▶ Efficiently implement converted objects using the deployment options (interactive deployment or automated deployment).
- ▶ Generate and run data movement scripts.
- ▶ Track the status of object conversions and data movement, including error messages, error location, and DDL change reports, using the detailed conversion log file and report.

We describe the instructions to download, install, and use the IBM Data Movement Tool in 5.3, “IBM Data Movement Tool installation and usage” on page 112.

For the rest of this chapter, we discuss how to convert the database structure to DB2. The next chapter discusses how to convert the data to DB2.

The following steps briefly describe the database structure conversion process when using the IBM Data Movement Tool:

1. Specify source and DB2 database server connection information.
2. Test the connection to the source and target database. Click **Connect to MySQL** to test the connection and **Connect to DB2** to test the DB2 connection.
3. Specify the working directory to where the DDL and the data will be extracted.
4. With the IBM Data Movement Tool, you have the option to extract only the database objects, only the data, or both. Choose if you want DDL and DATA.
5. Click **Extract Data** to extract the DDL and DATA and automatically convert to DB2 syntax. You can monitor the progress in the console window.
6. After the data extraction completes successfully, review the result output files for the status of the data movement, any warnings, errors, and other potential issues. Optionally, you can click **View Script/Output** to check the generated scripts, DDL, data, or the output log file.
7. Click **Deploy Data** to automatically create tables and indexes in DB2 and to load data that was extracted from the source database. Optionally, you can click **Interactive Deploy** to deploy database objects one at a time.

In this chapter, we only discuss the database conversion portion of the conversion process.

Other upgrade tools

There are other upgrade tools available to port MySQL database to DB2, such as SQLWays, which is a database conversion tool from Ispirer Systems Ltd.

6.4.2 Manual conversion

Sometimes, you must use a manual process to convert your database instead of a standard tool. For those few cases, we describe the manual conversion processes, with a focus on converting MySQL objects and features that are not automatically converted by the IBM Data Movement Tool.

In 6.1, “Data type mapping” on page 116 and 6.2, “Data definition language differences” on page 122, we demonstrated syntax and semantic differences between MySQL and DB2 that might require manual conversion. We have also discussed the creation, deletion, and alteration of various database objects, such as the database, tables, index, and views, and how these objects are related when converting from MySQL and DB2.

There are several steps involved in the manual process:

► Capture the database schema information from MySQL

MySQL offers a utility called `mysqldump`, which extracts the database structure and deposits this information into a text file. The structure is represented in DDL and can be used to recreate these database elements for your DB2 server. This example shows the syntax for `mysqldump`:

```
bash> mysqldump DatabaseName > mysqlobjects.ddl
```

When using this tool on an extremely large database, we recommend that you use the `--quick` or `--opt` option. Without these options, `mysqldump` loads the whole result set into memory before dumping the results.

For further information about this utility, refer to the *MySQL Reference Manual* at this Web site:

<http://dev.mysql.com/doc/refman/5.1/en/>

► Modify schema information for DB2

Now that you have captured the source of your MySQL database structure using the `mysqldump` utility, it is time to modify the schema information and make it work for DB2. You must make these manual changes:

– DDL changes

The first step in schema modification is the conversion of the create statement for various database objects, such as the database, tables, views, indexes, and other objects. Refer to 6.2, “Data definition language differences” on page 122 or to these conversions as reference. Also, you must write a DDL script for creating the database and table spaces.

– Data type changes

Check the data types that are used in the table definition. Change the MySQL data types to DB2 data types. Refer to 6.1, “Data type mapping” on page 116.

– Reserved words conversion

There are many reserved words in MySQL and DB2, which cannot be a valid name for the column and database element. Refer to the *MySQL Reference Manual* (<http://dev.mysql.com/doc/refman/5.1/en/>) and DB2.

For more detail about reserved words and to change the conflicting names in the DDL statements, visit the DB2 Information Center at this Web site:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp>

- Create database and database objects

Now that you have modified the DDL statements, you need to create the DB2 database and database objects. You can create them from the command line processor (CLP).

It is a common practice to place the database and table space creation statements in one file. You must place the database objects, such as tables, views, and other objects, that you create in a separate file. Note that you must create the database and table spaces prior to creating the database objects. You must base the database creation scripts on the logical and physical database design. In our example, we created a database using the `create-database.sql` script, as shown in Example 6-17.

Example 6-17 Create-database.sql script

```
-- Create the Initial Database
create database invent
    automatic storage yes ON '/home/db2inst1/invent;
update db cfg for invent using AUTO_MAINT ON;

connect to invent;

-- Create a Bufferpool with 8K and 16K Pages
create bufferpool bp8k size 2000 pagesize 8192;
create bufferpool bp16k size 1000 pagesize 16384;

-- Make the Bufferpool Change Take Effect
disconnect invent;
connect to invent;

-- Create a Tablespace with 8K and 16K Pages
create regular tablespace tblsp8k pagesize 8192 managed by automatic storage
extentsize 8
prefetchsize automatic
bufferpool bp8k;

reate regular tablespace tblsp16k pagesize 16384 managed by automatic storage
extentsize 8
prefetchsize automatic
bufferpool bp16k;

-- Create a System Temporary Tablespace with 8K and 16K Pages
create system temporary tablespace temp8k pagesize 8 K managed by automatic
storage
extentsize 32
prefetchsize automatic
bufferpool bp8k;
```

```
create system temporary tablespace temp16k pagesize 16 K managed by automatic
storage
extentsize 32
prefetchsize automatic
bufferpool bp16k;

disconnect invent;
```

To create the database, invoke this SQL script from the DB2 command-line window or bash shell by using this command:

```
db2>@create-database.sql
bash>db2 -f create-database.sql
```

We then use the database object creation statements from the output of mysqldump and the mysqlobjects.ddl file to create the db2objects.ddl script. Change the DDL statements and data types based on the discussion in 6.1, “Data type mapping” on page 116 and 6.2, “Data definition language differences” on page 122. You can create any additional statements that are required.

Now, we execute the newly created DDL scripts from the bash shell, as shown in Example 6-18.

Example 6-18 Create database object from the bash shell

```
bash>db2 CONNECT to invent
bash>db2 -tf db2objects.ddl
bash>db2 DISCONNECT invent
```

6.4.3 Metadata transport

In this section, we discuss using the database modeling tool for the conversion of a database structure from MySQL to DB2. There are a number of modeling tools that exist in the market that support model capturing of a MySQL database. Most of the tools use a logical model to define the database design and a physical model map to the target database. One tool is the IBM Rational Rose® Professional Data Modeler Edition.

The *IBM Rational Rose Data Modeler Edition* is a data design tool, which integrates application design with database design and maps the data model with the object model. It allows database designers, business analysts, and developers to work together through a common language. Use this tool for conversions that require your MySQL model file to be available for a conversion to a DB2 physical model.

Database structure conversion process using modeling tools

Database structure conversion using the modeling tool is an extremely neat technique for database conversion, because many applications already have entity-relationship diagrams. Follow these steps to convert the database structure easily:

1. Reverse-engineer database objects using a DDL script or an existing database using one of the modeling tools.
2. Switch to a physical model.
3. Select the DB2 database as a target database and generate a DDL script for the new target.
4. Create the DB2 database structure using DDL scripts.

Although this technique is good, it is extremely costly, because it requires modeling tools for the conversions.

6.5 Sample database conversion

In this section, we demonstrate the database structure conversion for our sample Inventory Management application from MySQL to DB2. The Inventory database is a small database consisting of tables, views, indexes, triggers, and a stored procedure. Before starting the database conversion, see the existing database structure in 4.2, “Database structure” on page 84.

For demonstration purposes, we describe a step-by-step conversion process using the IBM Data Movement Tool. Then, we describe the manual conversion method for objects that are not converted with the IBM Data Movement Tool. We also discuss additional database enhancements.

6.5.1 Converting database objects with the IBM Data Movement Tool

The following steps describe the process of converting the MySQL sample Inventory database to DB2 using the IBM Data Movement Tool.

STEP 1: Create the DB2 Database

DBAs normally like to create their databases according to their storage paths' information. Therefore, prior to using the IBM Data Movement Tool, you need to create the database manually. Consider reading the IBM best practice papers at this Web site:

<http://www.ibm.com/developerworks/data/bestpractices/>

For this example, we use the following command to create our new DB2 database:

```
db2> CREATE DATABASE invent AUTOMATIC STORAGE YES ON
'/home/db2inst1/invent'
```

STEP 2: Specify the source, connection information, and test connection

Specify the source, DB2 database server connection information, and test connection within the IBM Data Movement Tool:

1. Start the IBM Data Movement Tool.

For our conversion scenario, we use the IBM Data Movement Tool GUI:

```
<IBM Data Movement Tool Installation directory>/IBMDDataMovementTool.sh
db2inst1@db2server: /opt/ibm/IBMDDataMovementTool>
./IBMDDataMovementTool.sh
```

The IBM Data Movement window opens, as shown in Figure 6-8.

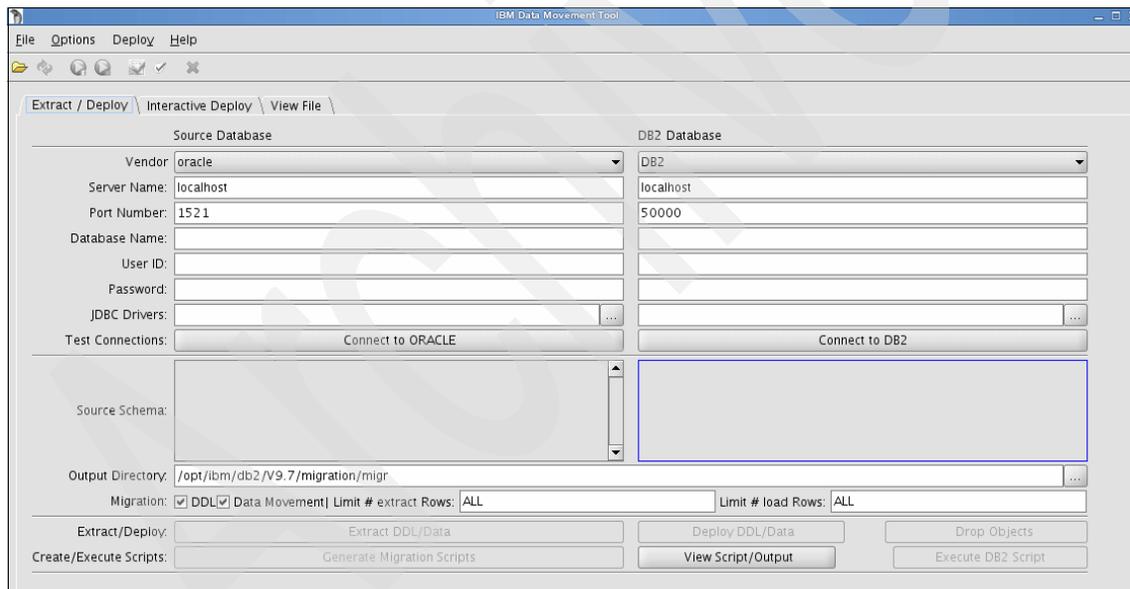


Figure 6-8 IBM Data Movement Tool startup window

You can also run the IBM Data Movement Tool in command-line mode. The tool automatically switches to command-line mode if it is not able to start the GUI. If you want to run the tool in command-line mode, you can run the following command:

```
<IBM Data Movement Tool Installation directory>/IBMDDataMovementTool.sh
-console
db2inst1@db2server:/opt/ibm/IBMDDataMovementTool>
./IBMDDataMovementTool.sh -console
```

2. Specify the source and target database information.

Enter the connection information in the Source Database and DB2 Database fields. Because the IBM Data Movement Tool that is used in this example is installed on the DB2 server, we connect to localhost for the DB2 server and specify the MySQL server IP address. You must know the following information:

- IP Address or host name of the source and DB2 servers
- Port numbers to connect
- Name of the databases
- A user ID with DBA privileges on the source database
- Password for that user
- Location of your source database and DB2 JDBC drivers
- Enough space or volume/mount point information where data will be stored

3. Test the connection.

After you have filled out the source and target database connection information, the next step is to test the connection to the source and target databases. Click **Connect to MySQL** to test the connection and **Connect to DB2** to test the DB2 connection. Connection results will show in the lower-left corner of the window.

Figure 6-9 on page 151 shows the values filled out for our sample conversion and the buttons to click to test the connections.

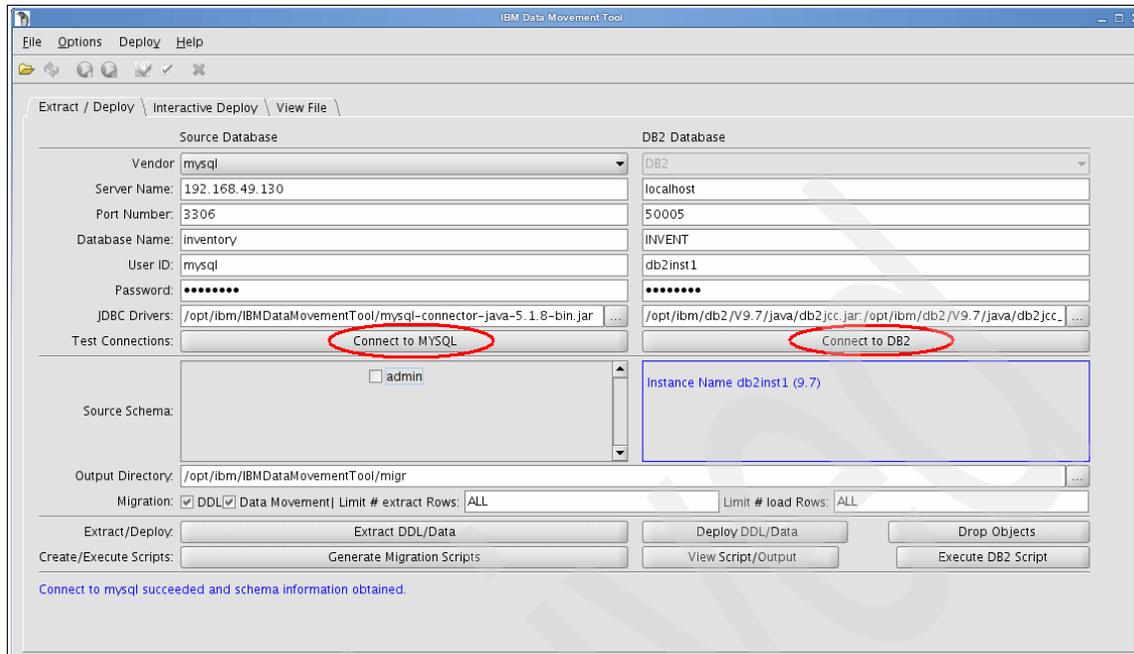


Figure 6-9 Testing database connection using IBM Data Movement Tool

STEP 3: Extract the DDL from the source database

In this step of the conversion process, we extract the DDL and data from the source MySQL database. The IBM Data Movement Tool connects to the specified source database, extracts the DDL and data, formats the DDL scripts to DB2 syntax, and generates conversion scripts:

1. Specify the output directory.

Specify the working directory where the DDL and data are to be extracted. The user running the IBM Data Movement tool must have full access to the directories specified in this step. For our sample conversion, we use the default directory:

```
<IBM Data Movement Tool Installation directory>/migr
```

2. Choose whether DDL and DATA are required.

With the IBM Data Movement Tool, you have the option to extract only the database objects, only the data, or both. Select or clear the **DDL** and **Data Movement** check boxes. We extract only the DDL at this point.

3. Extract the DDL and data.

Click **Extract Data** to extract the DDL or DATA and automatically convert it to DB2 syntax. Five threads are used to complete the extractions. For data

movement using the command-line approach, you can control the number of threads by modifying NUM_THREADS in the unload script. You can monitor the progress in the console window, as shown in Figure 6-10.

```
[2009-08-24 16.16.34.341] Driver com.mysql.jdbc.Driver loaded
[2009-08-24 16.16.36.396] Database Product Name :MySQL
[2009-08-24 16.16.36.397] Database Product Version :5.1.36-log
[2009-08-24 16.16.36.397] JDBC driver MySQL-AB JDBC Driver Version = mysql-cor
[2009-08-24 16.16.36.397] Database Version :5.1
[2009-08-24 16.16.36.811] Starting Blades
[2009-08-24 16.16.36.812] Starting Blade_0
[2009-08-24 16.16.36.815] Starting Blade_1
[2009-08-24 16.16.36.816] Starting Blade_2
[2009-08-24 16.16.36.834] Starting Blade_4
[2009-08-24 16.16.36.960] Starting Blade_3
[2009-08-24 16.16.37.616] Blade_2 unloaded 140 rows in 800 ms for ADMIN.locati
[2009-08-24 16.16.37.805] Blade_0 unloaded 6 rows in 993 ms for ADMIN.groups
[2009-08-24 16.16.37.859] Blade_0 unloaded 5 rows in 54 ms for ADMIN.severity
[2009-08-24 16.16.37.921] Blade_4 unloaded 808 rows in 1084 ms for ADMIN.servi
[2009-08-24 16.16.37.924] Blade_3 unloaded 504 rows in 964 ms for ADMIN.owners
[2009-08-24 16.16.37.988] Blade_1 unloaded 703 rows in 1173 ms for ADMIN.inver
[2009-08-24 16.16.38.086] Blade_1 unloaded 7 rows in 98 ms for ADMIN.status
[2009-08-24 16.16.38.089] ==== Total time: 1.0 sec
[2009-08-24 16.16.38.345] done Blade_0
[2009-08-24 16.16.38.601] done Blade_1
[2009-08-24 16.16.38.856] done Blade_2
[2009-08-24 16.16.39.113] done Blade_3
[2009-08-24 16.16.39.372] done Blade_4
[2009-08-24 16.16.39.388] Work completed
```

Figure 6-10 DDL and data extraction output

After the data extraction is complete, read through the result output files for the status of the data movement, warnings, errors, and other potential issues. You can click View Script/Output from the Extract/Deploy window to check the generated scripts, DDL, data, or output log file.

Table 6-3 on page 153 shows the command scripts that are regenerated each time that you run the tool in GUI mode. These scripts can also be issued in console mode without the GUI, which is helpful when you want to embed this tool as part of a batch process to accomplish an automated data movement.

Table 6-3 IBM Data Movement Tool scripts

Filename	Description
IBMExtract.properties	This file contains all input parameters that you specified through your GUI or command-line input values. You can edit this file manually to modify or correct parameters. Note this file will be overwritten each time that you run the GUI.
geninput	This script is the first data movement step where you will create an input file that contains the names of the tables to move. You can edit this file manually to exclude tables that you do not want to move.
genddl	This optional script is only generated if you choose to separate DDL generation from DATA. This step is the second step in data movement to generate all DDLs from your source database to DB2.
unload	This script is the last step of data movement. This script unloads data from the source database server to flat files. DB2 LOAD scripts will be generated after running this script. Note that if you did not choose to separate DDL from DATA, the genddl content is included in the unload script.
rowcount	This file will be used after you have moved the data to perform a sanity check for the row count for tables in source and target database servers.

STEP 4: Deploying database objects

Now that we have the database object definitions extracted, we have three options to deploy the database objects to DB2:

- ▶ Deploy DDL and DATA in one step using the Deploy DDL/DATA button from the GUI window.
- ▶ Deploy DDL and DATA in one step by using the command-line unload script.
- ▶ Deploy the database objects one by one by selecting the Interactive Deploy tab.

Choose an option that works best for your conversion project. The interactive deployment mode is best for deploying database objects that contain triggers, functions, or procedures. In most MySQL conversions, the first two options will suffice, because the conversion of the database objects will be performed outside of the IBM Data Movement tool. For our example, we select the Interactive Deploy mode to better explain the conversion process by separating the database object and the data deployment. Figure 6-11 on page 154 shows the Interactive Deploy tab window.

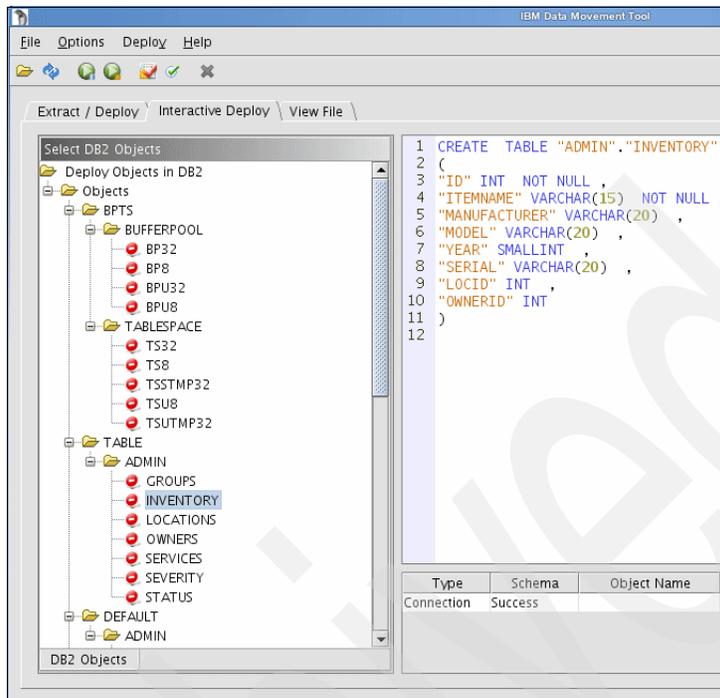


Figure 6-11 DDL and data extraction output

From the Interactive Deploy window, you can perform a number of tasks:

- ▶ Refresh the database object list.
Select the refresh button (circled in Figure 6-12 on page 155) to refresh the list of database objects in the DB2 Objects view on the left side of the window.
- ▶ Edit the object definition.
You can select the database object that you want to modify and edit in the right panel, as shown in Figure 6-12 on page 155. To save and deploy changes, deploy the object before selecting a new object. After deployment, you can return to refine any objects that failed to deploy.

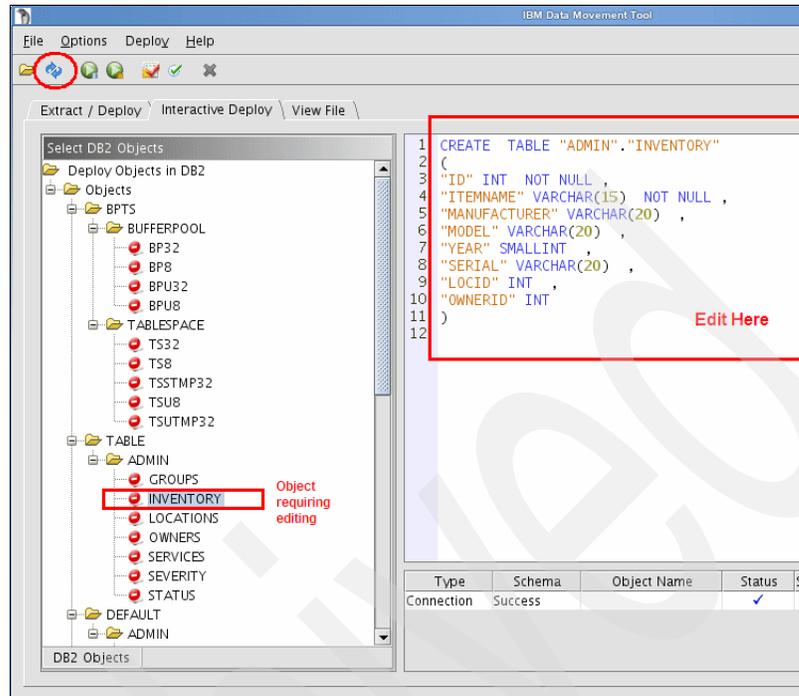


Figure 6-12 Editing objects

You can also edit the scripts that were extracted into the conversion directory. To change the table definition, edit the `db2tables.sql` file:

```
db2inst1@db2server:/opt/ibm/IBMDataMovementTool/migr> vi db2tables.sql
```

Example 6-19 shows the converted DB2 table creation file.

Example 6-19 Output DB2 statements - db2tables.sql

```
CONNECT TO INVENT;
-- Approximate Table Size 70
--#SET :TABLE:ADMIN:GROUPS
CREATE TABLE "ADMIN"."GROUPS"
(
"GROUPNAME" VARCHAR(30) NOT NULL ,
"EDITUSER" VARCHAR(10) ,
"EDITGRANTUSERPERM" VARCHAR(10) ,
"EDITINVT" VARCHAR(10) ,
"EDITSERVICE" VARCHAR(10)
)
;

-- Approximate Table Size 48
--#SET :TABLE:ADMIN:LOCATIONS
```

```

CREATE TABLE "ADMIN"."LOCATIONS"
(
  "ID" INT NOT NULL ,
  "ROOMNAME" VARCHAR(20) NOT NULL ,
  "FLOORNUM" INT NOT NULL ,
  "PASSCODE" VARCHAR(20)
)
;

-- Approximate Table Size 47
--#SET :TABLE:ADMIN:SEVERITY
CREATE TABLE "ADMIN"."SEVERITY"
(
  "ID" INT NOT NULL ,
  "TITLE" VARCHAR(15) NOT NULL ,
  "NOTES" VARCHAR(20) ,
  "ESTDAYS" INT ,
  "AVGDAYS" INT
)
;

-- Approximate Table Size 89
--#SET :TABLE:ADMIN:INVENTORY
CREATE TABLE "ADMIN"."INVENTORY"
(
  "ID" INT NOT NULL ,
  "ITEMNAME" VARCHAR(15) NOT NULL ,
  "MANUFACTURER" VARCHAR(20) ,
  "MODEL" VARCHAR(20) ,
  "YEAR" SMALLINT ,
  "SERIAL" VARCHAR(20) ,
  "LOCID" INT ,
  "OWNERID" INT
)
;

-- Approximate Table Size 64
--#SET :TABLE:ADMIN:SERVICES
CREATE TABLE "ADMIN"."SERVICES"
(
  "ID" INT NOT NULL ,
  "INVENTID" INT NOT NULL ,
  "DESCRIPTION" VARCHAR(20) NOT NULL ,
  "SEVERITY" INT NOT NULL ,
  "SERVICEOWNER" INT NOT NULL ,
  "OPENDATE" DATE NOT NULL ,
  "CLOSEDATE" DATE ,
  "TARGETCLOSEDATE" DATE ,
  "STATUS" INT
)
;

-- Approximate Table Size 39
--#SET :TABLE:ADMIN:STATUS

```

```

CREATE TABLE "ADMIN"."STATUS"
(
  "ID" INT NOT NULL ,
  "TITLE" VARCHAR(15) NOT NULL ,
  "NOTES" VARCHAR(20)
)
;

-- Approximate Table Size 212
--#SET :TABLE:ADMIN:OWNERS
CREATE TABLE "ADMIN"."OWNERS"
(
  "ID" INT NOT NULL ,
  "FIRSTNAME" VARCHAR(20) NOT NULL ,
  "LASTNAME" VARCHAR(20) NOT NULL ,
  "EMAIL" VARCHAR(50) ,
  "LOCID" INT ,
  "CUBENUM" INT ,
  "PHONENUM" VARCHAR(20) ,
  "LOGINNAME" VARCHAR(20) NOT NULL ,
  "PASSWORD" VARCHAR(20) NOT NULL ,
  "FAXNUM" VARCHAR(20) ,
  "GROUPS" VARCHAR(30)
)
;

TERMINATE;

```

Important: You must not reduce the size of any field, because it can cause an error while converting the data.

- ▶ Deploy objects to target.

The final step of the database conversion is to deploy the database objects. You use **Deploy All** (circled red in Figure 6-13 on page 158) to deploy all objects in the database object view or select **Deploy Selected** (circled blue in Figure 6-13 on page 158) to deploy the database object currently selected. You can view the status of the database object deployment and error messages in the lower-right panel, as shown in Figure 6-13 on page 158.

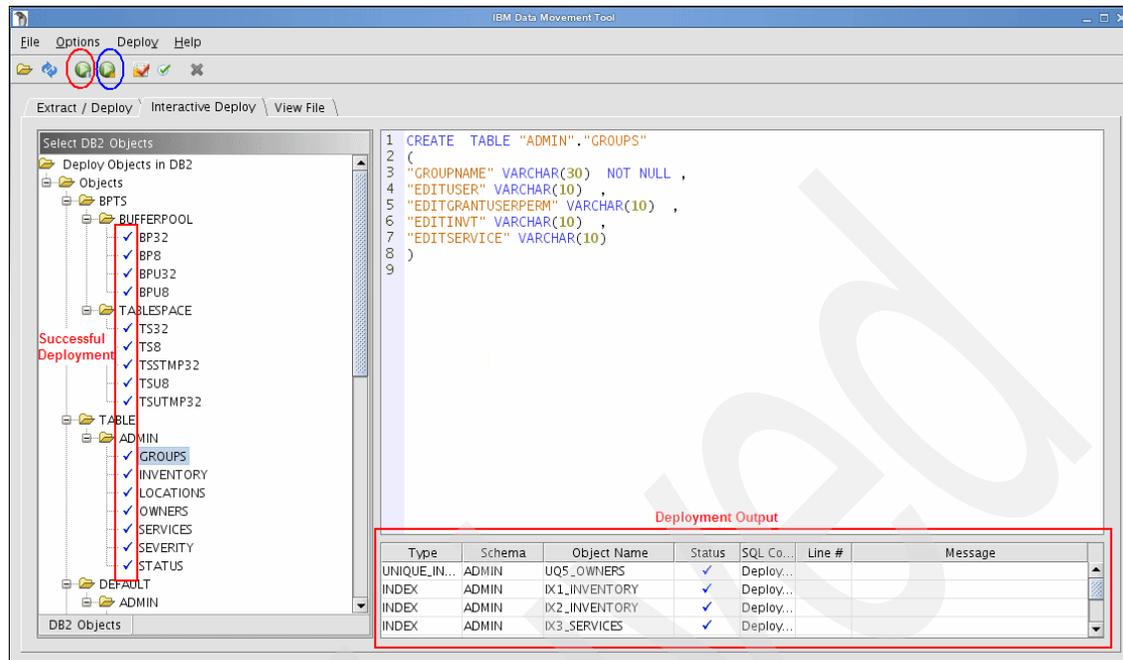


Figure 6-13 All database objects successfully deployed

6.5.2 Manual database object conversion and enhancements

Now that we have converted the major database objects to DB2 using the IBM Data Movement Tool, we can use manual methods for the objects that are not converted with the tool and add any additional database improvements.

Views

You must manually port views from MySQL to DB2. You can extract the MySQL view definition from the MySQL database using the mysqldump utility or selecting from the INFORMATION_SCHEMA.VIEWS table.

The syntax for a view in MySQL and DB2 is extremely similar, which makes it simple to convert the DDL for this database object. Example 6-20 on page 159 shows the CREATE VIEW syntax for the MySQL views.

For our example, we alter the CREATE VIEW commands to match the DB2 syntax, as shown in Example 6-21 on page 159.

Example 6-20 MySQL create view statement

```
CREATE VIEW managerGroup AS
SELECT o.id as newID, o.firstName, o.lastName, o.loginName, l.roomName, l.floorNum,
       o.phoneNum,
       (SELECT count(*) FROM inventory i, owners o
        WHERE o.id = newID and o.id = i.ownerID) as inventNum
FROM owners o, locations l
WHERE o.groups = 'manager' and o.locID = l.id;

CREATE VIEW empGroup AS
SELECT o.id as newID, o.firstName, o.lastName, o.loginName, l.roomName, l.floorNum,
       o.phoneNum,
       (SELECT count(*) FROM inventory i, owners o
        WHERE o.id = newID and o.id = i.ownerID) as inventNum
FROM owners o, locations l
WHERE o.groups = 'emp' and o.locID = l.id;

CREATE VIEW techGroup AS
SELECT o.id as newID, o.firstName, o.lastName, o.loginName, l.roomName, l.floorNum,
       o.phoneNum,
       (SELECT count(*) FROM inventory i, owners o
        WHERE o.id = newID and o.id = i.ownerID) as inventNum
FROM owners o, locations l
WHERE o.groups = 'tech' and o.locID = l.id;

CREATE VIEW bossGroup AS
SELECT o.id as newID, o.firstName, o.lastName, o.loginName, l.roomName, l.floorNum,
       o.phoneNum,
       (SELECT count(*) FROM inventory i, owners o
        WHERE o.id = newID and o.id = i.ownerID) as inventNum
FROM owners o, locations l
WHERE o.groups = 'boss' and o.locID = l.id;

CREATE VIEW generalGroup AS
SELECT o.id as newID, o.firstName, o.lastName, o.loginName, l.roomName, l.floorNum,
       o.phoneNum,
       (SELECT count(*) FROM inventory i, owners o
        WHERE o.id = newID and o.id = i.ownerID) as inventNum
FROM owners o, locations l
WHERE o.groups = 'general' and o.locID = l.id;
```

Example 6-21 DB2 create view statement

```
CREATE VIEW admin.managerGroup AS
SELECT o.id, o.firstName, o.lastName, o.loginName, l.roomName, l.floorNum,
       o.phoneNum,
       (SELECT count(*) FROM admin.inventory in, admin.owners u
        WHERE u.id = o.id and u.id = in.ownerID) AS inventNum
FROM admin.owners o, admin.locations l
WHERE o.groups = 'manager' and o.locID = l.id;

CREATE VIEW admin.empGroup AS
SELECT o.id, o.firstName, o.lastName, o.loginName, l.roomName, l.floorNum,
       o.phoneNum,
       (SELECT count(*) FROM admin.inventory in, admin.owners u
        WHERE u.id = o.id and u.id = in.ownerID) AS inventNum
FROM admin.owners o, admin.locations l
WHERE o.groups = 'emp' and o.locID = l.id;

CREATE VIEW admin.techGroup AS
SELECT o.id, o.firstName, o.lastName, o.loginName, l.roomName, l.floorNum,
```

```

        o.phoneNum,
        (SELECT count(*) FROM admin.inventory in, admin.owners u
         WHERE u.id = o.id and u.id = in.ownerID) AS inventNum
FROM admin.owners o, admin.locations l
WHERE o.groups = 'tech' and o.locID = 1.id;

CREATE VIEW admin.bossGroup AS
SELECT o.id, o.firstName, o.lastName, o.loginName, l.roomName, l.floorNum,
       o.phoneNum,
       (SELECT count(*) FROM admin.inventory in, admin.owners u
        WHERE u.id = o.id and u.id = in.ownerID) AS inventNum
FROM admin.owners o, admin.locations l
WHERE o.groups = 'boss' and o.locID = 1.id;

CREATE VIEW admin.generalGroup AS
SELECT o.id, o.firstName, o.lastName, o.loginName, l.roomName, l.floorNum,
       o.phoneNum,
       (SELECT count(*) FROM admin.inventory in, admin.owners u
        WHERE u.id = o.id and u.id = in.ownerID) AS inventNum
FROM admin.owners o, admin.locations l
WHERE o.groups = 'general' and o.locID = 1.id;

```

Trigger conversion

Triggers also require manual conversion to port them from MySQL to DB2. You can extract the MySQL trigger definition from the MySQL database using the `mysqldump` utility or selecting from the `INFORMATION_SCHEMA.TRIGGERS` table.

Example 6-22 on page 161 shows the MySQL `CREATE TRIGGER` statement and Example 6-24 on page 161 shows the DB2 `CREATE TRIGGER` statement. In these examples, note the difference between new and old data values that are declared and referenced. Refer to 6.2.4, “Trigger manipulation” on page 137 for more details about syntax differences between MySQL and DB2.

Also, notice the change in the date function. You must find the equivalent functions in DB2 when converting DDL and DML statements that contain built-in functions. We discuss and compare MySQL and DB2 built-in functions and operators in more detail in 8.1.10, “Built-in functions and operators” on page 221.

We have found a similar function in DB2 to replace the `curDate` function in MySQL. However, the default date format needs to be modified to the ISO format (`YYYY-MM-DD`), because the default format for the DB2 current date function is `MM/DD/YY`, whereas the MySQL `curDate` function is `YYYY-MM-DD`. The commands that are described in Example 6-23 on page 161 change the default format to the ISO format (`YYYY-MM-DD`).

Example 6-22 MySQL updateDate Trigger

```
CREATE TRIGGER updateDate BEFORE UPDATE on services
FOR EACH ROW
BEGIN
    IF NEW.status = 7 THEN
        SET NEW.closeDate = CURDATE();
    END IF;
END
```

Example 6-23 Commands to change to the ISO format

```
db2inst1@db2server:~> cd /home/db2inst1/sqllib/bnd/
db2inst1@db2server:~/sqllib/bnd> db2 CONNECT TO invent
```

Database Connection Information

```
Database server      = DB2/LINUX 9.7.0
SQL authorization ID = DB2INST1
Local database alias = INVENT
```

```
db2inst1@db2server:~/sqllib/bnd> db2 bind @db2ubind.lst datatype ISO blocking
all grant public
```

Example 6-24 DB2 updateDate trigger

```
CREATE TRIGGER updateDate BEFORE UPDATE on admin.services
REFERENCING NEW AS N OLD AS O
FOR EACH ROW
BEGIN
    IF N.status = 7 THEN
        SET N.closeDate = CURRENT DATE;
    END IF;
END
```

Stored procedure conversion

Porting stored procedures from MySQL to DB2 requires manual conversion. You can extract the MySQL stored procedure definition from the MySQL database by selecting the definition from the INFORMATION_SCHEMA.ROUTINES table.

The syntax for a procedure in MySQL and DB2 is similar. Example 6-25 on page 162 shows the CREATE PROCEDURE syntax for the MySQL procedure and Example 6-26 on page 162 shows the CREATE PROCEDURE syntax for the DB2 procedure. You might notice that the only difference between the statements is the date function to determine the number of days between the open date and the close date. For a description of MySQL built-in functions and DB2 equivalent

functions or solutions, refer to 8.1, “Data Manipulation Language differences and similarities” on page 206.

Example 6-25 MySQL create updateAvgDays procedure

```
CREATE PROCEDURE updateAvgDays(IN sevLevel INT)
BEGIN
    UPDATE severity SET avgDays = (SELECT SUM(Datediff(closeDate,
                                                openDate))/Count(*)
    FROM services where severity = sevLevel and status = 7)
    WHERE id = sevLevel;
END
```

Example 6-26 DB2 create updateAvgDays procedure

```
CREATE PROCEDURE updateAvgDays(IN sevLevel INT)
BEGIN
    UPDATE admin.severity
    SET avgDays = (SELECT SUM(TIMESTAMPDIFF(16,CHAR(TIMESTAMP(closeDate) -
TIMESTAMP(openDate))))/count(*)
    FROM admin.services WHERE severity = sevLevel and status = 7)
    WHERE id = sevLevel;
END
```

Foreign keys

Now, you can add any additional enhancement to your database using the DB2 features that might have not been supported in your existing MySQL storage engine. One example is referential integrity, which is essential to the database by ensuring consistency of data values between related columns in separate tables. Referential integrity is usually maintained by using the primary key, unique key, and foreign keys. MySQL only supports foreign keys in the InnoDB engine. Primary and unique keys are successfully converted using the IBM Data Movement Tool, but at the time of writing this book, foreign keys are not supported for a MySQL conversion project. If you want to create foreign keys in your database or to convert your foreign keys from an InnoDB database, you need to perform this task manually.

In our sample application, we create referential integrity between tables in a file called the `referential.ddl` file, which looks like Example 6-27.

Example 6-27 Add foreign keys

```
ALTER TABLE admin.owners ADD CONSTRAINT OWNERLOCI FOREIGN KEY (locid)
REFERENCES admin.locations (id) ON DELETE CASCADE;

ALTER TABLE admin.owners ADD CONSTRAINT OWNERGRPI FOREIGN KEY (groups)
REFERENCES admin.groups (GROUPNAME) ON DELETE CASCADE;
```

```

ALTER TABLE admin.inventory ADD CONSTRAINT INVENTLOCI FOREIGN KEY (locid)
    REFERENCES admin.locations (id) ON DELETE CASCADE;

ALTER TABLE admin.inventory ADD CONSTRAINT INVENTOWNERI FOREIGN KEY (ownerid)
    REFERENCES admin.owners (id) ON DELETE CASCADE;

ALTER TABLE admin.services ADD CONSTRAINT SERVINVENTI FOREIGN KEY (inventid)
    REFERENCES admin.inventory (id) ON DELETE CASCADE;

ALTER TABLE admin.services ADD CONSTRAINT SERVOWNERI FOREIGN KEY (serviceowner)
    REFERENCES admin.owners (id) ON DELETE CASCADE;

ALTER TABLE admin.services ADD CONSTRAINT SERVSEVERITYI FOREIGN KEY (severity)
    REFERENCES admin.severity (id) ON DELETE CASCADE;

ALTER TABLE admin.services ADD CONSTRAINT SERVSTATI FOREIGN KEY (status)
    REFERENCES admin.status (id) ON DELETE CASCADE;

```

Now, we have completed the DDL modification. We execute the changed scripts in Example 6-27 on page 162 to create the DB2 database and the objects, as shown in Example 6-28.

Example 6-28 Creation of tables and databases in DB2

```
db2inst1@db2server:~/DB2Scripts> db2 CONNECT TO invent
```

```
Database Connection Information
```

```

Database server      = DB2/LINUX 9.5.0
SQL authorization ID = DB2INST1
Local database alias = INVENT

```

```
db2inst1@db2server:~/DB2Scripts> db2 -tf referential.ddl
```

```

DB20000I The SQL command completed successfully.

```

Automatic maintenance

Performing maintenance activities on your databases is essential to ensure that they are optimized for performance and recoverability. The database manager provides automatic maintenance capabilities for performing database backups, keeping statistics current, and reorganizing tables and indexes as necessary.

For users, it can be time-consuming to determine when to run maintenance activities. Automatic maintenance takes the burden off of the users. With automatic maintenance, you can specify your maintenance objectives, including when automatic maintenance can run. The DB2 database manager uses the objectives that you have specified to determine if the maintenance activities need to be done and runs only the required maintenance activities during the next available maintenance window (a user-defined time period for running automatic maintenance activities). Example 6-29 shows the activities that can be controlled by the database manager's automatic maintenance feature.

Example 6-29 Automatic maintenance database manager variables

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

Enablement of the automatic maintenance features is controlled using the automatic maintenance database configuration parameters. These configuration parameters are a hierarchical set of switches to allow for simplicity and flexibility in managing the enablement of these features. You can automate database maintenance activities to run only when they are needed using the Configure Automatic Maintenance wizard. The DB2 database manager uses the objectives that you have specified using the Configure Automatic Maintenance wizard to determine whether the maintenance activities need to be done. Then, the DB2 database manager runs only the required maintenance activities during the next available maintenance window.

Self-tuning memory manager

A revolutionary memory tuning system called the *Self-Tuning Memory Manager* was introduced in DB2 9. It works on main database memory parameters, including buffer pools, sort heaps, locklist, package cache, and total database memory. It allows hands-off online memory tuning without DBA intervention by sensing the underlying workload and tunes the memory based on needs. When workload shifts, and memory redistribution is required to achieve optimal

performance, the Self-Tuning Memory Manager can adapt quickly to adjust the memory configuration.

Self-tuning memory simplifies the task of memory configuration by automatically setting values for memory configuration parameters and sizing buffer pools. When enabled, the memory tuner dynamically distributes available memory resources between several memory consumers, including sort, package cache and lock list areas, and buffer pools. Example 6-30 shows how to enable Self-Tuning Memory Manager and how to enable specific parameters to be automatically tuned.

Example 6-30 Setting up Self-Tuning Memory Manager

```
db2> UPDATE DATABASE CONFIGURATION FOR invent SELF_TUNING_MEM ON

db2> UPDATE DB CFG FOR invent USING locklist AUTOMATIC
      maxlocks AUTOMATIC
      pckcachesz AUTOMATIC
      sheapthres_shr AUTOMATIC
      sortheap AUTOMATIC
```

Archived

Data conversion

There are various considerations around converting data from a MySQL database to DB2. In this chapter, we focus on several of these considerations and describe the usage of tools and commands that aid in data movement from a MySQL database to a DB2 database.

In this chapter, we also discuss the differences in specific data formats and data types and ways in which they can be converted from MySQL to DB2.

We also describe how user account management (user data, access rights, and privileges) is implemented in MySQL and how this information can be ported to implement secure database access within DB2.

Finally, we discuss in detail the steps that we performed to convert the data in our sample project.

7.1 Data porting considerations

Data porting describes the necessary steps to get data from one database to another database. In general, you have to unload (also referred to as *dump* or *export*) the data from the source database into one or more files and load (also referred to as *import*) these files into the target database.

Database systems provide commands and tools for unloading and loading data. In MySQL, the `mysqldump` tool is used to unload a database. DB2 provides the `LOAD` and `IMPORT` commands for loading data from files into the database.

You have to be aware of the differences in how specific data types are represented by various database systems. For example, the representation of date and time values can differ in separate databases, and this representation often depends on the local settings of the system.

If the source and the target databases use different formats, you must convert the data either automatically using tools or manually. Otherwise, the loading tool is not able to understand the data that it needs to load due to improper formatting.

You must convert the binary data stored in binary large objects (BLOBs) manually, because binary data cannot be exported to files in a text format.

Porting the user account management is an extremely specific step in a conversion project. You must get the information about users, access rights, and privileges out of MySQL, convert it to DB2 specific security information, and create the users in DB2 according to the source data. Porting encrypted passwords is impossible in most cases.

7.1.1 Data porting commands and tools

MySQL and DB2 both provide tools that support data porting between the two systems. In MySQL, the `mysqldump` utility and the `mysqlhotcopy` scripts are used to retrieve data from the database; the `LOAD` and `IMPORT` commands can be used to get this data into DB2.

The IBM Data Movement Tool automates the use of the MySQL `SELECT` statements and the DB2 `LOAD` commands.

The mysqldump tool

When porting data, you use this tool to retrieve the data from MySQL tables. It is included with MySQL and is usually located in the bin directory of the MySQL installation.

The following example shows the syntax of the mysqldump tool:

```
mysqldump [OPTIONS] database [tables]
```

For a complete description of this tool, run `mysqldump --help`. Consider these important command-line options:

▶ `--no-data`

This option ensures that no data is extracted from the database, just the SQL statements for creating the tables and indexes. Therefore, this option is used for extracting DDL statements only.

▶ `--no-create-info`

This option ensures that no SQL statements for creating the exported table are extracted from the database. Therefore, it is used for exporting data only. The output file containing the data can be loaded into a DB2 table at a later time.

▶ `--tab=<outFilePath>`

This option creates a text file with the DDL (`<tablename>.sql`) and a tab separated text file with the data (`<tablename>.txt`) in the given path for each specified table. This option works only when the utility is run on the same machine as the MySQL daemon. If this option is not specified, INSERT statements for each row are created.

Example 7-1 shows the usage and output of the `mysqldump` command using only the `--user` and `--password` options. The output includes DDL statements for table creation and INSERT statements to insert data into the table.

Example 7-1 Usage of mysqldump with only the --user and --password options

```
mysqlServer:~ # mysqldump --user root --password inventory severity

-- MySQL dump 10.13  Distrib 5.1.36, for pc-linux-gnu (i686)
--
-- Host: localhost    Database: inventory
--
-----
-- Server version      5.1.36-log
--
--
-- Table structure for table `severity`
--
```

```

DROP TABLE IF EXISTS `severity`;

CREATE TABLE `severity` (
  `id` int(15) NOT NULL,
  `title` varchar(15) NOT NULL,
  `notes` varchar(20) DEFAULT NULL,
  `estDays` int(20) DEFAULT '14',
  `avgDays` int(20) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Dumping data for table `severity`
--

LOCK TABLES `severity` WRITE;

INSERT INTO `severity` VALUES
2,'high-med',NULL,4,4),
(4,'low-med',NULL,10,10),
(3,'medium',NULL,7,7),(
5,'low',NULL,14,12),(
1,'high',NULL,1,2);

UNLOCK TABLES;

-- Dump completed on 2009-08-25  2:11:48

```

Example 7-2 shows the usage and output of the `mysqldump` command with the `--no-create-info` option, but without the `--tab` option. The output has only `INSERT` statements for each row to insert data into the table.

Example 7-2 Usage of `mysqldump` with the `--no-create-info` but without the `--tab` option

```

mysqlServer:~# mysqldump--no-create-info --user root -password inventory severity

-- MySQL dump 10.13  Distrib 5.1.36, for pc-linux-gnu (i686)
--
-- Host: localhost    Database: inventory
-- -----
-- Server version    5.1.36-log
--
-- Dumping data for table `severity`
--

LOCK TABLES `severity` WRITE;

INSERT INTO `severity` VALUES
(2,'high-med',NULL,4,4),
(4,'low-med',NULL,10,10),
(3,'medium',NULL,7,7),
(5,'low',NULL,14,12),

```

```
(1, 'high', NULL, 1, 2);
```

```
UNLOCK TABLES;
```

```
-- Dump completed on 2009-08-25 2:28:23
```

Example 7-3 shows the usage and output of the `mysqldump` command with the `--no-create-info` and the `--tab` options. This command outputs a file in the current directory named `<tableName>.txt` that contains only the exported MySQL data. This file can be read by the DB2 LOAD command.

Example 7-3 Usage of mysqldump with the --no-create-info and the --tab option

```
mysqlServer:~ # mysqldump --no-create-info --tab=. --user root --password  
inventory severity
```

```
mysqlServer:~ # cat severity.txt
```

```
2      high-med      \N      4      4  
4      low-med       \N      10     10  
3      medium        \N      7      7  
5      low           \N      14     12  
1      high          \N      1      2
```

Example 7-4 shows the usage and output of `mysqldump` without the `--no-create-info` option but with the `--tab` option. This command outputs two files: one file contains the DDL statements for table creation (`<tableName>.sql`) and the other file contains the exported MySQL data (`<tableName>.txt`) in the current directory. The second file can be read by the DB2 LOAD command.

Example 7-4 Usage of mysqldump without the --no-create-info but with the --tab option

```
mysqlServer:~ # mysqldump --tab=. --user root --password inventory severity
```

```
mysqlServer:~ # cat severity.sql
```

```
-- MySQL dump 10.13 Distrib 5.1.36, for pc-linux-gnu (i686)
```

```
--
```

```
-- Host: localhost Database: inventory
```

```
-- -----
```

```
-- Server version 5.1.36-log
```

```
--
```

```
-- Table structure for table `severity`
```

```
--
```

```
DROP TABLE IF EXISTS `severity`;
```

```
CREATE TABLE `severity` (
```

```

`id` int(15) NOT NULL,
`title` varchar(15) NOT NULL,
`notes` varchar(20) DEFAULT NULL,
`estDays` int(20) DEFAULT '14',
`avgDays` int(20) DEFAULT NULL,
PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- Dump completed on 2009-08-25 2:49:30

```

```

mysqlServer:~ # cat severity.txt
2      high-med  \N      4      4
4      low-med   \N     10     10
3      medium   \N      7      7
5      low      \N     14     12
1      high     \N      1      2

```

The mysqlhotcopy script

When porting data, you can use the `mysqlhotcopy` Perl script to retrieve the data from MySQL tables. It is included with MySQL, located in the `bin` directory of the MySQL installation, and requires that the Perl DBI module is installed.

This example shows the syntax of the `mysqlhotcopy` script:

```
mysqlhotcopy database [/path/]
```

IBM Data Movement Tool SELECT script

The IBM Data Movement Tool generates a file with the `SELECT` statement that is required to extract the data from the MySQL database. This file, which is called `<database>.tables`, is located in the conversion output directory. It is generated during the DDL extraction phase, which is described in 6.5.1, “Converting database objects with the IBM Data Movement Tool” on page 148.

Example 7-5 illustrates the `inventory.tables` generated file for our sample conversion.

Example 7-5 *Inventory.tables* file

```

db2server:/opt/ibm/IBMDDataMovementTool/migr # cat inventory.tables
"ADMIN"."groups":SELECT * FROM groups
"ADMIN"."inventory":SELECT * FROM inventory
"ADMIN"."locations":SELECT * FROM locations
"ADMIN"."owners":SELECT * FROM owners
"ADMIN"."services":SELECT * FROM services
"ADMIN"."severity":SELECT * FROM severity
"ADMIN"."status":SELECT * FROM status

```

The DB2 LOAD command lets you specify the file-type-modifier clause *TIMESTAMPFORMAT*, which determines the formatting of the *TIMESTAMP* values. If you want to import MySQL *TIMESTAMP* values, you must change the LOAD command in the `deploy.sh` script to the following syntax:

```
db2 LOAD from "'<outFilePath>/<tablename>.dat'" |
of DEL
modified by
colde10x09
timestampformat=\ " YYYY-MM-DD HH:MM:SS\ "
insert into <schemaname>.<tablename>
```

If you use the IBM Data Movement Tool, you do not have to worry about this format, because the tool takes care of this formatting for you.

Manipulating data extraction

In certain cases, you might want to manipulate the data when it is extracted from the MySQL database. For example in our sample conversion, the application stores the application user passwords encrypted in the `owners` table. Because we know the key that is used to encrypt each password, these passwords can be decrypted during extraction. The IBM Data Movement Tool uses `SELECT` statements to extract the data from the MySQL database. You can find the `SELECT` commands in the `<database>.tables` file. To modify the extraction commands, open the file in an editor and make the necessary changes to the `SELECT` command. The highlighted text in Example 7-9 illustrates the changes that we made to the `inventory.tables` file.

Example 7-9 Edits made to IBM Data Movement Tool extraction script

```
"ADMIN"."groups":SELECT * FROM groups
"ADMIN"."inventory":SELECT * FROM inventory
"ADMIN"."locations":SELECT * FROM locations
"ADMIN"."owners":SELECT id, firstName, lastName, email, locID, cubeNum,
phoneNum, loginName, AES_DECRYPT(password,'password') as password, faxNum,
groups FROM owners
"ADMIN"."services":SELECT * FROM services
"ADMIN"."severity":SELECT * FROM severity
"ADMIN"."status":SELECT * FROM status
```

When extracting the data from the GUI tool, make sure to select **No** when requested to recreate the conversion directory and the `<database>.tables` file. Otherwise, your changes will be overwritten.

7.1.3 Differences in the user account management

The method in which the access rights and privileges are stored in MySQL differs completely from DB2.

MySQL user account management

Porting the user account management from MySQL to the DB2 security system requires knowledge about how the user account management data affects your application and how user data, passwords, access rights, and privileges are stored in MySQL.

User accounts

When assessing your application, be sure to distinguish between the following user types:

- ▶ Application user accounts

These users log on to the application, but they do not exist on the database level. Database access is through the application with the application's database user ID. Because the information about application users is usually stored in custom application tables, the porting of application user account data is done when porting the MySQL application data. In the sample inventory database, this custom application table is our owners table.

- ▶ Database user accounts

Database users connect directly to the MySQL database to retrieve and manipulate data. At least one database user must exist for applications to connect to the database. Database user accounts are created with the MySQL server and allow you to grant and restrict access to portions of the MySQL servers. A database user account is associated with a host name and user name. The user account information is stored with the *mysql.user* table and must be ported in data conversion step. Access rights and privileges for these users are stored in the *mysql.db*, *mysql.host*, *mysql.tables_priv*, *mysql.columns_priv*, and *mysql.procs_priv* tables.

Passwords

Database users have associated passwords, which are stored encrypted in the *mysql.user* table. Encrypted passwords cannot be ported and must be reset on the new system. The password of the database user, which is used by an application to access the database, is typically stored in a profile with restricted rights.

Important: The porting of encrypted database user passwords is impossible, because it is the intended purpose of encryption functions to make password decryption impossible.

Access rights

When accessing a MySQL database, there are two levels of access control. When you first connect to a MySQL server, you provide a user name and the associated password. Furthermore, the machine that you are connecting from must be associated with this user to allow the connection. This requirement is based on the assumption that a user with a specific user name from one host is different from a user with the same user name from a separate host.

This access information is stored in the `mysql.user` table in the fields: `user`, `password`, and `host`. The MySQL wildcard ampersand character (%) is often used in the `host` field to specify that this user can connect from any host. The wildcard underscore (_) is also sometimes used for single characters.

In Example 7-10, the `user1` user can connect from any host, the `user2` user can connect from only the `remoteHost.ibm.com` host, and `db2inst1` can connect from `localhost`, `myServer`, and `127.0.0.1`.

Example 7-10 Sample user data for connection verification

```
mysql> select user, password, host from mysql.user;
```

user	password	host
db2inst1	EE2FD1618DDEE42FD1618BB9JD33HD736H5HNNT757	myServer
db2inst1	GGFD1618BDJ473SKCNGU8JB9JD33HD73J5748SUFL	127.0.0.1
user1	2FD162470C0C06DEE418BDCA2EC9D1E1B99005A9K	%.com
user2	*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19	remoteHost.ibm.com
db2inst1	06DE005ADCA2EC2470C0C99005*ADCA2EC8BB9900	localhost

Refer to the *MySQL Reference Manual* at this Web site for a complete description of the MySQL connection verification:

<http://dev.mysql.com/doc/refman/5.1/en/index.html>

You can also find information about how the entries in the `mysql.user` table are ordered when the provided connection data meets more than one connection criteria.

Privileges

The second level of access control occurs after a connection to the MySQL database is established. Each time that a command is run against the database, MySQL checks if the connected user has sufficient privileges to run this command.

Privileges exist for selecting, inserting, updating, and deleting data for creating, altering, and dropping database objects and other operations performed at the database level.

All privileges are stored in either the `mysql.user`, `mysql.db`, `mysql.host`, `mysql.tables_priv`, `mysql.columns_priv`, or `mysql.procs_priv` tables.

Privileges in MySQL can be granted on the following levels:

- ▶ Global level
Global privileges provide control over the overall MySQL server and are stored in the `mysql.user` table.
- ▶ Database level
Database privileges provide control over a specific database and are stored in the `mysql.db` (database privileges) and `mysql.host` (host privileges) tables.
- ▶ Table level
Table privileges provide control over a specific table and are stored in the `mysql.tables_priv` table.
- ▶ Column level
Column privileges provide control over a specific column in a table and are stored in the `mysql.columns_priv` table.
- ▶ Routine level
Stored routine privileges provide control over a specific stored routine and are stored in the `mysql.procs_priv` table.

Privileges can be granted to users with the MySQL GRANT command; they can be revoked with the REVOKE command.

For more information about MySQL privileges, see the *MySQL Reference Manual* at this Web site:

<http://dev.mysql.com/doc/refman/5.1/en/>

DB2 security system

There are four major mechanisms within DB2 that allow DB2 to implement a security plan: authentication, administrative authorization, privileges, and *Label-Based Access Control (LBAC)*. DB2 works closely with the security features of the underlying operating system to authenticate users' specific database privileges and authorizations. The database privileges and authorizations are granted to operating system users, groups, or DB2 roles.

User accounts

To create a user for DB2 implies creation of a user in the server's operating system, assigning the user to a group, and granting specific database privileges to the user or group.

On Linux systems, you must have root access to the system to create groups and users. Group information is stored in the `/etc/group` file and user information is stored in the `/etc/passwd` file.

For example, if you want to create a new `db2app1` group with one user, `db2usr1`, to access a specific DB2 table, perform the necessary steps:

1. Log on to the Linux system with root privileges.
2. Create the group. Make sure that the provided group name does not already exist and ensure that it is not longer than eight characters:

```
groupadd [-g 995] db2app1
```

3. Create the user and assign it to the previously created group. Make sure that the ID for the user does not already exist and that it is not longer than eight characters:

```
useradd [-u 1001] -g db2app1 -m -d /home/db2usr1 db2usr1 [-p passwd1]
```

If the user will access the DB2 database locally, continue with the next two steps:

- a. Edit the profile of the created user:

```
vi /home/db2usr1/.profile
```

- b. Add the following line to the profile. Be sure to specify the path of your DB2instance owner's home directory and to specify a blank between the dot and the command:

```
. /home/db2inst1/sqllib/db2profile
```

DB2 9.5 introduced roles to simplify the management of authorization. *Roles* are equivalent to in-database groups and allow DBAs to group together one or more privileges, authorities, or security labels. Roles can be assigned to users, groups, *PUBLIC*, or other roles by using the GRANT statement.

Passwords

The passwords that are used for DB2 are the system passwords of the user. To set a password in Linux, use the `passwd <username>` command as root user.

Access rights

The first component in the DB2 security model is authentication. Access to DB2 databases is restricted to users that exist on the DB2 system. When connecting to a DB2 database, you have to provide a user name and password combination that is valid against the server's system. Authentication can occur at the DB2 server or the DB2 client using operating system authentication, Kerberos, or an external security manager.

Authorities and privileges

Privileges enable users to create or access database objects. *Authority levels* provide a method of grouping privileges and control over higher-level database manager maintenance and utility operations. Label-based access control (LBAC) provides a more granular approach to granting privileges on a row and column basis. Together, these methods control access to the database manager and its database objects. Users can access only those objects for which they have the appropriate authorization, that is, the required privilege or authority.

Administrative authority

A user or group can have one or more of the following administrative authorities.

System-level authorization

The system-level authorities provide varying degrees of control over instance-level functions:

▶ **SYSADM authority**

The *SYSADM* (*system administrator*) authority provides control over all the resources created and maintained by the database manager. The system administrator possesses all authorities for the SYSCTRL, SYSMOINT, and SYSMON authority. The user who has SYSADM authority is responsible for both controlling the database manager and ensuring the safety and integrity of the data.

▶ **SYSCTRL authority**

The *SYSCTRL* authority provides control over operations that affect system resources. For example, a user with SYSCTRL authority can create, update, start, stop, or drop a database. This user can also start or stop an instance, but cannot access table data. Users with SYSCTRL authority also have SYSMON authority.

► **SYSMAINT** authority

The *SYSMAINT* authority controls maintenance operations on all databases associated with an instance. A user with SYSMAINT authority can update the database configuration, back up a database or table space, restore an existing database, and monitor a database. Like SYSCTRL, SYSMAINT does not provide access to table data. Users with SYSMAINT authority also have SYSMON authority.

► **SYSMON** authority

The *SYSMON* (system monitor) authority controls the usage of the database system monitor.

Figure 7-1 illustrates the instance-level authorities that can be granted to a user or role.

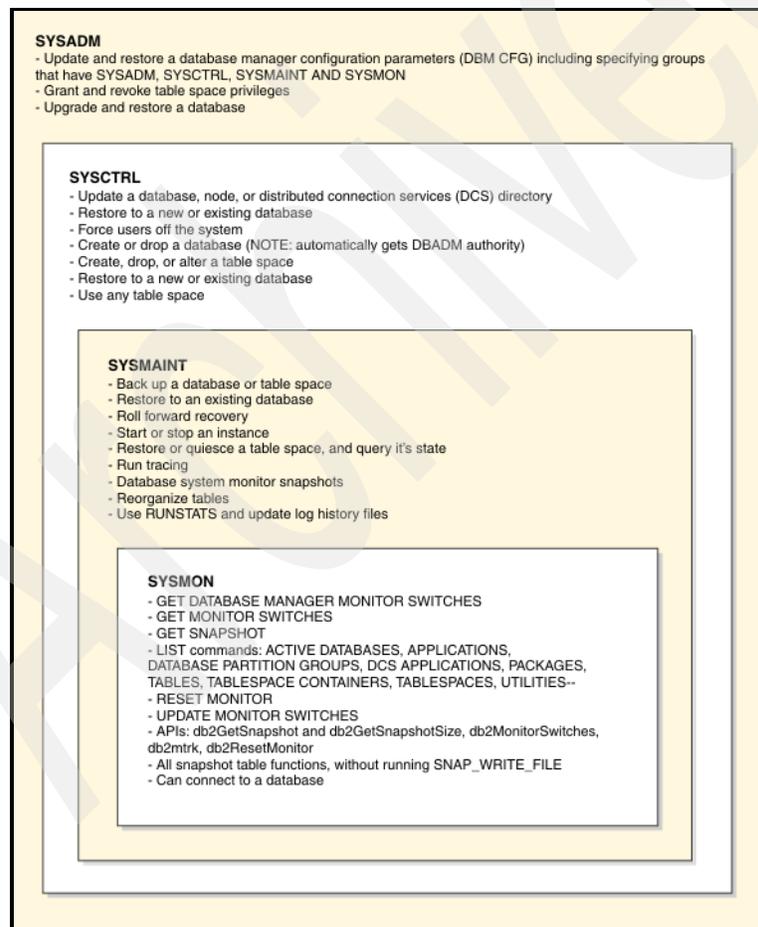


Figure 7-1 Instance-level authorities

Database-level authorization

The database-level authorities provide control within a database:

- ▶ **DBADM (database administrator)**

The *DBADM authority* level provides administrative authority over a single database. This database administrator possesses the privileges required to create objects and issue database commands.

The DBADM authority can only be granted by a user with SECADM authority and cannot be granted to PUBLIC.

- ▶ **SECADM (security administrator)**

The *SECADM authority* level provides administrative authority for security over a single database. The security administrator authority possesses the ability to manage database security objects (database roles, audit policies, trusted contexts, security label components, and security labels) and grant and revoke all database privileges and authorities. A user with SECADM authority can transfer the ownership of objects that they do not own. They can also use the `AUDIT` statement to associate an audit policy with a particular database or database object at the server.

The SECADM authority has no inherent privileges to access data stored in tables. This authority can only be granted by a user with SECADM authority. The SECADM authority cannot be granted to PUBLIC.

- ▶ **SQLADM (SQL administrator)**

The *SQLADM authority* level provides administrative authority to monitor and tune SQL statements within a single database. It can be granted by a user with ACCESSCTRL or SECADM authority.

- ▶ **WLMADM (workload management administrator)**

The *WLMADM authority* level provides administrative authority to manage workload management objects, such as service classes, work action sets, work class sets, and workloads. It can be granted by a user with ACCESSCTRL or SECADM authority.

- ▶ **EXPLAIN (explain authority)**

The *EXPLAIN authority* level provides administrative authority to explain query plans without gaining access to data. It can only be granted by a user with ACCESSCTRL or SECADM authority.

▶ **ACCESSCTRL** (access control authority)

The *ACCESSCTRL* authority level provides administrative authority to issue the following GRANT (and REVOKE) statements. ACCESSCTRL authority can only be granted by a user with SECADM authority. The ACCESSCTRL authority cannot be granted to PUBLIC.

- GRANT (Database authorities)
- GRANT (Global variable privileges)
- GRANT (Index privileges)
- GRANT (Module privileges)
- GRANT (Package privileges)
- GRANT (Routine privileges)
- GRANT (Schema privileges)
- GRANT (Sequence privileges)
- GRANT (Server privileges)
- GRANT (Table, view, or nickname privileges)
- GRANT (Table space privileges)
- GRANT (Workload privileges)
- GRANT (XML schema repository (XSR) object privileges)

▶ **DATAACCESS** (data access authority)

The *DATAACCESS* authority level provides the following privileges and authorities. It can be granted only by a user who holds SECADM authority. The DATAACCESS authority cannot be granted to PUBLIC.

- LOAD authority
- SELECT, INSERT, UPDATE, and DELETE privileges on tables, views, nicknames, and materialized query tables
- EXECUTE privilege on packages
- EXECUTE privilege on modules
- EXECUTE privilege on routines

▶ **Database authorities** (non-administrative)

Specific database authorities are required to perform activities, such as creating a table or a routine, or for loading data into a table. For example, the LOAD database authority is required for use of the load utility to load data into tables (a user must also have INSERT privilege on the table).

Figure 7-2 on page 185 illustrates the database-level authorities that can be granted to a user or role.

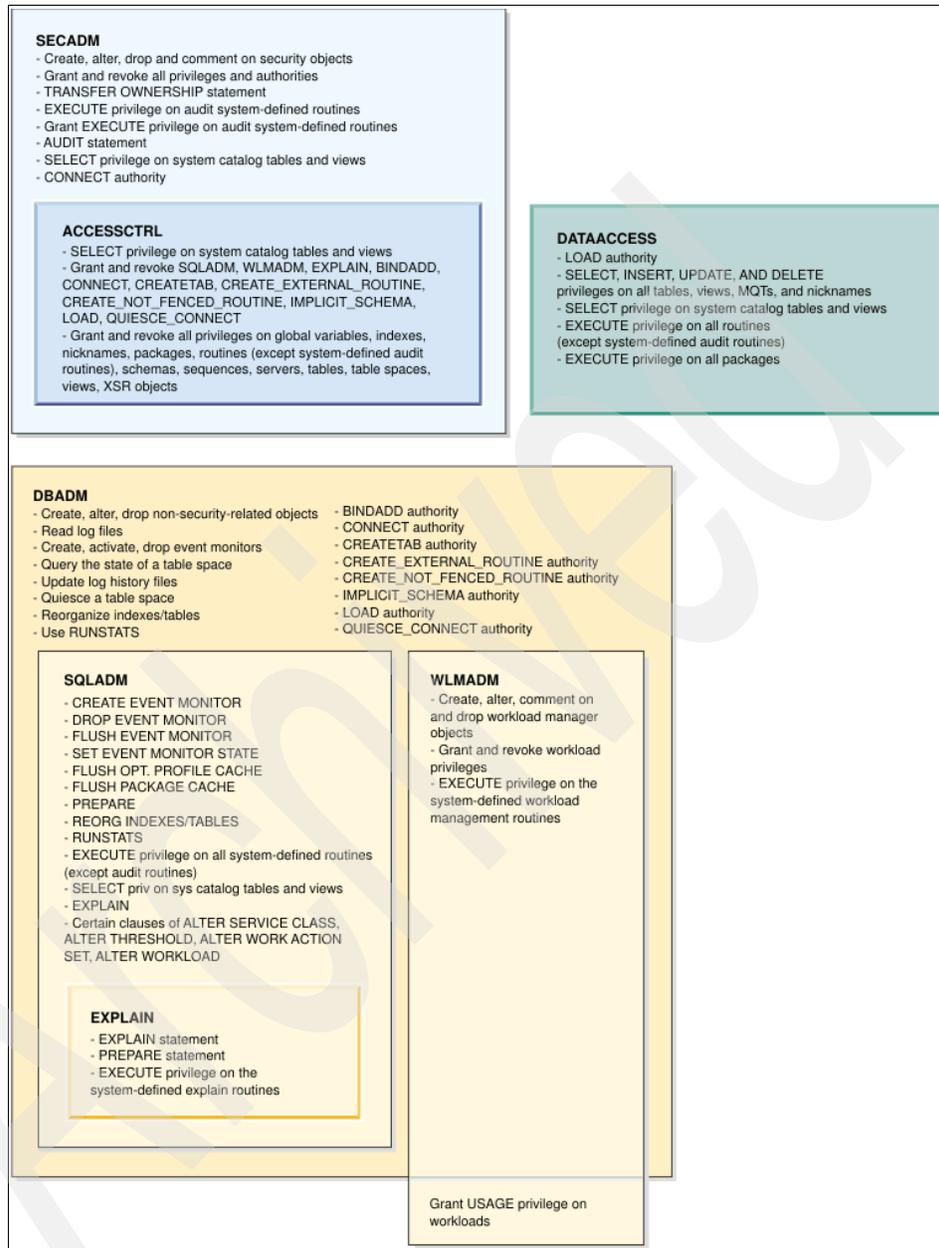


Figure 7-2 Database-level authorities

Privileges

A *privilege* is a permission to perform an action or a task. Authorized users can create objects, have access to objects they own, and can pass on privileges on their own objects to other users by using the GRANT statement.

Privileges can be granted to individual users, groups, or PUBLIC. *PUBLIC* is a special group that consists of all users, including future users. Users that are members of a group will indirectly take advantage of privileges granted to the group, where groups are supported.

Privileges can be assigned to a user in the following three ways:

- ▶ Explicit

Individual privileges can be explicitly granted to allow a user or group to carry out specific tasks on specific objects. These privileges can be given and revoked using the GRANT and REVOKE commands, for example:

```
db2 GRANT SELECT on TABLE db2inst1.t1 to employee
db2 REVOKE SELECT on TABLE db2inst1.t1 from employee
```

Users with the administrative authorities ACCESSCTRL or SECADM, or with the CONTROL privilege, can grant and revoke privileges to and from users.

Individual privileges and database authorities allow for a set of activities, but they do not include the right to grant the same privileges or authorities to other users. The right to grant table, view, schema, package, routine, and sequence privileges to other users can be extended to other users through the WITH GRANT OPTION on the GRANT statement. However, the WITH GRANT OPTION does not allow the person granting the privilege to revoke the privilege once it is granted. You must have SECADM authority, ACCESSCTRL authority, or the CONTROL privilege to revoke an explicitly granted privilege.

- ▶ Implicit

Implicit privileges are granted implicitly through the execution of a command. For example, if a user executes a CREATE statement, they inherently gain full access to the object they created, which is otherwise known as CONTROL privileges.

```
db2 CREATE TABLE t2
```

In this example, the user who issued the CREATE TABLE statement has full CONTROL privileges over table t2.

Possessing the CONTROL privilege on an object allows a user to access that database object, and to grant and revoke privileges to or from other users on that object. The creator of a base table, materialized query table, staging table, or nickname automatically receives the CONTROL privilege. The creator of a view automatically receives the CONTROL privilege if the user

holds the CONTROL privilege on all tables, views, and nicknames identified in the fullselect.

► Indirect

When a user has the privilege to execute a package or routine, the user does not necessarily require specific access privileges on the objects handled in the package or routine. If the package or routine contains static SQL or XQuery statements, the privileges of the owner of the package are used for those statements. If the package or routine contains dynamic SQL or XQuery statements, the authorization ID used for privilege checking depends on the setting of the DYNAMICRULES BIND option of the package issuing the dynamic query statements, and whether those statements are issued when the package is being used in the context of a routine.

A user or group can be authorized for any combination of individual privileges or authorities. When a privilege is associated with a resource, that resource must exist. For example, a user cannot be given the SELECT privilege on a table unless that table has previously been created.

Note: Care must be given to granting authorities and privileges to user names that do not exist in the system yet. At a later time, this user name can be created and automatically receive all of the authorities and privileges previously granted.

Privileges and authorities in DB2 can be granted on the following levels:

- Database level:
 - CONNECT privilege
 - CREATETAB privilege
 - LOAD privilege
 - IMPLICIT_SCHEMA privilege
 - BINDADD privilege
 - CREATE_EXTERNAL_ROUTINE privilege
 - CREATE_NOT_FENCED_ROUTINE privilege
 - IMPLICIT_SCHEMA privilege
 - LOAD privilege
 - QUIESCE_CONNECT privilege
 - ACCESSCTRL authority
 - DATAACCESS authority
 - EXPLAIN authority
 - DBADM authority
 - SECADM authority
 - SQLADM authority
 - WLMADM authority
 - Schema level

- CREATEIN privilege
- ALTERIN privilege
- DROPIN privilege
- ▶ Table space level:
 - USE privilege
- ▶ Table and view level:
 - CONTROL privilege
 - SELECT privilege
 - INSERT privilege
 - UPDATE privilege
 - DELETE privilege
 - INDEX privilege
 - ALTER privilege
 - REFERENCES privilege
 - ALL PRIVILEGES privilege
- ▶ Row or column level:
 - SECURITY LABEL privilege
 - LBAC Rule Exemption privileges
- ▶ Other privileges:
 - Package privileges
 - Index level privileges
 - Procedure, function, and method privileges
 - Sequence privileges

Label-Based Access Control

Label-based access control (LBAC) was introduced in DB2 9. LBAC provides the ability to have more control over who can access specific data in tables and views. LBAC provides particular privileges for users, groups, or roles on one or more columns and rows within a table.

LBAC controls access to table objects by attaching security labels to rows and columns. Users attempting to access an object must have a certain security label granted to them. Only users who have the proper labels when accessing the row are allowed to retrieve the data. Other users do not receive any indication that they cannot retrieve the row. This form of security differs from normal SELECT privileges, because users who attempt to access a table that they are not allowed to access receive an SQL error message.

All privileges and labels can be granted to users or groups with the GRANT command, and they can be revoked using the REVOKE command.

Note: Catalogs contain, among other things, statistics about data distribution in a table, which are needed by the query optimizer to determine the best way to execute a query. Users can indirectly gain knowledge about certain data values by accessing the catalogs. When using LBAC, you can add the new `RESTRICT_ACCESS` option on the `CREATE DATABASE` command to create a database where access to the catalogs is not granted to `PUBLIC`. Access to the catalogs can then be granted on a “need-to-know” basis.

Mapping the user information from MySQL to DB2

When converting from MySQL to DB2, you must port user privileges as well.

Table 7-1 shows the mapping of MySQL privileges to DB2 privileges, assuming that different MySQL databases are mapped to different DB2 schemas. For example, an `INSERT` privilege granted in MySQL on the global level means that you have to grant the `INSERT` privilege on all existing tables in the DB2 database to the specified user. If you create a new table in DB2, you have to grant the `INSERT` privilege on this table to the user.

Table 7-1 Mapping MySQL to DB2 privileges

MySQL privilege	MySQL scope	DB2 privilege or authority	DB2 scope
ALL [PRIVILEGES]	Global	SYSADMIN authority and DBADM authority with DATAACCESS and ACCESSCTRL authorities	Instance
	Database	DBADM authority with DATAACCESS and ACCESSCTRL authorities	Database
	Table	ALL WITH GRANT OPTION	Table
ALTER	Global	ALTERIN	For each schema in the database
	Database	ALTERIN	Schema
	Table	ALTER TABLE	Table
ALTER ROUTINE	Global	CREATE_EXTERNAL_ROUTINE	Database
	Database	CREATIN	Schema
CREATE	Global	CREATETAB	Database
	Database	CREATIN	Schema

	Table	CONTROL	Table
CREATE ROUTINE	Global	CREATE_EXTERNAL_ROUTINE	Database
	Database	CREATIN	Schema
CREATE TEMPORARY TABLES	Global	CREATETAB	Database
	Database	CREATIN	Schema
CREATE USER	Global	SECADMIN authority	Database
CREATE VIEW	Global	DATAACCESS authority	Database
	Database	CONTROL or SELECT	For all tables in the Schema
	Table	CONTROL or SELECT	Table
DELETE	Global	DATAACCESS authority	Database
	Database	DELETE	For each table in the schema
	Table	DELETE	Table
DROP	Global	DROPIN	For each schema in the database
	Database	DROPIN	Schema
	Table	CONTROL	Table
EVENT	Global	DBADM authority	Database
	Database	DBADM authority	Database
EXECUTE	Global	DATAACCESS authority	Database
	Database	EXECUTE	For each routine in the schema
	Routine	EXECUTE	Routine
FILE	Global	LOAD authority	Database
GRANT OPTION	Global	SECADMIN authority or DBADM WITH ACCESSCTRL	Database
	Database	SECADMIN authority or DBADM WITH ACCESSCTRL	Database

	Table	SECADMIN authority or DBADM WITH ACCESSCTRL or CONTROL	Database/Table
	Routine	SECADMIN authority or DBADM WITH ACCESSCTRL or CONTROL	Routine
INDEX	Global	CREATETAB	Database
	Database	CREATIN	Schema
	Table	INDEX	Table
INSERT	Global	DATAACCESS authority	Database
	Database	INSERT	For all tables in the Schema
	Table	INSERT	Table
	Column	LBAC	Column
PROCESS	Global	SYSADM, SYSCTRL, or SYSMAINT authority	Instance
RELOAD	Global	WLMADM	Database
REPLICATION CLIENT	Global	DBADM authority	Database
REPLICATION SLAVE	Global	DBADM authority	Database
SELECT	Global	DATAACCESS authority	Database
	Database	SELECT	For all tables in the Schema
	Table	SELECT	Table
	Column	LBAC	Column
SHOW DATABASES	Global	ALL users	All users
SHOW VIEW	Global	SELECT	On SYSCAT.VIEWS
	Database	SELECT	On SYSCAT.VIEWS
	Table	SELECT	On SYSCAT.VIEWS
SHUTDOWN	Global	SYSADMN, SYSCTRL, or SYSMAINT authority	Instance

SUPER	Global	Not available	
TRIGGER	Global	DBADM authority	Database
	Database	ALTERIN	Schema
	Table	CONTROL	Table
UPDATE	Global	DATAACCESS authority	Database
	Database	UPDATE	For all tables in the Schema
	Table	UPDATE	Table
	Column	LBAC	Column
USAGE	Global	Not applicable, because users are created through the operating system	
	Database	Not applicable, because users are created through the operating system	
	Table	Not applicable, because users are created through the operating system	

7.2 Sample project: Data porting

This section describes the steps that necessary to port data for our sample conversion project.

7.2.1 Export user data from MySQL

We use the `mysqlaccess` utility to get the user names that have access to the database inventory out of the MySQL database. See Example 7-11.

Example 7-11 shows retrieving users with access to the sample project database.

Example 7-11 Retrieve users with access to the sample project database

```
>mysqlaccess % % inventory -b -U root -P
mysqlaccess Version 2.06, 20 Dec 2000
.....
```


7.2.3 Create DB2 user

We use the script in Example 7-12 for user and group creation.

Example 7-12 Sample script to create DB2 users and groups

```
db2server:~ # cat db2addusr.sh
export DB2DIR='/home/db2inst1' export HOMEDIR='/home'

groupadd $2
useradd -g $2 -m -d $HOMEDIR/$1 $1
passwd $1
echo '. '${DB2DIR}.'/sql1lib/db2profile' >> $HOMEDIR/$1/.profile
```

The root user creates our user and group, as shown in Example 7-13.

Example 7-13 Creation of the user and group

```
db2server:~ # ./db2addusr.sh inventAppUser db2app1
Changing password for inventAppUser.
New password:
Re-enter new password:
Password changed
```

The instance owner, db2inst1, grants the privileges with the DB2 command that is shown in Example 7-14.

Example 7-14 Granting privileges

```
db2 => connect to invent
connect to invent

      Database Connection Information

Database server      = DB2/LINUX 9.7.0
SQL authorization ID = DB2INST1
Local database alias = INVENT

grant createin, dropin on schema db2inst1 to group db2app1
DB20000I The SQL command completed successfully.

grant select, insert, update, delete on table admin.inventory to user inventAppUser
DB20000I The SQL command completed successfully.

grant select, insert, update, delete on table admin.locations to user inventAppUser
DB20000I The SQL command completed successfully.

grant select, insert, update, delete on table admin.owners to user inventAppUser
DB20000I The SQL command completed successfully.

grant select, insert, update, delete on table admin.services to user inventAppUser
```

```
DB20000I The SQL command completed successfully.

grant select, insert, update, delete on table admin.BOSSGROUP to user inventAppUser
DB20000I The SQL command completed successfully.

grant select, insert, update, delete on table admin.EMPGROUP to user inventAppUser
DB20000I The SQL command completed successfully.

grant select, insert, update, delete on table admin.GENERALGROUP to user inventAppUser
DB20000I The SQL command completed successfully.

grant select, insert, update, delete on table admin.GROUPS to user inventAppUser
DB20000I The SQL command completed successfully.

grant select, insert, update, delete on table admin.MANAGERGROUP to user inventAppUser
DB20000I The SQL command completed successfully.

grant select, insert, update, delete on table admin.SEVERITY to user inventAppUser
DB20000I The SQL command completed successfully.

grant select, insert, update, delete on table admin.STATUS to user inventAppUser
DB20000I The SQL command completed successfully.

grant select, insert, update, delete on table admin.TECHGROUP to user inventAppUser
DB20000I The SQL command completed successfully.
```

7.2.4 Export MySQL application data

The database Data Definition Language (DDL) scripts were generated and ported to DB2 automatically with the IBM Data Movement Tool as described in 6.5, “Sample database conversion” on page 148.

The next step is to extract the MySQL application data using the Data Movement Tool. In 7.1.1, “Data porting commands and tools” on page 168 of this chapter, we discuss how to create data extraction and transfer files using the IBM Data Movement Tool. And in 7.1.2, “Differences in data formats” on page 175, we discuss how to modify the extraction scripts to extract special data. Now, we must execute these scripts and extract the data from the MySQL database. Open the Extract/Deploy tab window, clear the DDL check box, select the **Data Movement** check box, and click **Extract DDL/data**, as shown in Figure 7-3 on page 196. If you have made changes to the extraction script, be sure to select **No** when requested to recreate the output directory or `<tableName>.tables` file.

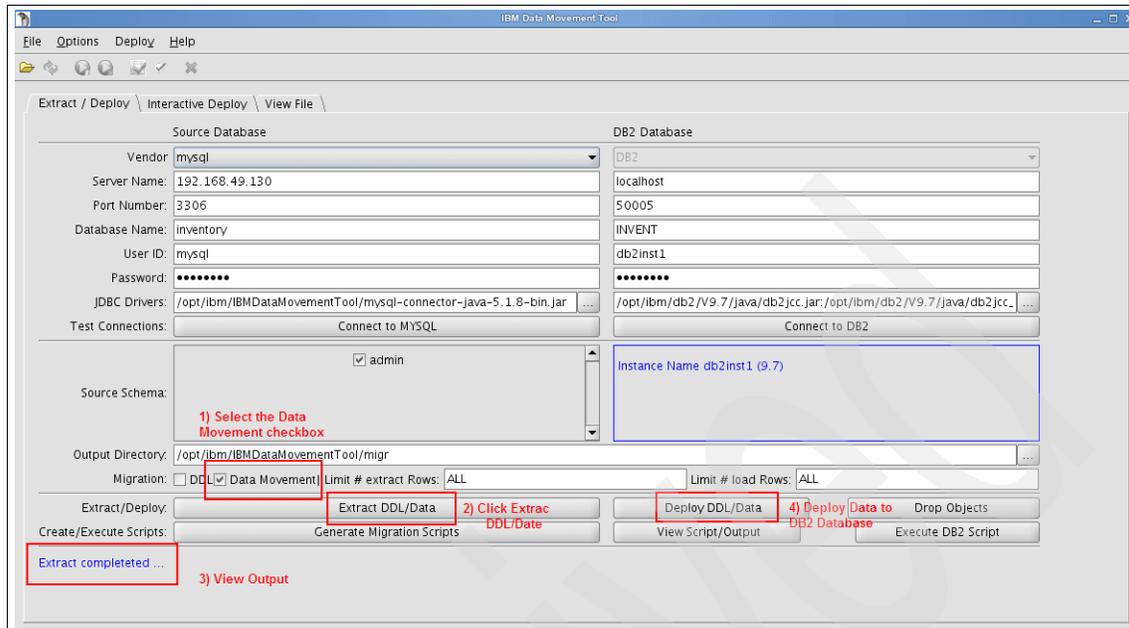


Figure 7-3 Extract/Deploy tab of the IBM Data Movement Tool

Now, you can go into your project directory to check the extracted data files. The data output files extracted from the MySQL database are located under the *<migration output directory>/data* directory. Example 7-15 shows the created scripts for our inventory scenario.

Example 7-15 IBM Data Movement Tool export files

```
db2server:/opt/ibm/IBMDataMovementTool/migr/data # ls -l
-rw-r--r-- 1 db2inst1 db2iadm1 114 Aug 28 01:03 admin_status.txt
-rw-r--r-- 1 db2inst1 db2iadm1 82 Aug 28 01:03 admin_severity.txt
-rw-r--r-- 1 db2inst1 db2iadm1 43188 Aug 28 01:03 admin_services.txt
-rw-r--r-- 1 db2inst1 db2iadm1 60137 Aug 28 01:03 admin_owners.txt
-rw-r--r-- 1 db2inst1 db2iadm1 4173 Aug 28 01:03 admin_locations.txt
-rw-r--r-- 1 db2inst1 db2iadm1 45489 Aug 28 01:03 admin_inventory.txt
-rw-r--r-- 1 db2inst1 db2iadm1 143 Aug 28 01:03 admin_groups.txt
```

Each of the files is tab-delimited, containing the data from the corresponding MySQL table. This format can be read by the DB2 LOAD command.

7.2.5 Convert MySQL application data to DB2 format

We do not have to manually convert any data in our export files, because no special data types are used in our sample database.

7.2.6 Import application data into DB2

You can import data into DB2 automatically by using the IBM Data Movement Tool. The IBM Data Movement Tool creates a script containing the DB2 LOAD commands and then executes the script. In case of any errors, you can edit the the script and run it again.

Example 7-16 shows the DB2 LOAD commands that are generated by the IBM Data Movement Tool for our sample project.

Example 7-16 DB2 LOAD commands for loading the data into the DB2 database

```
CONNECT TO INVENT;
--#SET :LOAD:ADMIN:GROUPS
LOAD FROM
"/opt/ibm/IBMDDataMovementTool/migr/data/admin_groups.txt" OF DEL
MODIFIED BY CODEPAGE=1208 COLDEL~ ANYORDER USEDEFAULTS CHARDEL"" DELPRIORITYCHAR
NOROWWARNINGS --DUMPFIL="/opt/ibm/IBMDDataMovementTool/migr/dump/admin_groups.txt"
METHOD P(1,2,3,4,5)MESSAGES "/opt/ibm/IBMDDataMovementTool/migr/msg/admin_groups.txt"
REPLACE INTO "ADMIN"."GROUPS"("GROUPNAME", "EDITUSER", "EDITGRANTUSERPERM", "EDITINVT",
"EDITSERVICE") --STATISTICS YES WITH DISTRIBUTION AND DETAILED INDEXES ALL
NONRECOVERABLE INDEXING MODE AUTOSELECT;

--#SET :LOAD:ADMIN:LOCATIONS
LOAD FROM
"/opt/ibm/IBMDDataMovementTool/migr/data/admin_locations.txt" OF DEL
MODIFIED BY CODEPAGE=1208 COLDEL~ ANYORDER USEDEFAULTS CHARDEL"" DELPRIORITYCHAR
NOROWWARNINGS --DUMPFIL="/opt/ibm/IBMDDataMovementTool/migr/dump/admin_locations.txt"
METHOD P(1,2,3,4)MESSAGES "/opt/ibm/IBMDDataMovementTool/migr/msg/admin_locations.txt"
REPLACE INTO "ADMIN"."LOCATIONS" ("ID", "ROOMNAME", "FLOORNUM", "PASSCODE")
--STATISTICS YES WITH DISTRIBUTION AND DETAILED INDEXES ALL NONRECOVERABLE
INDEXING MODE AUTOSELECT ;

--#SET :LOAD:ADMIN:SEVERITY
LOAD FROM
"/opt/ibm/IBMDDataMovementTool/migr/data/admin_severity.txt" OF DEL
MODIFIED BY CODEPAGE=1208 COLDEL~ ANYORDER USEDEFAULTS CHARDEL"" DELPRIORITYCHAR
NOROWWARNINGS --DUMPFIL="/opt/ibm/IBMDDataMovementTool/migr/dump/admin_severity.txt"
METHOD P(1,2,3,4,5) MESSAGES "/opt/ibm/IBMDDataMovementTool/migr/msg/admin_severity.txt"
REPLACE INTO "ADMIN"."SEVERITY" ("ID", "TITLE", "NOTES", "ESTDAYS", "AVGDAYS"
)--STATISTICS YES WITH DISTRIBUTION AND DETAILED INDEXES ALL NONRECOVERABLE INDEXING
MODE AUTOSELECT;
```

```

--#SET :LOAD:ADMIN:OWNERS
LOAD FROM
"/opt/ibm/IBMDataMovementTool/migr/data/admin_owners.txt" OF DEL
MODIFIED BY CODEPAGE=1208 COLDEL~ ANYORDER USEDEFAULTS CHARDEL"" DELPRIORITYCHAR
NOROWWARNINGS --DUMPFIL="/opt/ibm/IBMDataMovementTool/migr/dump/admin_owners.txt"
METHOD P (1,2,3,4,5,6,7,8,9,10,11) MESSAGES
"/opt/ibm/IBMDataMovementTool/migr/msg/admin_owners.txt" REPLACE INTO "ADMIN"."OWNERS"
("ID", "FIRSTNAME", "LASTNAME", "EMAIL", "LOCID", "CUBENUM",
"PHONENUM", "LOGINNAME", "PASSWORD", "FAXNUM", "GROUPS" ) --STATISTICS YES WITH
DISTRIBUTION AND DETAILED INDEXES ALL NONRECOVERABLE INDEXING MODE AUTOSELECT;

--#SET :LOAD:ADMIN:INVENTORY
LOAD FROM
"/opt/ibm/IBMDataMovementTool/migr/data/admin_inventory.txt" OF DEL
MODIFIED BY CODEPAGE=1208 COLDEL~ ANYORDER USEDEFAULTS CHARDEL"" DELPRIORITYCHAR
NOROWWARNINGS --DUMPFIL="/opt/ibm/IBMDataMovementTool/migr/dump/admin_inventory.txt"
METHOD P (1,2,3,4,5,6,7,8) MESSAGES
"/opt/ibm/IBMDataMovementTool/migr/msg/admin_inventory.txt"
REPLACE INTO "ADMIN"."INVENTORY"("ID", "ITEMNAME", "MANUFACTURER", "MODEL", "YEAR",
"SERIAL", "LOCID", "OWNERID")--STATISTICS YES WITH DISTRIBUTION AND DETAILED INDEXES ALL
NONRECOVERABLE INDEXING MODE AUTOSELECT ;

--#SET :LOAD:ADMIN:SERVICES
LOAD FROM
"/opt/ibm/IBMDataMovementTool/migr/data/admin_services.txt" OF DEL
MODIFIED BY CODEPAGE=1208 COLDEL~ ANYORDER USEDEFAULTS CHARDEL"" DELPRIORITYCHAR
NOROWWARNINGS --DUMPFIL="/opt/ibm/IBMDataMovementTool/migr/dump/admin_services.txt"
METHOD P (1,2,3,4,5,6,7,8,9) MESSAGES
/opt/ibm/IBMDataMovementTool/migr/msg/admin_services.txt" REPLACE INTO
"ADMIN"."SERVICES" ("ID", "INVENTID", "DESCRIPTION", "SEVERITY", "SERVICEOWNER",
"OPENDATE", "CLOSEDATE", "TARGETCLOSEDATE", "STATUS" )--STATISTICS YES WITH DISTRIBUTION
AND DETAILED INDEXES ALL NONRECOVERABLE INDEXING MODE AUTOSELECT;

--#SET :LOAD:ADMIN:STATUS
LOAD FROM
"/opt/ibm/IBMDataMovementTool/migr/data/admin_status.txt" OF DEL
MODIFIED BY CODEPAGE=1208 COLDEL~ ANYORDER USEDEFAULTS CHARDEL"" DELPRIORITYCHAR
NOROWWARNINGS --DUMPFIL="/opt/ibm/IBMDataMovementTool/migr/dump/admin_status.txt"
METHOD P (1,2,3) MESSAGES "/opt/ibm/IBMDataMovementTool/migr/msg/admin_status.txt"
REPLACE INTO "ADMIN"."STATUS" ("ID", "TITLE", "NOTES" )--STATISTICS YES WITH
DISTRIBUTION AND DETAILED INDEXES ALL NONRECOVERABLE INDEXING MODE AUTOSELECT;

TERMINATE;

```

To execute the DB2 LOAD commands, go to the Extract/Deploy tab window. Make sure that the DDL check box is still unselected and that **Data Movement** is selected. Click **Deploy DDL/Data**, as shown in Figure 7-3 on page 196. You can

also execute this load from the command line by running the db2load.sh script. If you made changes to the extraction script, be sure to select **No** when requested to recreate the output directory or <tableName>.tables file, because it overrides your changes.

After importing the data into the DB2 tables, execute the RUNSTATS command to recreate the statistics information about indexes. The query optimizer uses this statistics information. The IBM Data Movement Tool generates a custom RUNSTATS script for the new database, which is called db2runstats.sql. Example 7-17 shows the db2runstats.sql script for our sample conversion. You can run this script in the IBM Data Movement Tool GUI or command line.

Example 7-17 DB2 RUNSTATS commands for recreating the statistics information

```
CONNECT TO INVENT;  
RUNSTATS ON TABLE "ADMIN"."GROUPS" ON ALL COLUMNS WITH DISTRIBUTION  
ON ALL COLUMNS AND DETAILED INDEXES ALL ALLOW WRITE ACCESS ;  
  
RUNSTATS ON TABLE "ADMIN"."LOCATIONS" ON ALL COLUMNS WITH DISTRIBUTION  
ON ALL COLUMNS AND DETAILED INDEXES ALL ALLOW WRITE ACCESS ;  
  
RUNSTATS ON TABLE "ADMIN"."SEVERITY" ON ALL COLUMNS WITH DISTRIBUTION  
ON ALL COLUMNS AND DETAILED INDEXES ALL ALLOW WRITE ACCESS ;  
  
RUNSTATS ON TABLE "ADMIN"."OWNERS" ON ALL COLUMNS WITH DISTRIBUTION  
ON ALL COLUMNS AND DETAILED INDEXES ALL ALLOW WRITE ACCESS ;  
  
RUNSTATS ON TABLE "ADMIN"."INVENTORY" ON ALL COLUMNS WITH DISTRIBUTION  
ON ALL COLUMNS AND DETAILED INDEXES ALL ALLOW WRITE ACCESS ;  
  
RUNSTATS ON TABLE "ADMIN"."SERVICES" ON ALL COLUMNS WITH DISTRIBUTION  
ON ALL COLUMNS AND DETAILED INDEXES ALL ALLOW WRITE ACCESS ;  
  
RUNSTATS ON TABLE "ADMIN"."STATUS" ON ALL COLUMNS WITH DISTRIBUTION  
ON ALL COLUMNS AND DETAILED INDEXES ALL ALLOW WRITE ACCESS ;
```

7.2.7 Basic data checking

When the script executes, it creates a db2load.log file where all output messages are logged. Check the log file for the success of the DB2 LOAD commands. You can find this information in the log file, as shown in Example 7-18.

Example 7-18 Log file information about the DB2 LOAD command

```
Number of rows read      = 140  
Number of rows skipped  = 0  
Number of rows loaded   = 140  
Number of rows rejected = 0  
Number of rows deleted  = 0  
Number of rows committed = 140
```

```
LOAD FROM "/opt/ibm/IBMDataMovementTool/migr/data/admin_severity.txt" OF DEL MODIFIED BY
CODEPAGE=1208 COLDEL~ ANYORDER USEDEFAULTS CHARDEL"" DELPRIORITYCHAR NOROWWARNINGS METHOD P
(1,2,3,4,5) MESSAGES "/opt/ibm/IBMDataMovementTool/migr/msg/admin_severity.txt" REPLACE INTO
"ADMIN"."SEVERITY" ( "ID", "TITLE", "NOTES", "ESTDAYS", "AVGDAYS") NONRECOVERABLE INDEXING MODE
AUTOSELECT
```

```
Number of rows read      = 5
Number of rows skipped   = 0
Number of rows loaded    = 5
Number of rows rejected  = 0
Number of rows deleted   = 0
Number of rows committed = 5
```

```
LOAD FROM "/opt/ibm/IBMDataMovementTool/migr/data/admin_owners.txt" OF DEL MODIFIED BY
CODEPAGE=1208 COLDEL~ ANYORDER USEDEFAULTS CHARDEL"" DELPRIORITYCHAR NOROWWARNINGS METHOD P
(1,2,3,4,5,6,7,8,9,10,11) MESSAGES "/opt/ibm/IBMDataMovementTool/migr/msg/admin_owners.txt" REPLACE
INTO "ADMIN"."OWNERS" ( "ID", "FIRSTNAME", "LASTNAME", "EMAIL", "LOCID", "CUBENUM", "PHONENUM",
"LOGINNAME", "PASSWORD", "FAXNUM", "GROUPS" ) NONRECOVERABLE INDEXING MODE AUTOSELECT
```

```
Number of rows read      = 504
Number of rows skipped   = 0
Number of rows loaded    = 502
Number of rows rejected  = 2
Number of rows deleted   = 0
Number of rows committed = 504
```

SQL3107W There is at least one warning message in the message file.

```
LOAD FROM "/opt/ibm/IBMDataMovementTool/migr/data/admin_inventory.txt" OF DEL MODIFIED BY
CODEPAGE=1208 COLDEL~ ANYORDER USEDEFAULTS CHARDEL"" DELPRIORITYCHAR NOROWWARNINGS METHOD P
(1,2,3,4,5,6,7,8) MESSAGES "/opt/ibm/IBMDataMovementTool/migr/msg/admin_inventory.txt" REPLACE INTO
"ADMIN"."INVENTORY" ( "ID", "ITEMNAME", "MANUFACTURER", "MODEL", "YEAR", "SERIAL", "LOCID", "OWNERID"
) NONRECOVERABLE INDEXING MODE AUTOSELECT
```

```
Number of rows read      = 703
Number of rows skipped   = 0
Number of rows loaded    = 703
Number of rows rejected  = 0
Number of rows deleted   = 0
Number of rows committed = 703
```

SQL3107W There is at least one warning message in the message file.

```
LOAD FROM "/opt/ibm/IBMDataMovementTool/migr/data/admin_services.txt" OF DEL MODIFIED BY
CODEPAGE=1208 COLDEL~ ANYORDER USEDEFAULTS CHARDEL"" DELPRIORITYCHAR NOROWWARNINGS METHOD P
(1,2,3,4,5,6,7,8,9) MESSAGES "/opt/ibm/IBMDataMovementTool/migr/msg/admin_services.txt" REPLACE INTO
"ADMIN"."SERVICES" ( "ID", "INVENTID", "DESCRIPTION", "SEVERITY", "SERVICEOWNER", "OPENDATE",
"CLOSEDATE", "TARGETCLOSEDATE", "STATUS" ) NONRECOVERABLE INDEXING MODE AUTOSELECT
```

```
Number of rows read      = 808
Number of rows skipped   = 0
Number of rows loaded    = 808
Number of rows rejected  = 0
Number of rows deleted   = 0
Number of rows committed = 808
```

```
LOAD FROM "/opt/ibm/IBMDataMovementTool/migr/data/admin_status.txt" OF DEL MODIFIED BY
CODEPAGE=1208 COLDEL~ ANYORDER USEDEFAULTS CHARDEL"" DELPRIORITYCHAR NOROWWARNINGS METHOD P
(1,2,3) MESSAGES "/opt/ibm/IBMDataMovementTool/migr/msg/admin_status.txt" REPLACE INTO
"ADMIN"."STATUS" ( "ID", "TITLE", "NOTES" ) NONRECOVERABLE INDEXING MODE AUTOSELECT
```

```
Number of rows read      = 7
Number of rows skipped   = 0
Number of rows loaded    = 7
```

```
Number of rows rejected = 0
Number of rows deleted = 0
Number of rows committed = 7
```

```
TERMINATE
DB20000I The TERMINATE command completed successfully.
```

Database Connection Information

```
Database server = DB2/LINUX 9.7.0
SQL authorization ID = DB2INST1
Local database alias = INVENT
```

```
CONNECT TO INVENT
```

Database Connection Information

```
Database server = DB2/LINUX 9.7.0
SQL authorization ID = DB2INST1
Local database alias = INVENT
```

```
select count_big(*) "ADMIN.GROUPS" FROM "ADMIN"."GROUPS"
```

```
ADMIN.GROUPS
-----
6.
```

```
1 record(s) selected.
```

```
select count_big(*) "ADMIN.LOCATIONS" FROM "ADMIN"."LOCATIONS"
```

```
ADMIN.LOCATIONS
-----
140.
```

```
1 record(s) selected.
```

```
select count_big(*) "ADMIN.SEVERITY" FROM "ADMIN"."SEVERITY"
```

```
ADMIN.SEVERITY
-----
5.
```

```
1 record(s) selected.
```

```
select count_big(*) "ADMIN.OWNERS" FROM "ADMIN"."OWNERS"
```

```
ADMIN.OWNERS
-----
502.
```

```
1 record(s) selected.
```

```
select count_big(*) "ADMIN.INVENTORY" FROM "ADMIN"."INVENTORY"
```

```
ADMIN.INVENTORY
-----
703.
```

1 record(s) selected.

```
select count_big(*) "ADMIN.SERVICES" FROM "ADMIN"."SERVICES"
```

```
ADMIN.SERVICES
-----
                        808.
```

1 record(s) selected.

```
select count_big(*) "ADMIN.STATUS" FROM "ADMIN"."STATUS"
```

```
ADMIN.STATUS
-----
                        7.
```

1 record(s) selected.

```
TERMINATE
DB20000I The TERMINATE command completed successfully.
```

Database Connection Information

```
Database server      = DB2/LINUX 9.7.0
SQL authorization ID = DB2INST1
Local database alias = INVENT
```

```
TABLE_NAME          FK_CHECKED  CC_CHECKED  STATUS
-----
ADMIN.GROUPS        Y           Y           N
```

1 record(s) selected.

```
TABLE_NAME          FK_CHECKED  CC_CHECKED  STATUS
-----
ADMIN.LOCATIONS     Y           Y           N
```

1 record(s) selected.

```
TABLE_NAME          FK_CHECKED  CC_CHECKED  STATUS
-----
ADMIN.SEVERITY     Y           Y           N
```

1 record(s) selected.

```
TABLE_NAME          FK_CHECKED  CC_CHECKED  STATUS
-----
ADMIN.OWNERS       Y           Y           N
```

1 record(s) selected.

```
TABLE_NAME          FK_CHECKED  CC_CHECKED  STATUS
-----
```

```
ADMIN.INVENTORY                Y        Y        N
```

```
1 record(s) selected.
```

```
TABLE_NAME                      FK_CHECKED  CC_CHECKED  STATUS
-----
ADMIN.SERVICES                   Y           Y           N
```

```
1 record(s) selected.
```

```
TABLE_NAME                      FK_CHECKED  CC_CHECKED  STATUS
-----
ADMIN.STATUS                     Y           Y           N
```

```
1 record(s) selected.
```

```
DB20000I The TERMINATE command completed successfully.
```

Make sure that the number of rows read equals the number of rows committed, which also needs to equal the number of records in the MySQL source table. Example 7-19 shows the MySQL command for the record count.

Example 7-19 Retrieving the number of records from MySQL

```
mysql> select count(*) from groups;
+-----+
| count(*) |
+-----+
|         6 |
+-----+
1 row in set (0.04 sec)

mysql> select count(*) from inventory;
+-----+
| count(*) |
+-----+
|        703 |
+-----+
1 row in set (0.00 sec)

mysql> select count(*) from locations;
+-----+
| count(*) |
+-----+
|        140 |
+-----+
1 row in set (0.00 sec)

mysql> select count(*) from owners;
+-----+
| count(*) |
+-----+
|        504 |
+-----+
1 row in set (0.01 sec)

mysql> select count(*) from services;
+-----+
| count(*) |
```

```

+-----+
|      808 |
+-----+
1 row in set (0.00 sec)

mysql> select count(*) from status;
+-----+
| count(*) |
+-----+
|         7 |
+-----+
1 row in set (0.00 sec)

mysql> select count(*) from severity;
+-----+
| count(*) |
+-----+
|         5 |
+-----+
1 row in set (0.00 sec)

```

After you have checked that all the records were loaded into DB2, check the sample data in each ported table. Pay attention to ensure that the values are correct, especially if you have any time values or decimal values. Example 7-20 shows the table content checking.

Example 7-20 Sample data of MySQL data

```

mysql> select * from services where id = 20;
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | inventID | description | severity | serviceOwner | openDate | closeDate |
targetCloseDate | status |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 20 |      36 | test       |      1 |      108 | 2009-01-18 | 2009-01-20 |
2009-01-19      |      7 |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

```

db2inst1@db2server:~> db2 "select * from admin.services where id = 20"

ID INVENTID DESCRIPTION SEVERITY SERVICEOWNER OPENDATE CLOSEDATE TARGETCLOSEDATE
STATUS
-----
20      36 test       1      108 2009-01-18 2009-01-20 2009-01-19
7

1 record(s) selected.

```

Application conversion

The task of converting an application, its databases, and the associated data most often requires significant resources and commitment. Simultaneously to the meticulous planning of the porting project as a whole, it is important to assess the issues that might affect the highest level of resources. In many porting projects, evaluating the issues that might affect your resources is part of application porting. This chapter attempts to provide you with important considerations for the following areas:

- ▶ Differences in SQL Data Manipulation Language (DML), built-in functions, and SQL semantics
- ▶ Conversion of the application source, application programming interfaces (APIs), and condition handling
- ▶ Internals of the Database Management System that might affect the conversion, such as locking, isolation levels, transaction logging, and national language support

8.1 Data Manipulation Language differences and similarities

Modifying an application to work with DB2 is an important task during the conversion process. While a large portion of this step might encompass changing code to work with a separate development environment, it is likely that you will spend additional time testing the resulting code.

8.1.1 SELECT syntax

In this section, we focus on the SELECT statement syntax as it is supported in MySQL, and we attempt to show how MySQL extensions to the SQL standard might be implemented in DB2. Example 8-1 shows the MySQL syntax for the SELECT statement. Then, we discuss individual keywords.

Example 8-1 MySQL SELECT syntax

```
SELECT [STRAIGHT_JOIN]
[SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
[SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS] [HIGH_PRIORITY]
[DISTINCT | DISTINCTROW | ALL]
select_expression,...
[INTO {OUTFILE | DUMPFILE} 'file_name' export_options]
[FROM table_references
[WHERE where_definition]
[GROUP BY {unsigned_integer | col_name | formula} [ASC | DESC], ... [WITH
ROLLUP]]
[HAVING where_definition]
[ORDER BY {unsigned_integer | col_name | formula} [ASC | DESC], ...] [LIMIT
[offset,] row_count | row_count OFFSET offset]
[PROCEDURE procedure_name(argument_list)]
[FOR UPDATE | LOCK IN SHARE MODE]]
```

The STRAIGHT_JOIN keyword forces the MySQL optimizer to join tables in the order that is specified. In DB2, the join order is always determined by the optimizer. The optimizer choices can be limited by changing the default query optimization class to a lower level using SET CURRENT QUERY OPTIMIZATION. However, changing this class does not guarantee that the optimizer will evaluate the join in the order stated within the SQL statement, because the DB2 cost-based optimizer usually chooses the best access path for a given query. For additional information, see 10.5.5, “SQL execution plan” on page 366.

Options prefixed with SQL are MySQL-specific and do not require a DB2 equivalent.

The DISTINCTROW keyword is a synonym for the DISTINCT clause that is supported by DB2.

The INTO {OUTFILE | DUMPFILE} 'file_name' export_options selection allows you to write output data to a file without invoking the mysqldump utility. The DB2 command line processor allows you to direct the output of any SELECT statement to an operating system file.

The LIMIT [offset,] row_count | row_count OFFSET offset keyword translates to FETCH FIRST *n* ROWS ONLY in DB2. Implement an offset to retrieve rows through the WHERE clause if possible.

With the SQL_SMALL_RESULT or SQL_BIG_RESULT commands, the query developer can hint to the SQL optimizer the size of the expected result set, influencing the optimizer access strategy. Example 8-2 shows how the optimizer hint works.

Example 8-2 MySQL SELECT with optimizer hint

```
mysql> SELECT sql_small_result * FROM t1;
```

DB2 has a similar operator to guide SQL optimizer decisions with a different syntax, as shown in Example 8-3.

Example 8-3 DB2 SELECT with optimizer hint

```
db2 => SELECT * FROM t1 OPTIMIZE FOR 2 ROWS
DB20000I The SQL command completed successfully.
```

8.1.2 JOIN syntax

Join capabilities for commercial and industrial strength database management systems are extremely important. MySQL supports the linguistic elements for JOIN, as shown in Example 8-4. We discuss various aspects of JOINS next.

Example 8-4 MySQL JOIN Syntax

```
table_reference, table_reference
table_reference [INNER | CROSS] JOIN table_reference [join_condition]
table_reference STRAIGHT_JOIN table_reference
table_reference LEFT [OUTER] JOIN table_reference [join_condition]
table_reference NATURAL [LEFT [OUTER]] JOIN table_reference
{OJ table_reference LEFT OUTER JOIN table_reference ON conditional_expr }
```

```
table_reference RIGHT [OUTER] JOIN table_reference [join_condition]
table_reference NATURAL [RIGHT [OUTER]] JOIN table_reference
```

Where `table_reference` is defined as:

```
table_name [[AS] alias] [[USE INDEX (key_list)] | [IGNORE INDEX (key_list)] |
[FORCE IN and join_condition is defined as:
ON conditional_expr |
USING (column_list)
```

The `STRAIGHT_JOIN` keywords force the MySQL optimizer to join tables in the order that they are specified. In DB2, the join order is always determined by the optimizer. You can limit the optimizer choices by changing the default query optimization class by using the `SET CURRENT QUERY OPTIMIZATION` command.

A `NATURAL` join, as its name implies, can be invoked when two or more tables share exactly the same columns needed for a successful equijoin. It is semantically equivalent to DB2 `INNER JOIN` or `LEFT OUTER JOIN` with the respective join criteria specified in the `ON` clause.

According to the SQL ANSI standard when you must join tables that share more than one column naturally, you must use the `JOIN ... USING` syntax. You can compose an equivalent join using the DB2-supported join syntax in the `ON` clause.

Cartesian products do happen from time to time, usually as the result of an equijoin condition that has been missed in a query using DB2 syntax, even though the developers of applications and database queries spend a significant amount of time trying to avoid them. However, one of the advantages of the `CROSS JOIN` syntax is that a specific keyword is required to create a Cartesian product. Therefore, when the `CROSS JOIN` syntax is used in your conversion project, you can code a regular join in DB2 with a no join condition in the `WHERE` clause.

8.1.3 UNION syntax

MySQL support for the `UNION` feature (shown in Example 8-5) is extremely similar to the DB2 syntax and, therefore, does not require additional discussion.

Example 8-5 UNION syntax in MySQL and DB2

```
SELECT ...
UNION [ALL | DISTINCT] SELECT ...
[UNION [ALL | DISTINCT] SELECT ...]
```

8.1.4 Subquery syntax

A *subquery* is a SELECT statement within another query or statement. Therefore, subqueries can be found in other SELECT statements - either in the column list, the WHERE clause, or the HAVING clause, and in INSERT, UPDATE, and DELETE statements. Example 8-6 shows a sequence of subqueries in a DELETE statement.

Example 8-6 Example of subqueries in a DELETE statement

```
DELETE FROM t1
WHERE col1 > ANY
  (SELECT COUNT(*) FROM t2
   WHERE NOT EXISTS
     (SELECT col3 FROM t3
      WHERE col3 =
        (SELECT col4 FROM t4 UNION SELECT 1 FROM
         (SELECT col5 FROM t5) AS t5 )));
```

MySQL provides similar subquery implementation to that of DB2. We found nothing to cause serious issues when moving from MySQL to DB2.

8.1.5 Grouping, having, and ordering

All ANSI SQL 92 standard grouping functions available in MySQL Version 5.1 are also available in DB2. In general, we found no significant differences in the areas of grouping, having, and ordering. However, beyond the SQL 92 standard, DB2 provides interesting functionality that might enhance the ported application significantly. Refer to the DB2 manual *SQL Reference, Volume 1*, SC10-4249. Table 8-1 on page 210 contains a partial list of the differences in the two databases and provides conversion examples. For a full list of MySQL grouping functions and the DB2 equivalent, refer to A.1, “Grouping related functions” on page 400.

Table 8-1 MySQL and DB2 grouping related function

MySQL function	MYSQL example	DB2 function	DB2 example	Notes
AVG([DISTINCT] expression)	mysql> SELECT a, AVG(b) FROM t1 GROUP BY a	AVG ([DISTINCT ALL] expression)	db2 " SELECT a, AVG(b) FROM t1 GROUP BY a"	Returns the average set of numbers
COUNT([DISTINCT] expression, expression,...)	mysql> SELECT a, COUNT(b) FROM t1 GROUP BY a	COUNT([DISTINCT ALL] expression). DB2 allows only one expression: Use CONCAT for character data type or CHAR and CONCAT on numeric data types	db2 " SELECT a, COUNT(b) FROM t1 GROUP BY a"	Returns the number of rows or values in a set of rows or values.
MAX ([DISTINCT] expression)	mysql> SELECT a, MAX(b) FROM t1 GROUP BY a	MAX ([DISTINCT ALL] expression)	db2 "SELECT a, MAX(b) FROM t1 GROUP BY a"	Returns the maximum value in a set of values.
SUM([DISTINCT] expression)	mysql> SELECT a, SUM(b) FROM t1 GROUP BY a	SUM([DISTINCT ALL] expression)	db2 " SELECT a, sum(b) FROM t1 GROUP BY a"	Returns the sum of a set of numbers.
GROUP BY on alias	mysql> SELECT a as x FROM a GROUP BY x;	Use column name for grouping	db2 " SELECT a FROM t1 GROUP BY a"	Groups data by the alias name provided.
GROUP BY on position	mysql> SELECT a FROM t1 GROUP BY 1	Use column name for grouping	db2 " SELECT a FROM t1 GROUP BY a"	Groups data by the position provided.
HAVING on alias	mysql> SELECT a as x FROM t1 GROUP BY a HAVING x > 0	Use column name in having clause	db2 " SELECT a FROM t1 GROUP BY a HAVING a > 0"	Groups data meeting the HAVING expression.

8.1.6 Strings

Unless you start MySQL in ANSI mode (using `mysqld --ansi`), MySQL behaves differently than DB2. As Example 8-7 on page 211 illustrates, MySQL accepts single, as well as double quotation marks as a string delimiter when started in default mode.

Example 8-7 MySQL string handling

```
mysql> select 'redbook', "'redbook'", '"redbook"', 'red''book';
+-----+-----+-----+-----+
| redbook| "redbook"| "'redbook'"| red'book|
+-----+-----+-----+-----+
| redbook| "redbook"| "'redbook'"| red'book|
+-----+-----+-----+-----+
1 row in set (0.00 sec)
mysql> select "redbook", "'redbook'", '"redbook"', "red""book";
+-----+-----+-----+-----+
| redbook | 'redbook'| '"redbook"'| red"book|
+-----+-----+-----+-----+
| redbook | 'redbook'| '"redbook"'| red"book |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

DB2 is designed and implemented according to the ANSI standard and therefore accepts single quotation marks as a string delimiter. Double quotation marks are used in DB2 for delimiting SQL identifiers. Example 8-8 shows how DB2 handles strings. You achieve similar results when MySQL runs in ANSI mode.

Example 8-8 DB2 string handling

```
db2 => select "redbook", "'redbook'", '"redbook"', "red""book" from t1
SQL0206N "redbook" is not valid in the context where it is used.
SQLSTATE=42703

db2 => select 'redbook', "'redbook'", '"redbook"', 'red''book' from t1
1 2 3 4
-----
redbook "redbook" "'redbook'" red'book
```

Table 8-2 provides an overview of a few of the MySQL string related functions, and how these can be converted to DB2. For a full list of MySQL string functions and the DB2 equivalent, refer to A.2, “String functions” on page 402.

Table 8-2 MySQL and DB2 string-related function

MySQL function	MYSQL example	DB2 function	DB2 example	Notes
ASCII(string)	mysql> SELECT ascii('a'); +-----+ ascii('a') +-----+ 97 +-----+ 1 row in set (0.00 sec)	ASCII(string)	db2 "VALUES ascii('a') " 1 ----- 97 1 record(s) selected	Returns ASCII code value

CHAR_LENGTH(string) / CHARACTER_LENGTH(string)	mysql> SELECT CHAR_LENGTH('Orange'); +-----+ CHAR_LENGTH('Orange') +-----+ 6 +-----+ 1 row in set (0.00 sec)	CHARACTER_LENGTH(string, CODEUNITS32 OCTETS),/CHAR_LENGTH(string, CODEUNITS32 OCTETS)	db2 "VALUES CHAR_LENGTH('Orange', CODEUNITS32)" 1 ----- 6 1 record(s) selected.	Returns the number of bytes for expression. For double byte character set (DBCS) the number of DBCS characters is returned
CONCAT_WS(separator, string,...)	mysql> SELECT CONCAT_WS('-', firstname, lastname, loginname) as FULLNAME from owners where id = 501; +-----+ FULLNAME +-----+ Angela-Carlson-acarlson +-----+ 1 row in set (0.01 sec)	Use to implement CONCAT(list)	db2 "SELECT (firstName '-' lastName '-' loginName) as fullName from admin.owners where id = 501" FULLNAME ----- Angela-Carlson-acarlson 1 record(s) selected.	Returns the concatenation of string arguments with separator
FORMAT(double, integer)	FORMAT: select format(1234.5555, 2) returns 1,234.56	No equivalent function. Implement using UDF	Refer to UDF B.2, "Sample code for FORMAT function" on page 415	Returns the rounded string
INSERT(string, position, length, substring)	mysql> SELECT INSERT('original', 2, 2, 'NEW'); +-----+ INSERT('original', 2, 2, 'NEW') +-----+ oNEWginal +-----+ 1 row in set (0.00 sec)	INSERT(string, position, length, substring)	db2 "VALUES INSERT('original', 2, 2, 'NEW')" 1 ----- oNEWginal 1 record(s) selected.	Inserts a string into an existing string
LPAD(string, length, substring) / RPAD(string, length, substring)	mysql> SELECT LPAD('TEST',6,'!!!'); +-----+ LPAD('TEST',6,'!!!') +-----+ !!!TEST +-----+ 1 row in set (0.00 sec)	No equivalent function. Implement using UDF	Refer to UDF B.3, "Sample code for RPAD and LPAD functions" on page 416	Returns the a string of the given length, if the length is longer then the string the substring characters will be added to the left or right end.

REPLACE(string1, string2, string3)	mysql> SELECT REPLACE ('DINING', 'N', 'VID'); +-----+ REPLACE ('DINING', 'N', 'VID') +-----+ DIVIDIVIDG +-----+ 1 row in set (0.00 sec)	REPLACE(string1, string2, string3)	db2 "VALUES REPLACE ('DINING', 'N', 'VID') " 1 ----- DIVIDIVIDG 1 record(s) selected.	Returns as sting with all occurrences of <i>string2</i> in <i>string1</i> with <i>string3</i>
SUBSTRING (string, position, length) / MID (string, position, length) / SUBSTR(string, position, [length])	mysql> select substring('abcdef', 2, 3); +-----+ substring('abcdef', 2, 3) +-----+ bcd +-----+ 1 row in set (0.00 sec)	SUBSTR(string, position, length)	db2 "VALUES('abcdef', 2, 3)" 1 --- bcd 1 record(s) selected.	Returns a substring of a string.
TRIM([Both Leading trailing [substring] FROM] string)	mysql> select trim(trailing from trim(LEADING FROM ' abc ')) as OUTPUT; +-----+ OUTPUT +-----+ abc +-----+ 1 row in set (0.00 sec)	TRIM([Both Leading trailing [substring] FROM] string)	db2 "VALUES trim(trailing from trim(LEADING FROM ' abc '))" OUTPUT ----- abc 1 record(s) selected.	Removes blanks or occurrences of another specified character from the end or the beginning of a string expression

8.1.7 Implicit casting of data types

DB2 9.7 introduces support for *implicit casting*. Implicit casting is an automatic conversion of data from one data type to another data type based on an implied set of conversion rules, which occurs in support of weak typing.

Prior to Version 9.7, strong typing was used during comparisons and assignments. Strong typing requires matching data types, which means that you must explicitly convert one or both data types to a common data type prior to performing comparisons or assignments.

In Version 9.7, the rules that are used during comparisons and assignments have been relaxed. If reasonable interpretation can be made between two mismatched data types, implicit casting is used to perform comparisons or assignments. Implicit casting is also supported during function resolution. When the data types of the arguments of a function being invoked cannot be promoted to the data types of the parameters of the selected function, the data types of the arguments are implicitly cast to those of the parameters.

Implicit casting reduces the number of SQL statements that you must modify when enabling applications that run on data servers other than DB2 data servers to run on DB2 9.7. In many cases, you no longer have to explicitly cast data types when comparing or assigning values with mismatched data types.

Example 8-9 shows how MySQL implicitly casts the character value 5 to an integer value to resolve the query.

Example 8-9 MySQL performs implicit data type casting

```
mysql> create table t1 (c1 int);
Query OK, 0 rows affected (0.08 sec)
```

```
mysql> insert into t1 values(5);
Query OK, 1 row affected (0.02 sec)
```

```
mysql> select * from t1 where c1='5';
+-----+
| c1 |
+-----+
| 5 |
+-----+
1 row in set (0.00 sec)
```

As of DB2 9.7, DB2 now supports an implicit casting of incompatible data types. Example 8-10 shows implicit casting of a character value in DB2 9.7.

Example 8-10 DB2 9.7 implicit data type casting

```
db2 => create table t1 (c1 int)
DB20000I The SQL command completed successfully.
db2 => insert into t1 values(5)
DB20000I The SQL command completed successfully.
db2 => commit
DB20000I The SQL command completed successfully.
db2 => select * from t1 where c1='5'
```

```
C1
-----
      5
```

```
1 record(s) selected.
```

DB2 versions prior to 9.7 require explicit casting of the character value to an integer value, as illustrated in Example 8-11 on page 215.

Example 8-11 DB2 9.5 and prior versions require explicit casting

```
db2 => select * from t1 where c1 = '5'
SQL0401N The data types of the operands for the operation "=" are not
compatible.  SQLSTATE=42818
db2 => select * from t1 where c1 = cast ('5' as int)
```

```
C1
-----
5
1 record(s) selected.
```

Example 8-12 illustrates how MySQL implicitly casts numeric values and DATE, TIME, or TIMESTAMP values to strings when concatenated.

Example 8-12 MySQL implicit casting using concatenation for strings and DATE

```
mysql> select concat('ITS0SJ',1234) from t1;
+-----+
| stringcol |
+-----+
| ITS0SJ1234|
+-----+
1 row in set (0.02 sec)

mysql> select concat('ITS0SJ',current_date) as stringdate from t1;
+-----+
| stringdate |
+-----+
| ITS0SJ2009-08-31 |
+-----+
1 row in set (0.01 sec)
```

Example 8-13 illustrates how DB2 9.7 implicitly casts numeric values, as well as DATE, TIME, or TIMESTAMP values, to strings when concatenated.

Example 8-13 DB2 9.7 casting character strings and DATE implicitly

```
db2 => select concat('ITS0SJ',1234) from t1

1
-----
ITS0SJ1234

1 record(s) selected.

db2 => select concat('ITS0SJ',current_date) as stringdate from t1

STRINGDATE
```

ITSOSJ08/31/2009

1 record(s) selected.

DB2 9.5 and prior versions require compatible arguments for the concatenation built-in functions, as shown in Example 8-14. If the arguments are incompatible, for example, calling a function with a character data type argument using a numeric data type, the concatenation will fail with the error "SQL0440N No authorized routine named "CONCAT" of type "FUNCTION" having compatible arguments was found."

Example 8-14 DB2 9.5 and prior versions casting character strings and DATE explicitly

```
db2 => select concat('ITSOSJ',1234) from t1
SQL0440N  No authorized routine named "CONCAT" of type "FUNCTION" having
compatible arguments was found.  SQLSTATE=42884
db2 => select concat('ITSOSJ','1234') as stringcol from t1
```

```
STRINGCOL
-----
ITSOSJ1234
```

1 record(s) selected.

```
db2 => select concat('ITSOSJ', current date) as stringdate from t1
SQL0440N  No authorized routine named "CONCAT" of type "FUNCTION" having
compatible arguments was found.  SQLSTATE=42884
```

```
db2 => select concat('ITSOSJ',CAST(current date as char(20))) as stringdate
from t1
```

```
STRINGDATE
-----
ITSOSJ01/23/2004
1 record(s) selected.
```

8.1.8 String concatenation and NULL values

The ANSI92 standard for concatenation of multiple strings is stated to be executed by the SQL || operator. MySQL does not support the SQL "||" operator as concatenation, because this operator is used to represent an OR operator. MySQL uses the CONCAT(*string1*, *string2*, *string3*, *string4*, ...) function to concatenate strings, as shown in Example 8-15 on page 217.

Example 8-15 MySQL concatenation of strings

```
mysql> SELECT CONCAT('This ', 'is ', 'an ', 'example.') ;
+-----+
| CONCAT('This ', 'is ', 'an ', 'example.') |
+-----+
| This is an example.                        |
+-----+
1 row in set (0.00 sec)
```

DB2 follows the ANSI92 standard for concatenation of multiple strings. DB2 also has a `CONCAT(string1, string2)`, which can be used for concatenation of two strings. Example 8-16 shows how DB2 handles concatenating strings.

Example 8-16 DB2 concatenation of strings

```
db2 => VALUES CONCAT('This ', 'is ', 'an ', 'example.')
SQL0440N No authorized routine named "CONCAT" of type "FUNCTION" having
compatible arguments was found. SQLSTATE=42884
db2 => VALUES CONCAT('This is ', 'an example.')
```

```
1
-----
This is an example.

1 record(s) selected.
```

```
db2 => VALUES ('This ' || 'is ' || 'an ' || 'example.')
```

```
1
-----
This is an example.

1 record(s) selected.
```

The ANSI92 standard states that if you concatenate a NULL value onto an existing string, the result set is NULL. Example 8-17 shows you the behavior of MySQL.

Example 8-17 MySQL concatenation of strings and NULL values

```
mysql> create table t2 (col1 char(2));
Query OK, 0 rows affected (0.03 sec)

mysql> insert into t2 values(NULL);
Query OK, 1 row affected (0.07 sec)
mysql> select concat('abc',col1) as nullstring from t2;
+-----+
| nullstring |
```

```

+-----+
| NULL          |
+-----+
1 row in set (0.05 sec)
mysql> select concat('abc', coalesce(col1,'')) as nullstring from t2;
+-----+
| nullstring    |
+-----+
| abc          |
+-----+ 1 row in set (0.00 sec)

```

As shown, MySQL behaves as ANSI-92 compliant and, therefore, gives you the same result sets as Example 8-18 for DB2.

Example 8-18 DB2 string and NULL concatenation

```

db2 => create table t2 (col1 char(2))
DB20000I The SQL command completed successfully.
db2 => insert into t2 values(NULL)
DB20000I The SQL command completed successfully.
db2 => select concat('abc', col1) as nullstring from t2

NULLSTRING
-----
-
1 record(s) selected.

db2 => select concat('abc', coalesce(col1,'')) as nullstring from t2

NULLSTRING
-----
abc
1 record(s) selected.

```

8.1.9 Record deletion

Much like other competitive relational database management systems, MySQL uses the TRUNCATE statement. TRUNCATE is used to delete all rows from a table when there is no need to recover the deleted records. Therefore, there is no ROLLBACK where a TRUNCATE is used. Example 8-19 on page 219 shows using TRUNCATE remove all rows, which does not allow a rollback of the deleted rows.

Example 8-19 Using MySQL TRUNCATE to delete all records in a file

```
mysql> select * from t1;
+-----+
| col1 |
+-----+
|    5 |
|   10 |
+-----+
2 rows in set (0.02 sec)

mysql> truncate table t1;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from t1;
Empty set (0.00 sec)

mysql> rollback ;
ERROR 1196: Warning: Some non-transactional changed tables couldn't be rolled
back
mysql> select * from t1;
Empty set (0.00 sec)
```

The TRUNCATE option is primarily used to delete all records quickly from a table when no recovery of the deleted rows is required. As of DB2 9.5, you can enable the support of the TRUNCATE statement using the *DB2_COMPATIBILITY_VECTOR* registry variable.

The *DB2_COMPATIBILITY_VECTOR* registry variable is used to enable one or more DB2 compatibility features introduced since DB2 Version 9.5.

These features ease the task of converting applications written for other relational database vendors to DB2 Version 9.5 or later. This DB2 registry variable is represented as a hexadecimal value, and each bit in the variable enables one of the DB2 compatibility features. To enable the TRUNCATE statement, set the *DB2_COMPATIBILITY_VECTOR* registry variable to 8. Example 8-20 on page 220 shows the syntax to set the *DB2_COMPATIBILITY_VECTOR* registry variable and execute the TRUNCATE command. You can obtain more information about the *DB2_COMPATIBILITY_VECTOR* registry variable at the IBM Information Center at this Web site:

<http://publib.boulder.ibm.com/infocenter/db21uw/v9r7/index.jsp>

Example 8-20 DB2 TRUNCATE to delete all records in a file

```
db2inst1@db2rules:~> db2stop force
08/31/2009 16:09:09    0    0    SQL1064N  DB2STOP processing was successful.
SQL1064N  DB2STOP processing was successful.

db2inst1@db2rules:~> db2set DB2_COMPATIBILITY_VECTOR=8

db2inst1@db2rules:~> db2set
DB2_COMPATIBILITY_VECTOR=8
DB2RSHCMD=/usr/bin/ssh
DB2COMM=tcpip

db2inst1@db2rules:~> db2start
08/31/2009 16:13:19    0    0    SQL1063N  DB2START processing was successful.
SQL1063N  DB2START processing was successful.

db2inst1@db2rules:~> db2 create db itsodb
DB20000I  The CREATE DATABASE command completed successfully.

db2inst1@db2rules:~> db2 connect to itsodb

      Database Connection Information

Database server          = DB2/LINUX 9.7.0
SQL authorization ID    = DB2INST1
Local database alias    = ISSODB

db2inst1@db2rules:~> db2 "create table t1 (c1 int) "
DB20000I  The SQL command completed successfully.

db2inst1@db2rules:~> db2 "insert into t1 values(5)"
DB20000I  The SQL command completed successfully.
db2inst1@db2rules:~> db2 "insert into t1 values(10)"
DB20000I  The SQL command completed successfully.
db2inst1@db2rules:~> db2 "insert into t1 values(15)"
DB20000I  The SQL command completed successfully.
db2inst1@db2rules:~> db2 "insert into t1 values(20)"
DB20000I  The SQL command completed successfully.

db2inst1@db2rules:~> db2 "SELECT * from t1"

C1
-----
      5
     10
     15
     20

      4 record(s) selected.

db2inst1@db2rules:~> db2 TRUNCATE TABLE t1 IMMEDIATE
DB20000I  The SQL command completed successfully.
```

```
db2inst1@db2rules:~> db2 "SELECT * from t1"
```

```
C1
```

```
-----
```

```
0 record(s) selected.
```

You can also turn off logging with the following ALTER TABLE statement to achieve a similar result.

```
ALTER TABLE <tablename> ACTIVATE NOT LOGGED INITIALLY WITH EMPTY TABLE
```

8.1.10 Built-in functions and operators

We have already discussed string and aggregate functions. In this section, we complete the discussion of built-in functions, including date/time and numeric functions. We also discuss the differences between various operators.

A *function* is an operation that is denoted by a function name followed by a pair of parentheses enclosing the specification of arguments, if arguments are required.

The following group of tables is not an exhaustive list of differences in built-in functions and operators between MySQL Version 5.1 and DB2 9.7. The listed built-in functions attempt to highlight the differences and mapping functionalities between the two database management systems.

Numeric functions comparison

Many of the MySQL built-in numeric functions port easily to DB2 functions. A.3, “Numeric functions” on page 408 lists several functions and examples specific to manipulating numbers.

Functions converting date and time

Working with date and time functions and operators reveals many cases where the implementation differences exist between the various DBMSs. A.4, “Date and time functions” on page 409 lists date and time-related functions and examples.

Comparing operators and more functions

There are many functions, as listed in A.5, “Comparing operators and other functions” on page 410, where syntax or implementation differ between MySQL Version 5.1 and DB2 9.7. We summarize these differences, display code snippets for several selected functions, and show ways to convert these functions from MySQL to DB2. In many cases, the DB2 CASE expression is extremely helpful when function mapping is required.

8.2 Application source conversion

Every programming language provides various commands and possibilities to program a single functionality. It is almost impossible to provide a conversion example for every possible programming approach. Therefore, in this section, we discuss the most important concepts used to access a database:

- ▶ Connecting to a database
- ▶ Using query statements
- ▶ Disconnecting from a database

8.2.1 Converting MySQL Perl applications to DB2

Perl uses the *DBD::mysql* interface driver as a plug-in for the Database Independent (DBI) set of modules. DBI provides a common Perl API for all database connections. For more information about the *DBD::mysql* driver, visit this Web site:

<http://search.cpan.org/~capttofu/DBD-mysql-4.012/lib/DBD/mysql.pm>

You can also use the DBI interface to access DB2 using the *DBD::DB2* driver. You can obtain information about the DBI interface and the *DBD::DB2* driver, as well as installation instructions, at the following Web sites:

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

<http://search.cpan.org/~ibmtordb2/DBD-DB2-1.74/DB2.pod>

<http://www.perl.com/CPAN/modules/by-module/DBD/>

<http://aspn.activestate.com/ASPN/Modules/>

Figure 8-1 illustrates how the Perl interfaces and pluggable drivers connect to MySQL and DB2 databases.

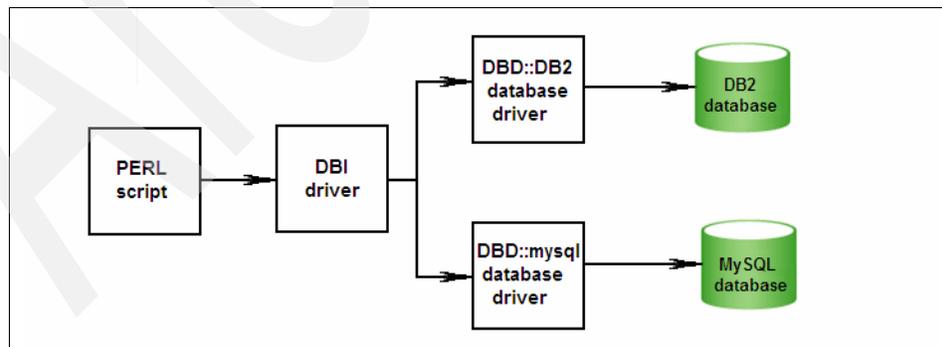


Figure 8-1 Perl interface and pluggable drivers

Converting DBI interface-supported code

Applications using the DBI interface to connect to MySQL generally can be adapted to DB2 by simply changing the database driver from `DBD::mysql` to `DBD::DB2` within the code.

Connecting to a database

Use the following code within your Perl application to connect to a MySQL database:

```
$dsn= "dbi:mysql:$database:$host:$port";
$dbh = DBI ->Connect($dsn, $user, $password);
```

The connection statement consists of the data source name, user ID, and password. The data source name consists of the vendor-specific database driver, the database name, the host name, and the port. Optionally, you can omit the host name and port from the data source name. Example 8-21 shows the connect syntax to a MySQL database. For simplicity reasons, we do not include the error handling in following examples.

Example 8-21 MySQL database connection with the Mysql.pm interface

```
use DBI;
use DBD::mysql;
my $host="localhost";
my $port="3306";
my $database="inventory";
my $user="mysql";
my $password="password";

my $dns="DBI:mysql:database=$database;host=$host;port=$port";

$dbh =
DBI->connect("$dns", "$user", "$password");
```

Connection to a DB2 database is similar. Instead of the `DBD::Mysql` interface, you must define the `DBD::DB2` interface. If DB2 constants are used (for example, `SQL_MODE_READ_ONLY` in the connection attributes, and others), you must announce the use of DB2 constants to the Perl interpreter by including “use `DBD::DB2::Constants`” in the Perl program (see Example 8-22).

Example 8-22 DB2 database connection with DBI interface

```
use DBI;
use DBI;
use DBD::DB2;
use DBD::DB2::Constants;
my $host="localhost";
my $port="50005";
my $database="invent";
```

```

my $user="db2inst1";
my $password="password";

my $dns = "dbi:DB2:DATABASE=$database; HOSTNAME=$host; PORT=$port;
PROTOCOL=TCPIP; UID=$user; PWD=$password;";

$dbh = DBI->connect("$dns", "$user", "$password");

```

Because DB2 is more powerful than MySQL, the DB2 connect statement might require a fourth argument `\%attr`, which contains the connection attributes, as shown in Example 8-23.

```
$dbh=DBI->connect($data_source, $user,$password, \%attr);
```

Example 8-23 DB2 specific connection attributes

db2_access_mode	SQLL_MODE_READ_ONLY or SQL_MODE_READ_WRITE
db2_clischema	Character string
db2_close_behavior	SQL_CC_NO_RELEASE or SQL_CC_RELEASE
db2_connect_node	Integer (must be set in DBI->connect method; it cannot be modified afterwards)
db2_set_schema	Character string
db2_db2estimate	Integer
db2_db2explain	One of: SQL_DB2EXPLAIN_OFF SQL_DB2EXPLAIN_SNAPSHOT_ON SQL_DB2EXPLAIN_MODE_ON SQL_DB2EXPLAIN_SNAPSHOT_MODE_ON
db2_info_acctstr	Character string
db2_info_applname	Character string
db2_info_programname	Character string
db2_info_userid	Character string
db2_info_wrkstnname	Character string
db2_longdata_compat	Boolean
db2_quiet_mode	Integer
db2_sqlerrp	Character string (read only)
db2_txn_isolation	One of the following: SQL_TXN_READ_UNCOMMITTED SQL_TXN_READ_COMMITTED SQL_TXN_REPEATABLE_READ SQL_TXN_SERIALIZABLE SQL_TXN_NOCOMMIT

The DBI connect statement returns a database handle when a successful connection is established. Otherwise, the value *undef* is returned. All further communication with the database server takes place through this object.

SELECT query statements

SELECT statements are handled in the same way for the two pluggable interfaces. The only conversion that is required is modification of the SQL syntax, if necessary, within the prepared statement. Example 8-24 shows how the two interfaces use the SELECT statement.

Example 8-24 SELECT statement that is used with the DBI interface

```
$sqlStatement = "SELECT * FROM owners;";  
$( $sqlStatement );  
$sth->execute();  
@arr = $sth->fetchrow;
```

Calling INSERT, UPDATE, or DELETE statements

Generally, You can use the same functions for INSERT, UPDATE, and DELETE statements as the functions that we discussed for the SELECT statement. For the non-SELECT statement, DBI provides a faster substitute for the *DBI::prepare/DBI::execute* pair with the *DBI::do* function:

```
$rows_affected = $dbh->do($sql_statement);
```

For further information, refer to the DBI documentation.

Disconnecting from a database

The DBI interface supports the disconnect function for the database handle, as shown in Example 8-25. There is no change required when converting.

Example 8-25 Disconnecting DB2 database using DBI

```
use DBI;  
use DBD::DB2::Constants;  
use DBD::DB2;  
my $user="db2inst1";  
my $password="password";  
$dbh = DBI->connect("dbi:DB2:invent", $username, $password);  
  
$dbh->disconnect;
```

8.2.2 Converting MySQL PHP applications to DB2

With PHP applications, you can use various approaches to access a MySQL database:

- ▶ **mysql for PHP**

The *mysql* extension interface provides a procedural API to read and write to a database.

- ▶ `mysqli` for PHP

The *mysqli* extension was introduced as a more advanced extension to the `mysql` extension to support features in MySQL 4.1.3 or newer versions. The `mysqli` extension provides both an object-oriented and procedural interface.

- ▶ PHP Data Objects (PDO)

You can access and manipulate MySQL databases with PDOs and the PDO MySQL driver.

Three approaches exist to connect to a DB2 application:

- ▶ IBM DB2 Extension for PHP

The *ibm_db2* extension interface provides an API to read and write from and to a database. The `mysql` and `mysqli` extensions can be easily mapped to `ibm_db2`.

- ▶ PHP Data Objects (PDO)

PDO_ODBC and *PDO_IBM* are object-oriented extension interfaces that are used to connect to a database. MySQL applications using PDO can be effortlessly converted to *PDO_ODBC* or *PDO_IBM*.

- ▶ Unified Open Database Connectivity (ODBC)

Unified ODBC is traditional procedural interface and supports multiple database servers. The Unified ODBC extension is not optimized for DB2 and therefore not recommended for new applications.

The following sections describe porting PHP applications from MySQL to DB2.

For more information about developing PHP applications with DB2, refer to *Developing PHP Applications for IBM Data Servers*, SG24-7218.

Converting from the `mysql` and `mysqli` extension library to IBM DB2 extension for PHP

Traditionally, `mysql` functions were used to connect to a MySQL database from PHP. To access MySQL V4.1 and V5 database servers from PHP 5 or higher code, use the `mysqli` functions. The `mysqli` functions have improved methods for accessing new features in MySQL.

If you use either the `mysql` or `mysqli` functions to access your MySQL database from PHP, it is relatively straightforward to convert your application to use the `ibm_db2` interface. You might also decide to perform a more complex code rewrite and use PDO instead.

Connecting to a database

Connecting to a MySQL database with the *mysql* extension consists of two parts. First, establish a connection to the MySQL server, and then, choose a database.

The following declaration shows the function that is specified to connect the MySQL server:

```
resource mysql_connect ( [string server [, string username [, string
password [, bool new_link [, int client_flags]]]])
```

Another way to connect to a MySQL server is to use the *mysql_pconnect()* function, which acts like the *mysql_connect()*, except this command opens a persistent connection on the database.

The server variable in the *mysql_connect()* function contains the host name or the IP address of the server that hosts the MySQL database.

The *mysql_select_db()* function chooses the MySQL database:

```
bool mysql_select_db ( string database_name [, resource link_identifier])
```

Example 8-26 shows the connection part of our sample application using the MySQL database.

Example 8-26 Connecting a MySQL database with the *mysql* extension

```
$host="localhost";
$user="mysql";
$pwd="password";
    $database="inventory";

$conn = mysql_connect($host, $user, $pwd) or die("Couldn't connect to server. "
. mysql_error() . "\n");

$dbConn = mysql_select_db($database, $conn) or die("Couldn't select database. "
. mysql_error() . "\n");
```

When connecting to a MySQL database using the *mysqli* extension, a single command, *mysqli_connect()*, can connect to the database:

```
mysqli mysqli_connect ([ string $host = ini_get("mysqli.default_host") [,
string $username = ini_get("mysqli.default_user") [, string $passwd =
ini_get("mysqli.default_pw") [, string $dbname = "" [, int $port =
ini_get("mysqli.default_port") [, string $socket =
ini_get("mysqli.default_socket") ]]]]] )
```

Example 8-27 on page 228 shows an example of connecting to a MySQL database using *mysqli*.

Example 8-27 Connecting a MySQL database with the mysqli extension

```
$host="localhost";
$user="mysql";
$pwd="password";
    $database="inventory";

$conn = mysqli_connect($host, $user, $pwd, $database)
or die("Couldn't connect to server. " . mysqli_connection_error() . "\n");
```

Just like the mysqli connections, a single *db2_connect()* command connects to a DB2 database with *ibm_db2* connections to databases:

```
resource db2_connect ( string $database , string $username , string
$password [, array $options ] )
```

Example 8-28 shows the converted connection part of the sample application. In our connection script, the command after the connection statement sets the current schema, which is used when querying the database. We use this approach, because when compared to DB2, MySQL does not have schemas or instances.

In MySQL, you can reference a table by *database.tablename*. When no database name is provided in the table name, the default is the database currently in use. In DB2, the table name is defined as *schema.tablename*. Referencing a DB2 table, when no schema is provided, uses the default schema associated with the user ID that is connected to the database. In an application, you can use the set schema statement to provide a global schema name for the tables, which do not have a full table name specified. To simplify application conversion, when porting the database, create the tables under the admin schema. By using the set schema after the database connection, we do not need to change every table name in the application.

Example 8-28 Connect to a DB2 database

```
$database = 'invent';
$user = 'db2inst1';
$password = 'password';

$conn = db2_connect($database, $user, $password) or die("Couldn't connect to
server. " . db2_conn_errormsg() . "\n");

$sql="SET CURRENT SCHEMA='ADMIN'";
$sqlResult = db2_exec($conn, $sql) or die ("Couldn't execute query.... " . $sql
. "\n" );
```

Note: The default subsystem name (DSN) is the database name, which is registered in the DB2 catalog. Because this database is cataloged, you do not have to declare any server information in the connect statement.

You can map the persistent connection functions *mysql_pconnect()* and *mysqli_pconnect()* to the *db2_pconnect()* function.

SELECT query statements

Example 8-29 shows querying a table in MySQL using the PHP *mysql* extension functions.

Example 8-29 MySQL select example using the mysql extension

```
$sql = "SELECT * from owners where id = $id";  
$sqlResults = mysql_query($sql)  
or die ("Couldn't execute query. " . $sql . "\n");  
$result = mysql_fetch_array($sqlResults);  
$username = $result[7];
```

Example 8-30 shows an example of querying a MySQL database using the *mysqli* extension.

Example 8-30 MySQL select example using mysqli

```
$sql = "SELECT * from owners where id = $id";  
$sqlResult = mysqli_query($conn, $sql)  
or die ("Couldn't execute query.... " . $sql . "\n");  
$result = mysqli_fetch_array($sqlResult);  
$username = $result[7];
```

Example 8-31 shows the corresponding *ibm_db2* DB2 query.

Example 8-31 DB2 SELECT example

```
$sql = "SELECT * from owners where id = $id";  
$sqlResults = db2_exec($conn, $sql)  
or die ("Couldn't execute query.... " . $sql . "\n");  
$result = db2_fetch_array($sqlResults);  
$username =  
$result[7];
```

You can map the *mysql_query()* function directly to the *db2_exec()* function, and the *db2_exec()* function nearly corresponds to the *mysql_query()* function. The only difference between the two functions is that *db2_exec()* requires two parameters. One parameter specifies the connection ID that is returned by the *db2_connect()* statement, and the other parameter is the SQL statement:

```
resource db2_exec ( resource $connection , string $statement [, array
$options ] )
```

To get each row in an array equal to the *mysql_fetch_array()* or the *mysql_fetch_array()* functions, you can use the *db2_fetch_array()* function without extensive modifications.

Calling INSERT, UPDATE, and DELETE statements

All three SQL commands, INSERT, UPDATE, and DELETE, use the same MySQL and *ibm_db2* functions that we discussed for the SELECT statement:

```
mysql_query($sql)
db2_exec($db,$sql)
```

Example 8-32, Example 8-33, and Example 8-34 on page 231 show the differences between the MySQL and *ibm_db2* functions for the INSERT statement.

Example 8-32 MySQL UPDATE statement using mysql

```
$updateCMD = "UPDATE services SET serviceOwner = " . $servOwner . " WHERE ID =
" . $srvID ;
$updateOutput = mysql_query($updateCMD) ;
if($updateOutput){
$textOutput = "Service Ticket updated. \n";
}else{
    $textOutput = "Service Ticket updated Failed. \n";
}
echo $textOutput . "\n";
```

Example 8-33 MySQL UPDATE statement using mysql

```
$updateCMD = "UPDATE services SET serviceOwner = " . $servOwner . " WHERE ID =
" . $srvID ;
$updateOutput = mysql_query($conn, $updateCMD)
if($updateOutput){
$textOutput = "Service Ticket updated. \n";
}else{
    $textOutput = "Service Ticket updated Failed. \n";
}
echo $textOutput . "\n";
```

Example 8-34 DB2 UPDATE statement

```
$updateCMD = "UPDATE services SET serviceOwner = " . $servOwner . " WHERE ID =  
" . $srvID ;  
$updateOutput = db2_exec($conn, $updateCMD);  
if($updateOutput){  
$textOutput = "Service Ticket updated. \n";  
}else{  
$textOutput = "Service Ticket updated Failed. \n";  
}  
echo $textOutput . "\n";
```

These functions differ in their return values. The *mysql_query()* and *mysqli_query()* functions return a resource identifier only for SELECT, SHOW, EXPLAIN, or DESCRIBE statements. The *mysql_query()* and *mysqli_query()* functions return a false if the query was executed incorrectly. For other types of SQL statements, *mysql_query()* and *mysqli_query()* return true on success and false on error. The *db2_exec()* function returns a result identifier if the SQL command was executed successfully and returns false if an error occurs.

Disconnecting from a database

Example 8-35, Example 8-36, and Example 8-37 show the disconnect functions using the *mysql*, *mysqli*, and *ibm_db2* extensions.

Example 8-35 Disconnecting from a MySQL database using *mysql*

```
$conn = mysql_connect($host, $user, $pwd)  
or die("Couldn't connect to server. " . mysql_error() . "\n");  
  
mysql_close($conn);
```

Example 8-36 Disconnecting from a MySQL database using *mysqli*

```
$conn = mysqli_connect($host, $user, $pwd, $database)  
or die("Couldn't connect to server. " . mysqli_connection_error() . "\n");  
  
mysqli_close($conn);
```

Example 8-37 Disconnecting from a DB2 database

```
$conn = db2_connect($database, $user, $password)  
or die("Couldn't connect to server. " . db2_conn_errormsg() . "\n");  
  
db2_close($conn);
```

These three functions execute the same task of disconnecting from the database, and both DB2 and MySQL return true on success and false on failure. In rare cases, the return value of the `mysql_close()` function is not used at all; therefore, you can perform conversions mostly by simply replacing the function or inserting the new function at the end of the program execution.

mysql, mysqli, and ibm_db2 function mapping

Table 8-3 maps common functions from `mysql` and `mysqli` to `ibm_db2`. Note that the `mysql` functions only offer a single interface for retrieving database errors. The `mysqli` and `ibm_db2` functions provide interfaces for handling both connections and statement errors.

Table 8-3 mysql, mysqli, and ibm_db2 function mapping

mysql	mysqli	ibm_db2
<code>mysql_close</code>	<code>mysqli_close</code>	<code>db2_close</code>
<code>mysql_connect</code>	<code>mysqli_connect</code>	<code>db2_connect</code>
<code>mysql_pconnect</code>	<code>mysqli_pconnect</code>	<code>db2_pconnect</code>
		<code>db2_pclose</code>
<code>mysql_query</code>	<code>mysqli_query</code>	<code>db2_exec</code>
	<code>mysqli_stmt_prepare</code>	<code>db2_prepare</code>
	<code>mysqli_stmt_execute</code>	<code>db2_execute</code>
<code>mysql_fetch_array</code>	<code>mysqli_fetch_array</code>	<code>db2_fetch_array</code>
<code>mysql_fetch_assoc</code>	<code>mysqli_fetch_assoc</code>	<code>db2_fetch_assoc</code>
<code>mysql_fetch_row</code>	<code>mysqli_fetch_row</code>	<code>db2_fetch_row</code>
<code>mysql_fetch_object</code>	<code>mysqli_fetch_object</code>	<code>db2_fetch_object</code>
<code>mysql_field_name</code>		<code>db2_field_name</code>
<code>mysql_errno</code>	<code>mysqli_connection_errno</code>	<code>db2_conn_error</code>
<code>mysql_error</code>	<code>mysqli_connection_error</code>	<code>db2_conn_errormsg</code>
<code>mysql_errno</code>	<code>mysqli_errno</code>	<code>db2_stmt_error</code>
<code>mysql_error</code>	<code>mysqli_error</code>	<code>db2_stmt_errormsg</code>
<code>mysql_affected_rows</code>	<code>mysqli_num_rows</code>	<code>db2_num_rows</code>

Converting from native MySQL library to PDO

If you currently use PDO with MySQL, it is simple to port your application to DB2. PDO provides a common API that you can use regardless of the database vendor to which the PHP application is trying to connect. In most cases, to convert databases, you only need to change the database driver and the connection string at which the connection to the database is established:

```
$db = new PDO("odbc:DSN=$database;UID=$user;PWD=$password;");
$db = new PDO("mysql:host=$hostname;dbname=$database", $user, $pass);
```

Connecting to a database

Use this syntax for connecting to a database using PDO:

```
$conn = new PDO( string $dsn [, string $username [, string $password [,
array $driver_options ]]] )
```

Example 8-38 and Example 8-39 show the difference between the syntax to connect to a MySQL and a DB2 database using PDO.

Example 8-38 Connecting to a MySQL database

```
$database = 'invent';
$user = 'db2inst1';
$password = 'password';

$conn = new PDO("mysql:host=$hostname;dbname=$database", $user, $password)
or die("Could not connect " . errorCode());
```

Example 8-39 Connecting to a DB2 database

```
$database = 'invent';
$user = 'db2inst1';
$password = 'password';

$conn = new PDO("odbc:$database", $user, $password)
or die("Could not connect " . errorCode());

$query="SET CURRENT SCHEMA='ADMIN'";
$output = $conn->exec($query);
```

SELECT query, INSERT, UPDATE, and DELETE statements

There are no changes required, because the syntax to execute queries is exactly the same for both data servers. Example 8-40 illustrates how to execute a query using PDO.

Example 8-40 DB2 SELECT example

```
$sql = "SELECT * from owners where id = '$id'";
$query = $conn->query($sql)
or die ("Couldn't execute query.... " . $sql . "\n");
foreach ($query as $query2){
    $username = $query[7];
}
```

Disconnecting from a database

Example 8-41 shows the disconnect function using the MySQL and IBM PDO API.

Example 8-41 Disconnecting from a DB2 database

```
$conn = new PDO("odbc:$database", $user, $password)
        or die("Could not connect " . errorCode());

$conn = null;
```

Converting from a native MySQL library to Unified ODBC

A third extension, *Unified ODBC*, has historically offered access to DB2 database systems. We do not recommend that you write new applications with this extension, because *ibm_db2* and *pdo_ibm* both offer significant performance and stability benefits over Unified ODBC. The *ibm_db2* extension API makes porting an application previously written for Unified ODBC almost as easy as globally changing the *odbc_* function name to *db2_* throughout the source code of your application.

Using the Unified ODBC support in PHP applications does not require a special load of library files, because support has been integrated during the compilation process of PHP. You can obtain a complete overview of the MySQL and the Unified ODBC functions in the PHP manual, which is available at this Web site:

<http://www.php.net/docs.php>

When we discuss ODBC in this section (which is always Unified ODBC), we refer to the native DB2 driver. Wide similarities between the syntax of Unified ODBC and other ODBC types and the performance advantages when using Unified ODBC support are the reasons that we discuss the application conversion with Unified ODBC support.

Connecting to a database

When connecting to a DB2 database with ODBC, you connect by using a single ODBC command (*odbc_connect()*):

```
resource odbc_connect ( string dsn, string user, string password [, int
cursor_type])
```

Example 8-42 on page 234 shows how to connect to our sample application database using the ODBC command.

Example 8-42 Connecting to a DB2 database

```
$database = 'invent';
$user = 'db2inst1';
```

```

$password = 'password';

$conn = odbc_connect($database,$user,$password)
        or die("Could not connect ". odbc_errormsg());

$query="SET CURRENT SCHEMA='ADMIN'";
odbc_exec($conn,$query) or die(odbc_errormsg($db));

```

The *odbc_pconnect()* function is a complementary function to the persistent connect function *mysql_pconnect()*.

SELECT query statements

Example 8-43 shows the ODBC DB2 query. Refer to Example 8-29 on page 229 and Example 8-30 on page 229 for the MySQL SELECT statements.

Example 8-43 DB2 query statement

```

$sql = "SELECT * FROM owners WHERE id = '$id'";
$sqlResults = odbc_exec($conn, $sql)
or die ("Couldn't execute query... " . $sql . "\n");
odbc_fetch_into($sqlResults, $result);
$username = $result[7];

```

The *odbc_exec()* function is almost identical to the *mysql_query()* function. The only difference is that the *odbc_exec()* function needs two parameters, one of which is the connection ID that is returned by the *odbc_connect* statement:

```
resource odbc_exec ( resource connection_id, string query_string)
```

To get each row in an array equal to the *mysql_fetch_row()* function, you can use the *odbc_fetch_into()* function without extensive modifications. There are only syntax differences in both statements in our case.

INSERT, UPDATE, and DELETE statements

Much like MySQL, the ODBC interface can execute all three types of SQL commands, INSERT, UPDATE, and DELETE, using the same function as that of the SELECT statement:

```

mysql_query($sql_statement) or mysql_queryi($db, sql_statement)
odbc_exec($db, $sql_statement)

```

Example 8-44 on page 236 illustrates the ODBC functions for the UPDATE statement. Refer to Example 8-32 on page 230 and Example 8-33 on page 230 for the MySQL UPDATE statements.

Example 8-44 DB2 UPDATE statement

```
$updateCMD = "UPDATE services SET serviceOwner = " . $servOwner . " WHERE ID =  
" . $srvID ;  
$updateOutput = odbc_exec($conn, $updateCMD);  
  
if($updateOutput){  
    $textOutput = "Service Ticket updated. \n";  
}else{  
    $textOutput = "Service Ticket updated Failed. \n";  
}  
echo $textOutput . "\n";
```

There is a minor difference between these functions in terms of return values:

- ▶ The *mysql_query()* function returns a resource identifier for SELECT, SHOW, EXPLAIN, or DESCRIBE statements, and false if the query is executed incorrectly. For any other SQL statements, a true is returned on success and false on error.
- ▶ The *odbc_exec()* function returns a result identifier if the SQL command was executed successfully and false if an error occurs.

Disconnecting from a database

To disconnect from a database, a difference in the return values exists between the *mysql_close()* and *odbc_close()* statements. The function *mysql_close()* returns true on success and false on failure. Whereas, the function *odbc_close()* does not return a value in either case.

Example 8-45 shows the disconnect function using the ODBC library. Refer to Example 8-35 on page 231 for the MySQL disconnect function.

Example 8-45 Disconnecting from a DB2 database

```
$conn = odbc_connect($database,$user,$password)  
        or die("Could not connect ". odbc_errormsg());  
  
odbc_close($conn);
```

8.2.3 Converting MySQL Ruby on Rails applications to DB2

Ruby is a fully-integrated object-oriented programming language that is used to develop applications. *Ruby on Rails* (RoR), which is also known as *Rails*, is a framework for developing Web applications that access databases according to the Model-View-Control architectural framework.

MySQL has two approaches to connect to a MySQL database from Ruby:

- ▶ MySQL/Ruby API
This API is a MySQL API.
- ▶ Ruby/MySQL API
This API is developed by Ruby.

The two APIs provide the same functions as the MySQL C API.

Collectively known as the *IBM_DB gem*, the *IBM_DB* Ruby driver and Rails adapter allow Ruby applications to access the following database management systems:

- ▶ DB2 Version 9 for Linux, UNIX, and Windows
- ▶ DB2 Universal Database (DB2 UDB) Version 8 for Linux, UNIX, and Windows
- ▶ DB2 UDB Version 5, Release 1 (and later) for AS/400® and iSeries, through DB2 Connect
- ▶ DB2 for z/OS, Version 8 and Version 9, through DB2 Connect
- ▶ Informix Dynamic Server, Version 11.10 and later

Converting DBI interface-supported code

You can easily connect to a DB2 database from Ruby on Rails by setting up the *IBM_DB* Ruby driver and Rails adapter. After you have configured the Ruby on Rails environment, you can install the *IBM_DB* gem by using one of these methods:

- ▶ On Linux, UNIX, and Mac OS X platforms, set the environment variables and optionally source the DB2 profile.
- ▶ On all supported platforms, issue the following gem command to install the *IBM_DB* adapter and driver:

```
db2server:~ # gem install ibm_db
```

For further information, visit this Web site:

<http://www.alphaworks.ibm.com/tech/db2onrails>

Connecting to a database

Connecting to a MySQL database with the MySQL/Ruby API consists of two parts. First, establish a connection to the MySQL server using the *connect()* function, and then, choose a database using the *select_db()* function.

Example 8-46 on page 238 illustrates the functions that are used to connect to a MySQL database from Ruby.

Example 8-46 Connecting a MySQL database from Ruby

```
require 'mysql'

conn = Mysql.init()
conn.connect('localhost', 'password')
conn.select_db('test')
```

You can create a connection to a DB2 database by using a single function, the *connect()* function. Use this syntax for the connect function:

```
resource IBM_DB::connect ( string database, string username, string
password [, array options] )
```

Example 8-47 illustrates how to connect to a DB2 database.

Example 8-47 Connecting to a DB2 database from Ruby

```
require 'rubygems'
require 'ibm_db'
conn = IBM_DB.connect("test", "db2inst1", "password")
```

SELECT query statements

Example 8-48 shows querying a table in MySQL using the MySQL/Ruby API functions.

Example 8-48 Issuing a SELECT statement on a MySQL database from Ruby

```
...
results = conn.query("SELECT * FROM owners;")

while row = results.fetch_row()
  puts "#{row[0]} : #{row[1]} "
end
```

The corresponding IBM_DB Ruby query is shown in Example 8-49.

Example 8-49 Issuing a SELECT statement on a DB2 database from Ruby

```
...
results = IBM_DB.exec(conn, "SELECT * FROM admin.owners")
  while row = IBM_DB.fetch_array(results)
    puts "#{row[0]} : #{row[1]} "
  end
```

You can map the MySQL *query()* function directly to the *DB2 exec()* function. Both functions have similar functionality. The only difference between the two functions is that *exec()* requires two parameters: one parameter specifies the connection ID that is returned by the *connect()* statement, and the other parameter is the SQL statement:

```
resource IBM_DB::exec ( resource connection, string statement [, array
options] )
```

To get each row in an array equal to the MySQL/Ruby *fetch_row()* function, you can use the *IBM_DB.fetch_array()* function without extensive modifications.

Calling INSERT, UPDATE, and DELETE statements

Generally, you can use the same functions for INSERT, UPDATE, and DELETE statements as the functions that we have discussed for the SELECT statement.

Disconnecting from a database

Both MySQL and DB2 use a *close()* function to disconnect from the database. However, the difference between the syntax is that the DB2 function requires the connection ID as a parameter for the function.

Example 8-50 and Example 8-51 show the disconnect functions using the MySQL/Ruby API and the IBM_DB API functions.

Example 8-50 Disconnecting from a MySQL database from Ruby

```
...
conn.close()
```

Example 8-51 Disconnecting from a Db2 database from Ruby

```
...
IBM_DB.close(conn)
```

For more information about developing DB2 applications with Ruby on Rails, review the *Developing Perl, PHP, Python, and Ruby on Rails Applications Manual*, which is available at this Web site:

<http://www.ibm.com/support/docview.wss?rs=71&uid=swg27015148>

8.2.4 Converting MySQL Java applications to DB2

DB2 supports the usage of Java programming at the following levels:

- ▶ DB2 server-side programming:
 - Java stored procedures on the DB2 server
 - Java user-defined functions (UDFs) on the DB2 server
- ▶ Java applications:
 - Java-enabled Web browser accessing DB2 using Java Database Connectivity (JDBC)
 - Java stand-alone application using JDBC and SQLJ
- ▶ Java 2 Platform, Enterprise Edition (J2EE™) application server:
 - JavaServer™ Pages (JSP™) using a JDBC connection
 - Servlets using JDBC or SQLJ
 - Enterprise JavaBeans™ (EJB™) using JDBC or SQLJ

DB2 provides the *IBM Data Server Driver for JDBC and SQLJ*. This driver is a single application driver to support the most demanding Java applications. You can use this agile driver in either type 4 or type 2 mode. This section provides an overview of JDBC, SQLJ, and the conversion of existing MySQL Java applications to DB2.

MySQL has an optional package, *MySQL Connector/J*, which is a type 4 JDBC driver. The latest version of MySQL Connector/J implements the SUN JDBC 4.0 API for relational database access.

Java database connectivity (JDBC)

JDBC is a vendor-neutral dynamic SQL interface that provides data access for your application through standardized Java methods. JDBC drivers provide the mechanics to the JDBC API to allow Java applications to access databases. Currently, the JDBC API is in its fourth revision. IBM DB2 supports JDBC 3 and JDBC 4.

A JDBC application can establish a connection to a data source using the JDBC DriverManager interface. In the following sections, we discuss the changes that are required within the code of a Java application when converting from MySQL to DB2.

IBM JDBC driver for DB2

A JDBC driver acts as an interface between a Java program and a database. DB2 9.7 includes two JDBC drivers:

- ▶ DB2 JDBC type 2 driver

The DB2 JDBC type 2 driver, which is also called the native-API/partly Java driver, allows Java applications to make JDBC calls that are translated to Java native methods. The Java applications that use this driver must run a DB2 client, which is used to communicate the JDBC requests to the DB2 server. Figure 8-2 on page 241 shows a call transfer for a DB2 JDBC type 2 driver. Only Java applications can use this driver. Implement this driver by using the DB2 call level interface (CLI) to communicate with DB2 servers.

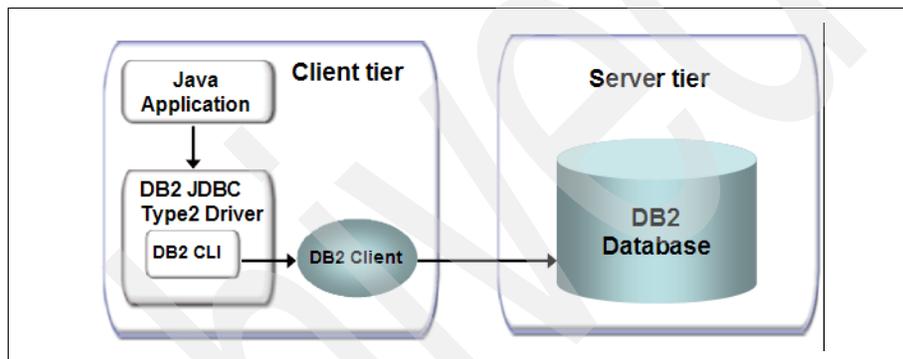


Figure 8-2 DB2 type 2 JDBC driver

In order to use the DB2 JDBC type 2 driver, you need following properties:

```
drivename="COM.ibm.db2.jdbc.app.DB2Driver"  
URL="java:db2:dbname"
```

The user ID and password are implicitly selected according to the DB2 client setup.

Note: The DB2 JDBC Type 2 Driver for Linux, UNIX, and Windows will not be supported in future releases. Consider switching to the IBM Data Server Driver for JDBC and SQLJ, which we describe next.

- ▶ IBM DB2 Driver for JDBC and SQLJ

The IBM DB2 Driver for JDBC and SQLJ is a single driver that includes both Type 2 and Type 4 connectivity. This type 4/Native-protocol all-Java driver is implemented in Java, and it uses the Distributed Relational Database Architecture™ (DRDA®) protocol for client/server communications. Figure 8-3 on page 242 shows the JDBC Driver usage in a Java application.

DB2 Universal JDBC Driver does not require any service on the client side. You can find it in the db2jcc.jar package.

To use the DB2 Universal JDBC driver, you need the following properties:

```
drivername="com.ibm.db2.jcc.DB2Driver"  
URL="java:db2://servername:serverport/dbname"
```

Note: *Servername* and *serverport* refer to the database server.

We recommend this driver for both applets and applications connecting to a DB2 database.

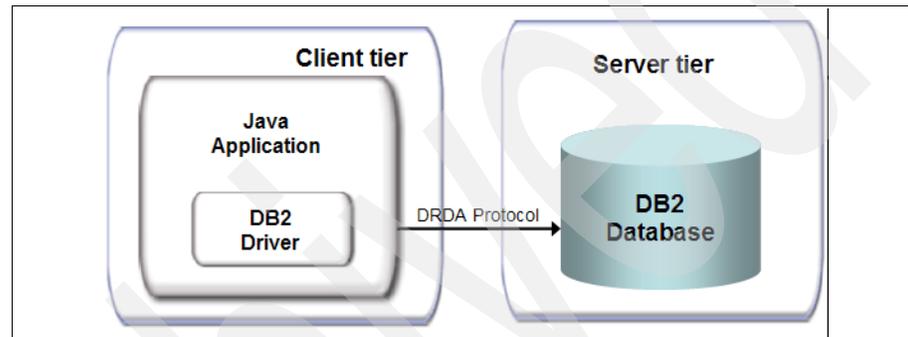


Figure 8-3 JDBC Driver

For more details, visit this Web site:

<http://www.ibm.com/software/data/db2/ad/java.html>

Embedded SQL for Java

DB2 provides embedded SQL (both static and dynamic) access to Java applications through SQLJ APIs. DB2 SQLJ support allows you to create, build, and run embedded SQL for Java applications, applets, stored procedures, and UDFs. The SQLJ API is an extension of JDBC. JDBC can only execute SQL statements dynamically. Statically bound SQL statements can run faster than dynamically bound statements; therefore, SQLJ applications have significant performance advantages over JDBC applications.

Conversion of JDBC applications

Because both MySQL and DB2 comply with JDBC specifications, most Java applications require minimal code changes. However, SQL statement changes might still be required, which we discussed as part of the data definition language (DDL) differences in 6.2, "Data definition language differences" on page 122.

In this section, we provide you with information about Java program conversion from MySQL to DB2.

Loading a JDBC driver

The first step in a Java program is to load the appropriate JDBC driver by calling `Class.forName(drivername)` within your Java program. We discuss appropriate values for the driver names in “IBM JDBC driver for DB2” on page 241. Example 8-52 on page 243 shows how to load the driver for MySQL, and Example 8-53 on page 243 shows the DB2 conversion.

Connecting to a database

In this part, the Java program tries to establish a connection to the given database by calling the function `DriverManager.getConnection` with the proper URL values as discussed within the driver description in “IBM JDBC driver for DB2” on page 241. After this call, `DriverManager` selects the appropriate driver from a set of registered drivers to connect to the database. Example 8-52 and Example 8-53 show these steps for MySQL and then DB2.

Example 8-52 MySQL JDBC driver loading and connection

```
import java.sql.*;
public class inventSample {
    public static void main(String[] args) throws Exception {
        try {
            //----- Load the driver -----//
            Class.forName("com.mysql.jdbc.Driver");

            //----- Connect to database -----//
            Connection conn = DriverManager.getConnection
                ("jdbc:mysql://localhost/inventory", "mysql", "password");

            //...
        }
    }
}
```

Example 8-53 DB2 JDBC driver loading and connection

```
import java.sql.*;
public class inventSample {
    public static void main(String[] args) throws Exception {
        try {
            //----- Load the driver -----//
            Class.forName("com.ibm.db2.jcc.DB2Driver");
            try {

                //----- Connect to database -----//
            }
        }
    }
}
```

```

        Connection conn = DriverManager.getConnection
("jdbc:db2:invent", "db2inst1", "password");
        //...
    }
}

```

Calling query statements

The query execution code does not change much, because both DB2 and MySQL follow the SQL standard and the JDBC specifications. When establishing a connection, code changes might only be required for SQL statements or return data types.

The JDBC API does not place any restrictions on the kind of SQL statements that can execute. Therefore, it becomes the responsibility of the application to pass SQL statements that are compatible with the database. The connection obtained in Example 8-52 on page 243 and Example 8-53 on page 243 can be used for one of the following three types of statements, depending upon the requirements:

- ▶ Statement: Simple single SQL statement

You can create the statement by using the `createStatement` method of the `Connection`. Example 8-54 shows the usage of `executeQuery` with a change for MySQL and DB2. It is evident that only changes to the SQL statement are required.

Example 8-54 Query statement changes from MySQL to DB2

```

Statement s = conn.createStatement();

//----- MySQL statement -----//
s.executeQuery("SELECT id, firstName, lastName, loginName FROM owners WHERE
id = 501");

//----- DB2 statement -----//
s.executeQuery("SELECT id, firstName, lastName, loginName FROM admin.owners
WHERE id = 501");

ResultSet rs = s.getResultSet ();
while (rs.next ()) {
    int id = rs.getInt ("id");
    String firstName = rs.getString ("firstName");
    String lastName = rs.getString ("lastName");
    String userName = rs.getString ("loginName");
    System.out.println (
        "id = " + id
        + ", firstName = " + firstName
        + ", lastName = " + lastName
    );
}

```

```

        + ", userName = " + userName);
    }
    rs.close ();
    s.close ();

```

- ▶ **PreparedStatement:** Precompiled SQL statements

PreparedStatements effectively execute particular statements multiple times. Example 8-55 shows the conversion of the prepared statements. Setting the appropriate values of host variables is vital when using prepared statements. We discuss prepared statements in more detail in “Java, JDBC, and SQL data type conversions”.

- ▶ **callableStatement:** Use this statement to call stored procedures.

Calling INSERT, UPDATE, and DELETE statements

You can execute any statement that updates the database or inserts and deletes a value within the database by using the *executeUpdate* or *execute* method. Example 8-55 shows the update of a record using the prepared statement for MySQL and its form of using the same call for DB2. As shown, these changes are due to the MySQL databases that have been merged into one DB2 database and grouped by the table schema. If you use DB2 SET SCHEMA = <schema_name> for a connection, you do not need to change anything.

Example 8-55 MySQL Prepared statement and executing UPDATE

```

//----- MySQL prepared statement -----//
//PreparedStatement smt = conn.prepareStatement
("UPDATE services SET serviceOwner = ? WHERE ID = ? ;");

//----- DB2 prepared statement -----//
PreparedStatement smt = conn.prepareStatement
("UPDATE admin.services SET serviceOwner = ? WHERE ID = ? ");

smt.setInt (1, 608);
smt.setInt (2, 108);
int count = smt.executeUpdate();
System.out.println (count + " rows were inserted");

conn.close();

```

Java, JDBC, and SQL data type conversions

In this section, we discuss how MySQL and DB2 handle column types for Java data type conversions. As we saw in 6.1, “Data type mapping” on page 116, MySQL and DB2 data types differ. While converting your Java application, we recommend that you check the Java data types that are used to fetch your data using JDBC. The JDBC driver converts the data exchanged between the

application and the database using the specified schema mapping, which is defined by both DB2 and MySQL for their data types.

Because MySQL does not enforce strict type conversions, the Java programmer has to take care of data lost because of round-off, overflow, or precision loss. For more details about how MySQL is mapped to Java data types, refer to the information at this Web site:

<http://dev.mysql.com/doc/refman/5.1/en/connector-j-reference-type-conversions.html>

However, DB2 sticks to the JDBC specification, providing a default and recommended data type mapping as shown in Table 8-4.

Table 8-4 DB2 data types mapping to Java types

DB2 data type	Java type	Data type description
SMALLINT	Short	16-bit signed integer
INTEGER	Int	32-bit signed integer
BIGINT	Long	64-bit signed integer
REAL	Float	Single precision floating point
DOUBLE	Double	Double precision floating point
DECIMAL	java.math.BigDecimal	Packed decimal
CHAR	java.lang.String	Fixed-length character string of length <i>n</i> where <i>n</i> is from 1 - 254
CHAR FOR BIT DATA	byte[]	Fixed-length character string of length <i>n</i> where <i>n</i> is from 1 - 254
VARCHAR	java.lang.String	Variable-length character string
VARCHAR FOR BIT DATA	byte[]	Variable-length character string
LONG VARCHAR	java.lang.String	Long variable-length character string
LONG VARCHAR FOR BIT DATA	byte[]	Long variable-length character string
BLOB	java.sql.Blob	Large object variable-length binary string

CLOB	java.sql.Clob	Large object variable-length character string
DBCLOB(n)	java.sql.Clob	Large object variable-length double-byte character string
GRAPHIC	java.lang.String	Fixed-length double-byte character string
VARGRAPHIC	java.lang.String	Non-null-terminating varying double-byte character string with 2-byte string length indicator
LONG VARGRAPHIC	java.lang.String	Non-null-terminating varying double-byte character string with 2-byte string length indicator
DB2 data type	Java type	Data type description
DATE	java.sql.Date	10-byte character string
TIME	java.sql.Time	8-byte character string
TIMESTAMP	java.sql.Timestamp	26-byte character string

8.2.5 Converting MySQL C/C++ applications to DB2

DB2 provides the following programming interfaces for developing applications in C/C++:

- ▶ Embedded SQL
- ▶ DB2 call level interface (CLI)

Apart from this, DB2 uses C/C++ for server side programming for creating:

- ▶ Stored procedures on DB2 server
- ▶ User-defined functions (UDF) on DB2 server.

DB2 provides precompilers for C, C++, COBOL, Fortran, and REXX to support embedded SQL applications. Embedded SQL applications support both static and dynamic SQL statements. Static SQL statements require information about all SQL statements, tables, and data types used at compile time. The application needs to be precompiled, bound, and compiled prior to execution. In contrast, dynamic SQL statements can be built and executed at runtime. For further details on embedded SQL refer to Getting Started with Database Application Development, GI11-9410-00. available at:

<http://www.ibm.com/support/docview.wss?rs=71&uid=swg27015148>

Another interface provided by DB2 is *DB2 call level interface (CLI)*. It is a standard based API following the Microsoft Open Database Connectivity (ODBC) specification and the International Organization for Standardization (ISO) SQL/CLI standard. Both embedded SQL and DB2 CLI support database administration and database manipulation from C/C++ applications.

MySQL provides a client library for accessing a MySQL database from C applications. The client library is shipped with the MySQL server or can be downloaded on its own using the MySQL connector/C. It provides features for:

- ▶ Connection mechanism
- ▶ Creation and execution of SQL queries
- ▶ Status and error reporting

The MySQL Connector/C++ (also commonly known as MySQL ++) is an additional library for accessing MySQL databases from C++ applications.

Converting applications

MySQL C API and *DB2 CLI* are similar in functionality and mechanisms to access databases. Both use the function call to pass dynamic SQL statements and do not need to be precompiled. We recommend converting MySQL C applications to DB2 CLI. This section describes conversion changes for various levels of the application:

Connecting to the server

The first step in converting MySQL C applications is to change the include information, initialize variables, and replace the MySQL connection with a DB2 connection. Example 8-56 shows a typical MySQL C program to initiate MySQL variables, create a connection, and terminate the connection.

Example 8-56 MySQL C application, initialize MySQL and create connection

```
#include <mysql/mysql.h> /* Include MySQL variable and function definition */
#include <stdio.h>

int main(){

MYSQL *conn; /* pointer to connection handler */
connection = mysql_init(NULL);

if(!mysql_real_connect (
    conn, /* pointer to connection handler */
    NULL, /* host to connect, default localhost*/
    "mysql", /* user name, default local user*/
    "password", /* password, default none*/
    "inventory", /* database name*/
    0, /* port */
    NULL, /* socket */
    0 /* flags*/
```

```

    ) )
    {
        printf("Connection Failed \n");
        fprintf(stderr, "%s\n", mysql_error(conn));
    } else{
        printf("Successful connection to database.\n");
    }
    }
mysql_close(conn);
}

```

Figure 8-4 shows a similar task using DB2 CLI. It shows the initialization tasks, which consist of: allocation and initializing the environment and connection handlers; creating the connection; processing of transactions; and terminating the connection and removing of handlers.

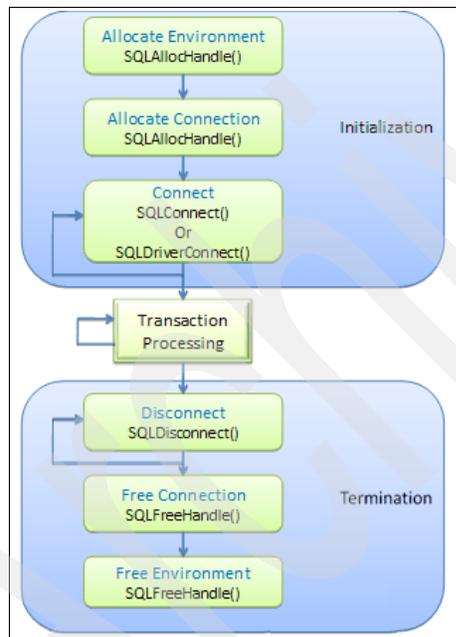


Figure 8-4 DB2 CLI activities

Example 8-57 shows the implementation of the task defined by the figure above.

Example 8-57 DB2 CLI application, connecting to a database

```

#include <sqlcli.h> /* Include DB2 CLI variable and function definition */ int main()
{
    SQLRETURN ret = SQL_SUCCESS; int rc = 0;
    SQLHANDLE henv; /* environment handle */
    SQLHANDLE hdbc; /* connection handle */

```

```

/* Allocate an environment handle */
ret = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);
if (ret != SQL_SUCCESS){
    /* handle error */
    return 1;
}

/* Allocate a connection handle */

ret = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);
if (ret != SQL_SUCCESS){
    /* handle error */
    return 1;
}

/* connect to the database */
ret = SQLConnect(hdbc,
    (SQLCHAR *)"invent", SQL_NTS,
    (SQLCHAR *)"db2inst1" /*user*/, SQL_NTS,
    (SQLCHAR *)"password" /* password*/, SQL_NTS);
if (ret != SQL_SUCCESS){
    /* handle error */
    return 1;
}

/* disconnect from the database */
ret = SQLDisconnect(hdbc);
if (ret != SQL_SUCCESS){
    /* handle error */
    return 1;
}

/* free connection handle */
ret = SQLFreeHandle(SQL_HANDLE_DBC, hdbc);
if (ret != SQL_SUCCESS){
    /* handle error */
    return 1;
}

/* free environment handle */
ret = SQLFreeHandle(SQL_HANDLE_ENV, henv);
if (ret != SQL_SUCCESS) {
    /* handle error */
    return 1;
}
exit(0);
}

```

Processing a query

A typical *MySQL C API* program involves three steps in query processing:

- ▶ Query construction

Depending upon your requirement you can construct a null terminated string or counted length string for the query:

```
char *query;
```

► Execution of the query

For executing the query you can use *mysql_real_query()* for a counted length query string or the *mysql_query()* for a null terminated query string.

Example 8-58 shows the processing of a query with both *mysql_real_query()* and *mysql_query()* method calls.

► Processing of the returned results

After executing the query, the final step is to process the results. All the statements except SELECT, SHOW, DESCRIBE, and EXPLAIN do not return results. For these statements, mysql provides the *mysql_affected_rows()* function for accessing the number of rows affected.

If your query returns a result set, follow these steps for the result processing:

- c. Generate the result set using *mysql_store_result()* or *mysql_use_result()*.
- d. Fetch each row using *mysql_fetch_row()*.
- e. Release the result set using *mysql_free_result()*.

Example 8-58 shows an example of both MySQL queries - some that return a set of results and others that do not.

Example 8-58 MySQL query processing

```
MYSQL_RES *result;
if (mysql_query(conn, "SELECT * FROM owners WHERE id = 501")) {
    fprintf(stderr, "%s\n", mysql_error(conn));
    return 1;
}
result = mysql_use_result(conn);

/* output table name */
printf("MySQL Tables in mysql database:\n");
while ((row = mysql_fetch_row(result)) != NULL){
    printf("firstName: %s \n", row[1]);
    printf("lastName: %s \n", row[2]);
    printf("email: %s \n", row[3]);
}

mysql_free_result(result);
}
```

On the other hand DB2 CLI provides a more comprehensive set of APIs for doing similar tasks. One of the essential parts of DB2 CLI is transaction processing,

which is supported by all the tables in DB2. Figure 8-5 on page 252 shows the typical order of function calls of query processing.

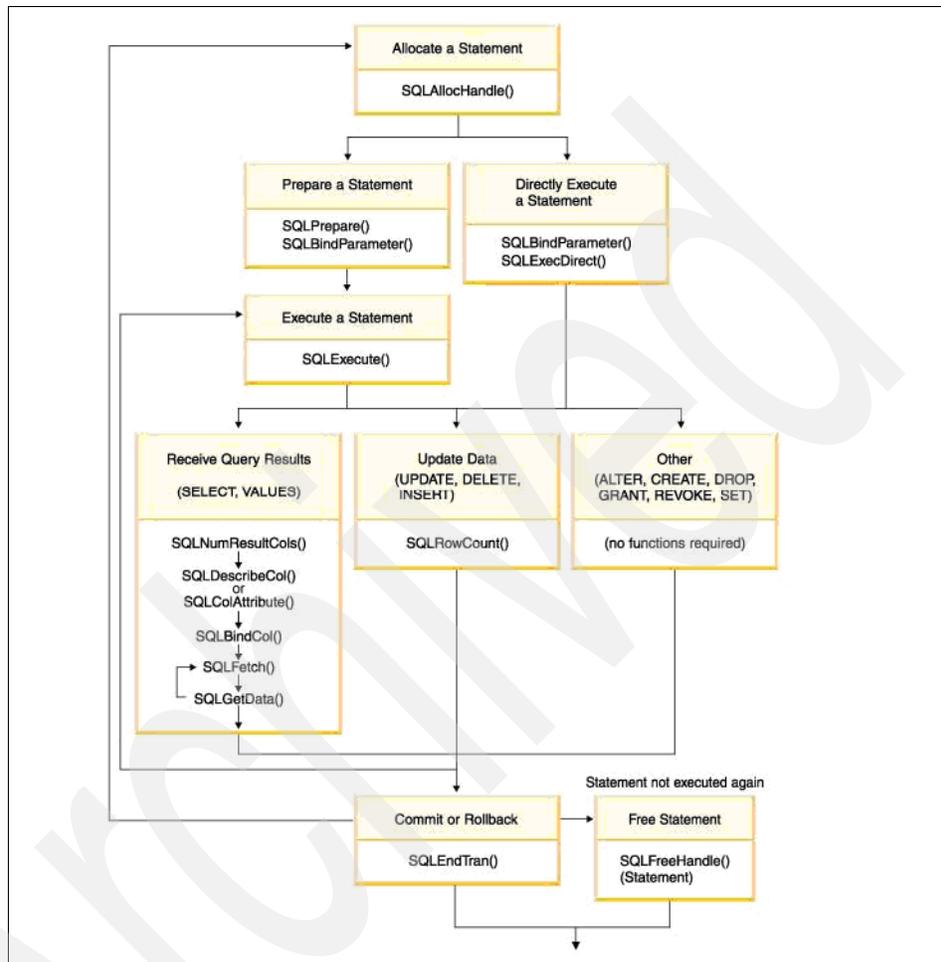


Figure 8-5 DB2 Query processing

DB2 CLI query processing involves the following steps:

1. Allocating a statement handle

A statement handle tracks execution of the query for a particular connection. This handle can be allocated using *SQLAllocHandle()* with a HandleType of *SQL_HANDLE_STMT* function.

2. Preparing and executing SQL statements

DB2 provides two ways for preparing and executing the query:

- Prepare and execute as separate steps

If you plan to execute the same query multiple times with different parameters, you can use this technique. This involves the following steps: Preparing a query using *SQLPrepare()*, binding of parameters using *SQLBindParameter()*, and executing this query using *SQLExecute()*. Example 8-59 shows an example for prepared statements.

- Prepare and execute in a single step

If your query is executed only once, then you can use *SQLExecDirect()* to directly call, prepare, and execute in a single step. Example 8-60 on page 255 shows the usage of this method.

3. Processing results

Processing query results involves binding application variables to columns of a result set, and then fetching the rows of data into the application variables. This is done by calling *SQLBindCol()* followed by *SQLFetch()* as shown in Example 8-59.

Another way to get data without binding the column is by calling *SQLFetch()* and *SQLGetData()*, this technique is used in Example 8-60 on page 255.

4. Committing or rolling back

DB2 supports two commit modes: auto-commit and manual commit. This can be set using *SQLSetConnectAttr()* with the parameter *SQL_AUTOCOMMIT_ON* or *SQL_AUTOCOMMIT_OFF*. If a transaction is set to *SQL_AUTOCOMMIT_OFF* it is the programmer's responsibility to end the transaction. This can be done using *SQLEndTran()* to either rollback or commit the transaction using parameter *SQL_COMMIT* or *SQL_ROLLBACK*.

5. Deallocating statement handle

This requires unbinding of variables, columns, or cursors (if allocated) using *SQLFreeStmt()* with the option of *SQL_CLOSE*, *SQL_UNBIND* or *SQL_RESET_PARAMS*, and then finally calling *SQLFreeHandle()* to deallocate the statement handle.

Example 8-59 DB2 CLI prepared statement with column binding, auto commit on

```
SQLHANDLE hstmt; /* statement handle */
SQLCHAR firstName [TEXT_SIZE];
SQLCHAR lastName [TEXT_SIZE];
SQLCHAR email [TEXT_SIZE];
SQLINTEGER id = 501;

/* SQL statements to execute */
SQLCHAR *stmt = (SQLCHAR *)"SELECT firstName, lastName, email FROM admin.owners WHERE
id = ?";

/* set AUTOCOMMIT on */
```

```

ret = SQLSetConnectAttr(hdbc, SQL_ATTR_AUTOCOMMIT, (SQLPOINTER)SQL_AUTOCOMMIT_ON,
SQL_NTS);

if (ret != SQL_SUCCESS) {
/* handle error */
}

/* allocate a statement handle */
ret = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt);
if (ret != SQL_SUCCESS) {
/* handle error */
}

/* prepare the statement */
ret = SQLPrepare(hstmt, stmt, SQL_NTS);
if (ret != SQL_SUCCESS) {
/* handle error */
}
/* bind parameter1 to the statement */
ret = SQLBindParameter(hstmt, 1, SQL_PARAM_INPUT, SQL_C_UBIGINT, SQL_CHAR, 0, 0, &id,
0, NULL);
if (ret != SQL_SUCCESS){
/* handle error */
}

ret = SQLBindCol(hstmt, 1, SQL_C_CHAR, (SQLPOINTER)firstName, sizeof(firstName) + 1,
NULL);
ret = SQLBindCol(hstmt, 2, SQL_C_CHAR, (SQLPOINTER)lastName, sizeof(lastName) + 1,
NULL);
ret = SQLBindCol(hstmt, 3, SQL_C_CHAR, (SQLPOINTER)email, sizeof(email) + 1, NULL);

/* execute the statement */
ret = SQLExecute(hstmt);
if (ret != SQL_SUCCESS){
/* handle error */
}

/* fetch each row and display */
ret= SQLFetch(hstmt);

if(ret == SQL_NO_DATA_FOUND){
printf("No data found");
}
while(ret != SQL_NO_DATA_FOUND){
printf("First name: %s \n",firstName);
printf("Last name: %s \n",lastName);
printf("email: %s \n", email);
ret=SQLFetch(hstmt);
}
ret = SQLFreeStmt(hstmt, SQL_UNBIND);
if (ret != SQL_SUCCESS) {
/* handle error */
}

```

```

/* free the statement handle */
ret = SQLFreeHandle(SQL_HANDLE_STMT, hstmt);
if (ret != SQL_SUCCESS) {
/* handle error */
return 1;
}

```

Example 8-60 DB2 CLI prepare/execute in one step with SQLGetData and manual commit

```

SQLHANDLE hstmt; /* statement handle */
SQLCHAR firstName [TEXT_SIZE];
SQLCHAR lastName [TEXT_SIZE];
SQLCHAR email [TEXT_SIZE];

/* SQL statements to execute */
SQLCHAR *stmt1 = (SQLCHAR *)"SELECT firstName, lastName, email FROM admin.owners
WHERE id = 501";

/* set AUTOCOMMIT on */
ret = SQLSetConnectAttr(hdbc,
SQL_ATTR_AUTOCOMMIT, (SQLPOINTER)SQL_AUTOCOMMIT_OFF, SQL_NTS);
if (ret != SQL_SUCCESS) {
/* handle error */
}

/* allocate a statement handle */
ret = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt);
if (ret != SQL_SUCCESS) {
/* handle error */
}

/* execute statement 1 directly */
ret = SQLExecDirect(hstmt, stmt1, SQL_NTS);
if (ret != SQL_SUCCESS) {

if(ret == SQL_NO_DATA_FOUND) {
printf("No data found");
}
int count = 0;
while(ret != SQL_NO_DATA_FOUND) {

/* get data from column 1 */
ret = SQLGetData(hstmt, 1, SQL_C_CHAR, (SQLPOINTER)firstName, sizeof(firstName) + 1,
NULL);
count = count + 1;
printf("COUNT %i \n", count);
printf("First Name: %s \n", firstName);
if (ret != SQL_SUCCESS) {
/* handle error */
}

}

/* get data from column 2 */

```

```

ret = SQLGetData(hstmt, 2, SQL_C_CHAR, (SQLPOINTER)lastName, sizeof(lastName) + 1,
NULL);
printf("Last Name: %s \n", lastName);
if (ret != SQL_SUCCESS) {
/* handle error */
}

/* get data from column 3 */
ret = SQLGetData(hstmt, 3, SQL_C_CHAR, (SQLPOINTER)email, sizeof(email) + 1, NULL);
printf("email : %s \n", email);
if (ret != SQL_SUCCESS) {
/* handle error */
}

ret=SQLFetch(hstmt);
}
ret = SQLEndTran( SQL_HANDLE_DBC, hdbc, SQL_ROLLBACK );
if (ret != SQL_SUCCESS) {
/* handle error */
return 1;
}

/* free the statement handle */
ret = SQLFreeHandle(SQL_HANDLE_STMT, hstmt);
if (ret != SQL_SUCCESS){
/* handle error */
return 1;
}

```

Consider converting all applications using the *MySQL Connector/C++* to *DB2 CLI*. The typical conversion process remains the same, because both MySQL C and MySQL C++ use the same flow of the program.

8.2.6 Converting Connector/ODBC applications to DB2

MySQL supports the ODBC database API to connect to a MySQL database server using an optional product called MySQL Connector/ODBC. The actual product name is MyODBC. Support for Connector/ODBC is available at two levels:

- ▶ Connector/ODBC 3.51 is a 32-bit ODBC driver supporting the ODBC 3.51 specification
- ▶ Connector/ODBC 5.1 supports 64-bit platforms and improves on Connector/ODBC 3.51 support

The conversion is easy, because DB2 CLI is also based on the ODBC specification and you can build ODBC applications without using any ODBC driver manager.

You only have to use DB2 ODBC driver to link to your application with libdb2. The DB2 CLI driver also acts as an ODBC driver when loaded by an ODBC driver manager. DB2 CLI conforms to ODBC 3.51.

Figure 8-6 shows the MySQL driver and DB2 ODBC driver in the ODBC scenario. Figure 8-6 shows the simplicity of converting an application using an ODBC driver to another driver. It also shows various components that are involved in the ODBC application and how they map from MySQL Connector/ODBC to DB2 ODBC.

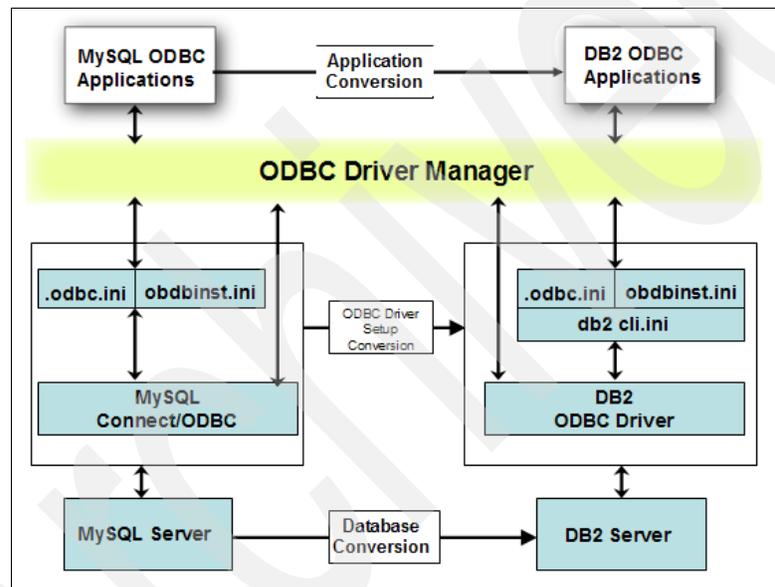


Figure 8-6 ODBC application conversion from MyODBC to DB2 ODBC driver

Typically, the MySQL Connector/ODBC connector has five components:

- ▶ Application

Because both MyODBC and DB2 CLI are based on the ODBC specification, applications do not require many changes. Although you do have to perform certain tasks, such as changing SQL queries, making changes in transaction management, and making proprietary method changes, where methods deviate from ODBC or other methods.

- ▶ Driver manager

The IBM Data Server CLI and ODBC Driver does not come with the ODBC driver manager. When using an ODBC application, you must ensure that an ODBC driver manager is installed and that users using ODBC have access to it. The following ODBC driver managers can be configured to work with the IBM Data Server CLI and ODBC Driver:

- unixODBC driver manager

The unixODBC driver manager is an open source ODBC driver manager supported for DB2 ODBC applications on all supported Linux and UNIX operating systems.

- Microsoft ODBC driver manager

You can use the Microsoft ODBC driver manager for connections to remote DB2 databases when using a TCP/IP network.

- DataDirect ODBC driver manager

You can use the DataDirect ODBC driver manager for DB2 for connections to the DB2 database.

- ▶ Connector/ODBC

As shown in Figure 8-6 on page 257, you are not required to use Connector/ODBC now; instead, you use the DB2 ODBC driver.

- ▶ ODBC configuration

The ODBC Driver Manager uses two initialization files:

- The `/etc/unixODBC/odbcinst.ini` file, in which you must add the following lines:

```
[IBM DB2 ODBC DRIVER]
Driver=/home/<instance name>/sql1lib/lib/db2.o
```

- The `/home/<instance name>/.odbc.ini` file, in which you must configure the data source. To set up a data source, you must add the following information:

```
In [ODBC Data Source] stanza add
invent= IBM DB2 ODBC DRIVER
Add [invent] stanza with
Driver=/home/<instance name>/sql1lib/lib/libdb2.so
Description= invent DB2 ODBC Database
```

- ▶ MySQL server

The MySQL database server is replaced by the DB2 server, which has been discussed in detail in previous chapters.

You can optionally configure the DB2 ODBC Driver to modify the behavior of the DB2 ODBC Driver by changing the `db2cli.ini` file.

8.2.7 Condition handling in DB2

It is critical that you foresee and handle exceptions in the application if you want a robust, industrial strength program. This section gives an introduction to condition handling in DB2.

DB2 error checking with **SQLCODE** and **SQLSTATE**

Each time that an SQL statement is executed, two values are returned by the DB2 engine and placed in the SQL communication area (*SQLCA*): *SQLCODE* and *SQLSTATE*.

Within your application program, you can retrieve these values to determine the state of the previously executed SQL statement. These identifiers provide more detailed information about the condition of the statement.

SQLCODE is the variable that is conventionally used for error handling in applications coded against the DB2 family. Therefore, *SQLCODE* is probably the most granular when it comes to DB2 exception handling.

However, IBM defines the value for *SQLCODE*. To achieve the highest portability of applications, only build dependencies on a subset of DB2 *SQLSTATE*s that are defined by ODBC Version 3 and ISO SQL/CLI specifications. Whenever you build your exception handling on IBM-supplied *SQLSTATE*s or *SQLCODE*s, carefully and thoroughly document the dependencies. You can access the specifications by using the search words ISO/IEC and standards 9075-1, 9075-2, and 9075-3 for SQL Foundation.

SQLSTATE is a five character string conforming to the American National Standards Institute (ANSI) SQL92 standard. The first two characters are known as the *SQLSTATE* class code, for example:

- ▶ 00 means successful completion.
- ▶ 01 is a warning.
- ▶ HY is generated by the DB2 CLI (call level interface) or ODBC driver.
- ▶ IM is generated by the ODBC driver manager.

For example, if your application signals *SQLSTATE* 23000, the DB2 description reports an integrity constraint violation, which is similar to MySQL's rudimentary description ER_NON_UNIQUE_ERROR or ER_DUP_KEY. Hence, condition handling for both database management systems can almost execute the same code.

SQLSTATE for DB2 CLI

Follow these guidelines for using SQLSTATES within your CLI application:

- ▶ Always check the function return code before calling *SQLGetDiagRec()* to determine if diagnostic information is available. Refer to the *IBM DB2 Call Level Interface Guide and Reference, Volume 1, SC10-4224-00*, for more information about this API.
- ▶ Use the SQLSTATES rather than the native error code.
- ▶ To increase your application's portability, only build dependencies on the subset of DB2 CLI SQLSTATES that is defined by the ODBC Version 3 and ISO SQL/CLI specifications, and return the additional dependencies as information only.

It might be useful to build dependencies on the class (the first two characters) of the SQLSTATES.

- ▶ For maximum diagnostic information, return the text message along with the SQLSTATE (if applicable, the text message will also include the IBM-defined SQLSTATE). It is also useful for the application to print the name of the function that returns the error.
- ▶ Ensure that the string allocated for the SQLSTATE includes space for the null termination character that is returned by DB2 CLI.

The code segment from *utilcli.c* in Example 8-61 shows how to retrieve and display diagnostic information, such as SQLSTATES.

Example 8-61 Handling SQLSTATE in CLI

```
void HandleDiagnosticsPrint(SQLSMALLINT htype, /* type identifier */
                           SQLHANDLE hndl /* handle */ )
{
    SQLCHAR message[SQL_MAX_MESSAGE_LENGTH + 1];
    SQLCHAR sqlstate[SQL_SQLSTATE_SIZE + 1];
    SQLINTEGER sqlcode;
    SQLSMALLINT length, i;

    i = 1;

    /* get multiple field settings of diagnostic record */
    while (SQLGetDiagRec(htype,
                        hndl,
                        i,
                        sqlstate,
                        &sqlcode,
                        message,
                        SQL_MAX_MESSAGE_LENGTH + 1,
                        &length) == SQL_SUCCESS)
    {
        printf("\n    SQLSTATE= %s\n", sqlstate);
        printf("Native Error Code = %ld\n", sqlcode);
    }
}
```

```
        printf("%s\n", message);
        i++;
    }

    printf("-----\n");
}
```

Note: We provide the code snippets in this chapter for illustrative purposes only. The `utilcli.c` code is sample code that is included with DB2, which you can find in the `SQLLIB/samples` directory.

Handling SQL errors in an SQLJ application

SQLJ clauses use the `java.sql.SQLException` JDBC class for error handling. SQLJ generates an `SQLException` under the following circumstances:

- ▶ When any SQL statement returns a negative SQL error code
- ▶ When a `SELECT INTO` SQL statement returns a +100 SQL error code

You can use the `getErrorCode` method to retrieve SQL error codes and the `getSQLState` method to retrieve `SQLSTATEs`.

To handle SQL errors in your SQLJ application, import the `java.sql.SQLException` class, and use the Java error handling try/catch blocks to modify the program flow when an SQL error occurs (see Example 8-62).

Example 8-62 SQL Exception with SQLJ

```
try {
#sql [ctxt] {SELECT LASTNAME INTO :empname
            FROM EMPLOYEE WHERE EMPNO='000010'};
}
catch(SQLException e) {
System.out.println("Error code returned: " + e.getErrorCode());
}
```

For exception handling in Java, it is important to know that DB2 provides several types of JDBC drivers with slightly different characteristics. With the DB2 Universal JDBC Driver, you can retrieve the `SQLCA`. For the DB2 JDBC type 2 driver for Linux, UNIX, and Windows (DB2 JDBC type 2 driver), use the standard `SQLException` to retrieve SQL error information.

SQLException under the IBM Data Server Driver for JDBC and SQLJ

As in all Java programs, error handling is done using try/catch blocks. Methods throw exceptions when an error occurs, and the code in the catch block handles those exceptions.

JDBC provides the *SQLException* class for handling errors. All JDBC methods throw an instance of *SQLException* when an error occurs during their execution. According to the JDBC specification, an *SQLException* object contains the following information:

- ▶ A string object that contains a description of the error or null
- ▶ A string object that contains the SQLSTATE or null
- ▶ An integer value that contains an error code
- ▶ A pointer to the next *SQLException* or null

IBM DB2 Driver for JDBC and SQLJ provides an extension to the *SQLException* class, which gives you more information about errors that occur when DB2 is accessed. If the JDBC driver detects an error, the *SQLException* class provides you with the same information as the standard *SQLException* class. However, if DB2 detects the error, the *SQLException* class provides you the standard information, along with the contents of the SQLCA that DB2 returns. If you plan to run your JDBC applications only on a system that uses the IBM DB2 Driver for JDBC and SQLJ, you can use this extended *SQLException* class.

Under the IBM DB2 Driver for JDBC and SQLJ, *SQLExceptions* from DB2 implement the *com.ibm.db2.jcc.DB2Diagnosable* interface. An *SQLException* from DB2 contains the following information:

- ▶ A *java.lang.Throwable* object that caused the *SQLException* or null if no such object exists. The *java.lang.Throwable* class is the superclass of all errors and exceptions in the Java language.
- ▶ The information that is provided by a standard *SQLException*
- ▶ An object of DB2-defined type *DB2Sqlca* that contains the SQLCA. This object contains the following objects:
 - An INT value that contains an SQL error code
 - A String object that contains the SQLERRMC values
 - A String object that contains the SQLERRP value
 - An array of INT values that contains the SQLERRD values
 - An array of CHAR values that contains the SQLWARN values
 - A String object that contains the SQLSTATE

Follow these steps to handle an *SQLException* in a JDBC program that runs under the IBM DB2 Driver for JDBC and SQLJ:

1. Import the required classes for DB2 error handling:
com.ibm.db2.jcc.DB2Diagnosable for getting diagnostic data and
com.ibm.db2.jcc.DB2Sqlca for receiving error messages.
2. In your code, catch *SQLException* and use it to get SQLCA, which is only allowed if the exception thrown is an instance of the *DB2Diagnosable* class.

- ▶ SQLWARNING identifies any condition where SQLWARN(0) = W or SQLCODE > 0 but is not equal to 100.
- ▶ NOT FOUND identifies any condition where SQLCODE = 100.

In each case, *action* specifies one of the following indicators:

- ▶ CONTINUE indicates to continue with the next instruction in the application.
- ▶ GO TO label indicates to go to the statement immediately following the label specified after GO TO (which is also commonly used as GOTO).

When the WHENEVER statement is not used, by default, execution continues even if an error, warning, or exception condition occurs.

You must use the WHENEVER statement prior to the SQL statements that will be affected. Otherwise, the precompiler does not know that additional error-handling code must be generated for executable SQL statements. You can have any combination of the three forms active at any time. The order in which you declare the three forms is insignificant.

To avoid an infinite looping situation, ensure that you undo the WHENEVER handling, prior to executing any SQL statements within the handler, by using the WHENEVER SQLERROR CONTINUE statement.

Declaring the SQLCA for error handling

You can declare the SQLCA in your application program so that the database manager can return information to your application. When you preprocess your program, the database manager inserts host language variable declarations in place of the INCLUDE SQLCA statement. The system communicates with your program using the variables for warning flags, error codes, and diagnostic information.

After executing each SQL statement, the system issues a return code in both SQLCODE and SQLSTATE. SQLCODE is an integer value that summarizes the execution of the statement, and SQLSTATE is a character field that provides error codes that are common across IBM relational database products. SQLSTATE also conforms to the ISO/ANSI SQL92 and Federal Information Processing Standard (FIPS) 127-2 standard.

Note that if SQLCODE is less than 0, it means that an error has occurred and the statement has not been processed. If the SQLCODE is greater than 0, it means that a warning has been issued, but the statement is still processed.

For a DB2 application that is written in C or C++, if the application is created using multiple source files, only one of the files can include the EXEC SQL

INCLUDE SQLCA statement to avoid multiple definitions of the SQLCA. The remaining source files need to use the following lines:

```
#include "sqlca.h"
extern struct sqlca sqlca;
```

Condition handling in a DB2 stored procedure

Similar to MySQL, DB2 defines condition handling by using a stored procedure. Although the support is similar, it seems appropriate to include a few examples of DB2 condition handling. For detailed information about condition handlers, refer to *SQL Procedural Languages: Application Enablement and Support*, SC09-4827.

This format is the general form of a handler declaration:

```
---DECLARE-----+---CONTINUE+----- HANDLER-- FOR---condition----->
+-EXIT-----+
+-UNDO-----+
>-----SQL-procedure-statement-----|
```

When DB2 raises a condition that matches a condition, DB2 passes control to the condition handler. The condition handler performs the action that is indicated by the handler-type and then executes the SQL-procedure-statement.

DB2 provides three general conditions:

- ▶ **NOT FOUND:**
This condition identifies a condition resulting in an SQLCODE of +100 or an SQLSTATE beginning with the characters '02'.
- ▶ **SQLEXCEPTION:**
This condition identifies any condition that results in a negative SQLCODE.
- ▶ **SQLWARNING:**
This condition identifies any condition that results in a warning condition (SQLWARN0 is 'W') or that results in a positive SQL return code other than +100. The corresponding SQLSTATE value will begin with the characters '01'.

You can also use the DECLARE statement to define your own condition for a specific SQLSTATE.

Example 8-64 on page 266 shows the general flow of the condition handler in a stored procedure.

Example 8-64 General example for condition handling

```
Begin
declare exit handler
    for sqlexception
    begin
statement3;
statement4;
end;

statement1;
statement2;
End
```

Example 8-65 shows a CONTINUE handler for delete and update operations on a table named EMP. Again, note that this code is solely intended for illustrative purposes.

Example 8-65 Example of a DB2 CONTINUE handler

```
CREATE PROCEDURE PROC1()
LANGUAGE SQL
BEGIN
    DECLARE SQLCODE, v_error INT;
    DECLARE CONTINUE HANDLER FOR
SQLEXCEPTION,
SET v_error = SQLCODE;

    DELETE FROM emp
    WHERE empno BETWEEN 100 and 200;
    IF (v_error = -147 ) THEN
        INSERT . . .
    UPDATE staff SET salary = salary * 1.25;
    IF (v_error <> 0 ) THEN
        RETURN -1;
    END IF;
END
```

8.2.8 Special conversions

MySQL provides access control on the host level, which means that specific privileges are granted depending on the host from which the user connects. DB2 does not provide this control mechanism. Therefore, if you use this MySQL feature, you have to implement a work-around in the application. You must change the code for any applications that use the MySQL host authentication feature to control user privileges on a database or global level.

There are many ways to implement this authentication mechanism on the application level. Here, we demonstrate a work-around using a simple example application that has two functions, SELECT and INSERT, which use the MySQL security feature to limit selecting and inserting data on the host level. Example 8-66 shows the MySQL host access data for four users that are controlled by our example.

Example 8-66 MySQL host access data

```
mysql> select user, host, select_priv, insert_priv from user;
```

user	host	select_priv	insert_priv
root	localhost	N	N
	localhost	N	N
user01	%	Y	N
user02	%.ibm.com	N	Y
inventAppUser	localhost	Y	Y
inventAppUser	%.ibm.com	Y	Y

This information has to be ported into a DB2 table. When a user attempts to access the data in the DB2 database, the application will verify each user's database access rights, along with the host system information for the host from which that user connects.

We need two tables for our DB2 conversion: one table to store user privilege information ported from MySQL and one working table. The table definitions and sample values are shown in Example 8-67.

Example 8-67 Creation of the tables for host authentication

```
-- script for creating the tables used by our example application

-- connect to the database
connect to invent user db2inst1 using password;

-- table ACCESSLIST
-- it stores access rights for specific users connecting from specific hosts
-- remark: there should be different access-flags for different functions

-- fields:
-- username, whom access to the function should be granted
-- hostname or ip-address, from which the user must connect
-- select access flag (Y/N), if SELECT is granted
-- insert access flag (Y/N), if INSERT is granted

drop table ACCESSLIST;
create table ACCESSLIST (
  USERNAME varchar(20),
```

```

HOST varchar(30),
ACCESS_SEL char(1),
ACCESS_INS char(1)
);
-- insert some sample values, according to the MySQL values (see above)
insert into ACCESSLIST values('user01', '%', 'Y', 'N');
insert into ACCESSLIST values('user02', '%.ibm.com', 'N', 'Y');
insert into ACCESSLIST values('inventAppUser', 'localhost', 'Y', 'Y');
insert into ACCESSLIST values('inventAppUser', '%.ibm.com', 'Y', 'Y');

-- table APPLACCESS
-- it stores the info about users and their host asking for access
-- this table is filled automatically by the sample application

-- fields:
-- username, who asks for access to the function
-- hostname, from which the user connects
-- ip-address, from which the user connects
-- timestamp, when the user asks for access

drop table APPLACCESS;
create table APPLACCESS (
  USERNAME varchar(8),
  HOSTNAME varchar(30),
  IPADDR varchar(30),
  TS timestamp
);

```

The authentication mechanism uses the ACCESSLIST and APPLACCESS tables:

► ACCESSLIST table

This table stores all the combinations of users and hosts (either name or IP address), and specific database access privileges are granted to user and host combinations. In our example, we control two access rights: SELECT and INSERT, so that we have two access fields: ACCESS_SEL and ACCESS_INS. In an actual client's application, more functions are controlled, so this table is expanded to have one ACCESS field for each function to be controlled, based on the corresponding MySQL privilege.

We insert sample values into this table for demonstration purposes.

► APPLACCESS table

This table is filled by the authentication application during run time. When a user asks for access, the application inserts the user ID, the host name, and the IP address from where the user connects. The timestamp is used as a key in this table, because records are not deleted from it.

Example 8-68 on page 269 lists the application code; remember that the code is just for demonstration purposes.

Example 8-68 Authentication mechanism example

```
import java.lang.*;
import java.io.*;
import java.sql.*;
import java.net.*;
public class AccessControl
{
    // in our example we use fixed values, you should make this variable
    private static final String DB2DB = "invent"; // database name
    private static final String DB2USR = "db2inst1"; // database user
    private static final String DB2PWD = "password"; // database password

    private static Connection db2Conn; // DB2 connection object

    public static void main(String[] args) throws SQLException, Exception
    {
        Class.forName("COM.ibm.db2.jdbc.app.DB2Driver").newInstance();

        db2Conn=DriverManager.getConnection("jdbc:db2:"+DB2DB,DB2USR,DB2PWD);

        // This example shows how to verify accesss to specific functions
        // depending on the host where the program is executed.
        // We assume that this program runs locally and connects directly to DB2
        // In a client/server-app, the server must detect the client's host.

        // Java provides two methods for this purpose:
        //     socket.getInetAddress() ... for socket-connections
        //     request.getRemoteHost() ... for HTTP-connections

        // In this example the username is the same as the DB2 user.
        // The application user and the database user could also be different.
        // This example implements no error handling.

        // This example shows access to two specific functions:
        // SELECT...this is indicated by mode='SEL'
        // INSERT...this is indicated by mode='INS'
        // mode should be a variable parameter

        String mode = "SEL";
        String insertStr = "";

        InetAddress addr = InetAddress.getLocalHost(); // appl executes locally
        String ipname = addr.getHostName(); // get the hostname
        String ipaddr = addr.getHostAddress(); // get the ip-address
        System.out.println("Host name = " + ipname);
        System.out.println("Host addr = " + ipaddr);

        // get records for the specified username with access to the function
        String query1 = "select HOST, current timestamp as TS from ACCESSLIST "+
            "where USERNAME='" + DB2USR + "' and ACCESS_" + mode + "='Y'";
        System.out.println("Query = " + query1);
        PreparedStatement ps1 = db2Conn.prepareStatement(query1);
        ResultSet rs = ps1.executeQuery(); // run the query
    }
}
```


is stored in a table, the SQL LIKE function can be used in the next step. The LIKE function handles wildcards (“%” and “_”) in the host information correctly.

- ▶ The third step is to verify if the host name or IP address has access rights by comparing the entry with the host name that was retrieved in the first step.
- ▶ If access is allowed, the function is executed. You can also implement a method that has a return code stating whether access is allowed.

8.3 Additional application considerations

After you convert to DB2 9.7, an application’s run time performance can be impacted by a number of factors. This section describes what locking and transaction isolation does to your application when running in a multiuser environment.

8.3.1 The purpose of locking

When many users access the same data source using your application, or any other interface that allows data manipulation, unwanted effects can occur:

- ▶ **Lost update:**
Two concurrent users retrieve and update the same data. The last successful change is kept while the first change is overridden.
- ▶ **Uncommitted (or dirty) read:**
User A can read or view data changed by User B, but those changes have not been committed yet.
- ▶ **Non-repeatable read:**
Within the same transaction, User A runs the identical SELECT statement multiple times with different results, because User B modified records in User A’s result set.
- ▶ **Phantom read:**
Within the same transaction, User A runs a SELECT statement multiple times and gets additional records, because user B added records in User A’s result set.

One of the more advanced features of a data management system is to define modification rules to control the use of data and to guarantee the integrity of the data to prevent these undesirable effects.

8.3.2 Concurrency control and transaction isolation

From an overview perspective, we can differentiate two methods for concurrency control:

- ▶ The optimistic concurrency approach:

This approach is a strategy to increase concurrency in which rows are not locked. Transactions are divided into read, validate, and write phases. Instead, during the validation phase before the records are updated or deleted, a cursor checks to see if the records have been changed since they were last read. If so, the update or delete fails.

- ▶ The pessimistic concurrency or locking approach:

This approach is a strategy in which rows are locked so that other transactions cannot change them. The transaction requests locks to the update resources. Other transactions have to wait or time out. The resource is released on the transaction completion or commit and rollback.

Both methods have their advantages and disadvantages, but by far, the most popular method is the latter approach. Both MySQL and DB2 follow this approach to various degrees of sophistication and with implementation differences.

8.3.3 DB2 isolation levels

The *isolation level* that is associated with an application process determines the degree to which the data that is being accessed by that process is locked or isolated from other concurrently executing processes. The isolation level is in effect for the duration of a unit of work.

The isolation level of an application process therefore specifies these conditions:

- ▶ The degree to which rows that are read or updated by the application are available to other concurrently executing application processes.
- ▶ The degree to which the update activity of other concurrently executing application processes can affect the application.

The isolation level for static SQL statements is specified as an attribute of a package and applies to the application processes that use that package. The isolation level is specified during the program preparation process by setting the ISOLATION bind or precompile option. For dynamic SQL statements, the default isolation level is the isolation level that was specified for the package preparing the statement. Use the SET CURRENT ISOLATION statement to specify a separate isolation level for dynamic SQL statements that are issued within a session. For both static SQL statements and dynamic SQL statements, the

isolation-clause in a select-statement overrides both the special register (if set) and the bind option value.

Because the isolation level determines how data is isolated from other processes while the data is being accessed, select an isolation level that balances the requirements of concurrency and data integrity. Table 8-5 gives you an overview of the DB2 isolation levels.

Table 8-5 DB2 isolation level

DB2 isolation level	Description
Uncommitted Read	<ul style="list-style-type: none"> ▶ Access to uncommitted data from other transactions ▶ No record locks unless updates occur
Cursor Stability	<ul style="list-style-type: none"> ▶ Addresses the <i>dirty read</i> issue ▶ Sees only committed data from other transactions ▶ Lock is only held on cursor position unless update occurs ▶ Update lock is held until transaction completed = commit ▶ Default isolation level
Read Stability	<ul style="list-style-type: none"> ▶ Addresses nonrepeatable read issue ▶ Sees only committed data from other transactions ▶ Locks are held on every row fetched (Inserts permitted) ▶ Locks are held for the duration of the transaction (commit/rollback)
Repeatable Read	<ul style="list-style-type: none"> ▶ Addresses phantom read issue ▶ All record locks held for the duration of the transaction ▶ A repeated query within the same transaction will get the same result set (Inserts are prevented)

Note: Only committed data is returned for the cursor stability isolation level, which is the currently *committed semantics* introduced in Version 9.7. Only committed data was returned in previous releases, but now readers do not wait for updaters to release the row locks. Instead, readers return data that is based on the currently committed version, that is, data prior to the start of the write operation.

The isolation levels that are listed in Table 8-5 on page 273 are ordered descendent according to the number and duration of the locks that are held during the transaction. Therefore, the degree of concurrency or locking is required to ensure the desired level of data integrity. However, too much locking drastically reduces concurrency. Poor application design and coding can cause locking problems, such as:

- ▶ Deadlocks
- ▶ Lock waits
- ▶ Lock escalation
- ▶ Lock timeouts

By default, DB2 operates with the cursor stability isolation level. You can specify transaction isolation at many levels, as we discussed in 8.3.5, “Specifying the isolation level in DB2” on page 276. For good performance, verify the lowest isolation level required for your converted application.

For additional information about the DB2 concurrency implementation, refer to the IBM DB2 Database for Linux, UNIX, and Windows Information Center at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp>

8.3.4 Locking

Certain MySQL applications, when ported to DB2, appear to behave identically, and you can ignore the topic of concurrency. However, if your applications involve frequent access to the same tables, the applications might behave differently. By default, MySQL runs in a mode that is called *autocommit*, which means that MySQL considers each and every SQL statement as an atomic unit of work or transaction.

In contrast, DB2, by default, considers a group of SQL statements with the corresponding unit of work boundaries set by a commit, a rollback statement as a single or atomic transaction, respectively. There are certain interfaces, such as the DB2 command line processor (CLP) or the JDBC interface, that run in autocommit mode. For other application interfaces, autocommit is turned off by default.

Another matter causing controversy among experts is the level of locking that is required for implementation on the database level. Do you implement the locking approach with the lowest level of overhead and, therefore, maintain locks on a table level? Or, is it better to lock on a lower level, for example, on the page level? Or, do you want even finer granularity with locking occurring on the row level? As usual, the correct answer to these questions is, “It depends.”

MySQL development decided to use the multistorage engine and decided to implement lock levels based on the type of table. Table types can be mixed within a database and even within a statement, and types can be altered. The default storage engine for MySQL supports only table-level locking. MyISAM table, Merge, and MEMORY tables use table-level locking. The InnoDB storage engine was released as a transactional table handler of MySQL with a lock manager for row-level locking mechanisms. Hence, the MySQL table type InnoDB defines tables that are the most like DB2 tables. Table 8-6 gives a high-level comparison of the MySQL tables and the DB2 tables.

Table 8-6 MySQL and DB2 table comparison

Characteristics	DB2 tables	MyISAM tables	InnoDB tables
Lock level	Row level, table level only on explicit request	None or table level	Row level and table level
Commitment control	Yes	No	Yes
Isolation level	Uncommitted Reads, Cursor Stability, Read Stability, and Repeatable Reads	No	Read Uncommitted, Read Committed, Repeatable Reads, Serializable
Rollback on DDL	Yes	No	Yes

Consider using table-level locking in these situations:

- ▶ Applications that use mostly reads, such as data warehouse and Business Intelligence applications
 - Applications reading and updating through key positioning, such as UPDATE... WHERE Custno =?
- ▶ Applications using INSERTs with subselects and only a small number of UPDATEs and DELETEs

Consider using row-level locking in these situations:

- ▶ Applications that require a high level of concurrency and online transaction processing (OLTP) capabilities
- ▶ Many SELECTs with only small result sets
- ▶ Applications with high UPDATE, INSERT, and DELETE frequency

However, there are concurrency issues that might arise when converting a MySQL application to DB2 based on the two MySQL table types that we consider significant:

- ▶ MyISAM tables provide a high level of concurrency, because SQL processing occurs in autocommit mode and no row-level locks are maintained. When converting to DB2, ensure that your application operates in autocommit mode. Verify the lowest isolation level required for your application and MyISAM tables.
- ▶ InnoDB tables provide concurrency control similar to DB2. Note that default transaction isolation for InnoDB is repeatable read.

8.3.5 Specifying the isolation level in DB2

Because the isolation level determines how data is locked and isolated from other processes, while it is being accessed, you need to select an isolation level that balances the requirements of concurrency and data integrity.

The isolation level that you specify is in effect for the duration of the *unit of work*. You can specify the isolation level in several ways. Use the following heuristics to determine which isolation level will be used when compiling an SQL or XQuery statement:

- ▶ Static SQL:
 - If an isolation clause is specified in the statement, the value of that clause is used.
 - If no isolation clause is specified in the statement, the isolation level used is the isolation level that is specified for the package at the time when the package was bound to the database.
- ▶ Dynamic SQL:
 - If an isolation clause is specified in the statement, the value of that clause is used.
 - If no isolation clause is specified in the statement, and a SET CURRENT ISOLATION statement has been issued within the current session, the value of the CURRENT ISOLATION special register is used.
 - If no isolation clause is specified in the statement, and no SET CURRENT ISOLATION statement has been issued within the current session, the isolation level used is the one specified for the package at the time that the package was bound to the database.
- ▶ For static or dynamic XQuery statements, the isolation level of the environment determines the isolation level that is used when the XQuery expression is evaluated.

Note: Many commercially written applications provide a method for choosing the isolation level. Refer to the application documentation for information.

SQL procedure and isolation level

This section discusses when to specify the isolation level for an SQL procedure.

At precompile or bind time

For an application that is written in a supported compiled language, use the ISOLATION option of the command line processor PREP or BIND commands. You can also use the sqlaprep or sqlabndx API to specify the isolation level:

- ▶ If you create a bind file at precompile time, the isolation level is stored in the bind file. If you do not specify an isolation level at bind time, the default is the isolation level that is used during precompilation.
- ▶ If you do not specify an isolation level, the default of cursor stability is used.

Tip: To determine the isolation level of a package, execute the following query where XXXXXXXX is the name of the package and YYYYYYYY is the schema name of the package. Both of these names must be in all capital letters:

```
SELECT ISOLATION FROM SYSCAT.PACKAGES
WHERE PKGNAME = 'XXXXXXX'
AND PKGSCHEMA = 'YYYYYYYY'
```

On database servers that support REXX

When a database is created, multiple bind files that support the isolation levels for SQL in REXX are bound to the database. Other command line processor packages are also bound to the database when a database is created.

REXX and the command line processor connect to a database using a default isolation level of cursor stability. Changing to another isolation level does not change the connection state.

To determine the isolation level that is used by a REXX application, check the value of the SQLISL predefined REXX variable. The value is updated each time that the CHANGE ISOLATION LEVEL command executes.

At the statement level

Use the WITH clause. The WITH clause cannot be used on subqueries. The WITH UR option applies to read-only operations only. In other cases, the statement is automatically changed from UR to CS.

This isolation level overrides the isolation level that is specified for the package in which the statement appears. You can specify an isolation level for the following SQL statements:

- ▶ DECLARE CURSOR
- ▶ Searched DELETE
- ▶ INSERT
- ▶ SELECT
- ▶ SELECT INTO
- ▶ Searched UPDATE

Note: Isolation levels for XQuery statements cannot be specified at the statement level.

From CLI or ODBC at run time

Use the CHANGE ISOLATION LEVEL command. With DB2 call level interface (CLI), you can change the isolation level as part of the CLI configuration. At run time, use the *SQLSetConnectAttr* function with the SQL_ATTR_TXN_ISOLATION attribute to set the transaction isolation level for the current connection referenced by the *ConnectionHandle* argument. You can also use the TXNISOLATION keyword in the `db2cli.ini` file.

When working with JDBC or SQLJ at run time

To create a package (and to specify its isolation level) in SQLJ, use the SQLJ profile customizer (the `db2sqljcustomize` command).

Note: JDBC and SQLJ are implemented with CLI on DB2, which means that the `db2cli.ini` settings might affect what is written and run using JDBC and SQLJ.

For dynamic SQL within the current session

Use the SET CURRENT ISOLATION statement to set the isolation level for dynamic SQL issued within a session. Issuing this statement sets the CURRENT ISOLATION special register to a value that specifies the isolation level for any dynamic SQL statements that are issued within the current session. When set, the CURRENT ISOLATION special register provides the isolation level for any subsequent dynamic SQL statement that is compiled within the session, regardless of which package issued the statement. This isolation level is in effect until the session ends or until the SET CURRENT ISOLATION...RESET statement is issued.

Database administration

In this chapter, we focus on the database administration features that are offered by DB2. We provide a general introduction and detailed description of DB2 administration programs, utilities, and tools. We also describe a few of the salient features that are available in DB2 but that are missing in MySQL.

We present key attributes of database administration for DB2, such as:

- ▶ Database configuration
- ▶ Data recovery
- ▶ Database replication
- ▶ Data movement utilities
- ▶ High availability
- ▶ Autonomics
- ▶ Workload Manager

We also explore the following graphical tools:

- ▶ DB2 Control Center
- ▶ IBM Data Studio
- ▶ Data Studio Administration Console

9.1 Database configuration

Database configuration is an extremely important task for database administrators. It involves setting up the optimal parameters for the database, system, database manager, and all other related components to achieve the best performance for your application. In this section, we discuss how DB2 database parameters are tuned.

9.1.1 DB2 configuration

DB2 has two levels of configuration:

- ▶ Database manager (instance)
- ▶ Database

DB2 environment variables and profile registry

DB2 environment and profile registry variables control the database environment. Information that is stored in these profile registries is used by the DB2 server instance and any DB2 applications that are started after the changes have been made.

Using the DB2 profile registry allows for centralized control of the environment variables. Through use of various profiles, multiple levels of support are provided. Remote administration of the environment variables is also available when using the DB2 Administration Server.

There are four profile registries:

- ▶ The DB2 instance-level profile registry
The majority of the DB2 environment variables are placed in this registry. The environment variable settings for a particular instance are kept in this registry. Values that are defined in this level override their settings on the global level.
- ▶ The DB2 global-level profile registry
If an environment variable is not set for a particular instance, this registry is used. This registry is visible to all instances pertaining to a particular copy of the DB2 data server; one global-level profile exists in the installation path.
- ▶ The DB2 instance node-level profile registry
This registry level contains variable settings that are specific to a database partition in a partitioned database environment. Values that are defined in this level override their settings at the instance and global levels.

► The DB2 instance profile registry

This registry contains a list of all instance names associated with the current copy. Each installation has its own list. You can see the complete list of all instances available on the system by running the **db2ilist** command.

You can set the variables by using the **db2set** command. The command immediately stores the updated variables in the profile registry. Example 9-1 shows the various modes in which the **db2set** command can be used.

Example 9-1 Changing registry and environment variables using the db2set command

```
//this shows the current registry variables//
db2inst1@db2server:~> db2set -all
[i] DB2_COMPATIBILITY_VECTOR=8
[i] DB2PROCESSORS=0,1
[i] DB2COMM=tcPIP
[g] DB2SYSTEM=db2server
[g] DB2INSTDEF=db2inst1
[g] DB2ADMINSERVER=dasusr1

//this sets registry for all databases in particular instance//
db2inst1@db2server:~>db2set DB2AUTOSTART=YES -i db2inst1

//this sets registry variable for all instances//
db2inst1@db2server:~>db2set DB2AUTOSTART=YES -g

//this sets registry variable for particular node//
db2inst1@db2server:~>db2set DB2AUTOSTART=YES -i db2inst1 65000
```

DB2 configures the operating environment by checking for registry values and environment variables, and DB2 resolves them in the following order:

1. Environment variables are set using the **set** command (or the **export** command on UNIX platforms).
2. Registry values are set with the instance node-level profile (using the **db2set -i <instance name> <nodenum>** command).
3. Registry values are set with the instance-level profile (using the **db2set -i** command).
4. Registry values are set with the global-level profile (using the **db2set -g** command).

DB2 Configuration parameters

In addition to the DB2 profile registry, DB2 also has instance and database configuration files, which provide users with the flexibility to configure the database and the database engine to fit business and application needs. These files contain parameters that define values, such as resources allocated to DB2, the diagnostic level, log file location, and other values.

When a DB2 database instance or a database is created, a corresponding configuration file is created with default parameter values. We strongly suggest that you modify these parameters to improve the performance of the instance or database.

Figure 9-1 illustrates the two DB2 configuration files and additional operating system configurations.

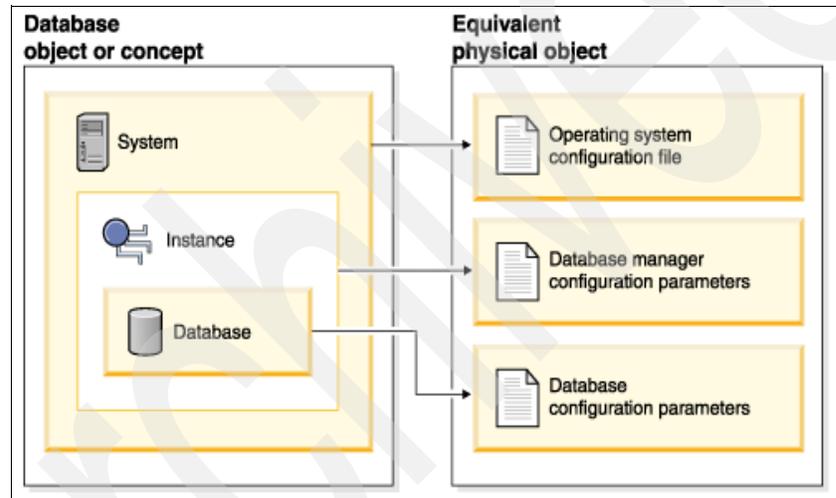


Figure 9-1 Configuration parameter files

There are two database configuration files:

- The database manager configuration file

This file is created automatically when creating a DB2 instance and affects the configuration on the instance level. It is stored in the `db2system` file under the `sql1ib` subdirectory of the instance. Use the following command to see the values in this file:

```
db2 GET DATABASE MANAGER CONFIGURATION
```

Set parameter values using this command:

```
db2 UPDATE DBM CFG USING <parameter> <value>
```

► The database configuration file

For each created database, a database configuration file is created, as well. It is stored in a file named SQLDBCONF and resides in the directory where the database resides. It defines the resources and variables for the particular database. Use the following command to obtain the values in this file:

```
db2 GET DATABASE CONFIGURATION FOR <database name>
```

Set parameter values using this command:

```
db2 UPDATE DATABASE CONFIGURATION FOR <database name> USING <parameter>  
<value>
```

Configuration tools

IBM has tools to assist you with configuring your database server. Two of these tools are the Configuration Assistant and the IBM Data Studio.

You can use the *DB2 Configuration Assistant* to configure and maintain the database objects that you or your application will use. The Configuration Assistant is a graphical tool that is tightly integrated with the DB2 Control Center. It allows you to update both the DB2 Profile Registry and the DB2 database manager configuration parameters on the local machine, as well as remotely. It can be launched from the DB2 Control Center or by calling the db2ca utility. The Configuration Assistant also has an advanced view, which uses a notebook to organize connection information by object: systems, instance nodes, databases, database connection services (DCS), and data sources. Figure 9-2 on page 284 shows how to change the database manager configuration using the Configuration Assistant.

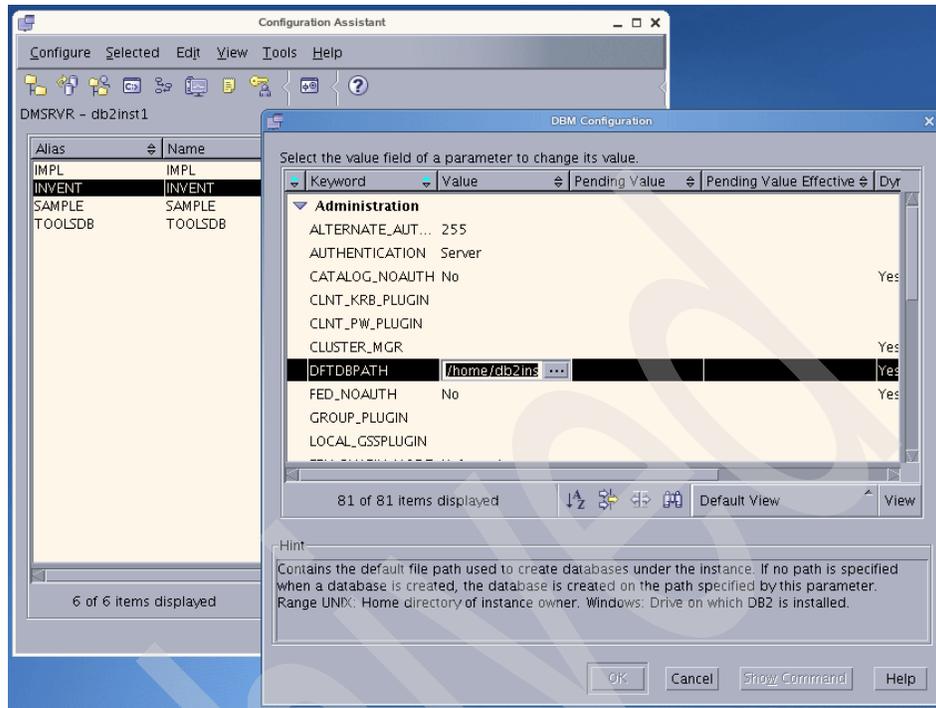


Figure 9-2 DB2 Configuration Assistant

Note: The Configuration Assistant has been deprecated in Version 9.7 and might be removed in a future release. We recommend that you use the IBM Integration Management solutions for managing DB2 for Linux, UNIX, and Windows data and data-centric applications.

IBM Data Studio is part of the IBM Integration Management solutions for managing your DB2 database. Data Studio simplifies the process of managing your database objects by supporting instance and database management and by providing the ability to run database commands and utilities. It provides a simple user interface to invoke the database administration commands that you use to maintain and manage your database environment. Figure 9-3 on page 285 shows how to change the database manager and database configuration using Data Studio.

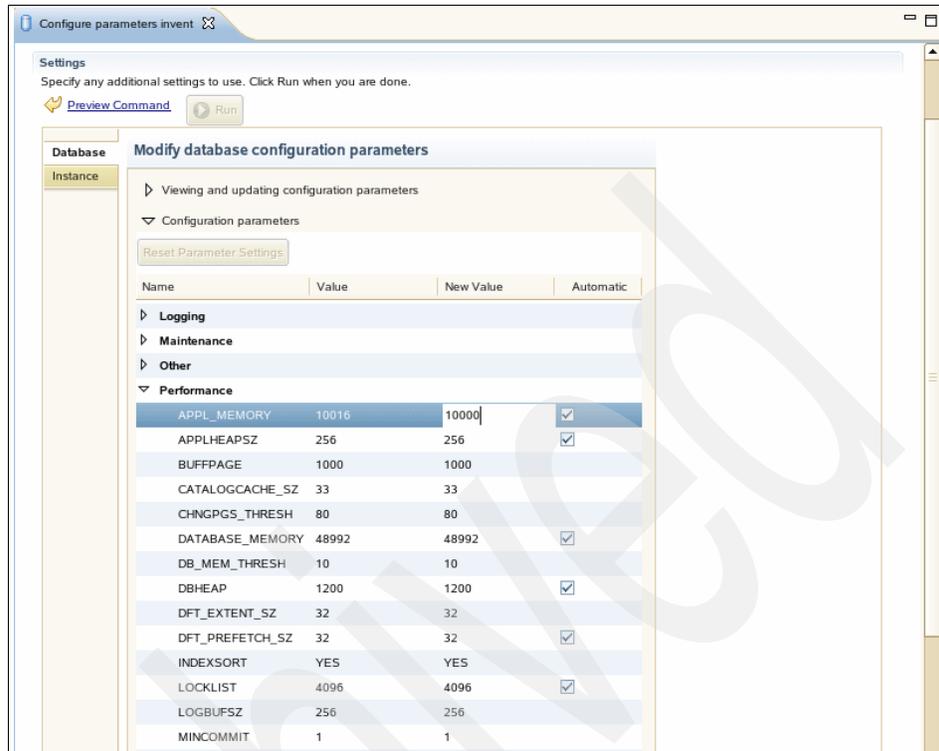


Figure 9-3 Data Studio configuration window

IBM offers a number of automatic tools and DB2 features to make database administration effortless. You can use the *Configuration Advisor* to assist with parameter configuration and to configure your database for optimal performance. The Configuration Advisor looks at your current database, asks for user input on the database workload, and suggests the best configuration parameters for buffer pool size, database configuration, and database manager configuration. Figure 9-4 on page 286 shows the suggested output for our sample inventory database.

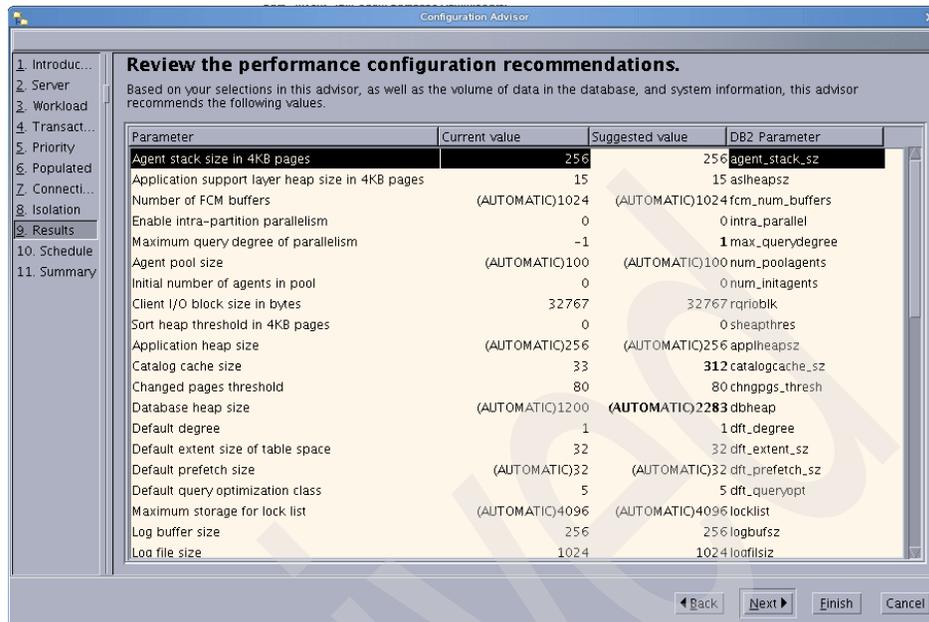


Figure 9-4 Configuration Advisor

9.2 Database recovery

Database recovery is the action that the database system or user takes to recuperate the database in case of system, hardware, software, or application failure. Database recovery includes precautionary measures and resolutions. The database systems use multiple recovery methods to avoid losing data.

9.2.1 DB2 database recovery

The DB2 recovery method is much more sophisticated than MySQL and allows you to build an optimized backup and recovery strategy for your database system. DB2 supports database recovery using database backup in conjunction with three recovery logs, as shown in Figure 9-5 on page 287. The recovery log files and the recovery history files are created automatically when a database is created. These log files are important if you need to recover data that is lost or damaged.

The three files work in the following ways (Figure 9-5):

- ▶ Recovery log files

You use the *recovery log files* to recover from application or system errors. In combination with the database backups, they are used to recover the consistency of the database right up to the point in time when the error occurred. Recovery log files exist for each database on the server.

- ▶ Recovery history files

The *recovery history files* contain a summary of the backup information that can be used to determine recovery options, if all or part of the database must be recovered to a certain point in time. The recovery history files track recovery-related events, such as backup and restore operations, among other events. The recovery history file is located in the database directory.

- ▶ Table space change history file

The *table space change history files* are also located in the database directory. These files contain information that can be used to determine which log files are required for the recovery of a particular table space.

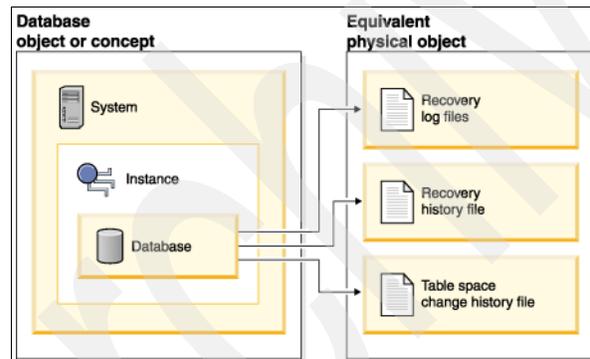


Figure 9-5 Database recovery files

Two types of databases exist for backup and recovery: non-recoverable databases and recoverable databases. As a database administrator, you need to decide in which category your database fits:

- ▶ Non-recoverable database

You can store data that is easily recreated in a *non-recoverable* database. This data includes data that is used for read-only applications and tables that are not often updated. These types of databases have a small amount of logging, which does not justify the added complexity of managing log files and rolling forward after a restore operation.

To set up a non-recoverable database, set both database configuration parameters `logarchmeth1` and `logarchmeth2` to `OFF`. Therefore, only the crash recovery logs are kept. These logs are known as *active logs*, and they contain current transaction data. Because non-recoverable databases only support active logs, they do not support roll-forward recovery.

► Recoverable database

Store data that cannot be easily recreated in a *recoverable* database. Data that cannot be easily recreated includes data whose source is destroyed after the data is loaded, data that is manually entered into tables, and data that is modified by application programs or users after it is loaded into the database.

To set up a recoverable database, set both `logarchmeth1` and `logarchmeth2` database configuration parameters to anything other than `OFF`. In addition to the active logs that are kept for crash recovery, recoverable databases also have archived logs. These archived logs contain committed data and therefore support crash, version, or roll-forward recovery. Recoverable databases also support backup, restore, and roll-forward individual table spaces.

Database backup

To back up a DB2 database, database partition, or selected table space, you can use the DB2 **backup** command. Use this command to create a backup to disk, tape, or named pipes in UNIX. DB2 supports both offline and online backup:

```
db2 backup database invent to /home/db2inst1/backup
```

You can back up an entire database, database partition, or only selected table spaces.

In addition to backing up the entire database every time, DB2 also supports incremental backups where you can back up large databases on a regular basis incrementally. Incremental backups require that the `trackmod` database configuration parameter is set to `yes`. Incremental backup can be a *cumulative backup*, which stores data changes since the last successful full backup, or a *delta backup*, which is the last successful backup irrespective of whether that backup was full, delta, or cumulative. Figure 9-6 on page 289 and Example 9-2 on page 289 show the cumulative and delta backup techniques.

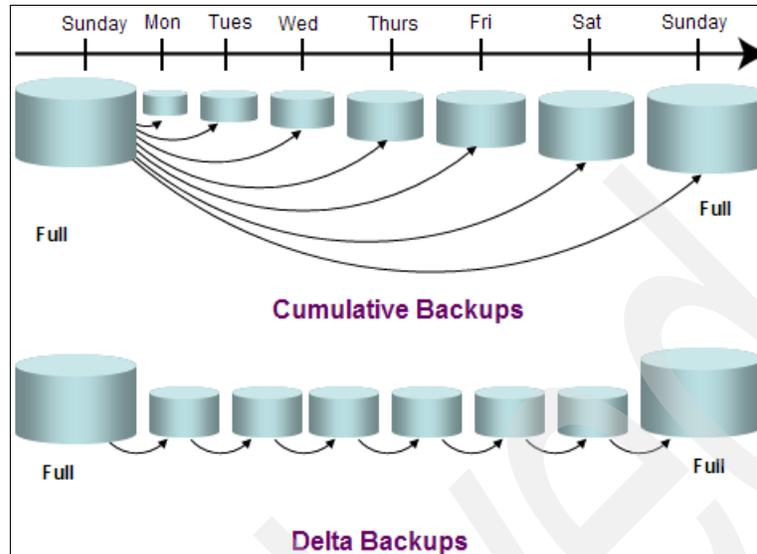


Figure 9-6 Incremental backup

Example 9-2 Incremental backup

```
db2inst1@db2server:~> db2 BACKUP DATABASE invent TO /home/db2inst1/backup
db2inst1@db2server:~> db2 BACKUP DATABASE invent INCREMENTAL TO
/home/db2inst1/incBackup
db2inst1@db2server:~> db2 BACKUP DATABASE invent INCREMENTAL DELTA TO
/home/db2inst1/deltaBackup
```

IBM has tools to assist you with the maintenance activities configuration, because it can be time-consuming to determine when the configuration is required and whether maintenance activities, such as backup operations, need to be run. You can use the Configure Automatic Maintenance wizard within Data Studio, as shown in Figure 9-7 on page 290, or the DB2 Control Center to configure the database maintenance activities. With automatic maintenance, you specify your maintenance objectives, including when automatic maintenance can run. DB2 then uses these objectives to determine if the maintenance activities need to be done and then runs only the required maintenance activities during the next available maintenance window (a user-defined time period for running automatic maintenance activities).

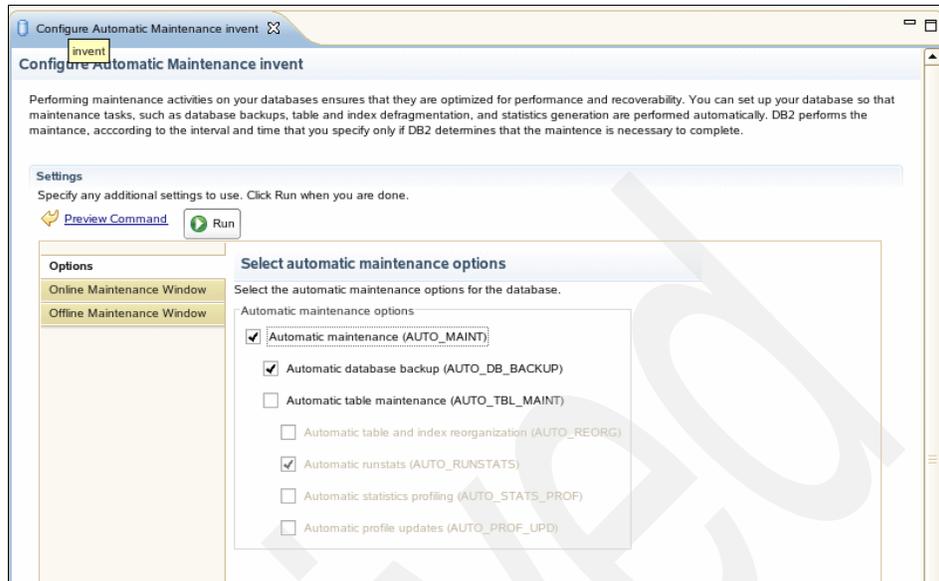


Figure 9-7 Data Studio: Configure Automatic Maintenance

Database recovery

The recover utility performs the necessary restore and roll-forward operations to recover a database to a specified time, based on information found in the recovery history file. When you use this utility, you specify that the database is recovered to a certain point in time or to the end of the log files. The utility will then select the best suitable backup image and perform the recovery operations. Example 9-3 shows how to use the RECOVER DATABASE command.

Example 9-3 Recover database

```
b2inst1@db2server:~> db2 RECOVER DB invent
```

Rollforward Status

```
Input database alias           = invent
Number of nodes have returned status = 1

Node number                   = 0
Rollforward status            = not pending
Next log file to be read      =
Log files processed           = S0000000.LOG - S0000001.LOG
Last committed transaction    = 2009-09-11-19.25.01.000000 Local
```

```
DB20000I The RECOVER DATABASE command completed successfully.
```

There are three types of recovery:

► **Crash recovery**

Crash recovery protects a database from being left in an inconsistent or unusable state, which occurs when transactions are interrupted unexpectedly. Crash recovery rolls back incomplete transactions and completes committed transactions that were still in memory when the crash occurred.

The database manager performs crash recovery automatically if you set the automatic restart (autorestart) database configuration parameter to ON with this command:

```
db2inst1@db2server:~> db2 UPDATE DATABASE CONFIGURATION FOR invent USING  
AUTORESTART ON
```

Or, you can restart the database when a database failure occurs by calling this command:

```
db2inst1@db2server:~> db2 RESTART DATABASE invent
```

DB2 maintains log files, the recovery history file, and the table space change history file to recover data that is lost or damaged.

► **Version recovery**

Version recovery is the restoration of a previous version of the database, using an image that was created during a backup operation. You can use this recovery option for non-recoverable databases (databases without archived logs). You can also use this method with recoverable databases by using the `WITHOUT ROLLING FORWARD` option in the `RESTORE DATABASE` command. A database restore operation will restore the entire database using a backup image that was created earlier. A database backup allows you to restore a database to a state identical that is to the state at the time that the backup was made. However, every unit of work from the time of the backup to the time of the failure is lost, as shown in Figure 9-8 on page 292.

Important: It is necessary to take a backup image on a regular basis, because this recovery method loses the changes made in the database after the backup operation. Version recovery restores the database using incremental, delta, or full backup.

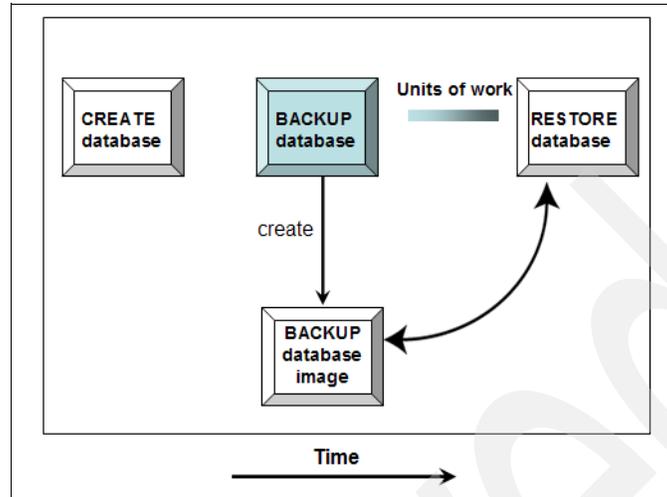


Figure 9-8 Version recovery

► Roll-forward recovery

Use *Roll-forward recovery* to reapply changes that were made by transactions that were committed after a backup was made.

There are two types of roll-forward recovery:

– Database roll-forward recovery

With this type of *roll-forward recovery*, you can tell DB2 to roll forward a database to the state immediately before the failure or you can specify the local time to which you want to roll forward your database. Figure 9-9 on page 293 shows the roll-forward recovery technique of a DB2 database.

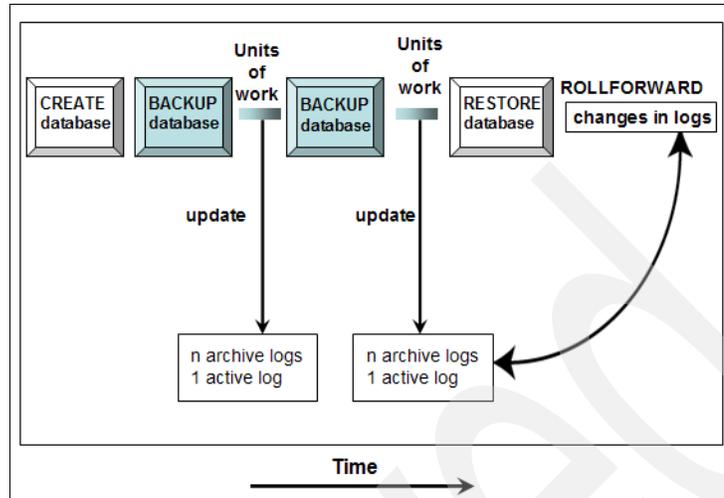


Figure 9-9 Version Recovery: Database roll-forward recovery

– Table space roll-forward recovery

With this type of *roll-forward recovery*, you can tell DB2 to roll forward a table space to either a state immediately before the failure or to a particular point in time. Figure 9-10 shows the roll-forward recovery technique of a DB2 table space.

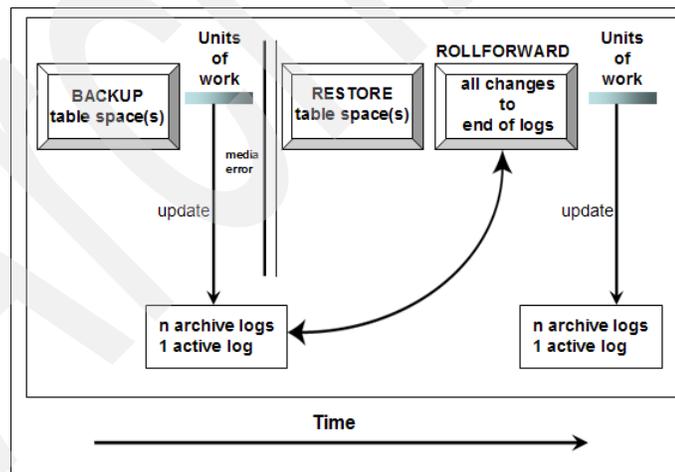


Figure 9-10 Version Recovery: Table space roll-forward recovery

Figure 9-11 on page 294 shows how you can use Data Studio to recover a DB2 database.

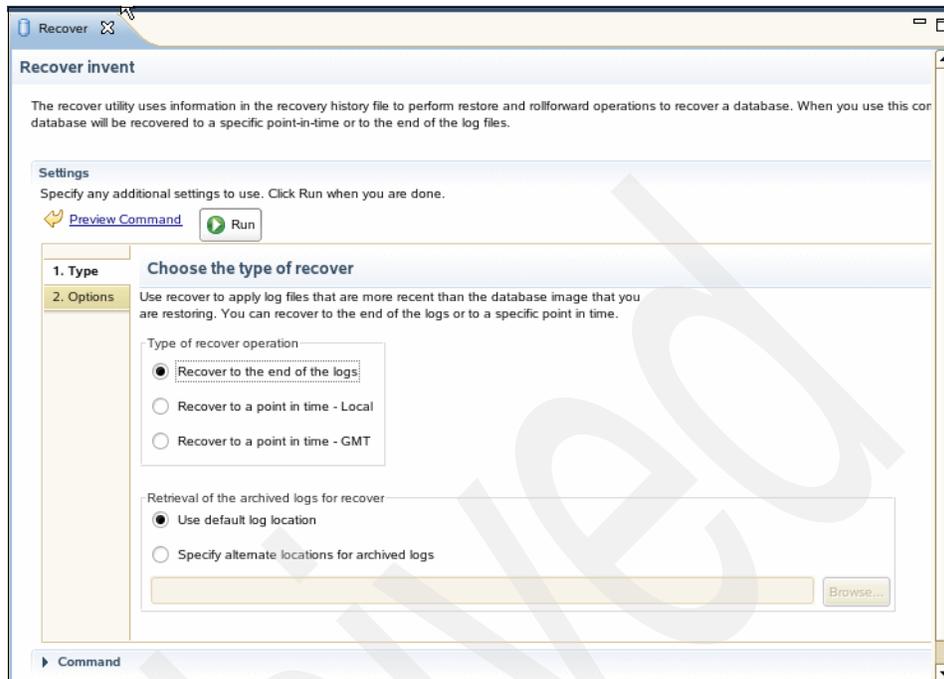


Figure 9-11 Data Studio: Recover option

Database restore

DB2 database restore is as easy as backing up the database. Use the RESTORE utility to perform DB2 database restore. The **restore database** command rebuilds the database data or table space to the state that it was in when the backup copy was made. This utility can overwrite a database with a separate image or restore the backup copy to a new database. You can also use the restore utility to restore backup images in DB2 Version 9.7 that were backed up on DB2 Universal Database Version 8, DB2 Version 9.1, or DB2 Version 9.5.

This RESTORE utility supports full and incremental database restore. Incremental database restore can be automatic or manual. Example 9-4 shows automatic incremental restore, and Example 9-5 on page 295 shows manual incremental restore.

Example 9-4 Automatic incremental restore

```
db2inst1@db2server:~ > db2 RESTORE DATABASE invent INCREMENTAL AUTOMATIC FROM
/home/db2inst1/backup TAKEN AT 20090911214318
```

Example 9-5 Manual incremental restore

```
db2inst1@db2server:~ > db2 RESTORE DATABASE invent INCREMENTAL FROM
/home/db2inst1/backup TAKEN AT 20090909
db2inst1@db2server:~ > db2 RESTORE DATABASE invent INCREMENTAL FROM
/home/db2inst1/backup TAKEN AT 20090910
db2inst1@db2server:~ > db2 RESTORE DATABASE invent INCREMENTAL FROM
/home/db2inst1/backup TAKEN AT 20090911
```

If at the time of the backup operation, the database was enabled for roll-forward recovery, you can take the database to its previous state by invoking the following ROLLFORWARD command after a successful completion of a restore operation:

```
db2inst1@db2server:~ > db2 ROLLFORWARD DATABASE invent COMPLETE
```

Tip: You can also perform DB2 backup and restore using the Data Studio.

You can also execute the RESTORE and ROLLFORWARD utilities from the Data Studio. Figure 9-12 on page 296 shows the restore database window.

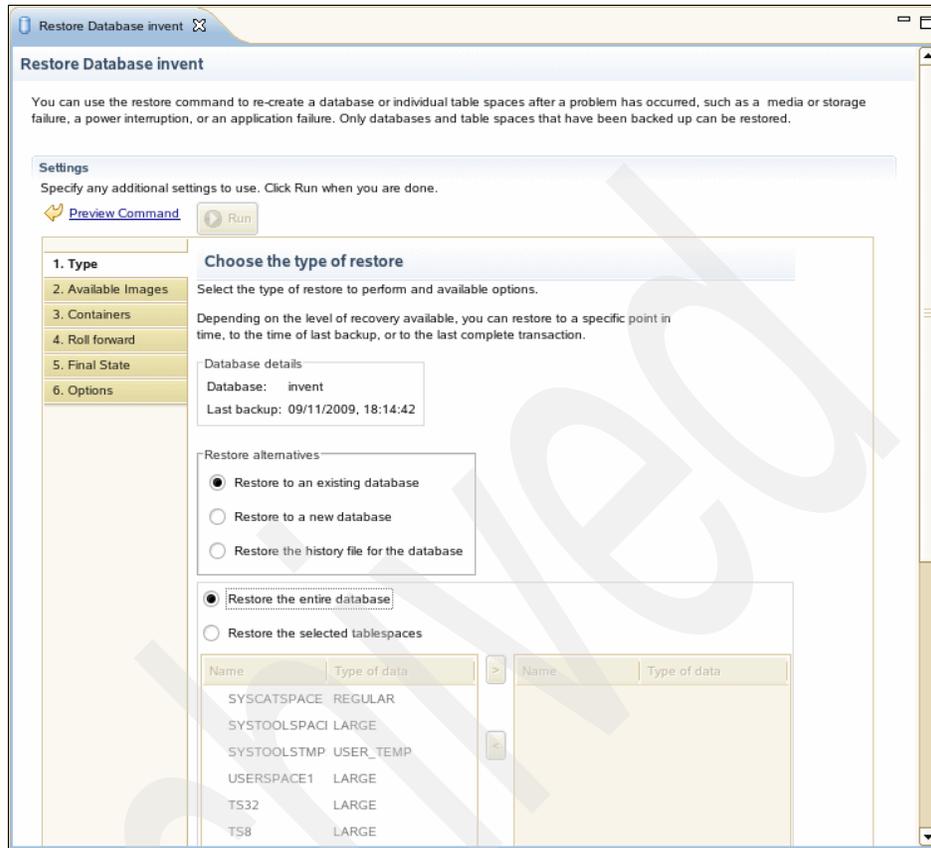


Figure 9-12 Data Studio: Restore option

9.3 Database replication

Database replication is the process of maintaining the same database under multiple servers or systems. It involves synchronizing changes from one database (a source) to another database (a target). This feature is extremely useful for load sharing, fast disaster recovery, and high availability.

IBM provides two solutions that you can use to replicate data from and to relational databases: *SQL replication* and *Q replication*. IBM also provides a solution called *event publishing* for converting committed source changes into messages in an XML or delimited format and for publishing those messages across WebSphere MQ queues to applications.

In SQL replication, committed source changes are staged in relational tables before being replicated to target systems. Q replication is a replication solution that can replicate large volumes of data at low levels of latency. In Q replication, committed source changes are written in messages that are transported through WebSphere MQ message queues to target systems.

Replication is not only supported between two DB2 systems running on separate platforms, but replication is also supported on the following non-DB2 databases: SQL replication supports replicating between DB2 on Linux, UNIX, Windows, z/OS, and iSeries; Informix; Microsoft SQL Server; Oracle; Sybase; and Teradata (target only). Q replication supports DB2 for Linux, UNIX, and Windows, DB2 for z/OS, Informix (target only), Microsoft SQL Server (target only), Oracle (target only), and Sybase (target only).

IBM provides three tools to assist with setting up replication:

► Replication Center

The *Replication Center* is a graphical user interface that you can use to define, operate, and monitor your replication and publishing environments. It comes with the DB2 Administration Client and runs on Linux and Windows systems. The Replication Center provides a single interface to administer your replication environments on various platforms across multiple systems. The Replication Center includes these features:

- Launchpads that show you step by step how to configure basic replication and publishing environments.
- Wizards that help you set up simple to highly customized replication and publishing configurations.
- Profiles that you can customize that let you create replication objects with schemas, names, and other attributes that conform to your own conventions and storage requirements.

You invoke the Replication Center through the DB2 Control Center or by using the `db2rc` command.

► ASNCLP command-line program

The *ASNCLP* program generates SQL scripts for defining and changing replication and publishing environments. The program runs on Linux, UNIX, Windows, and UNIX System Services for z/OS. The ASNCLP program does not run natively on z/OS or System i.

You can use the ASNCLP program to administer SQL replication, Q replication, Classic replication, event publishing, and the Replication Alert Monitor. You can build ASNCLP input files and run them to generate SQL scripts, or you can run ASNCLP commands interactively from an operating system prompt. You can also run the ASNCLP program in

execute-immediately mode, which is useful for operational commands, such as START QSUB, STOP QSUB, or LIST.

► Replication Dashboard

The *Replication Dashboard* is a Web-based, graphical user interface that helps you monitor and manage the health of replication and event publishing.

The dashboard provides an overall summary of replication and publishing configurations in a convenient tabular format with high-level status indicators. You can drill down for more detailed information about queues and queue depth, subscriptions, latency, and exceptions, and generate detailed reports to help track performance or diagnose problems. You can also view up to 24 moving graphs for near-real-time performance information.

You can change program parameters, start, stop, and reinitialize subscriptions, and start and stop queues. The dashboard also provides a convenient way to view alerts from the Replication Alert Monitor program.

9.4 Data movement

In this section, we discuss data movement support by DB2. You use the data movement utilities to transfer data between various tables, databases, or database systems.

9.4.1 DB2 data movement

DB2 provides fast and efficient tools and utilities for data movement across the various systems or for reorganizing data on the same system.

EXPORT utility

DB2 EXPORT is a powerful tool to export your DB2 data quickly from DB2 to the external file system. DB2 EXPORT uses SQL select or an XQuery statement to export tables, views, large objects, or typed tables to one of the three external file formats:

- .DEL: Delimited ASCII format file
- .WSF: Worksheet format, such as Lotus 1-2-3®
- .IXF: Integrated exchange format

You can invoke the EXPORT utility by using the following methods:

- ▶ Through the command line processor (CLP)

You can use the EXPORT utility from CLP by supplying an SQL SELECT or XQuery statement or by providing hierarchical information for typed tables, as shown below:

```
db2inst1@db2server:~ > db2 EXPORT TO invent.ixf of ixf SELECT * FROM  
admin.owners
```

- ▶ Data Studio

You can extract data from a DB2 database using the graphical user interface called Data Studio. This tool allows you to set export options for each table visually. Figure 9-13 shows the usage of the export through the DB2 Control Center.

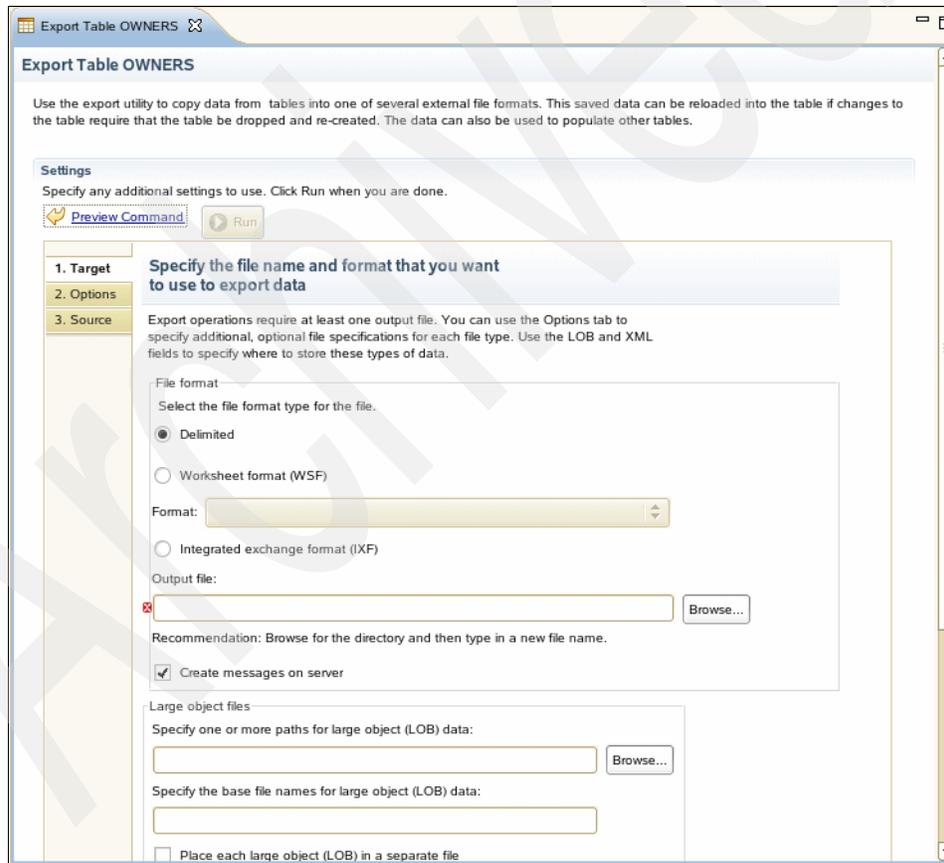


Figure 9-13 Data Studio: Export option

- ▶ DB2 Control Center
You can use the graphical user interface called the DB2 Control Center.
- ▶ ADMIN_CMD stored procedure
Applications can use the ADMIN_CMD stored procedure to run administrative calls, such as the **export** command.
- ▶ An application programming interface (API): db2Export
DB2 provides an API for exporting data that can be used to export data programmatically by importing *db2ApiDf.h* and the db2Export method.

IMPORT utility

You can use the files created with the same syntax as the EXPORT utility to populate data into a new DB2 database on the same system. Or, you can transfer these files to another platform and import or load them to the DB2 database that resides on that platform. The IMPORT utility supports the following file formats:

- ASC: Non-delimited ASCII format file
- .DEL: Delimited ASCII format file
- .WSF: Worksheet format, such as Lotus 1-2-3
- .IXF: Integrated exchange format

Similar to EXPORT, you can use the IMPORT utility with the following methods:

- ▶ The command line processor (CLP)
Use the IMPORT utility from CLP by supplying an SQL INSERT, INSERT UPDATE, or REPLACE option. The example shows the simple IMPORT statement:

```
db2inst1@db2server:~ > db2 IMPORT FROM invent.ixf OF ixf MESSAGES  
msg.txt INSERT INTO admin.owners
```
- ▶ Data Studio
Use Data Studio to import data graphically. Figure 9-14 on page 301 shows the Import wizard.

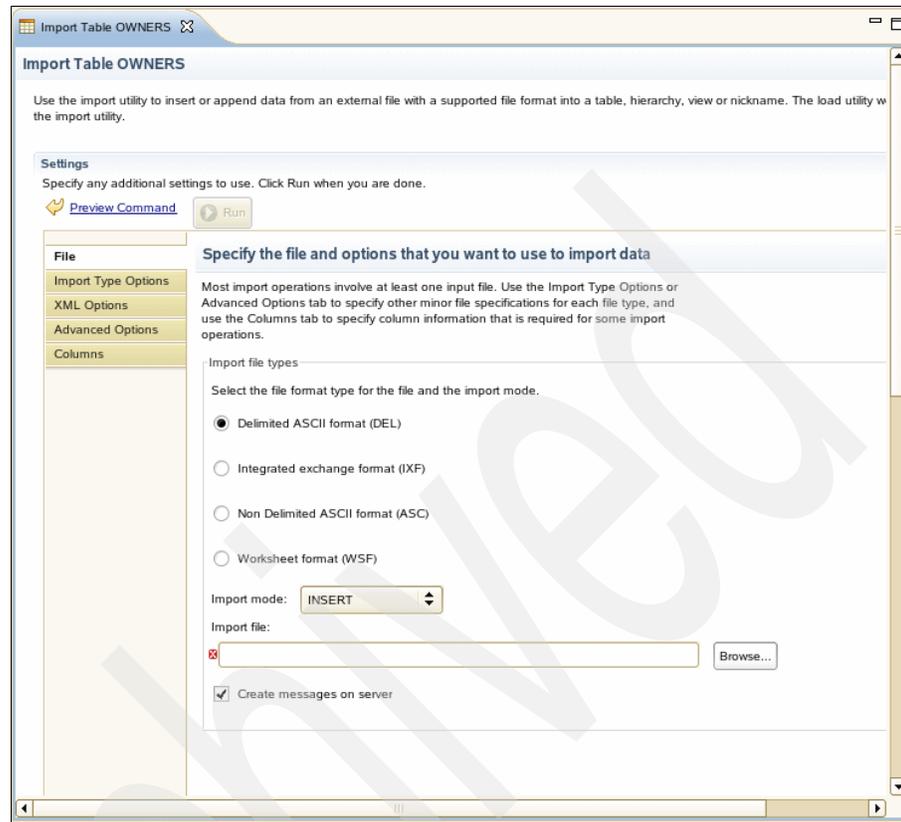


Figure 9-14 Data Studio: Import option

- ▶ DB2 Control Center
Use the DB2 Control Center for importing data visually.
- ▶ ADMIN_CMD stored procedure
Applications can use the ADMIN_CMD stored procedure to run administrative calls and run the **import** command.
- ▶ An application programming interface (API): db2Import
DB2 provides an API for importing data from files, such as files that have been exported using the export tool. This API provides an option to import programmatically by using *db2ApiDf.h* and the db2import method.

DB2MOVE utility

You can copy data using the DB2 EXPORT and IMPORT utilities. But a more efficient way to copy an entire DB2 database schema is by using the DB2 db2move utility. This utility queries the system catalog tables for a specified database and exports the table structure and the contents of each table found to a PC/IXF formatted file. You can use these files to populate another DB2 database. You can run the DB2 db2move utility in three modes:

- ▶ EXPORT mode

In this mode, the db2move utility invokes the DB2 EXPORT utility to extract data from one or more tables and write to PC/IXF formatted files. It also creates a db2move.lst file that contains the names of all of the exported tables and the names of the files to which the table data was written. Use EXPORT mode in this way:

```
db2inst1@db2server:~ > db2move invent EXPORT
```

- ▶ IMPORT mode

In this mode, the db2move utility invokes the DB2 IMPORT utility to recreate tables and indexes from data that is stored in PC/IXF formatted files. You can use the db2move.lst file that is generated in EXPORT mode to get information about tables in the exported files. You can import the exported files by using this command:

```
db2inst1@db2server:~ > db2move invent IMPORT
```

- ▶ LOAD mode

In this mode, the db2move utility invokes the DB2 LOAD utility to populate tables that already exist with data stored in PC/IXF formatted files. Use the db2move.lst file that is generated in EXPORT mode to get information about tables. Load these exported files by using this command:

```
db2inst1@db2server:~ > db2move invent LOAD -1 /home/db2inst1/export
```

The LOAD utility

Use the LOAD utility for moving large amounts of data into a newly created table or into a table that already contains data. The utility can handle most data types, including XML, large objects (LOBs), and user-defined types (UDTs). The LOAD utility is faster than the IMPORT utility, because it writes formatted pages directly into the database, while the IMPORT utility performs SQL INSERTs. The LOAD utility does not fire triggers and does not perform referential or table constraints checking (other than validating the uniqueness of the indexes). The LOAD utility supports the following import sources:

- ▶ ASC: Non-delimited ASCII format file
- ▶ .DEL: Delimited ASCII format file
- ▶ .IXF: Integrated exchange format

There are four modes in which you can execute the LOAD utility:

▶ **INSERT**

In this mode, the LOAD utility appends input data to the table without making any changes to the existing data.

▶ **REPLACE**

In this mode, the LOAD utility deletes existing data from the table and populates it with the input data.

▶ **RESTART**

In this mode, an interrupted load is resumed. In most cases, the load is resumed from the phase in which it failed. If that phase was the load phase, the load is resumed from the last successful consistency point.

▶ **TERMINATE**

In this mode, a failed load operation is rolled back.

You can use the LOAD utility with these methods:

▶ **The command line processor (CLP)**

Use the IMPORT utility from the CLP by supplying an SQL INSERT, INSERT UPDATE, or REPLACE option. This example shows the simple IMPORT statement:

```
db2inst1@db2server:~ > db2 LOAD FROM ownersdata.de1 OF de1 REPLACE INTO  
admin.owner
```

▶ **Data Studio**

Use the Data Studio to load data graphically. Figure 9-15 on page 304 shows the Load wizard.

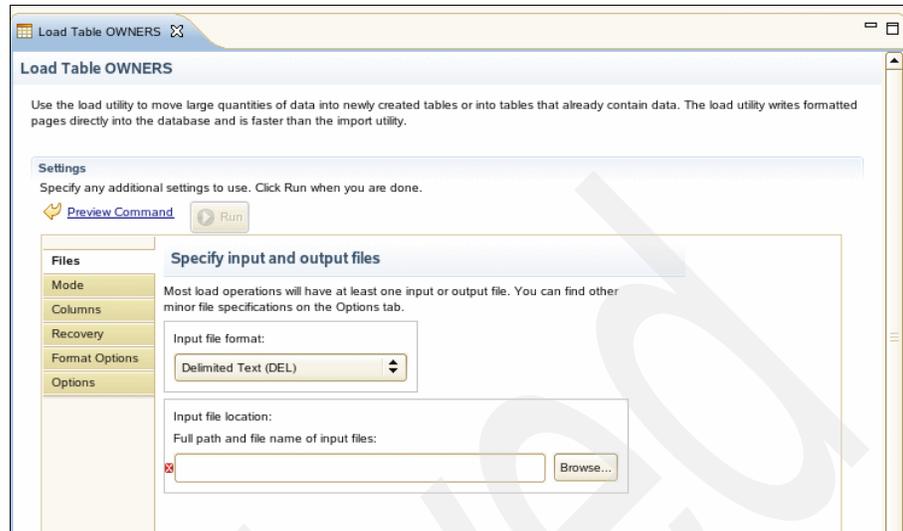


Figure 9-15 Data Studio LOAD option

- ▶ DB2 Control Center
Use the DB2 Control Center to load data visually.
- ▶ ADMIN_CMD stored procedure
Applications can use the ADMIN_CMD stored procedure to run administrative calls, and the ADMIN_CMD stored procedure can be used with the LOAD command.
- ▶ An application programming interface (API): db2load
DB2 provides an API for loading data from files. This API provides an option to perform a load programmatically by using *db2ApiDf.h* and a db2load method

9.5 High availability

High availability (HA) is a business requirement on the system to provide services at all times and to survive disaster, system crash, and glitches without or with minimal interruption of the service. You can achieve HA by using a number of techniques, such as online management of the database system, advanced instance management, suspended I/O, and clustered servers. *Failover* is an advanced feature used for HA, where a workload is transferred from one system to another system quickly and automatically in the case of a failure.

IBM provides software-based HA solutions, such as DB2 High Availability Disaster Recovery (HADR), WebSphere Replication Server Q Replication, SQL Replication, HA clustering software, such as High-Availability Cluster Multi-Processing (HACMP™), and Tivoli® System Automation.

The *DB2 High Availability Disaster Recovery (HADR)* feature is a database replication service that uses a TCP/IP network to propagate database changes (Data Definition Language (DDL) and Data Manipulation Language (DML) statements) from a primary to a standby database, which is monitored by a heartbeat. This HA solution is targeted toward database systems with 24x7 availability and failover times in mere seconds.

With HADR enabled, a second database copy on a separate server is constantly updated while transactions occur. Transactions are logged to ensure both database copies are protected from data loss caused by partial or complete site failures. In the case of an outage, the standby database on the redundant server can quickly take over. Database clients can automatically and transparently to the application be shifted over using the alternate client re-route feature built into the IBM DB2 Data Server client. This solution is extremely cost-effective and places no special requirements on the hardware. Shared storage is not required, because the network is used to replicate all changed data from one database to another database. The only requirement is that you need exactly two TCP/IP network-connected servers. For enhanced redundancy, you can use multiple network adapters. As of DB2 9.7, you can also use the standby server for read-only transactions, therefore utilizing all of the hardware at all times.

For the automatic failover, you use the DB2-integrated component called Tivoli System Automation for Multiplatforms (often referred to as *cluster manager*). This component monitors all resources involved in both systems, that is, DB2 instances, databases, network interfaces, and other components, and initiates the database takeover. At the same time, DB2 is aware of Tivoli System Automation for Multiplatforms and notifies it when a planned outage is necessary, for example, when the database is manually stopped by an administrator. This way, a database administrator can, for the most part, use the usual DB2 commands without having to involve the cluster manager. Every DB2 edition supporting HADR includes all of the necessary software packages, as well as licenses for Tivoli System Automation for Multiplatforms, and offers to install these products during the DB2 setup. DB2 also performs the configuration of Tivoli System Automation for Multiplatforms.

Figure 9-16 on page 306 illustrates an example high availability disaster recover scenario.

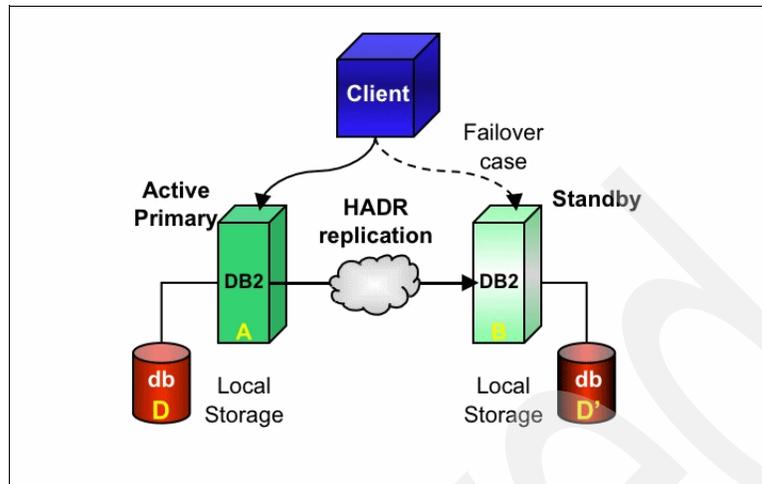


Figure 9-16 High availability disaster recovery scenario

For more information and implementation steps, refer to *High Availability and Disaster Recovery Options for DB2 on Linux, UNIX, and Windows*, SG24-7363.

9.6 Autonomics

Automated task management allows the automation of database management jobs by scheduling activities according to specific requirements. Automated task management is really useful for performing regular maintenance tasks, such as backup, space monitoring, error checking, maintenance, and so forth.

IBM strives for autonomic computing and the development of highly intelligent database systems. The DB2 autonomic computing environment is self-configuring, self-healing, self-optimizing, and self-protecting. By sensing and responding to situations that occur, autonomic computing shifts the burden of managing a computing environment from the database administrator to the technology. Autonomic computing provides users with improved resiliency, higher return on investment (ROI), and lower total cost of ownership (TCO) by accelerating the implementation of new capabilities to gain the highest value possible.

Automatic features provide a high-level summary of the capabilities that comprise the DB2 autonomic computing environment. Next, we provide a more detailed, categorized overview of the product's autonomic capabilities.

Self-Tuning Memory Manager

The *Self-Tuning Memory Manager* feature simplifies the task of memory configuration. This feature responds to significant changes in workload by automatically and iteratively adjusting the values of several memory configuration parameters, including the sizes of the buffer pools, to optimize performance. The memory tuner dynamically distributes available memory resources among several memory consumers, including the sort function, the package cache, the lock list, and buffer pools. By default, Self-Tuning Memory Manager is turned on. You can disable Self-Tuning Memory Manager after creating a database by setting the database configuration parameter `self_tuning_mem` to OFF.

Self-tuning memory manager tunes a database in two modes: DATABASE_MEMORY tuning and no DATABASE_MEMORY tuning. With DATABASE_MEMORY tuning, memory is taken and given back to the OS as necessary, meaning the total amount of memory allocated to a DB2 database can grow over time. Without DATABASE_MEMORY tuning, the overall amount of memory for a given database does not change. Memory tuning still occurs, but it only occurs between separate parameters within the database memory model.

There is a master switch to turn on Self-Tuning Memory Manager. To turn on Self-Tuning Memory Manager, set the database configuration parameter `self_tuning_mem` to ON (which is the default for newly created databases). Then, you can choose which buffer pools and memory-related database parameters to tune. You can automatically tune the following memory-related database configuration parameters:

- ▶ `database_memory`: Database shared memory size
- ▶ `locklist`: Maximum storage for lock list
- ▶ `maxlocks`: Maximum percent of lock list before escalation
- ▶ `pckcachesz`: Package cache size
- ▶ `sheapthres_shr`: Sort heap threshold for shared sorts
- ▶ `sortheap`: Sort heap size

Example 9-6 on page 308 shows how to enable Self-Tuning Memory Manager and how to configure each parameter to be managed by Self-Tuning Memory Manager.

Example 9-6 Self-tuning memory manager configuration

```
db2inst1@db2server:~ > db2 UPDATE DB CFG FOR DATABASE invent USING  
SELF_TUNING_MEM ON  
db2inst1@db2server:~ > db2 ALTER BUFFERPOOL bp32 SIZE AUTOMATIC  
db2inst1@db2server:~ > db2 UPDATE DB CFG FOR DATABASE invent USING locklist  
AUTOMATIC
```

After the `SELF_TUNING_MEM` parameter is set to `ON`, DB2 begins managing and tuning all buffer pools and memory-related database elements. DB2 senses the underlying workload on the system and adjusts the memory accordingly. Self-tuning memory manager can adjust allocated memory from 6-120 times an hour, depending on the changes and current settings. DB2 performs adjustments in a clever way, allowing parameters to have a maximum growth of 50% or a reduction of 20%. All changes are persistent and logged in the `db2diag.log` file and `STMM.log` file.

Self-tuning memory manager is most advantageous in these scenarios:

- ▶ **Unknown memory requirements**
Self-tuning memory manager can tune a database to the optimal configuration in less than an hour.
- ▶ **Tuning buffer pools of multiple page sizes**
Self-tuning memory manager works with buffer pools of multiple page sizes and can easily trade memory between the buffer pools.
- ▶ **Memory-varied workload**
Self-tuning memory manager constantly re-evaluates the memory requirements and will optimize the memory usage based on the currently running workload.

Automatic Storage

The *automatic storage* feature simplifies storage management for table spaces. When you create an automatic storage database, you specify the storage paths where the database manager will place your table space data. Then, the database manager manages the container and space allocation for the table spaces as you create and populate them. By default, automatic storage is turned on:

- ▶ **Automatic storage databases**
Automatic storage is intended to make storage management easier. Rather than managing storage at the table space level using explicit container definitions, storage is managed at the database level and the responsibility of creating, extending, and adding containers is taken over by the database manager.

To create a new database or to alter an existing database to use automatic storage, execute the following DB2 commands:

```
CREATE DATABASE <dbname> AUTOMATIC STORAGE YES ON <storagePaths>  
ALTER DATABASE <dbname> ADD STORAGE ON <storagePaths>
```

► **Automatic storage table spaces**

With automatic storage table spaces, storage is managed automatically. The database manager creates and extends containers as needed up to the limits imposed by the storage paths associated with the database.

To create a new table space or to alter an existing table space to use automatic storage, execute the following DB2 commands:

```
ALTER TABLESPACE <tablespacename> MANAGED BY AUTOMATIC STORAGE  
CREATE TABLESPACE <tablespacename> MANAGED BY AUTOMATIC STORAGE
```

Automatic maintenance

The database manager provides automatic maintenance capabilities for performing database backups, keeping statistics current, and reorganizing tables and indexes as necessary. Performing maintenance activities on your databases is essential in ensuring that they are optimized for performance and recoverability.

Maintenance of your database can include part, or all, of the following activities:

► **Backups**

When backing up a database, the database manager takes a copy of the data in the database and stores it on another medium in case of failure or damage to the original. Automatic database backups help to ensure that your database is backed up properly and regularly so that you do not have to worry about when to back up or know the syntax of the BACKUP command.

► **Data defragmentation (table or index reorganization)**

This maintenance activity can increase the efficiency with which the database manager accesses your tables. Automatic reorganization manages an offline table and index reorganization so that you do not need to worry about when and how to reorganize your data.

► **Data access optimization (statistics collection)**

The database manager updates the system catalog statistics on the data in a table, the data in indexes, or the data in both a table and its indexes. The optimizer uses these statistics to determine which path is necessary to access the data. Automatic statistics collection attempts to improve the performance of the database by maintaining up-to-date table statistics. The goal is to allow the optimizer to choose an access plan based on accurate statistics.

► **Statistics profiling**

Automatic statistics profiling advises when and how to collect table statistics by detecting outdated, missing, or incorrect statistics, and by generating statistical profiles based on query feedback.

It can be time-consuming to determine whether and when to run maintenance activities, which makes the automatic maintenance extremely convenient. You can manage the enablement of the automatic maintenance features simply and flexibly by using the automatic maintenance database configuration parameters.

The easiest way to set up automatic maintenance is to use the graphical administrative tools. You can access the Configure Automatic Maintenance wizard from either the Data Studio or the DB2 Control Center. We show configuring maintenance objectives by using the Data Studio in Figure 9-7 on page 290. The database manager uses these objectives to determine whether the maintenance activities need to be done and runs only the required maintenance activities during the next available maintenance window (a time period that you define in which the server will not be as heavily used).

Configuration Advisor

You can use the *Configuration Advisor* to obtain recommendations for values of the buffer pool size, database configuration parameters, and database manager configuration parameters.

To use the Configuration Advisor, specify the `AUTOCONFIGURE` command for an existing database, or specify `AUTOCONFIGURE` as an option of the `CREATE DATABASE` command.

You can display the recommended values or apply them by using the `APPLY` option of the `CREATE DATABASE` command. The recommendations are based on input that you provide and system information that the Configuration Advisor gathers.

When you create a database, this tool is automatically run to determine and set the database configuration parameters and the size of the default buffer pool (`IBMDEFAULTBP`). The values are selected based on system resources and the intended use of the system. This initial automatic tuning means that your database performs better than an equivalent database that you might create with the default values. It also means that you will spend less time tuning your system after creating the database.

You can run the Configuration Advisor at any time (even after your databases are populated) to have the tool recommend and optionally apply a set of configuration parameters to optimize performance based on the current system characteristics. You can use the graphical database administration tools to run

the Configuration Advisor. Figure 9-4 on page 286 shows the Data Studio Configuration Advisor Wizard.

The values that are suggested by the Configuration Advisor are relevant for only one database per instance. If you want to use the Configuration Advisor on more than one database, each database must belong to a separate instance.

Data compression

You can compress both tables and indexes to save storage. *Compression* is fully automatic. After you specify that a table or index must be compressed using the COMPRESS YES clause of the CREATE TABLE, ALTER TABLE, CREATE INDEX, or ALTER INDEX statements, there is nothing more you must do to manage compression. Temporary tables are compressed automatically; indexes for compressed tables are also compressed automatically, by default. We discuss data compression further in 11.2, “Data compression” on page 386.

Monitoring database health

The *health monitor* is a server-side tool that proactively monitors situations or changes in your database environment that can result in performance degradation or a potential outage. A range of health information is presented without any form of active monitoring on your part. If a health risk is encountered, the database manager informs and advises you about how to proceed. The health monitor gathers information about the system by using the snapshot monitor and does not impose a performance penalty. Furthermore, it does not turn on any snapshot monitor switches to gather information.

Utility throttling

This feature regulates the performance impact of maintenance utilities so that they can run concurrently during production periods. Although the impact policy for throttled utilities is defined by default, you must set the impact priority if you want to run a throttled utility. The throttling system ensures that the throttled utilities run as frequently as possible without violating the impact policy. Currently, you can throttle statistics collection, backup operations, rebalancing operations, and asynchronous index cleanup.

9.7 Workload management

More and more systems are consolidated due to the increasing calculating power of CPUs. Although hardware utilization is increased and costs are reduced, management complexity is increased, requiring more knowledge and efficient handling for the workloads that are running on the system. When consolidating databases from systems holding separate types of data and

accessed by differently behaving workloads, a sophisticated workload management solution is required to ensure a stable and predictable execution environment.

As of DB2 Data Server 9.5, *Workload Management* is built right into the DB2 engine, allowing administrators to monitor and control database activities, such as DDL and DML statements, over their full life cycle. Through definable rules, you program the engine to automatically filter certain workloads and apply execution priorities to control concurrency or activity through the setup of thresholds. Workload Management can explicitly control CPU usage among executing work, detect and prevent so-called “runaway” queries, which for example are exceeding the predicted or configured number of rows returned, the execution time, or the estimated execution costs. Specifically on the AIX platform, DB2 workload management does not stop at a database level and can be tightly integrated with the operating system’s workload management.

For more details and setup information, refer to the IBM DB2 9.7 for Linux, UNIX, Windows Information Center at this Web site:

<http://publib.boulder.ibm.com/infocenter/db21uw/v9r7/index.jsp>

9.8 Database management tools

In this section, we introduce the graphical tools that are available for managing DB2 database servers. Although you can perform all of these tasks using the command prompt, these tools play an important role in allowing administrator to manage the database graphically.

DB2 supports an impressive suite of tools for database management and services. You can perform all of the daily database operations easily and effectively by using the following GUI tools and wizards:

- ▶ Control Center
- ▶ Optim Development Studio
- ▶ Optim Database Administrator
- ▶ IBM Data Studio
- ▶ Data Studio Administration Console

9.8.1 DB2 Control Center

IBM *DB2 Control Center* is the central point from which you can manage your family of DB2 databases, which are running on an array of operating systems in your workplace. A user friendly graphical interface makes your job easier by guiding you through each of the steps in managing the DB2 database.

DB2 Control Center provides a common interface for managing DB2 databases on various platforms. You can run DB2 commands, create DDL statements, and execute DB2 utilities. DB2 Control Center provides point-and-click navigation capabilities to make it easy to find objects, whether you have hundreds or tens of thousands of objects in your database environment. Use it to administer the system, instances, tables, views, indexes, triggers, user-defined types, user-defined functions, packages, aliases, users, or groups.

DB2 Control Center is tightly coupled with other DB2 tools; Figure 9-17 shows a hierarchy of database objects on the leftmost panel and details on the rightmost panel. You start the Control Center by entering the `db2cc` command.

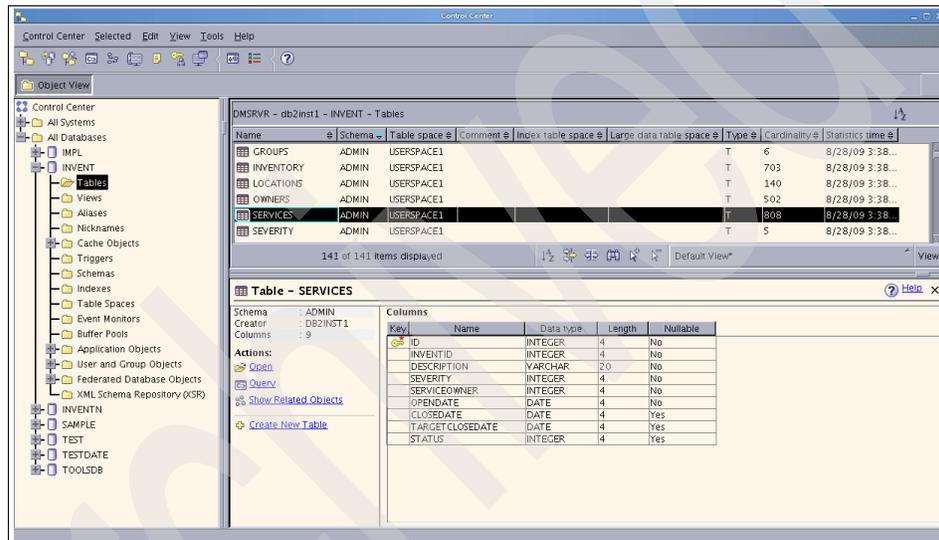


Figure 9-17 DB2 Control Center

You can start the following tools from the Control Center Tools menu:

- ▶ Replication Center
- ▶ Satellite Administration Center Command Center
- ▶ Command Editor
- ▶ Task Center
- ▶ Health Center
- ▶ Journal
- ▶ License Center
- ▶ Configuration Assistant

DB2 Control Center also provides a set of wizards for completing specific administration tasks by taking you through each step, one at a time. The following DB2 wizards are available through the Control Center:

- ▶ Add database partitions launchpad
- ▶ Backup wizard
- ▶ Create database wizard
- ▶ Create database with automatic maintenance
- ▶ Create table space wizard
- ▶ Create table wizard
- ▶ Design Advisor
- ▶ Load wizard
- ▶ Configuration Advisor
- ▶ Restore data wizard
- ▶ Configure database logging wizard
- ▶ Set up activity monitor wizard
- ▶ Set up high availability disaster recovery databases
- ▶ Configure automatic maintenance

9.8.2 IBM Optim and Data Studio tool suite overview

IBM Optim and Data Studio provide a new comprehensive suite of integrated tools for development database administrators and application developers. The tooling for each of these roles is essentially divided between the two main views in the Data Perspective, which are the Data Project Explorer view and the Data Source Explorer view, which is shown in Figure 9-18 on page 315. There are also features to help team members share resources.

A benefit to these new tools is that you can have a single environment to develop and manage your database, which required separate tools in the past. This tool also supports multiple database servers, such as DB2 for Linux, UNIX, Windows, i5, and z/OS, Apache Derby, Informix Dynamic Server, and other database servers. This flexibility makes managing multiple databases much easier.

The IBM Optim and Data Studio tool suite is built on the Eclipse platform and is an Eclipse-based development environment. The *Eclipse platform* is a framework that allows you to create integrated development environments (IDEs); plug-ins exist to allow development in Java, C/C++, PHP, COBOL, Ruby, and more products.

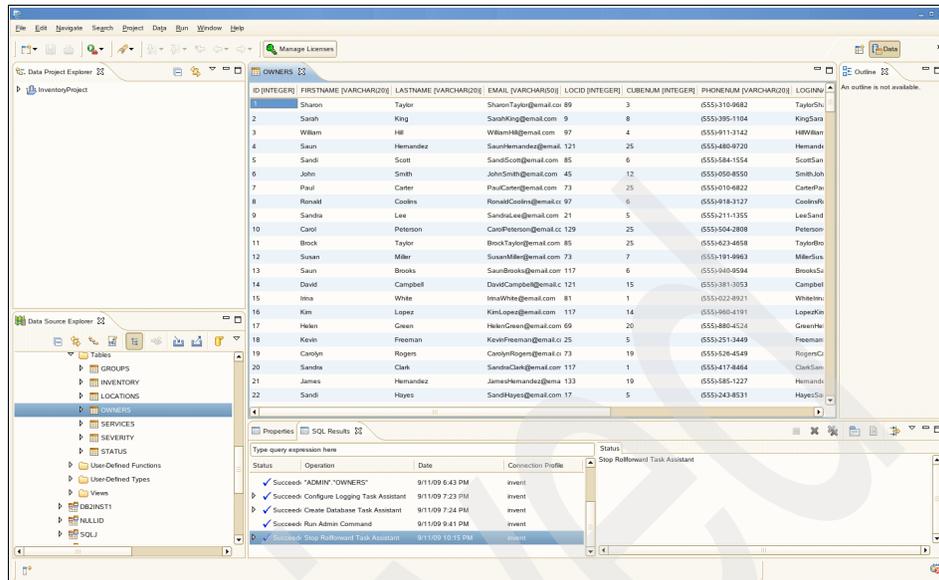


Figure 9-18 IBM Optim and Data Studio tool suite

There are two versions, the Optim Database Administrator and the Optim Development Studio, that you purchase. There is one no-charge version of IBM Data Studio.

Optim Database Administrator

Optim Database Administrator, formerly called Data Studio Administrator, is another GUI tool under the new suite of database management GUI tools. This tool is a powerful and flexible tool that helps you manage your database objects. Optim Database Administrator simplifies DB2 database administration by providing an easy-to-use interface and built-in analysis to automate complex database changes and conversions. Its capabilities make it easy to perform everyday tasks, such as object changes, utilities, commands, configuration changes, and granting or revoking security privileges. Optim Database Administrator enables immediate script execution or the ability to save the script for later execution.

You can use Optim Database Administrator to perform common database administration tasks within the Eclipse environment, such as:

- ▶ Starting or stopping a DB2 instance
- ▶ Backing up a database
- ▶ Creating database objects
- ▶ Modifying database objects
- ▶ Changing configuration settings

- ▶ Granting and revoking security privileges
- ▶ Running DB2 commands and utilities

Optim Database Administrator provides an integrated end-to-end solution that enables you to seamlessly work with multiple products simultaneously.

Optim Database Administrator 2.2 is on Eclipse 3.4. It can share a shell with the following products:

- ▶ Optim Development Studio 2.2
- ▶ Optim Query Tuner 2.2
- ▶ InfoSphere Data Architect 7.5.x.x

Optim Development Studio

Optim Development Studio, which was formerly called Data Studio Developer, is part of the new suite of GUI tools for managing DB2 data and database applications. Optim Development Studio enables database developers to rapidly develop high-quality code by providing a feature-rich database development environment. This environment simplifies database development by providing procedure building and debugging features along with database object management, project management, and optimization tools with Optim pureQuery Runtime.

For data application developers, Optim Development Studio provides the following key features. Working in a data development project in the Data Project Explorer, you can perform these tasks:

- ▶ Develop pureQuery applications in a Java project.
- ▶ Use wizards and editors to create, test, debug, and deploy routines, such as stored procedures and user-defined functions.
- ▶ Use the SQL builder and the SQL editor to create, edit, and run SQL queries.
- ▶ Use Visual Explain to tune routines and SQL queries.
- ▶ Use the Routine debugger to debug stored procedures.
- ▶ Create Web services that expose database operations (SQL SELECT and DML statements, XQuery expressions, or calls to stored procedures) to client applications.
- ▶ Use wizards and editors to develop XML applications.
- ▶ Develop SQLJ applications in a Java project.

Optim Development Studio provides the following key features for database object management. Typically, you perform these tasks on test databases that you use to test your applications.

Working in the Data Source Explorer, you can perform these tasks:

- ▶ Connect to data sources and browse data objects and their properties.
- ▶ Use editors and wizards to create and alter data objects.
- ▶ Modify privileges for data objects and authorization IDs.
- ▶ Copy database objects.
- ▶ Drop data objects from databases.
- ▶ Analyze the impact of your changes.
- ▶ Work with data, including extracting and loading data and inserting XML data into XML columns.
- ▶ Use data diagrams to visualize the relationships between data objects.

If you work on a large team, you can use the following features to enable team members to share resources:

- ▶ You can share data development projects using supported source code control systems.
- ▶ You can share database connection information by importing and exporting this information to XML files.
- ▶ You can customize the user interface to enable and disable visible controls and defaults.

Optim Development Studio 2.2 is on Eclipse Version 3.4, and here are several products with which it can share the shell:

- ▶ Rational Application Developer for WebSphere Software 7.5.x.x
- ▶ Optim Database Administrator 2.2
- ▶ InfoSphere Data Architect 7.5.x.x

IBM Data Studio

There is a free downloadable version called *IBM Data Studio*, which provides an integrated development environment for both administering and developing with IBM data servers. IBM Data Studio is available in two packaging options:

- ▶ The integrated development environment (IDE) package includes all administrative capabilities, as well as an integrated Eclipse development environment for Java, XML, and Web services.
- ▶ The new stand-alone package is a more basic offering that is designed specifically for administrators to get up and running quickly and easily. This package does not include several of the data development capabilities available in the IDE package and cannot be installed with other products to extend its capabilities.

For more information and to download the software, go to this Web site:

<http://www.ibm.com/software/data/optim/>

IBM Data Studio administration console

IBM Data Studio Administration Console provides a no-charge rich Web interface for monitoring your DB2 servers. Use its database health and availability monitoring features to get a summary of system health, view dashboard matrixes, investigate alerts, and troubleshoot problems using expert recommendations. The Data Server Administration Console monitors the health and availability of your connected DB2 data server databases on Linux, UNIX, Windows, and z/OS systems.

With the Data Studio Administration Console, you can perform these tasks:

- ▶ View system health at a glance:
 - Connect to and monitor multiple databases across various platforms from a single console.
 - Configure thresholds for warnings and alerts for key performance indicators.
- ▶ Drill down into alerts and warnings for a database:
 - Examine the details when problems occur with key performance indicators.
 - Filter the display of statistics by time period, database, or application.
- ▶ Browse alert history:
 - Collect and retain alert history for 72 hours.
 - Filter the display of statistics by time period, database, or alert type.
- ▶ Use expert recommendations for problem solving:
 - Review expert recommendations to help solve the problems that are causing a specific alert.
 - View the system and database parameters and statistics associated with a warning or alert.

You can also use Data Studio Administration Console to monitor Q replication and event publishing, generate replication health reports, and perform basic replication operations.

You can launch Data Studio Administration Console from the IBM Data Studio Developer user interface so that you can monitor IBM data servers for status, including database availability, dashboards, and alerts. Figure 9-19 on page 319 shows the Data Studio Administration Console dashboard.

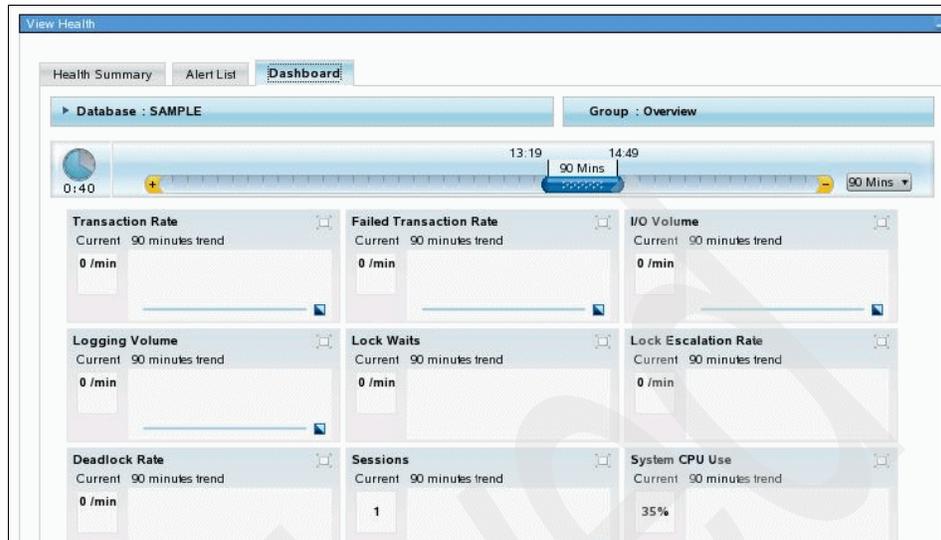


Figure 9-19 Data Studio Administration Console: Dashboard view

Archived

Testing and tuning

In this chapter, we discuss the steps to verify that data and application functionality were ported completely and correctly, including:

- ▶ Test planning
- ▶ Data checking
- ▶ Code and application testing
- ▶ Troubleshooting

We also provide information about how you can check that system behavior has not changed in an undesired way.

Furthermore, we discuss the methods and tools available for DB2 to tune the database in order to achieve optimal performance.

10.1 Test planning

The test planning stages detail the activities, dependencies, and efforts that are required to conduct the test of the converted solution.

10.1.1 Principles of software tests

Remember the following principles of software tests in general:

- ▶ It is not possible to test a non-trivial system completely.
- ▶ Tests are optimizing processes regarding completeness.
- ▶ Always test against expectations.
- ▶ Each test must have reachable goals.
- ▶ Test cases have to contain reachable and non-reachable data.
- ▶ Test cases must be repeatable.
- ▶ Test cases have to be archived in the configuration management system, as well as the source code and documentation.

10.1.2 Test documentation

The test documentation is an extremely important part of the project. The American National Standards Institute (ANSI)/Institute of Electrical and Electronics Engineers (IEEE) Standard 829-1998 for Software Test Documentation describes the exact content of the test document. Refer to this document for more details.

10.1.3 Test phases

A series of well-designed tests must validate all of the stages of the conversion process. A detailed test plan must describe all the test phases, the scope of the tests, and the validation criteria, and specify the time frame. To ensure that the applications operate in the same manner as they did in the source database, the test plan must include data conversion, functional and performance tests, as well as the following other post-conversion assessments:

- ▶ Data conversion testing
- ▶ Functional testing
- ▶ Integration testing
- ▶ Performance testing
- ▶ Volume/load stress testing

- ▶ Acceptance testing
- ▶ Post-conversion tests

10.1.4 Time planning and time exposure

The time planning must be based on realistic and validated estimates. If the estimates for the conversion of the application and database are inaccurate, the entire project plan will slip.

It is always best to tie all test dates directly to their related conversion activity dates, which prevents the test team from being perceived as the cause of a delay. For example, if system testing is to begin after the delivery of the final build, system testing begins the day after delivery. If the delivery is late, system testing starts from the day of delivery, not on a specific date. This approach is called *dependent* or *relative dating*.

Figure 10-1 shows the test phases during a typical conversion project. The definition of the test plans happen at an extremely early moment. The test cases, and all subsequent tasks, must be done for all test phases.

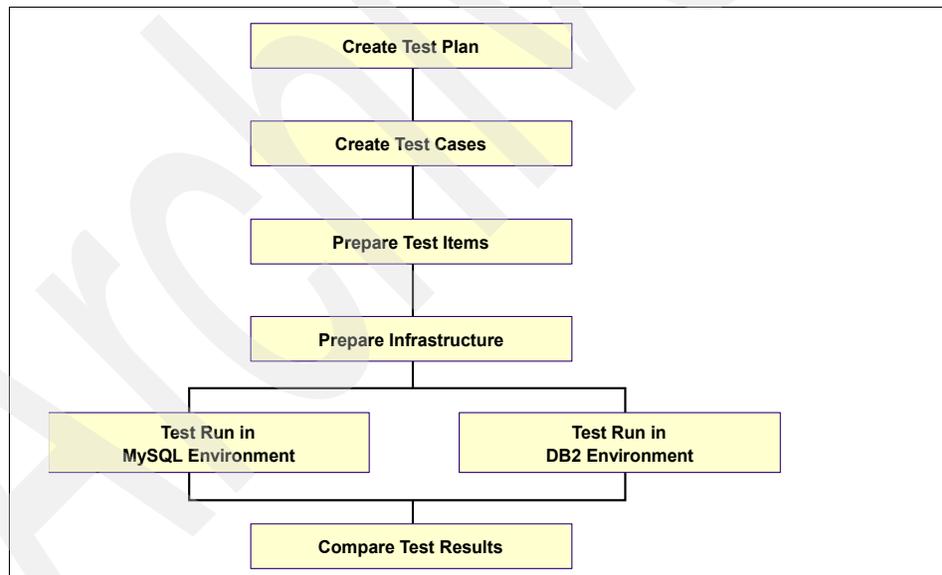


Figure 10-1 Test phases during a conversion project

The time exposure of tests depends on the availability of an existing test plan and already prepared test items. The efforts depend also on the degree of changes during the application and database conversion.

Note: Test efforts can range between 50% and 70% of the total conversion effort.

10.2 Data checking techniques

Data movement is the first focus for any conversion project. Without having all your tables and data properly moved over, all other conversion testing is in vain.

The testing process must detect if all rows were imported into the target database, ensure that all data type conversions were successful, and check random data byte-by-byte. The data checking process must be automated by appropriate scripts. When testing data conversion results, you must perform these steps:

- ▶ Check IMPORT/LOAD messages for errors and warnings.
- ▶ Count the number of rows in the source and target databases and compare them.
- ▶ Prepare scripts that perform data checks.
- ▶ Involve data administration staff familiar with the application and its data to perform random checks.

10.2.1 IMPORT/LOAD messages

You must always check the messages generated by the IMPORT or LOAD commands. Example 10-1 presents messages that are generated by the sample IMPORT command. Read not only the summary at the end of the listing, but also pay attention to the warning messages.

Example 10-1 Sample IMPORT message

```
db2inst1@db2server:~> db2 IMPORT FROM table01Data.txt OF DEL REPLACE INTO  
table01
```

```
SQL3109N The utility is beginning to load data from file "table01Data.txt".
```

```
SQL3148W A row from the input file was not inserted into the table. SQLCODE "-545" was  
returned.
```

```
SQL0545N The requested operation is not allowed because a row does not  
satisfy the check constraint "DB2INST1.TABLE01.SQL090915100543100".  
SQLSTATE=23513
```

SQL3185W The previous error occurred while processing data from row "2" of the input file.

SQL3117W The field value in row "3" and column "1" cannot be converted to a SMALLINT value. A null was loaded.

SQL3125W The character data in row "4" and column "2" was truncated because the data is longer than the target database column.

SQL3110N The utility has completed processing. "4" rows were read from the input file.

SQL3221W ...Begin COMMIT WORK. Input Record Count = "4".

SQL3222W ...COMMIT of any database changes was successful.

SQL3149N "4" rows were processed from the input file. "3" rows were successfully inserted into the table. "1" rows were rejected.

Number of rows read	= 4
Number of rows skipped	= 0
Number of rows inserted	= 3
Number of rows updated	= 0
Number of rows rejected	= 1
Number of rows committed	= 4

As shown in the summary, during the import process one record from the input file was rejected, and three records were inserted into the database. To understand the nature of the warnings, you must look into the data source file and the table definition (use the `db2look` command). The table definition for Example 10-1 on page 324 is shown in Example 10-2, and the data file for Example 10-1 on page 324 is shown in Example 10-3.

Example 10-2 Table definition for Example 10-1

```
db2> CREATE TABLE TABLE01 (  
C1 SMALLINT,  
C2 CHAR(3),  
C3 SMALLINT CHECK( C3 IN (1,2,3)))
```

Example 10-3 Data file for Example 10-1

```
1,"abc",1  
2,"abc",4  
32768,"abc",2  
4,"abcd",3
```

The first row from the input file (Example 10-3 on page 325) was inserted without any warnings. The second row was rejected, because it violated check constraints (warnings SQL3148W, SQL0545N, and SQL3185W). A value of 32768 from the third row was changed to null, because it was out of the SMALLINT data type range (warning SQL3117W) and string abcd from the last row was truncated to abc, because it was longer than the relevant column definition (warning SQL3125W).

The LOAD utility generates messages in a similar format, but because it is designed for speed, it bypasses the SQL engine and inserts data directly into table spaces without constraint checking. Inserting the same table01.unl file (Example 10-3 on page 325) into table01 (Example 10-2 on page 325) with the LOAD utility generates messages without SQL3148W, SQL0545N, and SQL3185W warnings, as shown in Example 10-4.

Example 10-4 LOAD messages

```
db2inst1@db2server:~> db2 LOAD FROM table01Data.txt OF DEL REPLACE INTO table02
```

```
SQL3109N The utility is beginning to load data from file  
"/home/db2inst1/DB2Scripts/chp10/table01Data.txt".
```

```
SQL3500W The utility is beginning the "LOAD" phase at time "09/15/2009  
10:15:23.814793".
```

```
SQL3519W Begin Load Consistency Point. Input record count = "0".
```

```
SQL3520W Load Consistency Point was successful.
```

```
SQL3117W The field value in row "F0-3" and column "1" cannot be converted to a SMALLINT  
value. A null was loaded.
```

```
SQL3125W The character data in row "F0-4" and column "2" was truncated  
because the data is longer than the target database column.
```

```
SQL3227W Record token "F0-3" refers to user record number "3".
```

```
SQL3227W Record token "F0-4" refers to user record number "4".
```

```
SQL3110N The utility has completed processing. "4" rows were read from the  
input file.
```

```
SQL3519W Begin Load Consistency Point. Input record count = "4".
```

```
SQL3520W Load Consistency Point was successful.
```

```
SQL3515W The utility has finished the "LOAD" phase at time "09/15/2009  
10:15:23.836019".
```

```
SQL3107W There is at least one warning message in the message file.
```

```
Number of rows read          = 4  
Number of rows skipped      = 0
```

```
Number of rows loaded      = 4
Number of rows rejected    = 0
Number of rows deleted     = 0
Number of rows committed  = 4
```

A table that has been created with constraints is left by the LOAD command in check pending state. Accessing the table with SQL queries generates a warning:

```
SQL0668N Operation not allowed for reason code "1" on table <TABLE_NAME>.
SQLSTATE=57016.
```

You need to use the SET INTEGRITY SQL statement to move loaded tables into a usable state. Example 10-5 shows a way to validate constraints. All rows that violated constraints will be moved to exception table table01_e.

Example 10-5 Turning integrity checking on

```
db2inst1@db2server:~> db2 CREATE TABLE table02_exp LIKE table02
db2inst1@db2server:~> db2 SET INTEGRITY FOR table02 IMMEDIATE CHECKED FOR
EXCEPTION IN table02 USE table02_exp
SQL3602W Check data processing found constraint violations and moved them to
exception tables.  SQLSTATE=01603
```

The SET INTEGRITY statement has many options, such as turning integrity on only for new data, turning integrity off, or specifying exception tables with additional diagnostic information. To read more about the SET INTEGRITY command, refer to this Web site:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp>

IBM Data Movement Tool LOAD messages and the SET INTEGRITY script

When using the IBM Data Movement Tool to convert, it is simple to review the data conversion logs. The IBM Data Movement Tool uses the DB2 LOAD command to load the converted data into DB2. All load output logs are located in the msg directory in the conversion project output directory. There is a file for each table that was loaded and the log output is in the same format as described in Example 10-4 on page 326.

The IBM Data Movement Tool also generates a script called *db2checkpending.sql* in the conversion project output directory, which, when executed, runs the SET INTEGRITY script on each of the tables.

10.2.2 Data checking

Scripts performing logical data integrity checks automate the data verification process and save administration efforts. For small tables (with fewer than 50,000 rows), you can write a program that compares data byte-by-byte. The program can extract sorted rows from MySQL and DB2 to files in the same ASCII format. Then, perform a binary comparison on the files (on Linux, use the **diff** command) and check to determine if the files are the same. For larger tables, comparing all rows byte-by-byte can be extremely inefficient. Evaluate the data conversion by comparing aggregate values, such as the number of rows; you can create a special table for storing the information about the number of rows in the source MySQL database. You can use table `ck_row_count`, which is shown in Example 10-6, for that purpose.

Example 10-6 Table for storing number of rows (MySQL)

```
CREATE TABLE ck_row_count (  
    tab_name VARCHAR(30), -- table name  
    row_count INT, -- number of rows  
    sys_name CHAR(3), -- code to distinguish the system: MYS or DB2  
    time_ins DATE -- time when the count was performed  
)
```

For each table, you must count the number of rows and store the information in the `CK_ROW_COUNT` table. You can use the following `INSERT` statement for that purpose:

```
INSERT INTO ck_row_count SELECT 'tab_name', COUNT(*), 'MYS', sysdate() FROM  
tab_name
```

You can manually convert the table `ck_row_count` and its data to the target DB2 database. Example 10-7 presents the DB2 version of the table.

Example 10-7 Table for storing number of rows (DB2)

```
CREATE TABLE ck_row_count (  
    tab_name VARCHAR(30),  
    row_count INT,  
    sys_name CHAR(3),  
    time_ins TIMESTAMP  
)
```

On the DB2 system, repeat the counting process with the equivalent `INSERT` statement:

```
INSERT INTO ck_row_count SELECT 'tab_name', COUNT(*), 'DB2', CURRENT  
TIMESTAMP FROM tab_name
```

After performing the described steps, DB2 table CK_ROW_COUNT will contain information about the number of rows counted in the MySQL and DB2 databases. The records in the table will look similar to Example 10-8.

Example 10-8 Sample table ck_row_count contents

```
SELECT * FROM ck_row_count
```

TAB_NAME	ROW_COUNT	SYS_NAME	TIME_INS
inventory	703	DB2	2009-09-15-10.53.46.573998
services	808	DB2	2009-09-15-10.55.18.782980
groups	6	DB2	2009-09-15-10.55.45.852181
owners	502	DB2	2009-09-15-10.56.02.341948
locations	140	DB2	2009-09-15-10.56.15.269177
severity	5	DB2	2009-09-15-10.56.31.337119
status	7	DB2	2009-09-15-10.56.51.599962

Having the information about the number of rows in an SQL table is convenient, because with a single query, you can get the table names that contain a different number of rows in the source and target database:

```
SELECT tab_name FROM (SELECT DISTINCT tab_name, row_count FROM
ck_row_count) AS t_temp GROUP BY t_temp.tab_name HAVING(COUNT(*) > 1)
```

You can extend this approach for comparing the number of rows for additional checking, such as comparing the sum of numeric columns. Here are the steps that summarize the technique:

1. Define check sum tables on the source database and characterize the scope of the computation.
2. Perform the computation and store the results in the appropriate check sum tables.
3. Convert the check sum tables just as you convert the other user tables.
4. Perform equivalent computations on the target system, and store the information in the converted check sum tables.
5. Compare the computed values.

Table 10-1 on page 330 provides computations for selected database types. The argument for the DB2 SUM() function is converted to DECIMAL type, because, in most cases, the SUM() function returns the same data type as its argument, which can cause arithmetic overflow. For example, when calculating the sum on an INTEGER column, if the result exceeds the INTEGER data type range, error SQL0802N is generated: Arithmetic overflow or other arithmetic exception occurred. Converting the argument to DECIMAL eliminates the error.

Table 10-1 Aggregations for data conversion verification

Data type	MySQL operation	DB2 operation
numeric(<precision>,<scale>)	sum(<u>val</u>)	sum(cast(<u>val</u> as decimal(31,<scale>)))
date	sum(trunc(<u>val</u> - to_date ('0001/01/02','yyyy/mm/dd')))	sum(cast(days(<u>val</u>) as decimal(31,1)))
variable length character	sum(length(<u>val</u>))	sum(cast(length(<u>val</u>) as decimal(31,0)))
fixed length character	sum(length(rtrim(<u>val</u>)))	sum(cast(length(rtrim(<u>val</u>))as decimal(31,0)))

Data checking with IBM Data Movement Tool

The IBM Data Movement Tool can simplify the process of data checking your converted data. The tool generates a script called rowcount in the conversion project output directory. When executed, this script connects to the MySQL and DB2 databases and counts the rows for each table in each database. The output is logged into a <database_name>.tables.rowcount file for easy comparison. Example 10-9 shows the output file for our sample conversion.

Example 10-9 Sample output inventory.tables.rowcount

```
db2inst1@db2server:/opt/ibm/IBMDDataMovementTool/migr> cat
inventory.tables.rowcount
mysql                               db2
mysql.groups                        : 6          "ADMIN"."GROUPS"           : 6
mysql.inventory                     : 703       "ADMIN"."INVENTORY"       : 703
mysql.locations                     : 140       "ADMIN"."LOCATIONS"       : 140
mysql.owners                        : 502       "ADMIN"."OWNERS"          : 502
mysql.services                      : 808       "ADMIN"."SERVICES"        : 808
mysql.severity                      : 5         "ADMIN"."SEVERITY"        : 5
mysql.status                        : 7         "ADMIN"."STATUS"          :
7
```

10.3 Code and application testing

The most important part of the testing process is to verify that each component of the system functions as it did prior to conversion. You must verify the functionality of all of the components of the relational database management system (RDBMS).

10.3.1 Checking the application code

The scope of application testing depends on the converted application. For self-built applications, you must start the testing process with the application queries. You must test all of the queries independently to ensure that they return the expected results. With the application queries successfully converted, you must rebuild the surrounding client programs and test the application against the target database. You need to run each module of the application and possibly each form on each window and check them for errors or improper functionality. You must check all supported interfaces also.

An important issue is documentation of all the test conditions, including operations performed, application windows opened, input data used for testing, and results. For larger projects, the documenting component can become overwhelming, increasing the need for specialized software. By definition, new applications cannot be fully tested. In the conversion project, the application testing is an iterative process of planning, designing test cases, executing the test cases, and finally evaluating and analyzing the results.

Along with the functional testing, you must also check the application against performance requirements. Because there are many architectural differences between MySQL and DB2, certain SQL operations might require further optimization. Observing the performance differences in the early testing stages increases the chance of preparing more optimal code for the new environment.

Before going into production, you must verify the converted database under high volumes and loads. These tests must emulate the production environment, and these tests can determine if further application or database tuning is necessary. The stress load can also reveal other hidden problems, such as locking issues, which can be observed only in a production environment.

10.3.2 Security testing

Before going into production, you must verify security. MySQL handles security differently than DB2, so it is not trivial to compare the user rights between the two systems.

MySQL users and privileges are resolved in DB2 with operating system users and groups. You must compare the list of MySQL users to the equivalent DB2 operating system users. You must verify all of DB2's authorities to allow the correct individuals to connect to the database. You must verify all of the privileges for all database objects.

10.4 Troubleshooting

The first step of problem determination is to know what information is available to you. When DB2 performs an operation, an associated return code is returned. The return code is displayed to the user in the form of an informational or error message. These messages are logged into diagnostic files depending on the diagnostic level that is set in the DB2 configuration. In this section, we discuss the DB2 diagnostic logs, error message interpretations, and tips, which might help with problem determination, troubleshooting, and resolutions to specific problems.

Perform the following actions when experiencing a DB2-related problem:

- ▶ Check related messages.
- ▶ Explain error codes.
- ▶ Check documentation.
- ▶ Search through available Internet resources.
- ▶ Review authorized program analysis reports (APARs) for the current fix pack level.
- ▶ Use the available tools to narrow the problem.
- ▶ Ask IBM for support.

10.4.1 Interpreting DB2 informational messages

Start your investigation from the return code. DB2 provides a return code for every operation performed in the form of *CCCnnnnnS*. The prefix *CCC* identifies the DB2 component that is returning the message; the *nnnnn* is a three to five digit error code, which is also referred to as the *SQLCODE*; and the *S* is a severity indicator. For example, in SQL0289N, the SQL identifier represents a message from the Database Manager, the SQLCODE is 0289, and N indicates an error message.

Here is the complete list for DB2 error messages to prefix your reference:

- ▶ ASN: Replication messages
- ▶ CCA: Client Configuration Assistant messages
- ▶ CLI: Call level interface messages
- ▶ DB2: Command line processor messages
- ▶ DBA: Control Center and Database Administration Utility messages
- ▶ DBI: Installation or configuration messages
- ▶ EXP: Explain utility messages
- ▶ FLG: Information Catalog Manager messages

- ▶ LIC: DB2 license manager messages
- ▶ SAT: Satellite messages
- ▶ SPM: Synch Point Manager messages
- ▶ SQJ: Embedded SQLJ in Java messages
- ▶ SQL: Database Manager messages

The two severity indicators are:

- ▶ W: Indicates warning or informational messages
- ▶ N: Indicates error messages

DB2 also provides detailed information for each message. The full error message describes the nature of the problem in detail along with potential user responses. To display the full message for the DB2 return code, you can use the DB2 command **db2 ? error-code** in Linux or AIX. Because the question mark (?) is a special character, you must separate the DB2 command and the error code with a double quotation mark ("). See Example 10-10.

Example 10-10 Explaining error codes

```
db2 "? sq10289"
SQL0289N Unable to allocate new pages in tablespace "<tablespace-name>".
```

Explanation:

Explanation:

One of the following conditions is true on one or more database partitions:

- 1 One of the containers assigned to this SMS table space has reached the maximum file size. This is the likely cause of the error.
- 2 All the containers assigned to this DMS table space are full. This is the likely cause of the error.

[...]

You can find the complete information about the DB2 message format and a listing of all the messages in the *Messages Reference, Volume 1*, SC27-2450-00, and *Messages Reference, Volume 2*, SC27-2451-00, which are available online at this Web site:

<http://www.ibm.com/support/docview.wss?rs=71&uid=swg27015148>

10.4.2 DB2 tools for troubleshooting

The following tools are available to help collect, format, or analyze diagnostic data. We discuss several of these tools in more detail in the next sections:

- ▶ **db2dart**

Use the **db2dart** command to verify the architectural correctness of databases and their associated objects. It can also be used to display the contents of database control files in order to extract data from tables that might otherwise be inaccessible.

- ▶ **db2diag**

The **db2diag** tool serves to filter and format the volume of information that is available in the `db2diag` log files. Filtering records in `db2diag` log files can reduce the time that is required to locate the records needed when troubleshooting problems.

- ▶ **db2greg**

You can view and edit the Global Registry with the **db2greg** tool.

- ▶ **db2level**

The **db2level** command helps you determine the version and service level (build level and fix pack number) of your DB2 instance.

- ▶ **db2look**

There are many times when it is advantageous to be able to create a database that is similar in structure to another database. For example, rather than testing new applications or recovery plans on a production system, it makes more sense to create a test system that is similar in structure and data and perform tests against the test system without adversely affecting the production system. You can use the **db2look** tool to extract the required DDL statements that are needed to reproduce the database objects of one database in another database. The tool can also generate the required SQL statements to replicate statistics from one database to another database, and any other statements needed to replicate the database configuration, database manager configuration, and registry variables.

- ▶ **db2ls**

With the ability to install multiple copies of DB2 products on your system and with the extended flexibility to install particular DB2 products and features in any paths of your choice, you need a tool to help you keep track of what is installed and where it is installed. On supported Linux and UNIX operating systems, the **db2ls** command lists the DB2 products and features that are installed on your system, including the DB2 Version 9 HTML documentation.

- ▶ db2pd
Use the **db2pd** tool for troubleshooting, because it can return quick and immediate information from the DB2 memory sets.
- ▶ db2support
When collecting information for a DB2 problem, the most important DB2 utility is db2support. The db2support utility automatically collects all of the DB2 and system diagnostic information that is available. It also has an optional interactive “Question and Answer” session, which poses questions about the circumstances of your problem.
- ▶ Traces
If you experience a recurring and reproducible problem with DB2, tracing sometimes allows you to capture additional information about it. Under normal circumstances, only use a trace when asked by IBM Software Support to use a trace. The process of taking a trace entails setting up the trace facility, reproducing the error, and collecting the data.

10.4.3 DB2 diagnostic logs

DB2 logs every return code in diagnostic logs based on the diagnostic level set in the database manager configuration. When investigating DB2 problems, you can obtain the essential information in diagnostic log files that are generated by DB2:

- ▶ Administration Notification log or Windows Event log
- ▶ The db2diag.log file
- ▶ Trap files
- ▶ DB2 dump files
- ▶ Core files (Linux/UNIX only)

Administration notification log

DB2 also provides diagnosis information to the administration notification log in the case of a failure. On Linux and UNIX platforms, the administration notification log is a text file called the *<instance>.nfy* file, where *<instance>* is the name of the DB2 instance. It is located in the `$HOME/sqllib/db2dump` directory. On Windows, all administration notification messages are written to the Event Log.

The DBM configuration parameter NOTIFYLEVEL specifies the level of information to be recorded:

- ▶ 0 - No administration notification messages captured (not recommended)
- ▶ 1 - Fatal or unrecoverable errors
- ▶ 2 - Immediate action required
- ▶ 3 - Important information, no immediate action required (default)
- ▶ 4 - Informational messages

DB2 is not the only product that can write to the notification logs, tools such as the Health Monitor, Capture and Apply programs, and user applications can also write to these logs using the *db2AdminMsgWrite* API function.

db2diag.log

The `db2diag.log` file is most frequently used file for DB2 problem investigation. You can find this file in the DB2 diagnostic directory, defined by the `DIAGPATH` variable in the database manager configuration. If the `DIAGPATH` parameter is not set, by default, the directory is located at this path for Linux and UNIX:

```
$HOME/sqllib/db2dump
```

`$HOME` is the home directory of the DB2 instance.

The directory is located at this path for Windows:

```
<INSTALL_PATH>\<DB2INSTANCE>
```

`<INSTALL_PATH>` is the directory where DB2 is installed, and `<DB2INSTANCE>` is the name of the DB2 instance.

The database manager configuration parameter `DIAGLEVEL` controls how much information is logged to the `db2diag.log` file. Valid values can range from 0 - 4:

- ▶ 0 - No diagnostic data captured
- ▶ 1 - Severe errors only
- ▶ 2 - All errors
- ▶ 3 - All errors and warnings (default)
- ▶ 4 - All errors, warnings, and informational messages

Most of the time, the default value is sufficient for problem determination. In certain cases, especially on development or test systems, you can set the parameter to 4 to collect all informational messages. However, ensure that you focus on the database activities and the size that is available on the file system, because this information can cause performance issues due to the large amounts of data recorded in the file. Setting `DIAGLEVEL` to 4 can also make the file extremely large and harder to read.

This `db2diag.log` file includes this information:

- ▶ A diagnostic message (beginning with `DIA`) explaining the reason for the error
- ▶ Application identifiers, which allow you to match error entries with corresponding application or DB2 server processes
- ▶ Any available supporting data, such as `SQLCA` data structures, and pointers to the location of any extra dump or trap files
- ▶ Administrative events, such as backup and restore start and finish

Example 10-11 contains an extract of a db2diag.log file taken at DIAGLEVEL 3.

Example 10-11 Example of a db2diag.log file

```
1 2007-05-18-14.20.46.973000-240 2 I27204F655 3 LEVEL: Info
4 PID : 3228 5 TID : 8796 6 PROC : db2syscs.exe
7 INSTANCE: DB2MPP 8 NODE : 002 9 DB : WIN3DB1
10 APPHDL : 0-51 11 APPID: 9.26.54.62.45837.070518182042
12 AUTHID : UDBADM
13 EDUID : 8796 14 EDUNAME: db2agntp (WIN3DB1) 2
15 FUNCTION: DB2 UDB, data management, sqlInitDBCB, probe:4820
16 DATA #1 : String, 26 bytes
Setting ADC Threshold to:
DATA #2 : unsigned integer, 8 bytes
1048576
```

We next explain the db2diag.log file entries. The numbers bolded in the example correspond to the following numbers:

1. A time stamp and time zone for the message.
2. The record ID field. The recordID of the db2diag.log file specifies the file offset at which the current message is being logged (for example, “27204”) and the message length (for example, “655”) for the platform where the DB2 diagnostic log was created.
3. The diagnostic level associated with an error message, for example, Info, Warning, Error, Severe, or Event.
4. The process ID.
5. The thread ID.
6. The process name.
7. The name of the instance generating the message.
8. For multi-partition systems, the database partition generating the message. (In a non-partitioned database, the value is “000”.)
9. The database name.
10. The application handle. This value aligns with that used in db2pd output and lock dump files. It consists of the coordinator partition number followed by the coordinator index number, separated by a dash.
11. Identification of the application for which the process is working. In this example, the process generating the message is working on behalf of an application with the ID 9.26.54.62.45837.070518182042.

The TCP/IP-generated application ID is composed of three sections:

- i. **IP address:** It is represented as a 32-bit number displayed as a maximum of 8 hexadecimal characters.

- ii. **Port number:** It is represented as 4 hexadecimal characters.
 - iii. A **unique identifier** for the instance of this application.
12. The authorization identifier.
 13. The engine dispatchable unit identifier.
 14. The name of the engine dispatchable unit.
 15. The product name (“DB2”), component name (“data management”), and function name (“sqlInitDBCB”) that is writing the message (as well as the probe point (“4820”) within the function).
 16. The information returned by a called function. There might be multiple data fields returned.

Trap files

The database manager generates a *trap file* if it cannot continue processing due to a trap, segmentation violation, or exception.

All signals or exceptions received by DB2 are recorded in the trap file. The trap file also contains the function sequence that was running when the error occurred. This sequence is sometimes referred to as the “*function call stack*” or “*stack trace*.” The trap file also contains additional information about the state of the process when the signal or exception was caught.

A trap file is also generated when an application is forced off the system while running a fenced thread-safe routine. The trap occurs as the process is shutting down. This trap file is not a fatal error, and it is nothing to be concerned about.

The files are located in the directory specified by the *DIAGPATH* database manager configuration parameter.

On all platforms, the trap file name begins with a *process identifier* (PID), followed by a *thread identifier* (TID), followed by the partition number (000 on single partition databases), and concludes with *.trap.txt*.

There are also diagnostic traps, which are generated by the code when certain conditions occur that do not warrant crashing the instance, but where it might be useful to verify values within the stack. These traps are named with the PID in decimal format, followed by the partition number (0 in a single partition database).

The following example resembles a trap file with a process identifier (PID) of 6881492 and a thread identifier (TID) of 2.

```
6881492.2.000.trap.txt
```

The following example is a trap file whose process and thread are running on partition 10.

```
6881492.2.010.trap.txt
```

You can generate trap files on demand using the **db2pd** command with the *-stack all* or *-dump* option. In general, though, only run this command as requested by IBM Software Support.

Dump files

When DB2 determines that extra information is required for collection due to an error, it often creates binary dump files in the diagnostic path. The binary dump file is named with the process or thread ID that failed, the node where the problem occurred, and ends with the `.dump.bin` extension, as shown in this example:

```
6881492.2.010.dump.bin
```

When a dump file is created or appended, an entry is made in the `db2diag.log` file indicating the time and the type of data that is written.

Core files (Linux/UNIX)

If a program terminates abnormally, a *core file* is created by the system to store a memory image of the terminated process. Errors, such as memory address violations, illegal instructions, bus errors, and user-generated quit signals, cause core files to be dumped.

These files are located in the directory that is specified by the `DIAGPATH` database manager configuration parameter.

10.4.4 DB2 support information

Identifying what information is required to resolve problems is another important step. All conditions that define the problem are essential when you try to find the solution by searching through the available Internet resources or contacting DB2 support.

Maintenance version

You can use the `db2level` utility to check the current version of DB2. As shown in Figure 10-2 on page 340, the utility returns information about the installed maintenance updates (fix packs), the word length used by the instance (32-bit or 64-bit), the build date, and other code identifiers. We recommend that you periodically check to determine if the newest available fix packs are installed.

DB2 maintenance updates are freely available at:

<ftp://ftp.software.ibm.com/ps/products/db2/fixes>

```
db2inst1@db2server:~> db2level
DB21085I Instance "db2inst1" uses "32" bits and DB2 code release "SQL09070"
with level identifier "08010107".
Informational tokens are "DB2 v9.7.0.0", "s090521", "LINUXIA3297", and Fix Pack
"0".
Product is installed at "/opt/ibm/db2/V9.7".
```

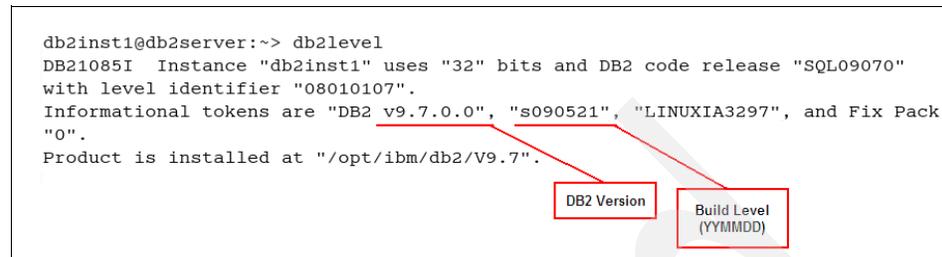


Figure 10-2 Sample db2level output

The 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 that is related to the configuration and the system environment.

The output of the program is stored in a single compressed file named db2support.zip, which is located in the directory specified as part of the **db2support** command options.

In one simple step, the tool can gather database manager snapshots, configuration files, and operating system parameters, which can help you determine the problem more quickly. This example is a sample call of the utility:

```
db2support . -d invent -c
```

The dot represents the current directory where the output file is stored. The rest of the command options are not required and can be omitted. The -d and -c clauses instruct the utility to connect to the invent database and to gather information about database objects, such as table spaces, tables, or packages.

DB2 Technical Support site

An invaluable place to look when experiencing a problem is the DB2 Technical Support site for Linux, Windows, and UNIX, which is located on the Web at this Web site:

http://www.ibm.com/software/data/db2/support/db2_9/

The Web site has the most recent copies of documentation, a knowledge base to search for technical recommendations or DB2 defects, links for product updates, the latest support news, and other useful DB2-related links.

To find related problems, prepare words that describe the issues, such as the commands that were run, symptoms, or tokens from the diagnostic messages. You can use these words as search terms in the DB2 Knowledge Base. The Knowledge Base offers an option to search through DB2 documentation, TechNotes, and DB2 defects (APARs). TechNotes are recommendations and solutions for specific problems.

Authorized Program Analysis Reports (APARs) are defects in the DB2 code that have been discovered by clients and that require a fix. APARs have unique identifiers and are always specific to a particular version, but they can affect multiple products in the DB2 family that run on multiple platforms. Fixes for APARs are provided through the DB2 fix packs.

On the DB2 support site, you can search for closed, open, and HIPER APARs. A closed status for an APAR indicates that a resolution for a problem has been created and included in a specific fix pack. Open APARs represent DB2 defects that are currently being addressed or are waiting to be included in the next available fix pack. High-Impact or PERvasive (HIPER) APARs are critical problems that you must review to assess the potential affect of staying at a particular fix pack level.

The DB2 Technical Support site offers e-mail notification of critical or pervasive DB2 client support issues, including HIPER APARs and fix pack alerts. To subscribe to it, follow the DB2 Alert link on the Technical Support main page.

You can also send DB2 for Linux, UNIX, and Windows questions to:
askdata@ca.ibm.com.

Calling the IBM Software Support Center

If the problem appears too complex to solve on your own, you can contact the *IBM Software Support Center*. In order to understand and resolve your support service request in the most expedient way, it is important that you gather information about the problem and have it available when you talk to the software specialist.

Guidelines and reference materials (which you might need when calling IBM support), as well as the telephone numbers, are available in the IBM Software Support Guide at this Web site:

<http://techsupport.services.ibm.com/guides/handbook.html>

10.4.5 Monitoring tools

Tuning and troubleshooting a database can be a complex process. DB2 comes with various tools, functions, and applications that make this task much easier. One of the monitoring tools is the *DB2 monitoring utility*, which can collect information about many system activities, such as the usage of buffer pools, locks held by applications, sorts performed by the system, activities on tables, connections, transaction statistics, or statements run on the system. There are three major methods of monitoring:

- ▶ Monitoring table functions
- ▶ Snapshot™ monitoring
- ▶ Event monitoring

Monitoring table functions

Starting with DB2 Version 9.7, you can access monitor data through a simpler alternative to the traditional system monitor. You can use monitor table functions to collect and view data for systems, activities, or data objects.

Data for monitored elements continually accumulates in memory and is available for querying. You can choose to receive data for a single object (for example, service class A or table TABLE1) or for all objects.

When using these table functions in a database partitioned environment, you can choose to receive data for a single partition or for all partitions. If you choose to receive data for all partitions, the table functions return one row for each partition. Using SQL, you can sum the values across partitions to obtain the value of a monitor element across partitions.

Monitor table functions can be divided into three categories, depending on the information that they monitor:

- ▶ **System:** The *system monitoring perspective* includes the complete volume of work and effort expended by the data server to process application requests. This perspective helps you determine all of the data server activities, as well as the activities for a particular subset of application requests. The following list shows the table functions for retrieving system information:
 - MON_GET_SERVICE_SUBCLASS
 - MON_GET_SERVICE_SUBCLASS_DETAILS
 - MON_GET_WORKLOAD
 - MON_GET_WORKLOAD_DETAILS
 - MON_GET_CONNECTION
 - MON_GET_CONNECTION_DETAILS
 - MON_GET_UNIT_OF_WORK
 - MON_GET_UNIT_OF_WORK_DETAILS

- ▶ **Activities:** The *activity monitoring perspective* focuses on the subset of data server processing that is related to executing activities. In the context of SQL statements, the term *activity* refers to the execution of the section for an SQL statement. You use the following table functions to access current data for activities:
 - MON_GET_ACTIVITY_DETAILS
 - MON_GET_PKG_CACHE_STMT
- ▶ **Data objects:** The *data object monitoring perspective* provides information about operations that are performed on data objects, such as tables, indexes, buffer pools, table spaces, and containers. Use the following table functions to access the current details for data objects:
 - MON_GET_BUFFERPOOL
 - MON_GET_TABLESPACE
 - MON_GET_CONTAINER
 - MON_GET_TABLE
 - MON_GET_INDEX

Example 10-12 shows an example of how you can use the MON_GET_TABLE function to retrieve the rows read, inserted, updated, and deleted from all tables in the ADMIN schema.

Example 10-12 Monitor table function

```
db2inst1@db2server:~> db2 "SELECT varchar(tabschema,20) as tabschema,
      varchar(tabname,20) as tabname,
      sum(rows_read) as total_rows_read,
      sum(rows_inserted) as total_rows_inserted,
      sum(rows_updated) as total_rows_updated,
      sum(rows_deleted) as total_rows_deleted
FROM TABLE(MON_GET_TABLE('ADMIN','",-1)) AS t
GROUP BY tabschema, tabname
ORDER BY total_rows_read
DESC"
```

Snapshot monitoring

Snapshot monitoring describes the state of database activity at a particular point in time when a snapshot is taken. Snapshot monitoring is useful in determining the current state of the database and its applications. Taken at regular intervals, the snapshots are useful for observing trends and foreseeing potential problems.

You can take snapshots from the command line, by using custom APIs, or through SQL by using table functions. Example 10-13 on page 344 shows an extract from a sample snapshot invoked from the command line.

Example 10-13 Example snapshot

```
db2inst1@db2server:~> db2 GET SNAPSHOT FOR DATABASE ON invent
```

```
Database Snapshot

Database name                = INVENT
Database path                =
/home/db2inst1/invent/db2inst1/NODE0000/SQL00001/
Input database alias        = INVENT
Database status              = Active
Catalog database partition number = 0

[...]

High water mark for connections = 9
Application connects          = 767
Secondary connects total     = 7
Applications connected currently = 1
Appls. executing in db manager currently = 0
Agents associated with applications = 7
Maximum agents associated with applications = 9
Maximum coordinating agents   = 9

[...]

Buffer pool data logical reads = Not Collected
Buffer pool data physical reads = Not Collected
Buffer pool temporary data logical reads = Not Collected
Buffer pool temporary data physical reads = Not Collected

[...]
```

In Example 10-13, the snapshot has collected database-level information for the *INVENT* database. Several of the returned parameters display point-in-time values, such as the number of currently connected applications:

```
Applications connected currently = 1
```

Certain parameters represent cumulative values, such as the number of connect statements that are issued against the database:

```
Application connects = 767
```

Other parameters can contain historical values, such as the maximum number of concurrent connections that have been observed on the database:

```
High water mark for connections = 9
```

Cumulative or historical values are used to relate to the point in time during the last initialization of counters. The counters can be reset to zero by the RESET MONITOR command or by the appropriate DB2 event. In Example 10-13,

database deactivation and activation reset all of the database-level counters. Example 10-14 shows how to reset monitors for an entire instance and for the specific database.

Example 10-14 Resetting snapshot monitor counters

```
db2 RESET MONITOR ALL
db2 RESET MONITOR FOR DATABASE invent
```

To optimize database performance in a default DB2 configuration, most of the snapshot monitor elements are not collected. In Example 10-12 on page 343, the value `Not Collected` was displayed for the buffer pool statistics. DB2 contains monitor switches to provide database administrators with the option of constraining the collection of monitor elements. You can display the current monitor switches that are set for the session from the command line with the `GET MONITOR SWITCHES` command, as shown in Example 10-15.

Example 10-15 Displaying monitor switches

```
db2inst1@db2server:~> db2 GET MONITOR SWITCHES

Monitor Recording Switches

Switch list for db partition number 0
Buffer Pool Activity Information (BUFFERPOOL) = OFF
Lock Information (LOCK) = OFF
Sorting Information (SORT) = OFF
SQL Statement Information (STATEMENT) = OFF
Table Activity Information (TABLE) = OFF
Take Timestamp Information (TIMESTAMP) = ON 09/11/2009 22:10:34.801069
Unit of Work Information (UOW) =
OFF
```

The monitor switches can be turned on at the instance level or the application level. To switch the monitors at the instance level, modify the appropriate database manager parameter. After modifying the `DFT_MON_BUFPOOL` parameter, as shown in Example 10-16, all users with `SYSMAINT`, `SYSCTRL`, or `SYSADM` authorities are able to collect buffer pool statistics on any database in the instance.

Example 10-16 Updating monitor switches at the instance level

```
db2 UPDATE DBM CFG USING DFT_MON_BUFPOOL ON
```

To switch the monitors at the application level, issue the UPDATE MONITOR SWITCHES command using the command line. The changes only are applicable to that particular prompt window. Example 10-17 shows how to update the suitable monitor switch for collecting buffer pool information.

Example 10-17 Updating monitor switches at the application level

```
db2 UPDATE MONITOR SWITCHES USING BUFFERPOOL ON
```

Table 10-2 shows the complete list of monitor switches and related database manager (DBM) parameters.

Table 10-2 List of monitor switches and related DBM parameters

Database manager parameter	Monitor switch	Information provided
DFT_MON_BUFPOOL	BUFFERPOOL	Number of reads and writes and time taken
DFT_MON_LOCK	LOCK	Number of locks held and the number of deadlocks
DFT_MON_SORT	SORT	Number of heaps used, overflows, and sort performance
DFT_MON_STMT	STATEMENT	Start/stop time and statement identification
DFT_MON_TABLE	TABLE	Measure of activity (rows read/written)
DFT_MON_UOW	UOW (Unit Of Work)	Start/end times and completion status
DFT_MON_TIMESTAMP	TIMESTAMP	Timestamps for operations

Sample snapshots

The database manager snapshot (Example 10-18) captures information specific to the instance level. The information centers around the total amount of memory that is allocated to the instance and the number of agents that are currently active on the system.

Example 10-18 Database manager snapshot

```
db2 GET SNAPSHOT FOR DATABASE MANAGER
```

The lock snapshot (Example 10-19 on page 347) is useful in determining what locks an application currently holds and the locks that other applications are waiting on. The snapshot lists all applications on the system and the locks that each of these applications holds. Each lock is given a unique identifier number, and each application is given a unique identifier number.

Example 10-19 Lock snapshot

```
db2 GET SNAPSHOT FOR LOCKS ON invent
```

The table snapshot (Example 10-20) contains information about the usage and creation of all tables. This information is useful in determining how much work is being run against a table and how much the table data changes. You can use this information to decide how to lay out your data physically.

Example 10-20 Table snapshot

```
db2 GET SNAPSHOT FOR TABLES ON invent
```

The table space and buffer pool snapshots (Example 10-21) contain similar information. The table space snapshot returns information regarding the layout of the table space and the amount of space that is used. The buffer pool snapshot contains information about the amount of space currently allocated for buffer pools and the amount of space that is required when the database is next reset. Both snapshots contain a summary of the way in which data is accessed from the database. This access can be done from a buffer pool, directly from tables on disk, or through a direct read or write for LOBs or LONG objects.

Example 10-21 Table space and buffer pool snapshots

```
db2 GET SNAPSHOT FOR TABLESPACES ON invent  
db2 GET SNAPSHOT FOR BUFFERPOOLS ON invent
```

The dynamic SQL snapshot (Example 10-22) is used extensively to determine how well SQL statements perform. This snapshot summarizes the behavior of the various dynamic SQL statements that are run. The snapshot does not capture static SQL statements, so anything that was pre-bound does not show up in this list. The snapshot is an collection of the information concerning the SQL statements. If an SQL statement is executed 102 times, there is one entry, which encapsulates a summary of the behavior of all 102 executions.

Example 10-22 Dynamic SQL snapshot

```
db2 GET SNAPSHOT FOR DYNAMIC SQL ON invent
```

Snapshot table functions

Authorized users can also capture snapshot monitoring information for an instance by using snapshot table functions or snapshot administrative views. The snapshot table functions allow you to request data for specific database partitions, globally aggregated data, or data from all database partitions. Certain snapshot table functions allow you to request data from all active databases. The

snapshot administrative views provide a simple means of accessing data for all database partitions of the connected database.

Example 10-23 and Example 10-24 show how to get similar monitoring information using the table functions and views as we did from Example 10-17 by using the GET SNAPSHOT command. Example 10-23 demonstrates a query that captures the snapshot of lock information for the currently connected database. Example 10-24 is a query that captures a snapshot of lock information about the SAMPLE database for the currently connected database partition.

Example 10-23 Sample snapshot table function

```
db2inst1@db2server:~> SELECT * FROM SYSIBMADM.SNAPLOCK
```

Example 10-24 Sample snapshot table function

```
db2inst1@db2server:~>SELECT * FROM TABLE(SNAP_GET_LOCK('invent',-1)) AS  
SNAPLOCK
```

Table 10-3 lists the snapshot table functions, administrative views, and return information that can be used to monitor your database system. All administrative views belong to the SYSIBMADM schema.

Table 10-3 Common snapshot table functions and administrative views

Monitor level	Snapshot table function	Administrative view	Information returned
DBM	SNAP_GET_DBM_V95	SNAPDBM	Retrieve the database manager-level information.
	SNAP_GET_FCM	SNAPFCM	Retrieve the fcm logical data group snapshot information.
	SNAP_GET_FCM_PART	SNAPFCM_PART	Retrieve the fcm_node logical data group snapshot information.
	SNAP_GET_SWITCHES	SNAPSWITCHES	Retrieve database snapshot switch state information.
	SNAP_GET_DBM_MEMORY_POOL	SNAPDBM_MEMORY_POOL	Retrieve database manager-level memory usage information.
DB	SNAP_GET_DB_V95	SNAPDB	Retrieve snapshot information from the database logical group.
	SNAP_GET_DB_MEMORY_POOL	SNAPDB_MEMORY_POOL	Retrieve database-level memory usage information.
	SNAP_GET_HADR	SNAPHADR	Retrieve HADR logical data group snapshot information.

APP	SNAP_GET_APPL_V95	SNAPAPPL	Retrieve application logical data group snapshot information.
	SNAP_GET_APPL_INFO_V95	SNAPAPPL_INFO	Retrieve appl_info logical data group snapshot information.
	SNAP_GET_LOCKWAIT	SNAPLOCKWAIT	Retrieve lockwait logical data group snapshot information.
	SNAP_GET_STMT	SNAPSTMT	Retrieve statement snapshot information.
	SNAP_GET_AGENT	SNAPAGENT	Retrieve agent logical data group application snapshot information.
	SNAP_GET_SUBSECTION	SNAPSUBSECTION	Retrieve subsection logical monitor group snapshot information.
	SNAP_GET_AGENT_MEMORY_POOL	SNAPAGENT_MEMORY_POOL	Retrieve memory_pool logical data group snapshot information.
Table	SNAP_GET_TAB_V91	SNAPTAB	Retrieve table logical data group snapshot information.
	SNAP_GET_TAB_REORG	SNAPTAB_REORG	Retrieve table reorganization snapshot information.
Lock	SNAP_GET_LOCK	SNAPLOCK	Retrieve lock logical data group snapshot information.
Table space	SNAP_GET_TBSP_V91	SNAPTBSP	Retrieve table space logical data group snapshot information.
	SNAP_GET_TBSP_PART_V91	SNAPTBSP_PART	Retrieve tablespace_nodeinfo logical data group snapshot information.
	SNAP_GET_TBSP QUIESCER	SNAPTBSP QUIESCER	Retrieve quiescer table space snapshot information.
	SNAP_GET_CONTAINER_V91	SNAPCONTAINER	Retrieve tablespace_container logical data group snapshot information.
	SNAP_GET_TBSP_RANGE	SNAPTBSP_RANGE	Retrieve range snapshot information.
Buffer pool	SNAP_GET_BP_V95	SNAPBP	Retrieve buffer pool logical group snapshot information.
	SNAP_GET_BP_PART	SNAPBP_PART	Retrieve bufferpool_nodeinfo logical data group snapshot information.
Dynamic SQL	SNAP_GET_DYN_SQL_V95	SNAPDYN_SQL	Retrieve dynsql logical group snapshot information.
DB	SNAP_GET_UTIL	SNAPUTIL	Retrieve utility_info logical data group snapshot information.
	SNAP_GET_UTIL_PROGRESS	SNAPUTIL_PROGRESS	Retrieve progress logical data group snapshot information.
	SNAP_GET_DETAILLOG_V91	SNAPDETAILLOG	Retrieve snapshot information from the detail_log logical data group.
	SNAP_GET_STORAGE_PATHS	SNAPSTORAGE_PATHS	Retrieve automatic storage path information.

The SQL table functions have two input parameters:

- ▶ database name
VARCHAR(255). If you enter NULL, the name of the currently connected database is used.
- ▶ partition number
SMALLINT. For the partition number parameter, enter the integer (a value between 0 and 999) corresponding to the partition number that you must monitor. To capture a snapshot for the currently connected partition, enter a value of -1 or a NULL. To capture a global snapshot, enter a value of -2.

For the following list of snapshot table functions, if you enter NULL for the currently connected database, you get snapshot information for all databases in the instance:

- ▶ SNAP_GET_DB_V95
- ▶ SNAP_GET_DB_MEMORY_POOL
- ▶ SNAP_GET_DETAILLOG_V91
- ▶ SNAP_GET_HADR
- ▶ SNAP_GET_STORAGE_PATHS
- ▶ SNAP_GET_APPL_V95
- ▶ SNAP_GET_APPL_INFO_V95
- ▶ SNAP_GET_AGENT
- ▶ SNAP_GET_AGENT_MEMORY_POOL
- ▶ SNAP_GET_STMT
- ▶ SNAP_GET_SUBSECTION
- ▶ SNAP_GET_BP_V95
- ▶ SNAP_GET_BP_PART

The database name parameter does not apply to the database manager-level snapshot table functions; they have an optional parameter for database partition number.

Event monitoring

Event monitors are used to monitor the performance of DB2 over a fixed period of time. The information that can be captured by an event monitor is similar to the snapshots, but in addition to snapshot-level information, event monitors also examine transition events in the database, and they consider each event as an object. Event monitors can capture information about DB2 events in the following areas:

- ▶ Statements: A *statement event* is recorded when an SQL statement ends. The monitor records the statement's start and stop time, CPU used, text of dynamic SQL, the return code of the SQL statement, and other matrixes, such as the fetch count.

- ▶ **Connections:** A *connection event* is recorded whenever an application disconnects from the database. The connection event records all application-level counters.
- ▶ **Database:** An *event of database* information is recorded when the last application disconnects from the database. This event records all database-level counters.
- ▶ **Buffer pools:** A *buffer pool event* is recorded when the last application disconnects from the database. The information captured contains the type and volume of use of the buffer pool, use of prefetchers and page cleaners, and whether direct I/O was used.
- ▶ **Table spaces:** A *table space event* is recorded when the last application disconnects from the database. This monitor captures information regarding counters for buffer pool, prefetchers, page cleaners, and direct I/O for each table space.
- ▶ **Tables:** All active *table events* are recorded when the last application disconnects from the database. An active table is a table that has been altered or created since the database was activated. The monitor captures the number of rows read and written to the table.
- ▶ **Activities:** An *activity event* is recorded on the completion of an activity that executed in a service class, workload, or work class that has had its COLLECT ACTIVITY DATA option turned on. Data is also collected for the targeted activity in the instant that the WLM_CAPTURE_ACTIVITY_IN_PROGRESS stored procedure is executed. Data can also be collected if the activity violates a threshold that has the COLLECT ACTIVITY DATA option enabled. The activity event monitor records activity-level data. If the WITH DETAILS option was specified as part of the COLLECT ACTIVITY DATA statement, this collected data includes statement and compilation environment information for those activities that have it. If AND VALUES was also specified, this collected data also includes input data values for those activities that have it.
- ▶ **Statistics:** A *statistics event* is recorded every *xx* minutes, where *xx* is the length of time over which statistics are gathered. This period is defined in the WLM_COLLECT_INT database configuration parameter. Data can also be collected when the WLM_COLLECT_STATS stored procedure is called. The statistics captured are activities that executed with each service class, workload, or work class that exists on the system.
- ▶ **Threshold violations:** A *threshold violation event* is recorded upon detection of a threshold violation. The threshold violations event monitor records a description of the threshold that was violated (the identifier, maximum value, and so on), identification of the activity that violated the threshold, the unit of work identifier, and the time that the threshold was violated.

- ▶ Locking: The *locking event monitor* is new and can replace the deprecated deadlock event monitor. A locking event is recorded upon detection of any of the following event types, depending on the configuration: lock timeout, deadlock, or lock wait beyond a specified duration. The information captured from this event monitor focuses on the locks involved in the failure and the applications that own them.
- ▶ Unit of work: The *unit of work* is a new event monitoring type to DB2 9.7 and replaces the deprecated transaction event monitor. A unit of work event is recorded upon the completion of a unit of work. The unit of work event records a variety of information, including attributes at the database level, connection level, and unit of work level.

Event monitors are created with the CREATE EVENT MONITOR SQL statement. Information about event monitors is stored in the system catalog table, and it can be reused later.

Example 10-25 shows a sequence of statements that illustrate how to collect Event Monitor information using commands.

Example 10-25 Working with event monitors

```

1 db2inst1@db2server:~> db2 "CREATE EVENT MONITOR mymon1 FOR DATABASE,
STATEMENTS WRITE TO FILE '/home/db2inst1/temp'"
DB20000I The SQL command completed successfully.

2 db2inst1@db2server:~> db2 SET EVENT MONITOR mymon1 STATE=1

3 db2inst1@db2server:~/> db2 "SELECT * FROM amdin.services"
[...]

4 db2inst1@db2server:~/> db2 SET EVENT MONITOR mymon1 STATE=0

5 db2inst1@db2server:~/> db2 DROP EVENT MONITOR mymon1
DB20000I The SQL command completed successfully.

6 db2inst1@db2server:~/> db2evmon -path /home/db2inst1/temp

```

The numbers bolded in Example 10-25 correspond to the following descriptions:

1. This line is the statement used to create the event monitor mymon1, which specifically collects DATABASE and STATEMENT events.
2. This line is turning the event monitor on. Values for this statement can be 0 for off and 1 for on.
3. This line is used as an SQL statement that must be captured by the event monitor.
4. This line turns off the event monitor.

5. This line drops the event monitor mymon1.
6. The db2evmon utility converts the event monitor binary files to a user readable form. Example 10-26 shows the output from this utility.

Example 10-26 db2evmon output

EVENT LOG HEADER

Event Monitor name: MYMON1
Server Product ID: SQL09070
Version of event monitor data: 10
Byte order: LITTLE ENDIAN
Number of nodes in db2 instance: 1
Codepage of database: 1208
Territory code of database: 1
Server instance name: db2inst1

Database Name: INVENT
Database Path: /home/db2inst1/invent/db2inst1/NODE0000/SQL00001/
First connection timestamp: 09/16/2009 15:16:33.138148
Event Monitor Start time: 09/17/2009 10:12:14.573251

3) Connection Header Event ...

Appl Handle: 4195
Appl Id: *LOCAL.db2inst1.090917034642
Appl Seq number: 00029
DRDA AS Correlation Token: *LOCAL.db2inst1.090917034642
Program Name : db2bp
Authorization Id: DB2INST1
Execution Id : db2inst1
Codepage Id: 1208
Territory code: 1
Client Process Id: 12152
Client Database Alias: INVENT
Client Product Id: SQL09070
Client Platform: Unknown
Client Communication Protocol: Local
Client Network Name: db2server
Connect timestamp: 09/16/2009 23:46:42.964073

[...]

10.4.6 Visual Explain

An *access plan* is a cost estimation of resource usage for a query, based on available information, such as statistics for tables and indexes, instance and database configuration parameters, bind options and query optimization level, and so on. An access plan also specifies the order of operations for accessing the data.

The access plan that is acquired from Visual Explain helps you to understand how individual SQL or XQuery statements are executed. You can use the information displayed in the Visual Explain graph to tune SQL and XQuery queries to optimize performance.

You can start Visual Explain from the Control Center or from the Optim Data Studio toolset. From Data Studio, create or open SQL or XQuery statement. In the Main panel view, right-click and select **Open Visual Explain**, as shown in Figure 10-3.

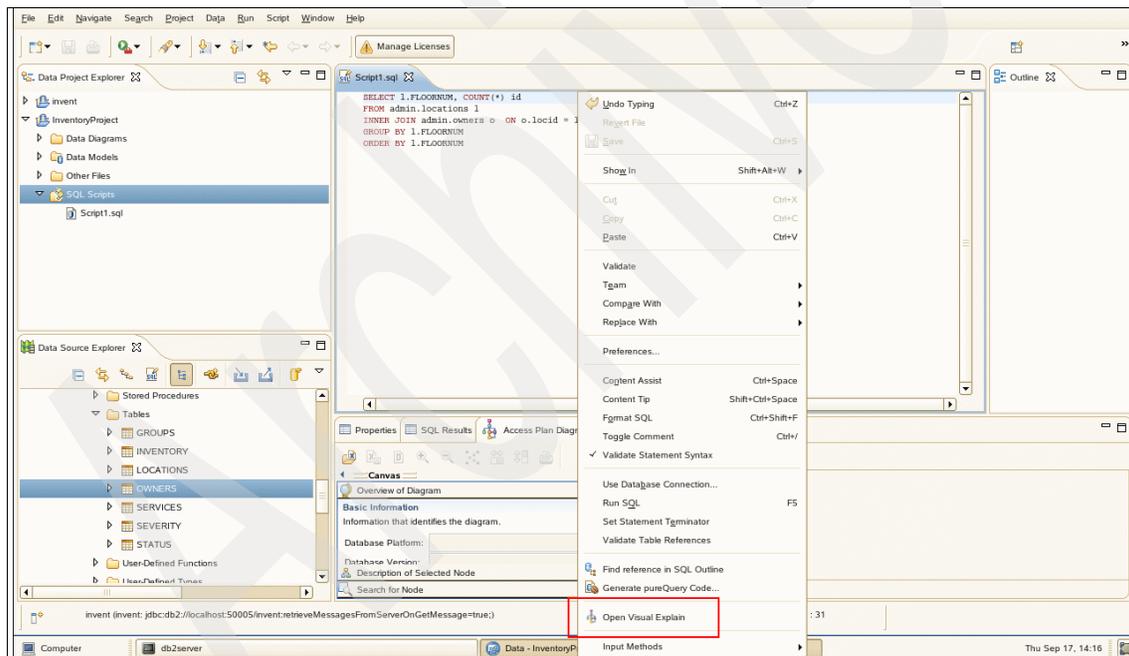


Figure 10-3 Opening Visual Explain in Data Studio

A configuration window appears where you can specify the general settings and the values for Visual Explain to use for special registers when fetching explain data.

Figure 10-4 shows an example of an access plan graph. To get the details, right-click the desired graph element.

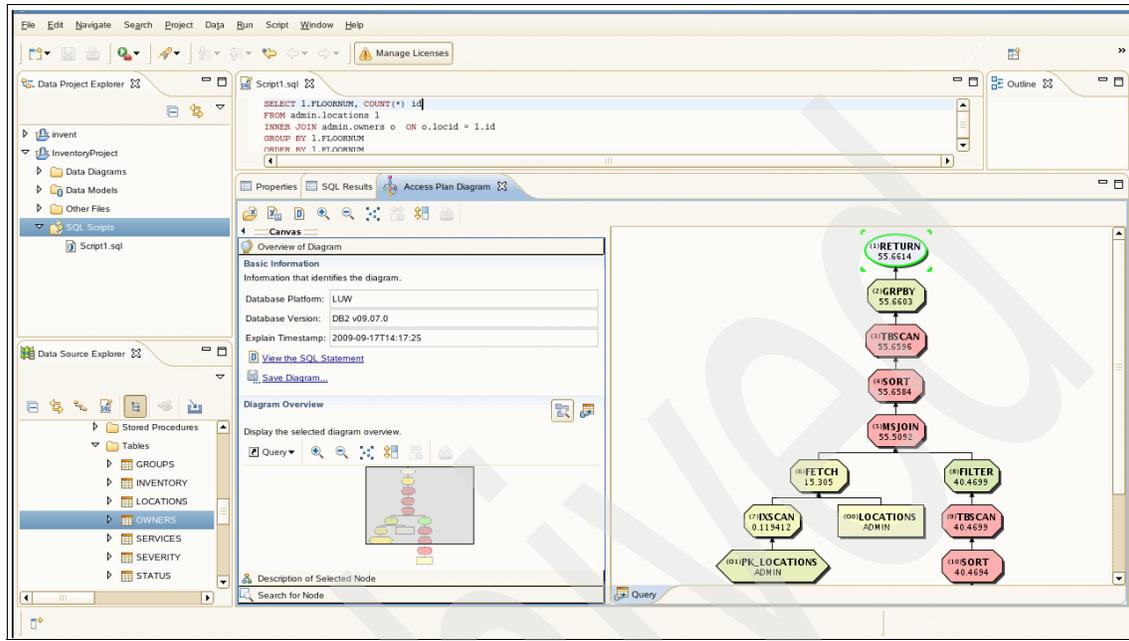


Figure 10-4 Visual Explain access plan graph

You can use Visual Explain to perform these tasks:

- ▶ View the statistics that were used at the time of optimization. You can compare these statistics to the current catalog statistics to help you determine whether rebinding the package might improve performance.
- ▶ Determine whether or not an index was used to access a table. If an index was not used, Visual Explain can help you determine which columns might benefit from being indexed.
- ▶ View the effects of performing various tuning techniques by comparing the before and after versions of the access plan graph for a query.
- ▶ Obtain information about each operation in the access plan, including the total estimated cost and the number of rows retrieved (*cardinality*).

10.5 Initial tuning

Performance of a DB2 database application can be influenced by many factors, such as the type of workload, application design, database design, capacity planning, and instance and database configuration. Initial tuning has become easier with the introduction of autonomic computing and automatic features that are turned on by default with DB2. However, you still might need to modify a few settings to fit your environment. This section focuses on a number of DB2 performance tuning tips that can be used for initial configuration.

10.5.1 Table space design

At database creation time, three table spaces are created:

- ▶ SYSCATSPACE: Catalog table space for storing information about all the objects in the database.
- ▶ TEMPSPACE1: System temporary table space for storing internal temporary data required during SQL operations, such as sorting, reorganizing tables, creating indexes, and joining tables.
- ▶ USERSPACE1: For storing user-defined tables.

By default, SYSCATSPACE and USERSPACE1 table spaces are created as *Database Managed Space (DMS)*, which means that the database manager controls the storage space. The TEMPSPACE1 table space is created as *System Managed Space (SMS)*, meaning that the regular operating system functions are used for handling I/O operations.

For simple databases, the default configurations might be sufficient for your needs. However, in most cases, you might want to add additional table spaces. The benefit of multiple table spaces is that you can assign separate database objects to separate table spaces and assign the table spaces to dedicated physical devices. Using this method allows each table object to utilize the hardware allocated to the table space to which it belongs. This approach essentially allows for table-level backup.

You can create an SMS table space using the `MANAGED BY SYSTEM` clause in the create table space definition. The benefit of using an SMS table space is that it does not require initial storage; space is not allocated by the system until it is required. Creating a table space with SMS requires less initial work, because you do not have to predefine the containers.

However, with SMS table spaces, the file system of the operating system decides where each logical file page is physically stored. Pages might not be stored contiguously on disk, because the storage placement depends on the file

system algorithm and on the level of activity on the file system. And, the performance of an SMS table space can be negatively affected. Therefore, SMS table spaces are ideal for small databases that require low maintenance and monitoring and that grow and shrink rapidly.

With DMS, the database manager can ensure that pages are physically contiguous, because it bypasses operating system I/O and interfaces with the disk directly. This approach can improve performance significantly. You can create a DMS table space by using the `MANAGED BY DATABASE` clause in the create table space definition.

The disadvantage is that a DMS table space requires more tuning and administrative effort, because you must add more storage containers as the table space fills with data. However, you can easily add new containers, drop, or modify the size of existing containers. The database manager then automatically rebalances existing data into all the containers belonging to the table space. Therefore, DMS table spaces are ideal for performance-sensitive applications, particularly applications that involve a large number of `INSERT` operations.

If a database is enabled for automatic storage (which is enabled by default), there is a third option when creating a table space. You can specify automatic management by using the `MANAGED BY AUTOMATIC STORAGE` clause in the `CREATE TABLESPACE` definition. With this option, DB2 decides what type of table space to create. There is no need to specify container details, because DB2 assigns the containers and manages table space creation automatically.

For optimal performance, you must place large volume data and indexes within DMS table spaces; if possible, split them to separate raw devices. Initially, system temporary table spaces need to be of the SMS type. In an online transaction processing (OLTP) environment, there is no need to create large temporary objects to process SQL queries, so the SMS system temporary table space is a good starting point. The easiest way to optimize your table spaces is to use table spaces that are managed by automatic storage.

10.5.2 Physical placement of database objects

When creating a database, the first important decision is the storage architecture. The ideal situation is to have the fastest disks possible and at least five to ten disks per processor (for a high I/O OLTP workload, use even more disks per processor). The reality is that hardware is often chosen based on other considerations, so to achieve optimal performance, carefully plan the placement of database objects.

Refer to Figure 10-5 on page 358 for an explanation of logical logs.

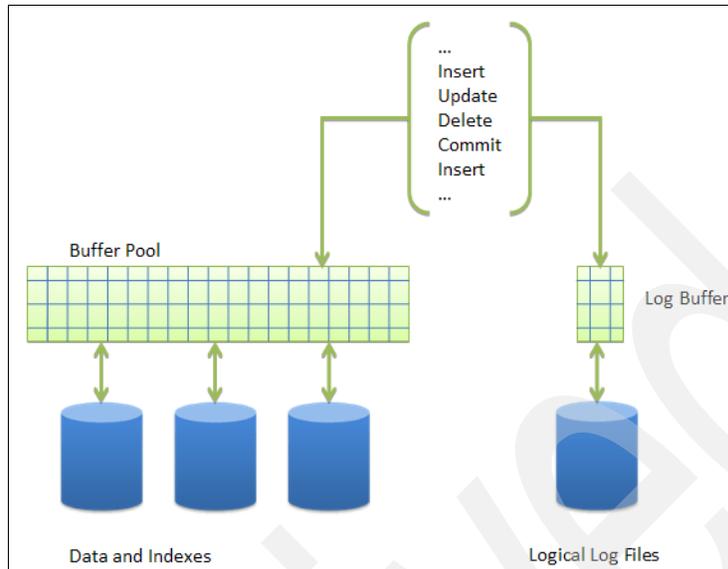


Figure 10-5 Explaining logical logs

As shown in Figure 10-5, all data modifications are not only written to table space containers, but they are also logged to ensure recoverability. Because every INSERT, UPDATE, or DELETE statement is replicated in the transactional log, the flushing speed of the logical log buffer can be crucial for the entire database performance. To understand the importance of logical log placement, remember that the time necessary to write data to disk depends on the physical data distribution on disk. The more random reads or writes that are performed, the more disk head movements are required, and therefore, the slower the writing speed. Flushing the logical log buffer to disk is by its nature sequential, and other operations must not interfere with it. Locating logical log files on separate devices isolates them from other processes and ensures uninterrupted sequential writes.

To change logical log files to a new location, you must modify the `NEWLOGPATH` database parameter, as shown in Example 10-27. The logs are relocated to the new path on the next database activation (it takes time to create the files).

Example 10-27 Relocation of logical logs

```
db2 UPDATE DB CFG FOR SAMPLE USING NEWLOGPATH /db2/1ogs
```

When creating a DMS table space with many containers, DB2 automatically distributes the data across the containers in a round-robin fashion, similar to the striping method that is available in disk arrays. To achieve the best possible performance, place each table space container on a dedicated physical device. For parallel asynchronous writes and reads from multiple devices, you must adjust the number of database page cleaners (NUM_IO_CLEANERS) and I/O servers (NUM_IOSERVERS). The best value for these two parameters depends on the type of workload and available resources. You can start your configuration with the following values:

- ▶ NUM_IOSERVERS = Number of physical devices, not less than three and no more than five times the number of CPUs
- ▶ NUM_IO_CLEANERS = Number of CPUs

However, the most effective way to configure these parameters is to set them to automatic and let DB2 manage them, as shown in Example 10-28.

Example 10-28 Updating I/O-related processes

```
db2 UPDATE DB CFG FOR sample USING NUM_IOSERVERS AUTOMATIC
db2 UPDATE DB CFG FOR sample USING NUM_IOCLEANERS AUTOMATIC
```

If there are a relatively small number of disks available, it can be difficult to keep logical logs, data, indexes, system temporary table spaces (more important for processing large queries in a data warehousing environment), backup files, or the operating system paging file on separate physical devices. A compromise solution is to have one large file system striped by a disk array (RAID device) and create table spaces with only one container. The load balancing is shifted to the hardware, and you do not have to worry about space utilization. If you want parallel I/O operations on a single container, you must set the DB2_PARALLEL_IO registry variable before starting the DB2 engine.

If this registry variable is set, and the prefetch size of the table is not AUTOMATIC, the degree of parallelism of the table space is the prefetch size divided by the extent size. If this registry variable is set, and the prefetch size of the table space is AUTOMATIC, DB2 automatically calculates the prefetch size of a table space. Table 10-4 on page 360 summarizes the available options and how parallelism is calculated for each situation.

Table 10-4 Calculating parallelism

Prefetch size of table space	DB2_PARALLEL_IO setting	Parallelism is equal to
AUTOMATIC	Not set	Number of containers
AUTOMATIC	Table space ID	Number of containers x 6
AUTOMATIC	Table space ID:n	Number of containers x n
Not AUTOMATIC	Not set	Number of containers
Not AUTOMATIC	Table space ID	Prefetch size/extent size
Not AUTOMATIC	Table space ID:n	Prefetch size/extent size

10.5.3 Buffer pools

The default size for buffer pools is relatively small: only 250 pages (~ 1 MB) for Windows and 1,000 pages (~ 4 MB) for Linux and UNIX platforms. The overall buffer size has a great effect on DB2 performance by significantly reducing I/O, which is the most time-consuming operation. We recommend that you increase these default values. However, the total buffer pool size must not be set too high, because there might not be enough memory to allocate the desired size. To calculate the maximum buffer size, all other DB2 memory-related parameters, such as database heap, the agent's memory, and storage for locks, as well as the operating system and any other applications, must be considered.

Initially, set the total size of buffer pools to 10% to 20% of available memory. You can monitor the system later and correct it. DB2 allows changing buffer pool sizes without shutting down the database. The ALTER BUFFERPOOL statement with the IMMEDIATE option takes effect right away, except when there is not enough reserved space in the database-shared memory to allocate new space. This feature can be used to tune database performance according to periodical changes in use, for example, switching from daytime interactive use to nighttime batch work.

When the total available size is determined, this area can be divided into separate buffer pools to improve utilization. Having more than one buffer pool can preserve data in the buffers. For example, let us suppose that a database has many frequently used smaller tables, which normally reside in the buffer in their entirety, and thus, are accessible quickly. Now, let us suppose that there is a query running against a very large table using the same buffer pool and involving reads of more pages than the total buffer size. When this query runs, the pages from the small, frequently used tables will be lost, making it necessary to reread them when they are needed again.

At the start, you can create additional buffer pools for caching data and leave the IBMDEFAULTBP for system catalogs.

Creating an extra buffer pool for system temporary data can also be valuable for the system performance, especially in an OLTP environment where the temporary objects are relatively small. Isolated temporary buffer pools are not influenced by the current workload, so it takes less time to find free pages for temporary structures, and it is likely that the modified pages will not be swapped out to disk.

In a data warehousing environment, the operations on temporary table spaces are considerably more intensive, so the buffer pools need to be larger, or combined with other buffer pools if there is not enough memory in the system (one pool for caching data and temporary operations).

Example 10-29 shows how to create buffer pools, assuming that an additional table space named DATASPACE for storing data and indexes was already created and that there is enough memory in the system. Use this example as a starting buffer pool configuration for a 2 GB RAM system.

Example 10-29 Increasing buffer pools

```
connect to sample;
-- creating two buffer pools 256 MB and 64 MB
CREATE BUFFERPOOL data_bp IMMEDIATE SIZE 65536 pagesize 4k;
CREATE BUFFERPOOL temp_bp IMMEDIATE SIZE 16384 pagesize 4k;

-- changing size of the default buffer pool
ALTER BUFFERPOOL ibmdefaultbp IMMEDIATE SIZE 16384;

-- binding the tablespaces to buffer pools
ALTER TABLESPACE dataspace BUFFERPOOL data_bp;
ALTER TABLESPACE temp space1 BUFFERPOOL temp_bp;

-- checking the results
SELECT
  substr(bs.bpname,1,20) AS bpname
  ,bs.npages
  ,bs.pagesize
  ,substr(ts.tb space,1,20) as TBSPACE
FROM syscat.bufferpools bs JOIN syscat.tablespaces ts ON
bs.bufferpoolid = ts.bufferpoolid;
```

The results:

BPNAME	NPAGES	PAGESIZE	TBSPACE
IBMDEFAULTBP	16384	4096	SYSCATSPACE
IBMDEFAULTBP	16384	4096	SYSTOOLSPACE

IBMDEFAULTBP	16384	4096	USERSPACE1
DATA_BP	65536	4096	DATASPACE
TEMP_BP	16384	4096	TEMPSPACE1

Although you can tune your buffer pools manually, using the Self-Tuning Memory Manager is an easier and more effective way of tuning the buffer pools for optimal performance. As we discussed in 9.6, “Autonomics” on page 306, the Self-Tuning Memory Manager can tune database memory parameters and buffer pools without any DBA intervention. The Self-Tuning Memory Manager works with buffer pools of multiple page sizes and can easily trade memory between the buffer pools as needed. You can turn on the Self-Tuning Memory Manager for a specific buffer pool by issuing commands in Example 10-30.

Example 10-30 Self-tuning memory manager and tuning buffer pools

```
db2inst1@db2server:~ > db2 UPDATE DB CFG FOR DATABASE invent USING  
SELF_TUNING_MEM ON  
db2inst1@db2server:~ > db2 ALTER BUFFERPOOL bp32 SIZE AUTOMATIC
```

The first command in Example 10-30 turns the Self-Tuning Memory Manager on, which is the default. The second command tells DB2 to automatically tune the buffer pool BP32. You can tune individual buffer pools or all of the buffer pools with the Self-Tuning Memory Manager.

The `CHNGPGS_THRESH` parameter specifies the percentage of changed pages at which the asynchronous page cleaners will be started. Asynchronous page cleaners write changed pages from the buffer pool to disk. The default value for the parameter is 60%. When that threshold is reached, certain users might experience a slower response time. Having larger buffer pools means more modified pages in memory and more work to be performed by page cleaners, as shown in Figure 10-6 on page 363.

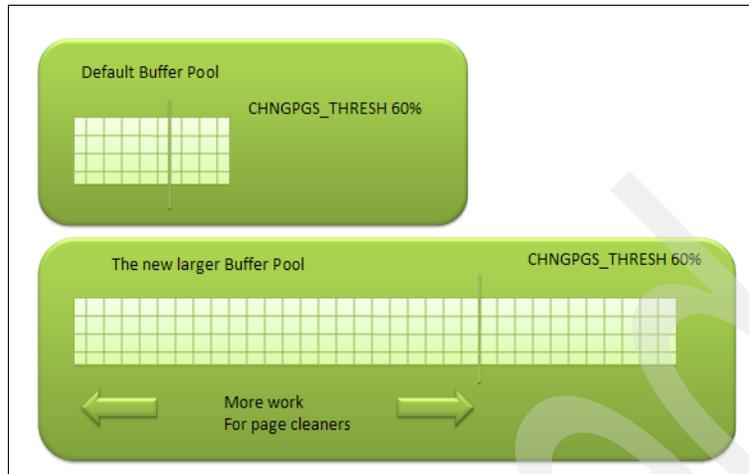


Figure 10-6 Visualizing `CHNGPGS_THRESH` parameter

For databases with a heavy update transaction workload, you can generally ensure that there are enough clean pages in the buffer pool by setting the parameter value to be equal-to or less-than the default value. A percentage larger than the default can help the performance of your database if there are a small number of extremely large tables. To change the default parameter, you can use the following command:

```
db2 update db cfg for sample using CHNGPGS_THRESH 40
```

10.5.4 Large transactions

By default, databases are created with a relatively small space for transactional logs. There are only three log files of 250 pages each on Windows and of 1,000 pages on Linux and UNIX.

A single transaction must fit into the available log space to be completed; if it does not fit, the transaction is rolled back by the system (SQL0964C The transaction log for the database is full). To process transactions that modify large numbers of rows, adequate log space is needed.

Currently, you can calculate the total log space that is available for transactions by multiplying the size of one log file (database parameter `LOGFILSIZ`) and the number of logs (database parameter `LOGPRIMARY`).

From a performance perspective, it is better to have a larger log file size because of the cost of switching from one log to another log. When log archiving is switched on, the log size also indicates the amount of data for archiving. In this case, a larger log file size is not necessarily better, because a larger log file size can increase the chance of failure or cause a delay in archiving or log shipping scenarios. You need to balance the log size and the number of logs.

Example 10-31 allocates 400 MB of total log space.

Example 10-31 Resizing the transactional log

```
db2 UPDATE DB CFG FOR sample USING logfilsiz 5120
db2 UPDATE DB CFG FOR sample USING logprimary 20
```

Locking is the mechanism that the database manager uses to control concurrent access to data in the database by multiple applications. Each database has its own list of locks (a structure stored in memory that contains the locks held by all applications concurrently connected to the database). The size of the lock list is controlled by the LOCKLIST database parameter.

The default storage for LOCKLIST on Windows and UNIX is set to AUTOMATIC. On 32-bit platforms, each lock requires 48 or 96 bytes of the lock list, depending on whether other locks are held on the object. On 64-bit platforms, each lock requires 64 or 128 bytes of the lock list, depending on whether other locks are held on the object.

When this parameter is set to AUTOMATIC, it is enabled for self-tuning, which allows the memory tuner to dynamically size the memory area controlled by this parameter as the workload requirements change. Because the memory tuner trades memory resources among separate memory consumers, there must be at least two memory consumers enabled for self-tuning in order for self-tuning to be active.

The value of LOCKLIST is tuned together with the MAXLOCKS parameter. Therefore, disabling the self-tuning of the LOCKLIST parameter automatically disables the self-tuning of the MAXLOCKS parameter. Enabling the self-tuning of the LOCKLIST parameter automatically enables the self-tuning of the MAXLOCKS parameter.

Automatic tuning of this configuration parameter only occurs when self-tuning memory is enabled for the database (the SELF_TUNING_MEM database configuration parameter is set to ON).

When the maximum number of lock requests has been reached, the database manager replaces existing row-level locks with table locks (*lock escalation*). This operation reduces the requirements for lock space, because transactions will

hold only one lock on the entire table instead of many locks on every row. Lock escalation has a negative performance impact, because it reduces concurrency on shared objects. Other transactions must wait until the transaction holding the table lock commits or rolls back work. Setting LOCKLIST to AUTOMATIC avoids this situation, because the lock list will increase synchronously to avoid lock escalation or a “*lock list full*” situation.

To check the current usage of locks, use snapshots, as shown in Example 10-32.

Example 10-32 Invoking a snapshot for locks on the invent database

```
db2 get snapshot for locks on invent
```

The snapshot collects the requested information at the time that the command was issued. Issuing the **get snapshot** command later can produce other results, because, in the mean time, the applications might commit the transaction and release the locks. To check lock escalation occurrences, look at the `db2diag.log` file.

Log buffer

Log records are written to disk when one of the following situations occurs:

- ▶ A transaction commits or a group of transactions commit, as defined by the `mincommit` configuration parameter.
- ▶ The log buffer is full.
- ▶ Another internal database manager event occurs, which results in log records being written to disk.

This log buffer size must also be less than or equal to the `dbheap` parameter. Buffering the log records results in more efficient logging file I/O, because the log records are written to disk less frequently and a greater quantity of log records are written out at each time.

The default size for the log buffer is 256 4 KB pages. In most cases, the log records are written to disk when one of the transactions issues a COMMIT or when the log buffer is full. We recommend that you increase the size of this buffer area if there is considerable read activity on a dedicated log disk or if there is high disk utilization. Increasing the size of the log buffer can result in more efficient I/O operations, especially when the buffer is flushed to disk. The log records are written to disk less frequently, and more log records are written each time.

When increasing the value of this parameter, also consider increasing the `DBHEAP` parameter, because the log buffer area uses space that is controlled by the `DBHEAP` parameter.

At a later time, you can use the **get snapshot for applications** command to check the current usage of log space by transactions, as shown in Example 10-33.

Example 10-33 Current usage of log space by applications

```
$db2 UPDATE MONITOR SWITCHES USING UOW ON
$db2 GET SNAPSHOT FOR APPLICATIONS ON sample | grep "UOW log"

UOW log space used (Bytes) = 478
UOW log space used (Bytes) = 21324
UOW log space used (Bytes) = 110865
```

Before running the application snapshot, switch on the *Unit Of Work* monitor. In Example 10-33, at the time that the snapshot was issued, you can see that there are only three applications running on the system. The first transaction uses 478 bytes of log space, the second transaction uses 21,324 bytes of log space, and the last transaction uses 110,865 bytes of log space, which is roughly 28 pages more than the default log buffer size. The snapshot gives only the current values from the moment that the command was issued. To get more valuable information about the usage of log space by transactions, run the snapshot multiple times.

10.5.5 SQL execution plan

DB2 prepares an *access plan* whenever an SQL or XQuery statement is issued against the database. This plan specifies the order of operation for accessing the data. The access plan is created at prep/bind time for static statements and at run time for dynamic statements.

You can use an access plan to view statistics for selected tables, indexes, or columns; properties for operators; global information, such as table space and function statistics; and configuration parameters that are relevant to optimization. With *Visual Explain*, you can view the access plan for an SQL or XQuery statement in graphical form.

It is important to understand that an access plan is an estimate based on the information that is available. The optimizer bases its estimations on the following types of information:

- ▶ Statistics in system catalog tables (if statistics are not current, update them using the RUNSTATS command)
- ▶ Configuration parameters
- ▶ Bind options
- ▶ The query optimization class

After many changes to table data, logically sequential data might reside on non-sequential data pages, so that the database manager must perform additional read operations to access data.

Additional read operations are also required if many rows have been deleted. In this case, consider reorganizing the table to match the index and to reclaim space. You can also reorganize the system catalog tables.

Because reorganizing a table usually takes more time than updating statistics, you can execute the RUNSTATS command to refresh the current statistics for your data and then rebind your applications. If refreshed statistics do not improve performance, reorganization might help.

You can execute the RUNSTATS command against a table from the command line. Example 10-34 shows how to execute the RUNSTATS command against our sample inventory table.

Example 10-34 Executing RUNSTATS on the inventory table

```
db2inst1@db2server:~> db2 "RUNSTATS ON TABLE ADMIN.INVENTORY"  
DB20000I The RUNSTATS command completed successfully.
```

It is also possible to update statistics using the Data Studio tool. Within the Database Explorer View, connect to your database and drill down the database object folders until you find the table for which you want to update the statistics. In our example, we connect to the invent database, and then, we drop down the invent database folder, the schema folder, the ADMIN schema folder, and the Tables folder to a list of tables in the invent database in the ADMIN schema. To pull up the table options, we right-click the **INVENTORY table** icon, as shown in Figure 10-7 on page 368.

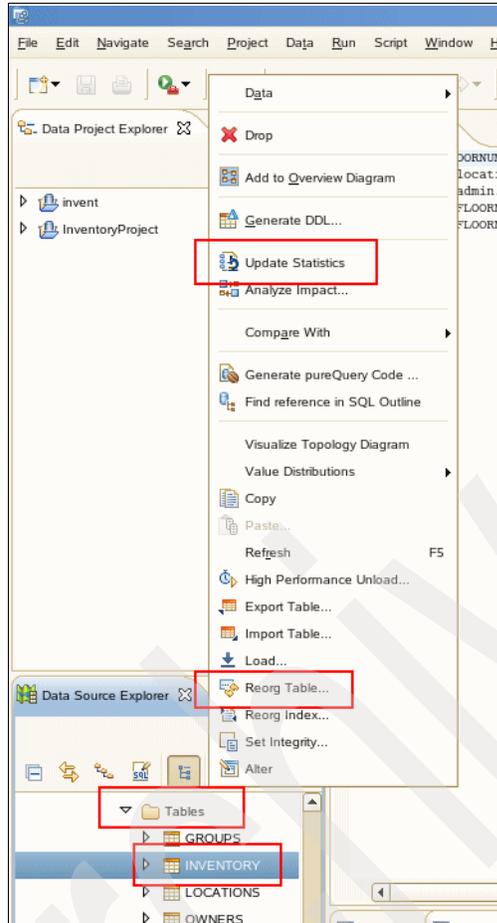


Figure 10-7 Running RUNSTATS on the INVENT table

Figure 10-8 shows the output of the update statistics.

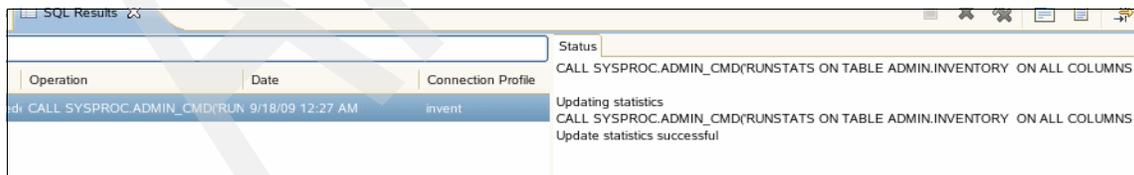


Figure 10-8 Output of RUNSTATS on the INVENT table

In certain scenarios, you might need more than current statistics on a table to improve performance. The following factors can indicate if your database will benefit from table reorganization:

- ▶ A high volume of insert, update, and delete activity has occurred against tables that are accessed by queries.
- ▶ Significant changes have occurred in the performance of queries that use an index with a high cluster ratio.
- ▶ Executing the RUNSTATS command to refresh table statistics does not improve performance.
- ▶ Output from the REORGCHK command indicates a need for table reorganization.

You can access the table reorganization option from the Data Studio by right-clicking the table, as shown in Figure 10-7 on page 368. After selecting the **REORG Table** option in the drop-down menu, the reorganization table wizard opens in the main view. Figure 10-9 on page 370 shows the reorganization table wizard. You can select the parameters for the REORG command, and select **Run** to execute the command.

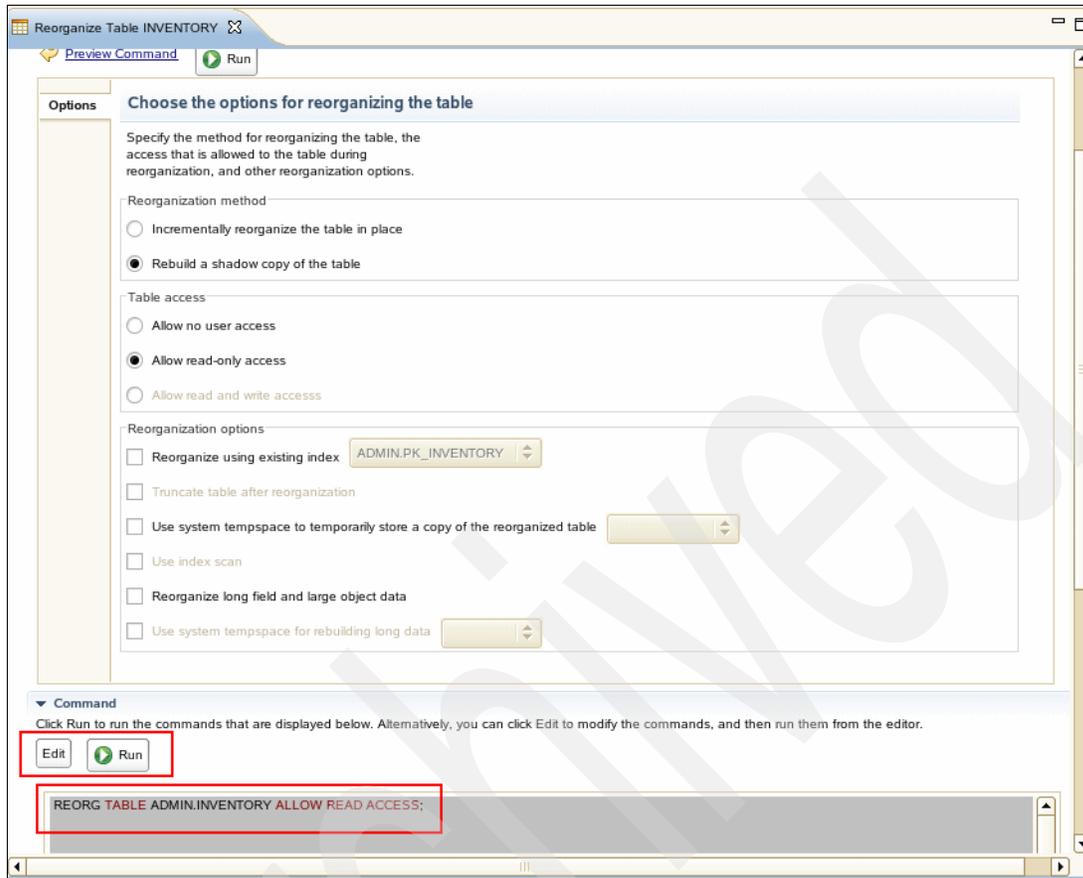


Figure 10-9 Reorganization table wizard

It can be time-consuming to constantly monitor the database to see when statistics need to be refreshed. As discussed in 9.6, “Autonomics” on page 306, DB2 autonomics can this job easier for you. With autonomics, you can create a schedule for DB2 to perform regular table or index reorganization, statistics collection, and profiling. Figure 10-10 on page 371 shows the Configure Automatic Maintenance wizard within the Data Studio GUI tool. To open this wizard, right-click the database for which you want to configure automatic maintenance, and select **Update statistics**, as highlighted in Figure 10-7 on page 368.

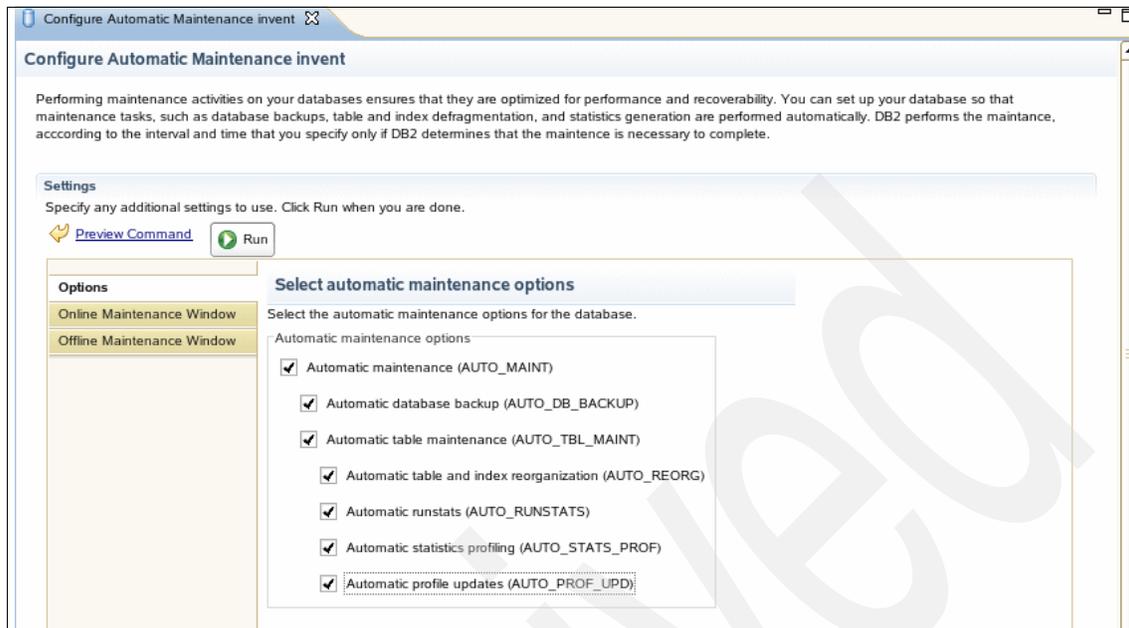


Figure 10-10 Automatic maintenance wizard

DB2 comes with an extremely powerful query optimization algorithm. This cost-based algorithm attempts to determine the least expensive way to perform a query against a database. Items, such as the database configuration, database physical layout, table relationships, and data distribution are all considered when finding the optimal access plan for a query. To check the current execution plan, you can use the Explain utility.

10.5.6 Configuration Advisor

The *Configuration Advisor* wizard is a tool that can be helpful in preparing the DB2 initial configuration. The wizard requests information about the database, its data, and the purpose of the system, and then, it recommends new configuration parameters for the database and the instance. You can run the Configuration Advisor wizard manually, through a GUI or command-line interface, and automatically during database creation.

By default in DB2 9, any new databases run the Configuration Advisor in the background and have the configuration recommendations automatically applied. To disable this feature, or to explicitly enable it, you must use the **db2set** command, as shown in Example 10-35 on page 372.

Example 10-35 Configuring the default behavior of the Configuration Advisor

```
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=NO
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=YES
```

In any case, you can run the Configuration Advisor manually at any time against a database to update the current configuration, regardless of the `DB2_ENABLE_AUTOCONFIG_DEFAULT` setting. All recommendations are based on the input that you provide and the system information that the Configuration Advisor gathers. The generated recommendations can be applied or simply displayed.

It is important to point out that the values that are suggested by the Configuration Advisor are relevant for only one database per instance. If you want to use this advisor on more than one database, each database must belong to a separate instance.

The Configuration Advisor can be manually invoked with the `AUTOCONFIGURE` command from the command line processor (CLP), either stand-alone or as part of the `CREATE DATABASE` command. Additionally, it can also be run via a GUI that is available in the Control Center, by calling the `db2AutoConfig` API, or finally, by using the `ADMIN_CMD` stored procedure.

To invoke this wizard from the DB2 Control Center, expand the object tree until you find the database that you want to tune. Select the icon for the database, right-click, and select **Configuration Advisor**. Through several dialog windows, the wizard collects the following information:

- ▶ Percentage of memory that is dedicated to DB2
- ▶ Type of workload
- ▶ Number of statements per transaction
- ▶ Transaction throughput
- ▶ Trade-off between recovery and database performance
- ▶ Number of applications
- ▶ Isolation level of applications that are connected to the database

Based on the supplied answers, the wizard proposes configuration changes and gives you the option to apply the recommendations or to save them as a task for the Task Center for later execution, as shown in Figure 10-11 on page 373. Figure 10-12 on page 374 shows the resulting recommendations.

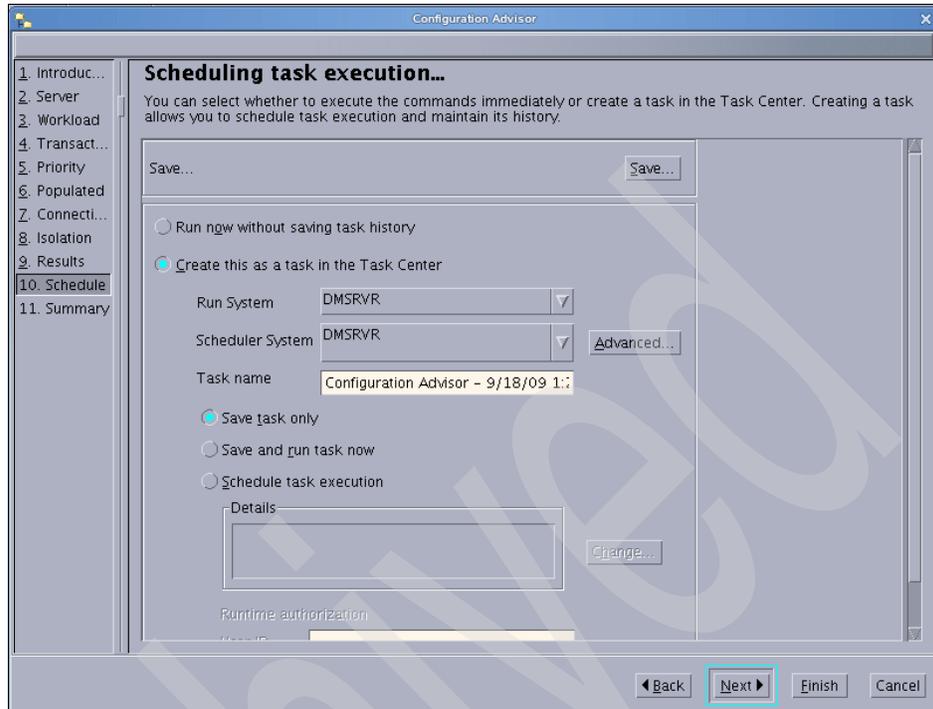


Figure 10-11 Scheduling Configuration Advisor

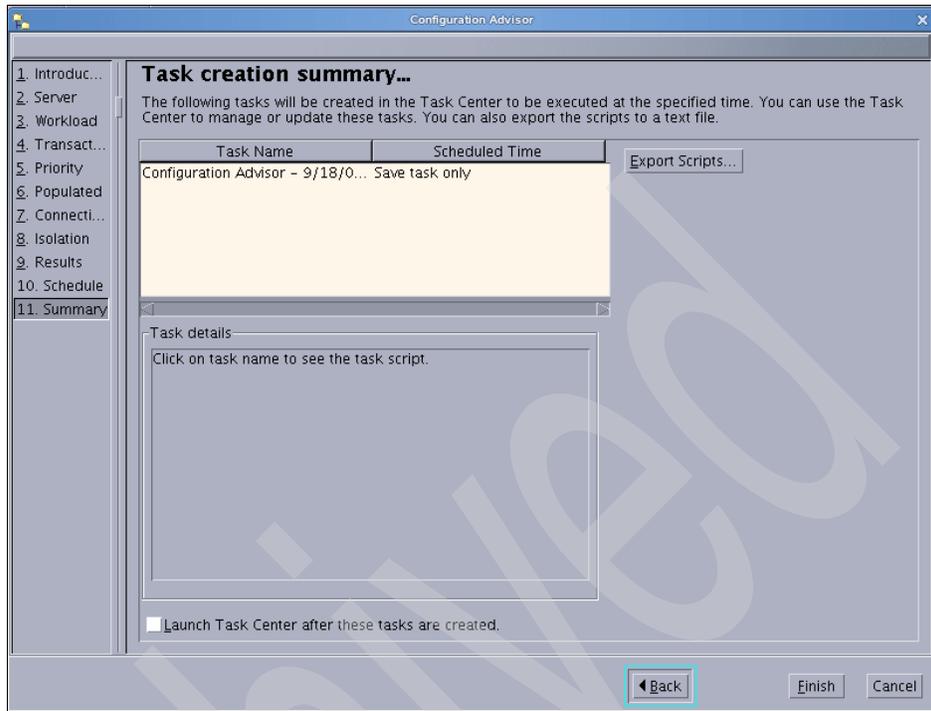


Figure 10-12 Configuration Advisor recommendations

You can also acquire the initial configuration recommendations through the text-based AUTOCONFIGURE command (Example 10-36).

Example 10-36 Sample AUTOCONFIGURE command

```
db2 AUTOCONFIGURE USING mem_percent 40 tpm 300 num_local_apps 80 isolation CS
apply none
```

[...]

Current and Recommended Values for Database Configuration

Description	Parameter	Current Value	Recommended Value
Default application heap (4KB)	(APPLHEAPSZ) =	256	256
Catalog cache size (4KB)	(CATALOGCACHE_SZ) =	33	91
Changed pages threshold	(CHNGPGS_THRESH) =	80	80
Database heap (4KB)	(DBHEAP) =	1200	3552
Degree of parallelism	(DFT_DEGREE) =	1	1
Default tablespace extentsize (pages)	(DFT_EXTENT_SZ) =	32	32

[...]

Table 10-5 lists all of the AUTOCONFIGURE command parameters.

Table 10-5 Parameters for the AUTOCONFIGURE command

Keyword	Valid values	Default value	Explanation
mem_percent	1–100	25	Percentage of memory to dedicate to DB2
Workload_type	simple, mixed, or complex	mixed	Type of workload: simple for transaction processing or complex for data warehousing
num_stmts	1–1 000 000	10	Number of statements per unit of work
Tpm	1–200 000	60	Transactions per minute
admin_priority	performance, recovery, or both	both	Optimize for better performance or better recovery time
is_populated	yes, no	yes	Is the database populated with data?
num_local_apps	0–5 000	0	Number of connected local applications
num_remote_apps	0–5 000	10	Number of connected remote applications
Isolation	RR, RS, CS, UR	RR	Isolation levels: Repeatable Read, Read Stability, Cursor Stability, or Uncommitted Read
bp_resizeable	yes, no	yes	Are buffer pools re-sizeable?

10.5.7 Design Advisor

Well-designed indexes are essential to database performance. DB2 comes with the Index Advisor utility, which recommends indexes for specific SQL queries. Index Advisor can be invoked either using the `db2adv is` command or using the Design Advisor wizard from the Command Center or Control Center. This utility accepts one or more SQL statements and their relative frequency, which is known as a *workload*.

The Index Advisor utility provides several benefits:

- ▶ Finding the best indexes for a problem query
- ▶ Finding the best indexes for a specified workload. When specifying the workload, you can use the frequency parameter to prioritize the queries. You can also limit the disk space for the target indexes.
- ▶ Testing an index on a workload without having to create the index

To execute the index advisor against a specific database, we first must specify the workload that will be run against the database. From the command line, we create a file that defines the workload. Example 10-37 shows the queries to run against the SAMPLE database.

Example 10-37 The db2advise.in input file

```
--#SET FREQUENCY 100
SELECT COUNT(*) FROM EMPLOYEE;
SELECT * FROM EMPLOYEE WHERE LASTNAME='HAAS';
--#SET FREQUENCY 1
SELECT AVG(BONUS), AVG(SALARY) FROM EMPLOYEE
      GROUP BY WORKDEPT ORDER BY WORKDEPT;
--#SET FREQUENCY 50
select FIRSTNAME, lastname, deptname from department d, employee e where
d.deptno = e.workdept and e.lastName like 'W%'
```

We can then run the **db2advise** command and specify the `db2advise.in` file as the workload input script. Example 10-38 shows the syntax and output to execute the index advisor. For more options, run **db2advise -h** from the command line.

Example 10-38 Finding indexes for a particular query

```
db2inst1@db2server:~/ > db2advise -d sample -i db2advise.in -t 5

Using user id as default schema name. Use -n option to specify schema
execution started at timestamp 2009-09-18-02.38.35.785903
found [3] SQL statements from the input file
Recommending indexes...
total disk space needed for initial set [ 0.000] MB
total disk space constrained to [ 29.459] MB
Trying variations of the solution set.
  0 indexes in current solution
[813.2663] timerons (without recommendations)
[813.2663] timerons (with current solution)
[0.00%] improvement
--
--
-- LIST OF RECOMMENDED INDEXES
-- =====
-- no indexes are recommended for this workload.
--
--
-- RECOMMENDED EXISTING INDEXES
-- =====
-- RUNSTATS ON TABLE "DB2INST1"."EMPLOYEE" FOR SAMPLED DETAILED INDEX "DB2INST1"."XEMP2"
;
-- COMMIT WORK ;
--
--
-- UNUSED EXISTING INDEXES
```

```

-- =====
-- DROP INDEX "DB2INST1"."XEMP2";
-- =====
--
-- ====ADVISOR DETAILED XML OUTPUT=====
-- ==(Benefits do not include clustering recommendations)==
--
--<?xml version="1.0"?>
--<design-advisor>
--<statement>
--<statementnum>0</statementnum>
--<statementtext>
--
--                SELECT COUNT(*) FROM EMPLOYEE
--</statementtext>
--<objects>
--<identifier>
--<name>EMPLOYEE</name>
--<schema>DB2INST1</schema>
--</identifier>
--<identifier>
--<name>XEMP2</name>
--<schema>DB2INST1</schema>
--</identifier>
--</objects>
--<benefit>0.000000</benefit>
--<frequency>100</frequency>
--</statement>
--<statementnum>1</statementnum>
--<statementtext>
-- SELECT * FROM EMPLOYEE WHERE LASTNAME='HAAS'
--</statementtext>
--<objects>
--<identifier>
--<name>EMPLOYEE</name>
--<schema>DB2INST1</schema>
--</identifier>
--</objects>
--<benefit>0.000000</benefit>
--<frequency>100</frequency>
--</statement>
--<statementnum>2</statementnum>
--<statementtext>
--
--                SELECT AVG(BONUS), AVG(SALARY)
-- FROM EMPLOYEE
-- WORKDEPT
-- GROUP BY WORKDEPT ORDER BY
--</statementtext>
--<objects>
--<identifier>
--<name>EMPLOYEE</name>
--<schema>DB2INST1</schema>
--</identifier>
--</identifier>

```

```
--<name>XEMP2</name>
--<schema>DB2INST1</schema>
--</identifier>
--</objects>
--<benefit>0.000000</benefit>
--<frequency>1</frequency>
--</statement>
--</design-advisor>
-- ====ADVISOR DETAILED XML OUTPUT=====
--
```

27 solutions were evaluated by the advisor
DB2 Workload Performance Advisor tool is finished.

Launching the Index Advisor in a GUI environment

You can also invoke the *Index Advisor* as a GUI tool. From the Control Center, expand the object tree to find the Database folder and right-click the desired database to select **Design Advisor**. The wizard guides you through all of the necessary steps and also helps you to construct a workload by looking for recently executed SQL and XQuery queries or by looking through the recently used packages. In order to get accurate recommendations, it is important to have the current catalog statistics. With the Design Advisor, there is an option to collect the required basic statistics. However, this option increases the total calculation time. Figure 10-13 on page 379 presents a sample Design Advisor window.

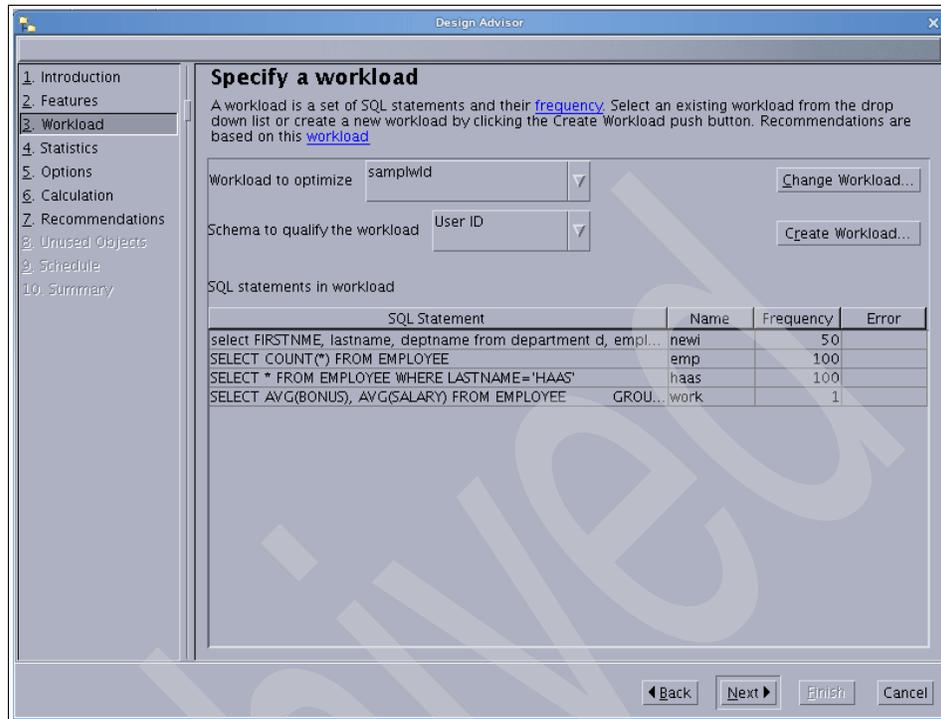


Figure 10-13 Design Advisor

Archived

Advanced DB2 features

In this chapter, we discuss the advanced features that are offered by DB2 in detail. During conversion of your database from MySQL to DB2, consider the DB2 features and functions that are explained within this chapter to enhance your application.

We discuss these features in this chapter:

- ▶ XML
- ▶ Compression
- ▶ Partitioning
- ▶ Multidimensional clustering (MDC)
- ▶ Materialized query tables (MQT)
- ▶ User-defined data types (UDT)

11.1 DB2 pureXML

DB2 *pureXML* allows native storage of XML data in a pre-parsed tree format within a database table by using a special XML column data type. However, unlike other databases on the market, DB2 does not store XML data simply as character strings, as the character large object (CLOB) data type, or shred it into relational data. When inserting XML data into a DB2 database, the data gets parsed and fragmented with node-level granularity, preserving the tree structures of the original XML data. Moreover, during this process, you can enable DB2 to validate XML data against an XML schema that is registered with the database and thus make sure that the data is always in the format that is required by the applications.

DB2 pureXML provides superior performance when managing and manipulating XML data. XML documents do not need to be parsed at query run time, and the advanced indexing technology that is available with XML in DB2 speeds up the searches across and within the already parsed documents.

DB2 pureXML is fully integrated with relational processing, allowing you to seamlessly query various types of data at one time using SQL, XML, or SQL/XML mixed query languages. And unlike other database vendors, there is no internal translation of XQuery into SQL.

Figure 11-1 on page 383 illustrates a simple example of how relational and XML data are integrated within a single table. As shown in Figure 11-1 on page 383, when creating a table within a DB2 database, you can specify both relational and XML data types in separate column definitions. This specification creates a table, where for every row of relational data, there will be an XML document associated with that row.

Because it is possible that the large size of XML documents will be stored, they are physically stored in separate objects, by default. These objects are called the *XML Data Area (XDA) objects*. For most application scenarios, XDA objects provide excellent performance. However, you have the option of storing the XML documents in the same physical space as the relational data by using the SET INLINE LENGTH clause in the CREATE or ALTER statement. For more information, refer to the IBM DB2 Information Center:

<http://publib.boulder.ibm.com/infocenter/db21uw/v9r7/index.jsp>

Storing XML: Native XML Storage

- DB2 stores XML in **parsed hierarchical format (~DOM)**

```
create table dept (deptID char(8), ..., deptdoc xml);
```

- Relational columns are stored in relational format (tables)

- XML columns are stored **natively**

- No XML parsing for query evaluation!

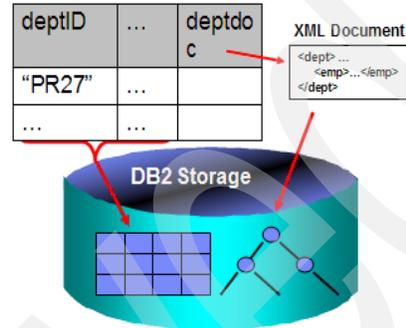


Figure 11-1 SQL and XML data integration

In order to query and update SQL and XML data, you can use SQL and XQuery statements (SQL/XML: International Organization for Standardization (ISO) standard ISO/International Electrotechnical Commission (IEC) 9075-14:2003). Several operations are available to directly modify not only full documents, but also parts or subtrees of XML documents without having to read, modify, and reinsert them. Using the XQuery language, you can directly modify single values and nodes within the XML document. XQuery is a fairly new, standardized query language supporting path-based expressions. You can obtain more information about XQuery at this Web site:

<http://www.w3.org/TR/2007/REC-xquery-20070123/>

With pureXML, applications are not only able to combine statements from both languages to query SQL and XML data; you can express many queries in plain XQuery, in SQL/XML, or XQuery with embedded SQL. In certain cases, one of the options to express your query logic might be more intuitive than another option. In general, you must choose the correct approach for querying XML data on a case-by-case basis, taking the application's requirements and characteristics into account. Example 11-1 on page 384 shows a simple XQuery command.

Example 11-1 XQuery command

```
db2 => XQUERY db2-fn:xmlcolumn('CUSTOMER.INFO')/customerinfo/name
```

1

```
-----  
<name>Kathy Smith</name>  
<name>Kathy Smith</name>  
<name>Jim Noodle</name>  
<name>Robert Shoemaker</name>  
<name>Matt Foreman</name>  
<name>Larry Menard</name>
```

6 record(s) selected.

Example 11-2 illustrates, in a table format, a combined SQL/XML statement that is used to retrieve the XML data.

Example 11-2 SQL/XQuery command

```
db2 => SELECT T.* FROM CUSTOMER C, XMLTABLE('$cu/customerinfo' PASSING C.INFO  
as "cu" COLUMNS "NAME" VARCHAR (30) PATH 'name',"STREET" VARCHAR (30) PATH  
'addr/street', "CITY" VARCHAR (30) PATH 'addr/city') AS T
```

NAME	STREET	CITY
Kathy Smith	5 Rosewood	Toronto
Kathy Smith	25 EastCreek	Markham
Jim Noodle	25 EastCreek	Markham
Robert Shoemaker	1596 Baseline	Aurora
Matt Foreman	1596 Baseline	Toronto
Larry Menard	223 NatureValley Road	Toronto

6 record(s) selected.

Example 11-3 shows how to use a combined XQuery/SQL statement to retrieve XML data based on specific relational data.

Example 11-3 XQuery with embedded SQL

```
db2 => XQUERY db2-fn:sqlquery("SELECT INFO FROM CUSTOMER WHERE  
CID=1001")/customerinfo/name
```

1

```
<name>Kathy Smith</name>
```

1 record(s) selected.

DB2 pureXML technology

DB2 pureXML technology includes these features:

- ▶ Reduced development time (no XML parsing any longer, validation and schema repository building)
- ▶ DB2 pureXML data type and storage techniques for efficient management of hierarchical structures that are common in XML documents
- ▶ DB2 pureXML indexing technology to speed up searches of subsets of XML documents
- ▶ New query language support (for XQuery and SQL/XML) that is based on industry standards and new query optimization techniques
- ▶ Industry-leading support for managing, validating, and evolving XML schemes
- ▶ Comprehensive administrative capabilities, including extensions to popular database utilities
- ▶ Integration with popular application programming interfaces (APIs) and development environments
- ▶ XML shredding and publishing facilities for working with existing relational models
- ▶ Enterprise-proven reliability, availability, scalability, performance, security, and maturity that you have come to expect from DB2

Benefits of DB2 pureXML technology

DB2 pureXML technology provides these benefits:

- ▶ Reduces development time and costs (no XML parsing any longer, validation and schema repository building)
- ▶ Avoids breaking down XML to relational data (directly query and update XML)
- ▶ Increases agility through versatile XML schema evolution
- ▶ Improves insight by exploiting previously unmanaged XML data and process queries more quickly through XML-optimized storage and indexing
- ▶ Improves application performance
- ▶ Simplifies the operating environment
- ▶ Results in lower storage costs

11.2 Data compression

Data compression is a feature in DB2, which compresses row data to reduce storage requirements, improve I/O efficiency, and provide quicker data access from the disk. By enabling data compression with DB2, you can save up to 80% of the storage space and greatly increase performance in database systems with high disk I/O activity. Considering the fact that disk storage can often be the most expensive component of a database system, this feature can result in tremendous cost savings. These savings also extend to backup disk space and more.

Two forms of data compression are currently available to you:

- ▶ **Value compression** involves removing duplicate entries for a value, storing only one copy, and keeping track of the location of any references to the stored value.
- ▶ **Row compression** involves replacing repeating patterns that span multiple column values within a row with shorter symbol strings. The row compression logic scans a table that is to be compressed for repetitive and duplicate data. A compression dictionary contains short, numeric keys to that data, and in a compressed row, these keys replace the actual data. We discuss this type of compression in this section.

Before you consider turning on row compression, you can inspect your tables to see what potential savings to expect. Compressing data and decompressing data are effortless.

To enable row compression, you can use the COMPRESS YES keywords in either the CREATE or ALTER TABLE statement, as shown in Example 11-4.

Example 11-4 Enabling row compression

```
db2inst1@db2server:~> db2 ALTER TABLE admin.owners COMPRESS YES
DB20000I The SQL command completed successfully.
```

After enabling row compression and loading or inserting 1 - 2 MB of data, DB2 automatically creates the compression dictionary. This dictionary contains the frequently occurring patterns with an associated shorter 12-bit symbol. These symbols then are used to replace the original data.

On heavily CPU-utilized systems, the compression and decompression can cause overhead and decrease performance. In cases where the system is mainly I/O-bound, fewer pages need to be read from disk, which results in a performance gain. Data compression can also reduce the amount of buffer pool memory that is needed, because pages fetched from disk stay compressed in main memory while the page is actively used.

Figure 11-2 illustrates the mapping of repeating patterns in two table rows to dictionary symbols representing those patterns. The end result is a compressed data record that is shorter in length than the original uncompressed record - which is depicted by the yellow rectangles representing the rows beneath the table.

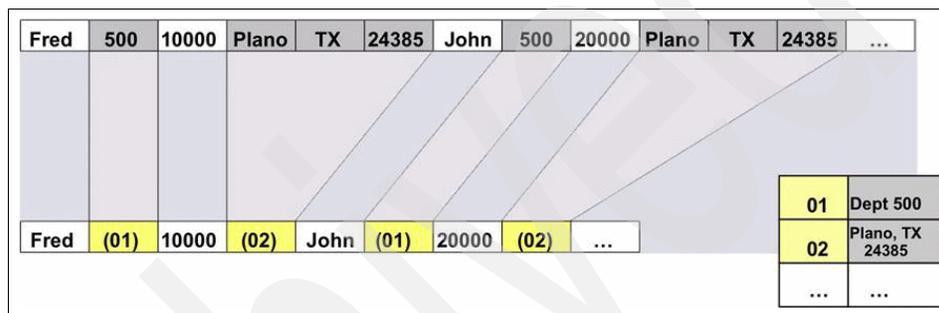


Figure 11-2 Row compression

As of DB2 9.7, data compression had been extended to include all temporary tables. Data compression for temporary tables reduces the amount of temporary disk space that is required for large and complex queries, increasing query performance.

Index objects and indexes on compressed temporary tables can also be compressed to reduce storage costs. This compression is especially useful for large online transaction processing (OLTP) and data warehousing environments, where it is common to have many large indexes. In both cases, index compression can cause significant performance improvements in I/O-bound environments and little or no performance decrements in CPU-bound systems.

If compression is enabled on a table with an XML column, the XML data that is stored in the XDA object is also compressed. A separate compression dictionary for the XML data is stored in the XDA object. XDA compression is not supported for tables whose XML columns were created prior to this version; for such tables, only the data object is compressed.

11.3 Partitioning features

In this section, we introduce DB2 partitioning features.

11.3.1 Database partitioning feature

If you require increased processing power or scalability of your data beyond the capabilities of a single server, you can achieve this increased processing power or scalability with database partitioning. *Database Partitioning Feature (DPF)* allows partitioning of a database across multiple physical servers or within a large symmetric multiprocessor (SMP) server, as shown in Figure 11-3. DPF offers scalability, because you can add new servers and spread your database across them. DPF offers more CPU, more memory, and more disk and is ideal for the larger databases that are used for data warehousing, data mining, and online analytical processing (OLAP) or for working with online transaction processing (OLTP) workloads.

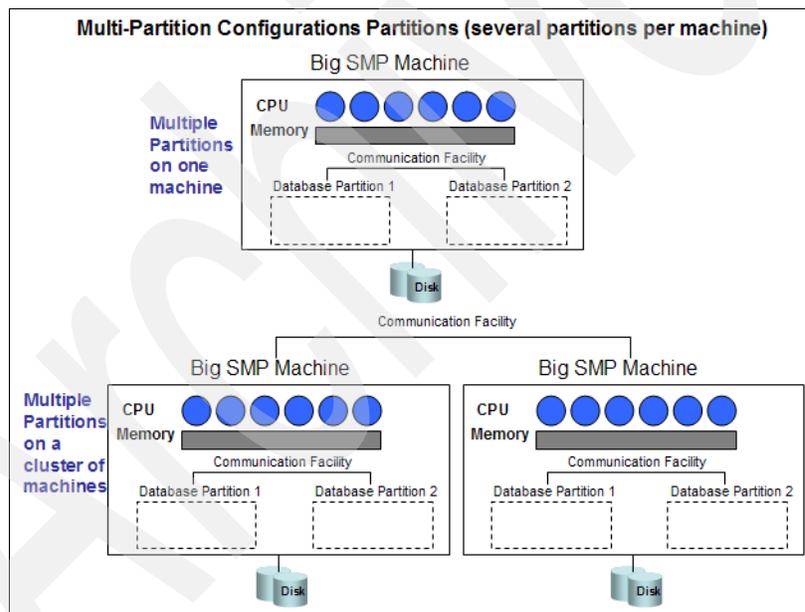


Figure 11-3 Database partitioning feature

The database partitioning feature uses a shared-nothing architecture. All data can be distributed and split across the participating nodes. From a client perspective, when a DB2 database is split across separate nodes using DPF, applications and administrators will still see a single database copy and access data within it as though it was never partitioned. When scanning through data,

DB2 will automatically parallelize queries by splitting them up and sending them directly to the nodes that hold the parts of the data that was requested.

These systems can also be made highly available when using shared storage. In this case, two or more nodes can share the file systems holding the table spaces. If an outage occurs, the surviving node can immediately access the failed node's table spaces and continue processing.

Remember you can use database partitioning efficiently in combination with table partitioning and multidimensional clustering.

11.3.2 Table partitioning

Table partitioning allows for the organization of table data into multiple storage objects called *data partitions*. The table can be organized based on one or more column values. The major benefit of table partitioning is that the partitions can be distributed onto multiple table spaces and can greatly increase the table capacity limit. While the data externally appears as a single table, internally DB2 can increase query performance by optimizing the access plans to use the method of partition elimination. Most importantly, table partitioning makes managing table data easier using roll-in and roll-out operations.

To partition a particular table, specify the `PARTITION BY RANGE` clause and the partitioning columns. You can specify multiple columns and generated columns. The column must be a base type, no large objects (LOBs), `LONG VARCHARs`, and so forth. Figure 11-4 on page 390 describes the various syntax for creating the same table partitions. The first `CREATE` table statement creates a table with three partitions on the `c1` column. This statement creates a partition to hold data for each of the following ranges: 1 - 33, 34 - 66 and 67 - 99. We refer to this `CREATE` table statement as *short form*, because it allows DB2 to create, name, and distribute the partitions over three table spaces. In the second `CREATE TABLE` statement, the user specifies the partition names by using the `PARTITION` or `PART` key word. In this example, the user also specifies the table spaces, in which each partition must be stored.

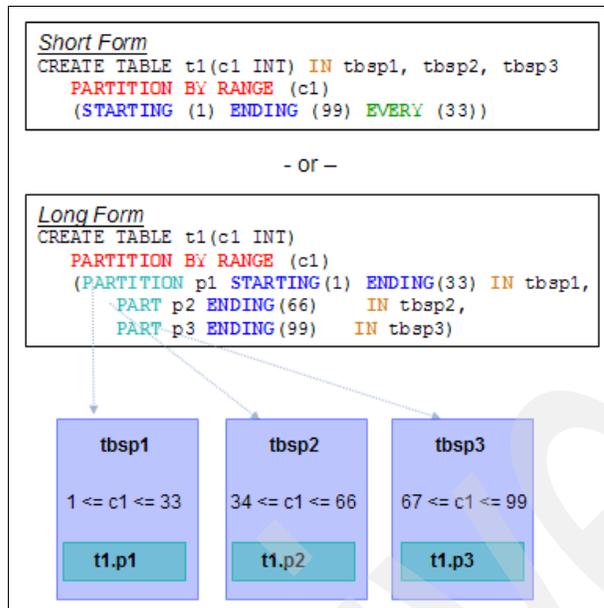


Figure 11-4 Syntax for creating a range-partitioned table

After creating a partitioned table, open INSERT, UPDATE, or LOAD into the table, and DB2 automatically inserts rows into the appropriate table partition according to the specified range. If the inserted data does not fit within the ranges of any of the partitions, DB2 produces an error.

Traditionally, in order to archive older data, you moved data to the archived locations, and you issued delete statements to remove the data from the current table. This effort results in a full table scan to find all rows belonging to the requested range. By using table partitioning, each table partition can be quickly separated from the table using the DETACH PARTITION key words in the ALTER TABLE statement. Example 11-5 describes the syntax for dropping a particular table partition.

Example 11-5 Detaching a table partition

```
db2> ALTER TABLE sales DETACH PARTITION Q4_2009 INTO TABLE 01dMonthSales

db2> COMMIT

db2> EXPORT 01dMonthSales; DROP 01dMonthSales
```

Physically, there is no impact to the system when using the ALTER table DETACH PARTITION command. The command is extremely fast, because no

data movement takes place. As you can see in Figure 11-5, the table containing the detached partition resides in the same table space as the original table partition. The DETACH only changes catalog entries to let DB2 know that the table partition is no longer a part of the main table. After the statement is committed, the detached data is available from the new table name. From now on, the table is a regular table, and you can perform actions on it.

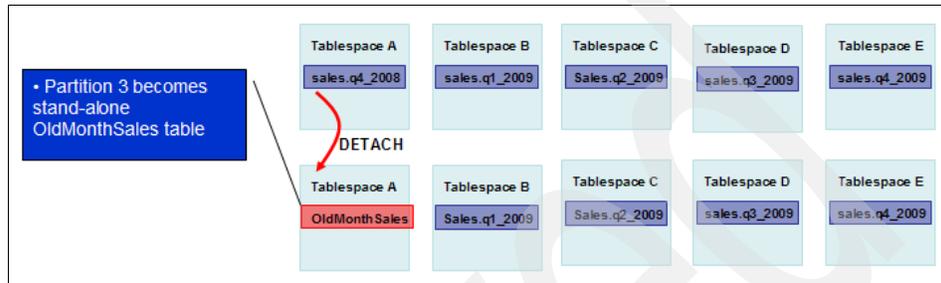


Figure 11-5 Detaching a table partition

The ATTACH command is similar to DETACH. For more details, visit the IBM DB2 Information Center:

<http://publib.boulder.ibm.com/infocenter/db21uw/v9r7/index.jsp>

Another advantage to table partitioning is *table partition elimination*. Table partition elimination is possible, because DB2 uses the partition definition to see which partitions contain the requested values. DB2 only queries those partitions with matching ranges. Figure 11-6 illustrates how DB2 uses query definitions for partition elimination.

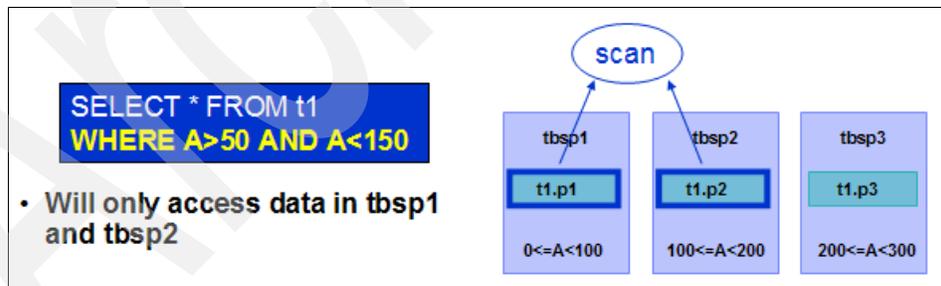


Figure 11-6 Example of a range-partitioned table

Remember that you can use table partitioning efficiently in combination with database partitioning and multidimensional clustering.

11.3.3 Multidimensional clustering

Multidimensional Clustering (MDC) provides a powerful method for improving the performance of SELECT, INSERT, UPDATE, and DELETE statements. Multidimensional clustering provides a way to organize data physically on disk according to provided dimensions. The data stays clustered on disk, which can significantly improve query performance and maintenance operations (table/index reorganization). This feature is best suited for data warehousing, OLTP, and large database environments.

The indexes for each dimension are block-based, not record-based, thus reducing their size (and the effort that is needed for logging and maintaining) dramatically. Reorganization of the table in order to re-cluster is unnecessary.

Example 11-6 shows the CREATE TABLE statement of an MDC table clustered in three columns: itemId, nation, and orderDate. The block indexes for each dimension are created automatically.

Example 11-6 Creating an MDC table

```
CREATE TABLE Orders(  
    itemId INT, nation varchar(30), orderDate date,  
    A INT, B INT, C Date, D INT ...)  
    IN Tablespace X, Tablespace Y, Tablespace Z ...  
    INDEX IN Tablespace Y  
    ORGANIZE BY DIMENSIONS (itemId, nation, orderDate)
```

Figure 11-7 on page 393 shows the data clustering according to the three dimensions as defined in Example 11-6.

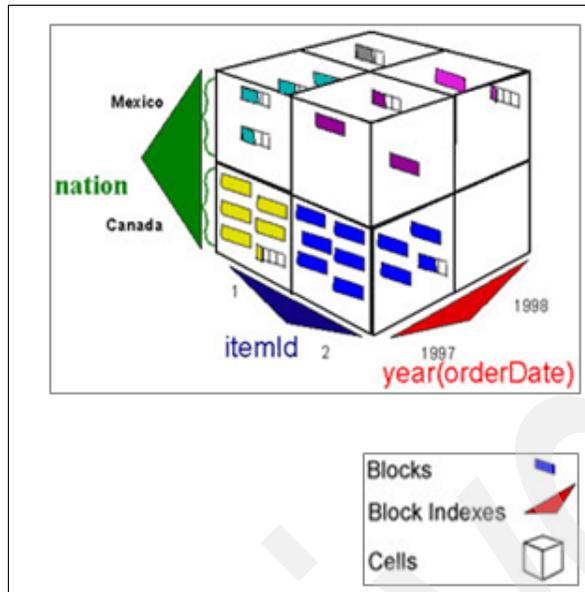


Figure 11-7 MDC table and indexes

When organizing by dimensions, you can specify one or more table columns. DB2 places all inserted rows with the same values for specific columns into a physical location close to one another. This special physical location is called a *block*. A block (*extent*) is a set of contiguous pages on disk, so access to these records is sequential and accessed with minimal I/O operations. If an existing block is filled, a new block is allocated. All blocks with the same combination of dimension values are grouped into *cells*. With this internal organization, DB2 can quickly find data along dimensions or find all rows for a specific combination of dimension values.

Benefits of multidimensional clustering

Multidimensional clustering tables include these advantages:

- ▶ Dimension block index lookups can identify the required portions of the table and quickly scan only the required blocks.
- ▶ Block indexes are smaller than record identifier (RID) indexes, thus, lookups are faster.
- ▶ Index ANDing and ORing can be performed at the block level and combined with RIDs.
- ▶ Data is guaranteed to be clustered on extents, which makes retrieval faster.
- ▶ Rows can be deleted faster when rollout can be used.

Note that you can use MDC efficiently in combination with database partitioning and table partitioning.

11.4 Materialized query tables

A *materialized query table (MQT)* is a table whose definition is based on the result of a query, and whose data is in the form of precomputed results that are taken from one or more tables on which the materialized query table definition is based. The DB2 optimizer can use these tables to determine whether a query can best be served by accessing an MQT instead of base tables.

MQTs are a powerful way to improve response time for complex queries, especially queries that might require several of the following operations:

- ▶ Aggregates data over one or more dimensions
- ▶ Joins and aggregates data over a group of tables
- ▶ Includes data from a commonly accessed subset of data, that is, from a “hot” horizontal or vertical database partition
- ▶ Repartitions data from a table, or part of a table, in a partitioned database environment

Knowledge of MQTs is integrated into the SQL and XQuery compiler. During compilation, the query rewrites phases, and the optimizer matches queries with MQTs to determine whether substitution of an MQT for a query that accesses the base tables is required. If an MQT is used, the EXPLAIN facility can provide information about which MQT was selected.

Because MQTs behave like regular tables in many ways, the same guidelines as MQTs apply for optimizing data access using table space definitions, creating indexes, and issuing RUNSTATS.

As an example, assume a database warehouse contains a set of customers and a set of credit card accounts. The data warehouse records the set of transactions that are made with the credit cards. Each transaction contains a set of items that are purchased together. This schema is classified as a *multi-star* schema, because it has two large tables: one table containing transaction items and the other table identifying the purchase transactions.

We use three hierarchical dimensions to describe a transaction in our example: product, location, and time. The product hierarchy is stored in two normalized tables representing the product group and the product line. The location hierarchy contains city, state, and country or region information and is represented in a single denormalized table. The time hierarchy contains day,

month, and year information and is encoded in a single date field. The date dimensions are extracted from the date field of the transaction using built-in functions. Other tables in this schema represent account information for customers and customer information.

An MQT is created with the sum and count of sales for each level of the following hierarchies:

- ▶ Product
- ▶ Location
- ▶ Time, composed of year, month, day

Many queries can be satisfied from this stored aggregate data. The following example shows how to create an MQT that computes the sum and the count of sales along the product group and line dimensions; along the city, state, and country dimension; and along the time dimension. It also includes several other columns in its GROUP BY clause. Example 11-7 is an example of the CREATE TABLE statement that will create an MQT table.

Example 11-7 Create MQT statement

```
CREATE TABLE dba.PG_SALESSUM
AS (
  SELECT l.id AS prodline, pg.id AS pgroup,
         loc.country, loc.state, loc.city,
         l.name AS linename, pg.name AS pgroupname,
         YEAR(pdate) AS year, MONTH(pdate) AS month,
         t.status,
         SUM(ti.amount) AS amount,
         COUNT(*) AS count
  FROM   cube.transitem AS ti, cube.trans AS t,
         cube.loc AS loc, cube.pgroup AS pg,
         cube.prodline AS l
  WHERE  ti.transid = t.id
         AND ti.pgid = pg.id
         AND pg.lineid = l.id
         AND t.locid = loc.id
         AND YEAR(pdate) > 1990
  GROUP BY l.id, pg.id, loc.country, loc.state, loc.city,
          year(pdate), month(pdate), t.status, l.name, pg.name
)
DATA INITIALLY DEFERRED REFRESH DEFERRED;

REFRESH TABLE dba.PG_SALESSUM;
```

The cost of computing the answer using the MQT can be significantly less than using a large base table, because a portion of the answer is already computed. MQTs can reduce expensive joins, sorts, and the aggregation of base data.

The larger the base tables become, the greater the improvements in response time can be, because the MQT grows more slowly than the base table. MQTs can effectively eliminate overlapping work among queries by performing the computation after the MQTs are built and refreshed, as well as reusing their content for many queries.

11.5 User-defined data types

In certain cases, existing built-in data types might not meet the needs of your application. User-defined types (UDTs) allow you to create or extend existing data types to your application needs.

There are six types of UDTs:

- ▶ Distinct type

A *distinct type* is a user-defined data type that is based on an existing built-in data type. Internally, it is stored as an existing data type, but it is considered as a separate and incompatible type. The major advantages of using distinct types are extensibility, strong typing, encapsulation, and customization.

A distinct type can be created by issuing the CREATE DISTINCT TYPE statement. The following statement defines a new distinct type for our sample application where we want all of the identification numbers to have common properties and functions. To achieve this goal, we create a distinct type ID, which contains INTEGER values:

```
db2> CREATE DISTINCT TYPE id AS integer with comparisons
```

- ▶ Structured type

A *structured type* is a user-defined data type that has a well defined structure consisting of existing built-in or user-defined data types. A structured type has the attributes and methods defined. The attribute defines its data storage properties, and methods define its behavior.

A structured type can be used as the type of a table, view, or column. When used as a type for a table or view, that table or view is known as a *typed table* or *typed view*. For typed tables and typed views, the names and data types of the attributes of the structured type become the names and data types of the columns of this typed table or typed view. Rows of the typed table or typed view can be thought of as a representation of instances of the structured type. When used as a data type for a column, the column contains values of that

structured type (or values of any of that type's subtypes, as defined next). Methods are used to retrieve or manipulate attributes of a structured column object.

A structured type can be created using the CREATE TYPE statement. For example, we can define a *product* and *sku* type, which can be used to create typed tables, as shown in Example 11-8. Figure 11-8 shows its hierarchy.

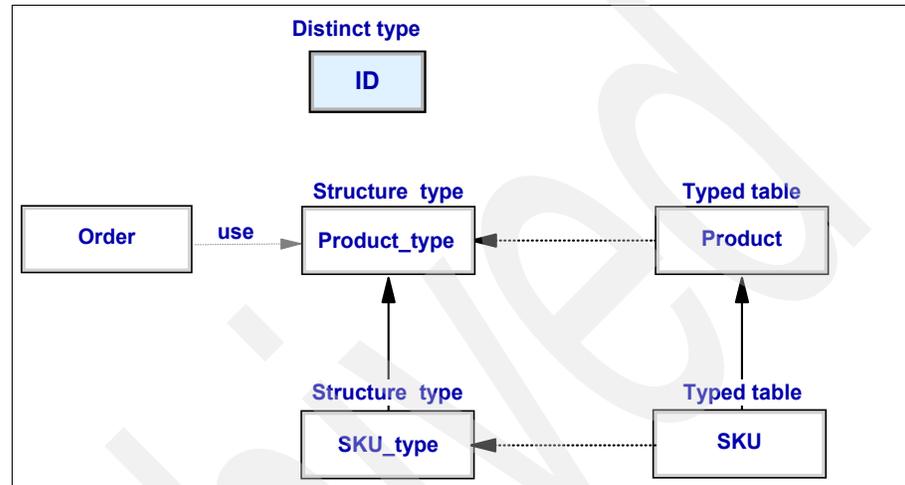


Figure 11-8 User-defined data types

Example 11-8 Structured types and typed tables

```

db2>CREATE TYPE product_type AS (
    name VARCHAR(20), description VARCHAR(200), brand VARCHAR(30))
instantiable ref using integer mode db2sql

db2>CREATE TYPE sku_type
    UNDER product_type AS (quantity integer)
    instantiable mode db2sql

db2>CREATE TABLE product
    OF product_type (ref is id user generated)

db2>CREATE TABLE sku of sku_type
    UNDER product inherit select privileges

db2>CREATE TABLE order(id ID,sku sku_type)
  
```

We can also use this type as a type for a column, as shown in the last statement of Example 11-8.

▶ Reference type

A *reference type* is a companion type to a structured type. Similar to a distinct type, a reference type is a scalar type that shares a common representation with one of the built-in data types. This same representation is shared for all types in the type hierarchy. The reference type representation is defined when the root type of a type hierarchy is created. When using a reference type, a structured type is specified as a parameter of the type. This parameter is called the *target type of the reference*.

The target of a reference is always a row in a typed table or a typed view. When a reference type is used, it can have a scope defined. The scope identifies a table (called the *target table*) or view (called the *target view*) that contains the target row of a reference value. The target table or view must have the same type as the target type of the reference type. An instance of a scoped reference type uniquely identifies a row in a typed table or typed view, which is called the *target row*.

▶ Array type

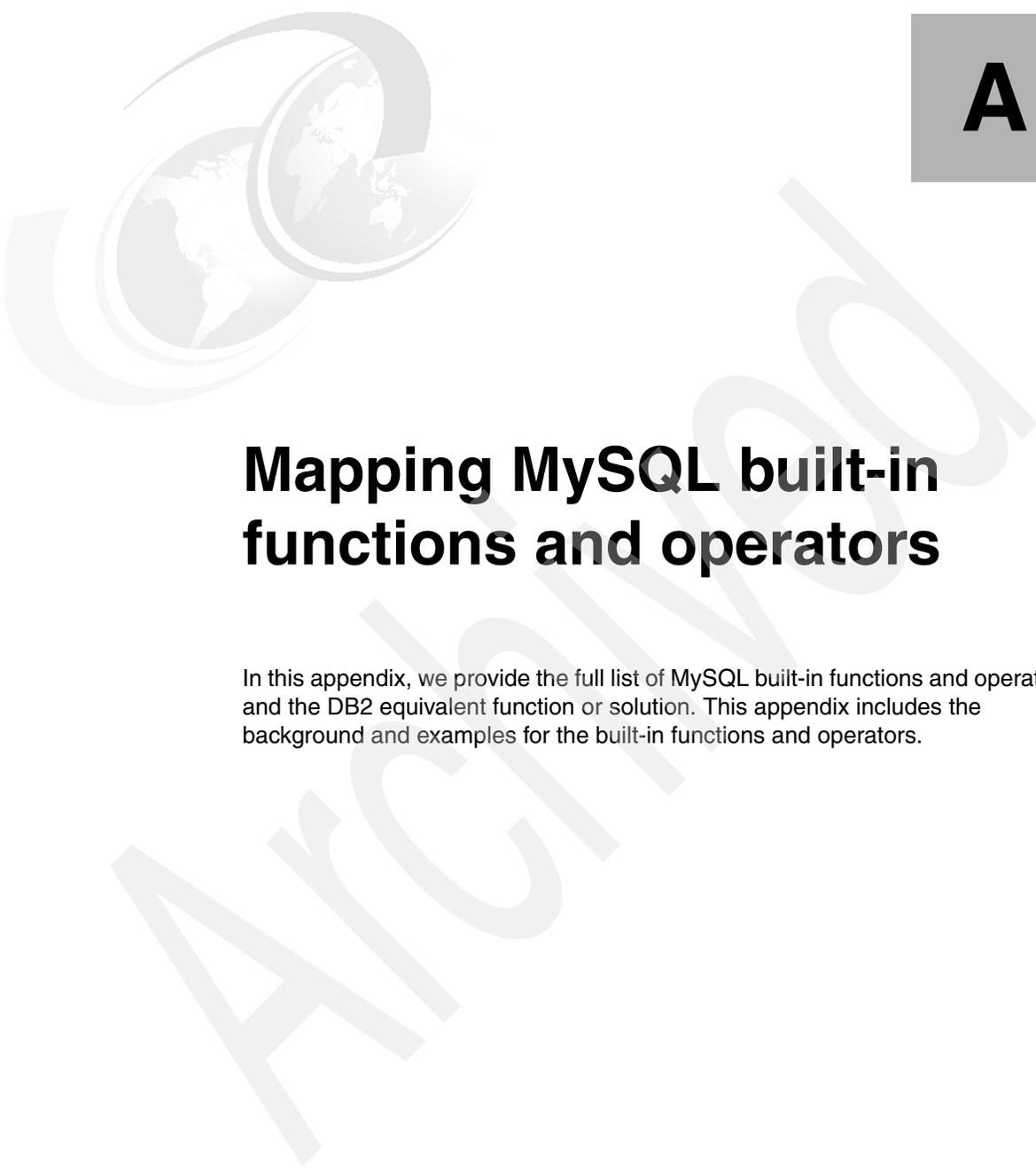
A *user-defined array type* is a data type that is defined as an array with elements of another data type. Every ordinary array type has an index with the data type of INTEGER and has a defined maximum cardinality. Every associative array has an index with the data type of INTEGER or VARCHAR and does not have a defined maximum cardinality.

▶ Row type

A *row type* is a data type that is defined as an ordered sequence of named fields, each with an associated data type, which effectively represents a row. A row type can be used as the data type for variables and parameters in SQL PL to provide simple manipulation of a row of data.

▶ Cursor data type

A *user-defined cursor type* is a user-defined data type that is defined with the keyword CURSOR and optionally with an associated row type. A user-defined cursor type with an associated row type is a strongly typed cursor type; otherwise, it is a weakly typed cursor type. A value of a user-defined cursor type represents a reference to an underlying cursor.



Mapping MySQL built-in functions and operators

In this appendix, we provide the full list of MySQL built-in functions and operators and the DB2 equivalent function or solution. This appendix includes the background and examples for the built-in functions and operators.

A.1 Grouping related functions

Table A-1 lists the differences in two databases and provides conversion examples.

Table A-1 MySQL and DB2 grouping related functions

MySQL function	MYSQL example	DB2 function	DB2 example	Notes
AVG([DISTINCT] expression)	mysql> SELECT a, AVG(b) FROM t1 GROUP BY a	AVG ([DISTINCT ALL] expression)	db2 " SELECT a, AVG(b) FROM t1 GROUP BY a"	Returns the average set of numbers
COUNT([DISTINCT] expression, expression,...)	mysql> SELECT a, COUNT(b) FROM t1 GROUP BY a	COUNT([DISTINCT ALL] expression). DB2 allows only one expression: Use CONCAT for character data type or CHAR and CONCAT on numeric data types	db2 " SELECT a, COUNT(b) FROM t1 GROUP BY a"	Returns the number of rows or values in a set of rows or values
MAX ([DISTINCT] expression)	mysql> SELECT a, MAX(b) FROM t1 GROUP BY a	MAX ([DISTINCT ALL] expression)	db2 "SELECT a, MAX(b) FROM t1 GROUP BY a"	Returns the maximum value in a set of values
MIN ([DISTINCT] expression)	mysql> SELECT a, MIN(b) FROM t1 GROUP BY a	MIN ([DISTINCT ALL] expression)	db2 "SELECT a, MIN(b) FROM t1 GROUP BY a"	Returns the minimum value in a set of values
STDDEV (expression) / STDDEV_POP (expression)	mysql> SELECT STDDEV (a), a FROM t1 GROUP BY a	STDDEV ([DISTINCT ALL] expression)	db2 " SELECT stddev(a), a FROM t1 GROUP BY a"	Returns the standard deviation (/n) of a set of numbers
SUM([DISTINCT] expression)	mysql> SELECT a, SUM(b) FROM t1 GROUP BY a	SUM([DISTINCT ALL] expression)	db2 " SELECT a, sum(b) FROM t1 GROUP BY a"	Returns the sum of a set of numbers
VAR_POP(expression) / VARIANCE(expression)	mysql> SELECT VAR_POP(a) FROM t1 GROUP BY a	VARIANCE ([DISTINCT ALL] expression)	db2 " SELECT VARIANCE(a) FROM t1 GROUP BY a"	Returns the variance of a set of numbers
BIT_AND (expression) <i>This function is an extension to SQL standards</i>	mysql> SELECT BIT_AND(a), a FROM t1 GROUP BY a	No equivalent function. Implement using user-defined function (UDF).	Refer to UDF B.1, "Sample code for BIT_AND" on page 414.	Returns the value of the bitwise logical AND operation
BIT_OR (expression) <i>This function is an extension to SQL standards</i>	mysql> SELECT BIT_OR(a), a FROM t1 GROUP BY a	No equivalent function. Implement using UDF.	Refer to UDF B.1, "Sample code for BIT_AND" on page 414.	Returns the value of the bitwise logical OR operation
BIT_XOR (expression) <i>This function is an extension to SQL standards</i>	mysql> SELECT BIT_XOR(a), a FROM t1 GROUP BY a	No equivalent function. Implement using UDF.	Refer to UDF B.1, "Sample code for BIT_AND" on page 414.	Returns the value of the bitwise logical XOR operation

GROUP_CONCAT(<i>expression</i>) This function is an extension to SQL standards	mysql> SELECT a, GROUP_CONCAT(b) FROM t1 group by a	No equivalent function. Implement using UDF.		Returns a concatenated variable for all values
STD <i>This function is an extension to SQL standards</i>	mysql> SELECT STD(a),a FROM t1 GROUP BY a	STDDEV ([DISTINCT ALL] expression)	db2 "SELECT stddev(a), a FROM t1 GROUP BY a"	Returns the standard deviation (\sqrt{n}) of a set of numbers
Not Available		CORRELATION (expression, expression)	db2 " SELECT CORRELATION(a, b) FROM t1 WHERE c = 52"	Returns the coefficient of correlation of a set of number pairs
Not Available		COVARIANCE (expression, expression)	db2 "SELECT COVARIANCE(a, b) FROM t1 WHERE c = 52 "	Returns the (population) covariance of a set of number pairs
Not Available		GROUPING(<i>expression</i>)	db2 " SELECT a, b, SUM(c) AS x, GROUPING(a) AS y, GROUPING(b) AS z FROM t1 GROUP BY CUBE (a, b) ORDER BY a, b"	Returns a value that indicates whether a row returned in a GROUP BY answer set is a row generated by a grouping set that excludes the column represented by <i>expression</i>
GROUP BY on alias	mysql> SELECT a as x FROM a GROUP BY x;	Use column name for grouping.	db2 " SELECT a FROM t1 GROUP BY a"	
GROUP BY on position	mysql> SELECT a FROM t1 GROUP BY 1	Use column name for grouping.	db2 " SELECT a FROM t1 GROUP BY a"	
HAVING on alias	mysql> SELECT a as x FROM t1 GROUP BY a HAVING x > 0	Use column name in having clause.	db2 " SELECT a FROM t1 GROUP BY a HAVING a > 0"	

A.2 String functions

Table A-2 provides an overview of MySQL string-related functions and how to convert these functions to DB2.

Table A-2 MySQL and DB2 string functions

MySQL function	MySQL example	DB2 function	DB2 example	Notes
ASCII(string)	mysql> SELECT ascii('a'); +-----+ ascii('a') +-----+ 97 +-----+ 1 row in set (0.00 sec)	ASCII(string)	db2 "VALUES ascii('a') " 1 ----- 97 1 record(s) selected.	Returns ASCII code value
BIN(integer)	mysql> SELECT bin(5); +-----+ bin(5) +-----+ 101 +-----+ 1 row in set (0.00 sec)	No equivalent function. Alternative use UDF		Returns the binary value of the string
BIT_LENGTH(string)	mysql> SELECT bit_length('Hello') ; +-----+ bit_length('Hello') +-----+ 40 +-----+ 1 row in set (0.00 sec)	LENGTH(string, CODEUNITS16 CODEUNITS32 OCTETS)	db2 "VALUES length('Hello')*8 " 1 ----- 40 1 record(s) selected.	Result depends on the encoding scheme used for character data: single byte, double byte, UTF-8
CHAR_LENGTH(string) / CHARACTER_LENGTH(string)	mysql> SELECT CHAR_LENGTH('Orange'); +-----+ CHAR_LENGTH('Orange') +-----+ 6 +-----+ 1 row in set (0.00 sec)	CHARACTER_LENGTH(string, CODEUNITS16 CODEUNITS32 OCTETS),/CHAR_LENGTH(string, CODEUNITS16 CODEUNITS32 OCTETS)	db2 " VALUES CHAR_LENGTH('Orange', CODEUNITS32)" 1 ----- 6 1 record(s) selected.	Returns the number of bytes for expression. For double-byte character set (DBCS), the number of DBCS characters is returned
CHAR(int, [USING character set])	mysql> SELECT char(97); +-----+ char(97) +-----+ a +-----+ 1 row in set (0.00 sec)	CHR(string)	db2 "VALUES chr('97') " 1 - a 1 record(s) selected.	Returns the character that has the ASCII code value specified by the argument

CONCAT_WS(separator, string, string,...)	mysql> SELECT CONCAT_WS('-', firstname, lastname, loginname) as FULLNAME from owners where id = 501; +-----+ FULLNAME +-----+ Angela-Carlson-acarlson +-----+ 1 row in set (0.01 sec)	Use to implement CONCAT(list).	db2 "SELECT (firstName '-' lastName '-' loginName) as fullName from admin.owners where id = 501" FULLNAME ----- Angela-Carlson-acarlson 1 record(s) selected.	Returns the concatenation of string arguments with separator
CONCAT(strin g, string,...)	mysql> SELECT CONCAT(firstname, '', lastname) as FULLNAME from owners where id = 501; +-----+ FULLNAME +-----+ Angela Carlson +-----+ 1 row in set (0.00 sec)	Use CONCAT(string, string) or to implement CONCAT(list).	db2 "SELECT (firstName '' lastName) as fullName from admin.owners where id = 501" FULLNAME ----- Angela Carlson 1 record(s) selected.	Returns the concatenation of string arguments
ELT(int,str1,str 2,str3,...)	mysql> SELECT ELT(2, firstname, lastname, loginname) as FULLNAME from owners where id = 501; +-----+ FULLNAME +-----+ Carlson +-----+ 1 row in set (0.00 sec)	No equivalent function. Alternative use CASE expression or UDF.		Returns the <i>n</i> th string or NULL
EXPORT_SET(bit, onString, offString, length)	mysql> SELECT EXPORT_SET(12,'ON','OFF',' ',4); +-----+ EXPORT_SET(12,'ON','OFF',' ',4) +-----+ OFFIOFFIONION +-----+ 1 row in set (0.00 sec)	No equivalent function. Alternative use UDF		Returns a string representation for each of the bit values of a binary value
FIELD(string,st ring1,string2,str ing3,...)	mysql> SELECT FIELD('Carlson', firstname, lastname, loginname) as FULLNAME from owners where id = 501; +-----+ FULLNAME +-----+ 2 +-----+ 1 row in set (0.00 sec)	No equivalent function. Alternative use CASE expression or UDF		Returns the position of the matching string
FIND_IN_SET(expr)	mysql> SELECT FIND_IN_SET('Doe','John,Doe,Stas ie,Smith') as POSITION; +-----+ POSITION +-----+ 2 +-----+ 1 row in set (0.01 sec)	No equivalent function. Implement using UDF.		Returns the position in the set of the matching string

FORMAT(double, integer)	<pre> FORMAT: select format(1234.5555, 2) returns 1,234.56 </pre>	No equivalent function. Implement using UDF.	Refer to UDF B.2, "Sample code for FORMAT function" on page 415.	Returns the rounded string
HEX(string)	<pre> mysql> SELECT HEX(255); +-----+ HEX(255) +-----+ FF +-----+ 1 row in set (0.00 sec) </pre>	HEX(string)	<pre> db2 "VALUES HEX(00000255)" 1 ----- FF000000 1 record(s) selected. </pre>	Returns a hexadecimal representation of a value as a character string
INSERT(string, position, length, substring)	<pre> mysql> SELECT INSERT('original', 2, 2, 'NEW'); +-----+ INSERT('original', 2, 2, 'NEW') +-----+ oNEWginal +-----+ 1 row in set (0.00 sec) </pre>	INSERT(string, position, length, substring)	<pre> db2 "VALUES INSERT('original', 2, 2, 'NEW')" 1 ----- oNEWginal 1 record(s) selected. </pre>	Inserts a string into an existing string
INSTR(substring, string) / LOCATE(substring, string, [position]) / POSITION(substring, string)	<pre> mysql> SELECT LOCATE('N', 'DINING') ->; +-----+ LOCATE('N', 'DINING') +-----+ 3 +-----+ 1 row in set (0.00 sec) </pre>	LOCATE(substring, string, [start], [CODEUNITS16 CODEUNITS32 OCTETS])	<pre> db2 " SELECT LOCATE('N', 'DINING') FROM SYSIBM.SYSDUMMY1" 1 ----- 3 1 record(s) selected. </pre>	Returns the starting position of the first occurrence of one string within another string
LCASE(string) / LOWER(string)	<pre> mysql> SELECT LCASE('JOB'); +-----+ LCASE('JOB') +-----+ job +-----+ 1 row in set (0.00 sec) </pre>	LCASE(string)	<pre> db2 " SELECT LCASE('JOB') FROM SYSIBM.SYSDUMMY1" 1 --- job 1 record(s) selected. </pre>	Returns a string in which all characters have been converted to lowercase characters
LEFT(string1, string2)	<pre> mysql> SELECT LEFT('Inventory', 6); +-----+ LEFT('Inventory', 6) +-----+ Invent +-----+ 1 row in set (0.00 sec) </pre>	LEFT(string1, integer)	<pre> db2 "SELECT LEFT('Inventory', 6) FROM SYSIBM.SYSDUMMY1" 1 ----- Invent 1 record(s) selected. </pre>	Returns a string consisting of the leftmost <i>String</i> up to the integer position
LENGTH(string) / OCTET_LENGTH(string)	<pre> mysql> select LENGTH('Jürgen') ; +-----+ LENGTH('Jürgen') +-----+ 7 +-----+ 1 row in set (0.00 sec) </pre>	LENGTH(string) / OCTET_LENGTH(string)	<pre> db2 "select LENGTH('Jürgen') from admin.owners where id = 501" 1 ----- 7 1 record(s) selected. </pre>	Returns the length of <i>string</i> in the implicit or explicit string unit
LOAD_FILE(directory)	<pre> update blobTBL SET data = LOAD_FILE('/tmp/AquaBlue.jpg') WHERE id = 6; </pre>	Use the LOAD command with LOBS FROM <lob_directory>.		Inserts the file into the database

LPAD(string, length, substring) / RPAD(string, length, substring)	mysql> SELECT LPAD('TEST',6,'!!!'); +-----+ LPAD('TEST',6,'!!!') +-----+ !!!TEST +-----+ 1 row in set (0.00 sec)	No equivalent function. Implement using UDF.	Refer to UDF B.3, "Sample code for RPAD and LPAD functions" on page 416.	Returns the string of the given length. If the length is longer than the string, the substring characters will be added to the left or right end.
LTRIM(string)	mysql> SELECT LTRIM(' Apple'); +-----+ LTRIM(' Apple') +-----+ Apple +-----+ 1 row in set (0.00 sec)	LTRIM(string)	db2 "SELECT LTRIM(' Apple') FROM SYSIBM.SYSDUMMY1" 1 ----- Apple	Removes blanks from the beginning of <i>string-expression</i>
MAKE_SET(bits, string, string,...)	mysql> SELECT MAKE_SET(214, "HELLO", "BYE", "GOOD DAY", "GOOD NIGHT") as OUTPUT; +-----+ OUTPUT +-----+ BYE,GOOD DAY +-----+ 1 row in set (0.00 sec)	No equivalent function. Alternative use UDF		Returns a set of strings based on the bit
OCT(expression)	mysql> select oct(22); +-----+ oct(22) +-----+ 26 +-----+	No equivalent function. Alternative use UDF		Returns the octal value of the expression
QUOTE(string)	mysql> SELECT quote(firstname) from owners where id = 501; +-----+ quote(firstname) +-----+ 'Angela' +-----+ 1 row in set (0.00 sec)	SELECT with	db2 "select ('" firstname "'") from admin.owners where id = 501" 1 ----- 'Angela' 1 record(s) selected.	Returns string with single quotes
REPEAT(string , integer)	mysql> select REPEAT('REPEAT THIS', 5); +-----+ REPEAT('REPEAT THIS', 5) +-----+ REPEAT THIS REPEAT THIS REPEAT THIS REPEAT THIS REPEAT THIS +-----+ 1 row in set (0.00 sec)	REPEAT(string, integer)	db2 "VALUES REPEAT('REPEAT THIS', 5)" 1 ----- REPEAT THIS REPEAT THIS REPEAT THIS REPEAT THIS REPEAT THIS 1 record(s) selected.	Returns the repeated string <i>N</i> times

REPLACE(string1, string2, string3)	mysql> SELECT REPLACE('DINING', 'N', 'VID'); +-----+ REPLACE('DINING', 'N', 'VID') +-----+ DIVIDIVIDG +-----+ 1 row in set (0.00 sec)	REPLACE(string1, string2, string3)	db2 "VALUES REPLACE('DINING', 'N', 'VID') " 1 ----- DIVIDIVIDG 1 record(s) selected.	Returns as string with all occurrences of <i>string2</i> in <i>string1</i> with <i>string3</i>
REVERSE(string1, S)	mysql> SELECT REVERSE('abc'); +-----+ REVERSE('abc') +-----+ cba +-----+ 1 row in set (0.00 sec)		SET (RESTSTR, LEN) = (INSTR, LENGTH(INSTR)); WHILE LEN > 0 DO SET (REVSTR, RESTSTR, LEN)= (SUBSTR(RESTSTR, 1, 1) REVSTR, SUBSTR(RESTSTR, 2, LEN - 1), LEN - 1); END WHILE ;	Returns the reverse order of the string. Implement UDF. For the complete code of the example, refer to <i>IBM DB2 SQL Reference, Volume 1, V8, SC10-4249</i> .
RIGHT(string, length)	mysql> SELECT RIGHT('Inventory', 4); +-----+ RIGHT('Inventory', 4) +-----+ tory +-----+ 1 row in set (0.00 sec)	RIGHT(string, length)	db2 "VALUES RIGHT('Inventory', 4) " 1 ----- tory 1 record(s) selected.	Returns a string consisting of the rightmost <i>String</i> starting from the integer position
RTRIM(string)	mysql> SELECT RTRIM('PEAR '); +-----+ RTRIM('PEAR ') +-----+ PEAR +-----+ 1 row in set (0.00 sec)	RTRIM(string)	db2 "VALUES RTRIM('PEAR ') " 1 ----- PEAR 1 record(s) selected.	Removes blanks from the end of <i>string</i>
SOUNDEX(string)	mysql> SELECT SOUNDEX('apple'); +-----+ SOUNDEX('apple') +-----+ A140 +-----+ 1 row in set (0.00 sec)	SOUNDEX(string)	db2 "VALUES SOUNDEX('apple') " 1 ----- A140 1 record(s) selected.	Returns a 4-character code representing the sound of the words in the argument. MySQL: String1 SOUNDS LIKE string2 can be ported to SOUNDEX(String 1) = SOUNDEX(string 2) in DB2
SPACE(expression)	mysql> SELECT space(30); +-----+ space(30) +-----+ +-----+ 1 row in set (0.00 sec)	SPACE(expression)	db2 " VALUES space(3) " 1 -----	Returns a character string consisting of blanks with length specified by the second argument

STRCMP(string, string)	mysql> SELECT STRCMP('test', 'testing'); +-----+ STRCMP('test', 'testing') +-----+ -1 +-----+ 1 row in set (0.00 sec)	CASE	Implement using CASE expression and VALUES statement.	Returns -1 if the first string is smaller, 0 if the strings are the same length, 1 if the second string is smaller
SUBSTRING_INDEX()	mysql> SELECT SUBSTRING_INDEX('This is a test...', 't', 2); +-----+ + SUBSTRING_INDEX('This is a test...', 't', 2) +-----+ + This is a test +-----+ + 1 row in set (0.00 sec)	No equivalent function. Implement using UDF.	Refer to UDF B.7, "Sample code for SUBSTRING_INDEX" on page 432.	
SUBSTRING(string, position, length) / MID(string, position, length) / SUBSTR(string, position, length)	mysql> select substring('abcdef', 2, 3); +-----+ substring('abcdef', 2, 3) +-----+ bcd +-----+ 1 row in set (0.00 sec)	SUBSTR(string, position, length)	db2 "VALUES('abcdef', 2, 3)" 1 --- bcd 1 record(s) selected.	Returns a substring of a string
TRIM([Both Leading trailing substring] FROM] string)	mysql> select trim(trailing from trim(LEADING FROM ' abc ')) as OUTPUT; +-----+ OUTPUT +-----+ abc +-----+ 1 row in set (0.00 sec)	TRIM([Both Leading trailing substring] FROM] string)	db2 "VALUES trim(trailing from trim(LEADING FROM ' abc '))" OUTPUT ----- abc 1 record(s) selected.	Removes blanks or occurrences of another specified character from the end or the beginning of a string expression
UCASE(string) / UPPER(String)	mysql> SELECT UPPER('jobs'); +-----+ UPPER('jobs') +-----+ JOBS +-----+ 1 row in set (0.00 sec)	UCASE(string) / UPPER (String)	db2 "VALUES UPPER('jobs')" 1 --- JOBS 1 record(s) selected.	Returns a string in which all characters have been converted to uppercase characters
UNHEX()	mysql> SELECT UNHEX('546F646179'); +-----+ UNHEX('546F646179') +-----+ Today +-----+ 1 row in set (0.00 sec)	No equivalent function. Alternative use UDF		Converts hexadecimal digits to a character string

A.3 Numeric functions

Table A-3 lists numeric functions that are specific to manipulating numbers.

Table A-3 Numeric functions

MySQL command	MySQL example	DB2 command	DB2 example
ABS(expression)	mysql> SELECT ABS(-51234);	ABS(expression)	db2 "VALUES ABS(-51234)"
ACOS(expression)	mysql> SELECT COS(1);	ACOS(expression)	db2 "VALUES COS(1)"
ASIN(expression)	mysql> SELECT ASIN(1);	ASIN(expression)	db2 "VALUES ASIN(1)"
ATAN(expression, expression), ATAN2(expression, expression)	mysql> SELECT ATAN2(1,-1);	ATAN(expression), ATAN2(expression, expression),	db2 "VALUES ATAN2(-1,1)"
CEIL(expression), CEILING(expression)	mysql> SELECT CEILING(3.35);	CEIL(expression), CEILING(expression)	db2 "VALUES Ceiling(3.35)"
COS(expression)	mysql> SELECT cos(0);	COS(expression)	db2 "VALUES COS(0)"
COT(expression)	mysql> SELECT coT(1);	COT(expression)	db2 "VALUES COT(1)"
DEGREES(expression)	mysql> SELECT DEGREES(3);	DEGREES(expression)	db2 "VALUES DEGREES(3)"
EXP(expression)	mysql> SELECT EXP(3);	EXP(expression)	db2 "VALUES EXP(3)"
FLOOR(expression)	mysql> SELECT FLOOR(3.35);	FLOOR(expression)	db2 "VALUES FLOOR(3.35)"
LN(expression)	mysql> SELECT LN(100);	LN(expression)	db2 "VALUES LN(100)"
LOG10(expression)	mysql> SELECT LOG10(100);	LOG10(expression)	db2 "VALUES LOG10(100)"
LOG(expression, [expression])	mysql> SELECT LOG(100);	LOG(expression)	db2 "VALUES LOG(100)"
MOD(expression, expression)	mysql> SELECT MOD(125, 50);	MOD(expression, expression)	db2 "VALUES MOD(125, 50)"
POW(expression, expression), POWER(expression, expression)	mysql> SELECT POWER(5, 2);	POWER(expression, expression)	db2 "VALUES POWER(5, 2)"
RADIANS(expression)	mysql> SELECT RADIANS(180);	RADIANS(expression)	db2 "VALUES RADIANS(180)"
RAND([expression])	mysql> SELECT RAND();	RAND([expression])	db2 "VALUES RAND()"
ROUND(expression, expression)	mysql> SELECT ROUND(873.726, 1);	ROUND(expression, expression)	db2 "VALUES ROUND(873.726, 1)"
SIGN(expression)	mysql> SELECT SIGN(6453);	SIGN(expression)	db2 "VALUES SIGN(6453)"

MySQL command	MySQL example	DB2 command	DB2 example
SIN(expression)	mysql> SELECT SIN(3);	SIN(expression)	db2 "VALUES SIN(3)"
SQRT(expression)	mysql> SELECT SQRT(25);	SQRT(expression)	db2 "VALUES SQRT(25)"
TAN(expression)	mysql> SELECT TAN(2);	TAN(expression)	db2 "VALUES TAN(2)"
TRUNCATE(expression, expression)	mysql> SELECT TRUNCATE(873.726,2);	TRUNCATE(expression, expression)	db2 "VALUES TRUNCATE(873.726,2)"

A.4 Date and time functions

Table A-4 lists date and time-related functions.

Table A-4 Date and time-related functions

MySQL command	MySQL example	DB2 command	DB2 example
ADDDATE(), DATE_ADD(), DATE_SUB(), SUBDATE()	mysql> SELECT DATE_ADD('2009-8-31', INTERVAL 15 DAY);	DATE + expression	db2 "VALUES DATE('2009-8-31') + 15 days"
CURDATE(), CURRENT_DATE(), CURRENT_DATE	mysql> SELECT CURDATE();	CURRENT DATE	db2 "VALUES CURRENT DATE"
CURRENT_TIME(), CURRENT_TIME(), CURTIME()	mysql> SELECT CURRENT_TIME();	CURRENT TIME	db2 "VALUES CURRENT TIME"
DAYOFMONTH (expression), DAY(expression)	mysql> SELECT DAYOFMONTH('2009-08-31 05:06:00');	Day(expression)	db2 "VALUES DAY('2009-08-31 05:06:00')"
EXTRACT (unit FROM expression)	mysql> SELECT EXTRACT(YEAR_MONTH TH from '2009-08-31 05:06:00');	Use concatenate different date stripping functions (DAY, YEAR, MONTH, DAYNAME, DAYOFWEEK, and so on).	db2 "VALUES (YEAR('2009-08-31 05:06:00') MONTH('2009-08-31 05:06:00'))"
HOURL(expression)	mysql> SELECT HOUR('2009-08-31 05:06:00');	MIDNIGHT_SECONDS (expression)/60/60	db2 "VALUES MIDNIGHT_SECONDS('2009-08-31 05:06:00')/60/60"
SEC_TO_TIME()		UDF OR SELECT with time Arithmetic with HOUR, MINUTE, SECOND	SEC_TO_TIME(arg INTEGER) RETURNS TIME CONTAINS SQL NO EXTERNAL ACTION DETERMINISTIC RETURN TIME('00:00:00') + (arg / 3600) HOURS + MOD(arg / 60, 60) MINUTES + MOD(arg, 3600) SECONDS "
STR_TO_DATE(expression, format)	mysql> SELECT STR_TO_DATE('15,03,2009','%d,%m,%Y');	DATE and TO_DATE	db2 "VALUES DATE(TO_DATE('15,3,2009','DD,MM,YYYY'))"
WEEK(expression) WEEKOFYEAR (expression, mode)	mysql> SELECT week('2009-08-31 05:06:00');	WEEK(expression)	db2 "VALUES WEEK('2009-08-31 05:06:00')"

MySQL command	MySQL example	DB2 command	DB2 example
WEEKDAY(expression)	mysql> SELECT weekday('2009-08-31 05:06:00');	DAYOFWEEK(expression)	db2 " VALUES DAYOFWEEK('2009-08-31 05:06:00')" db2 " VALUES DAYOFWEEK_ISO ('2009-08-31 05:06:00')"

A.5 Comparing operators and other functions

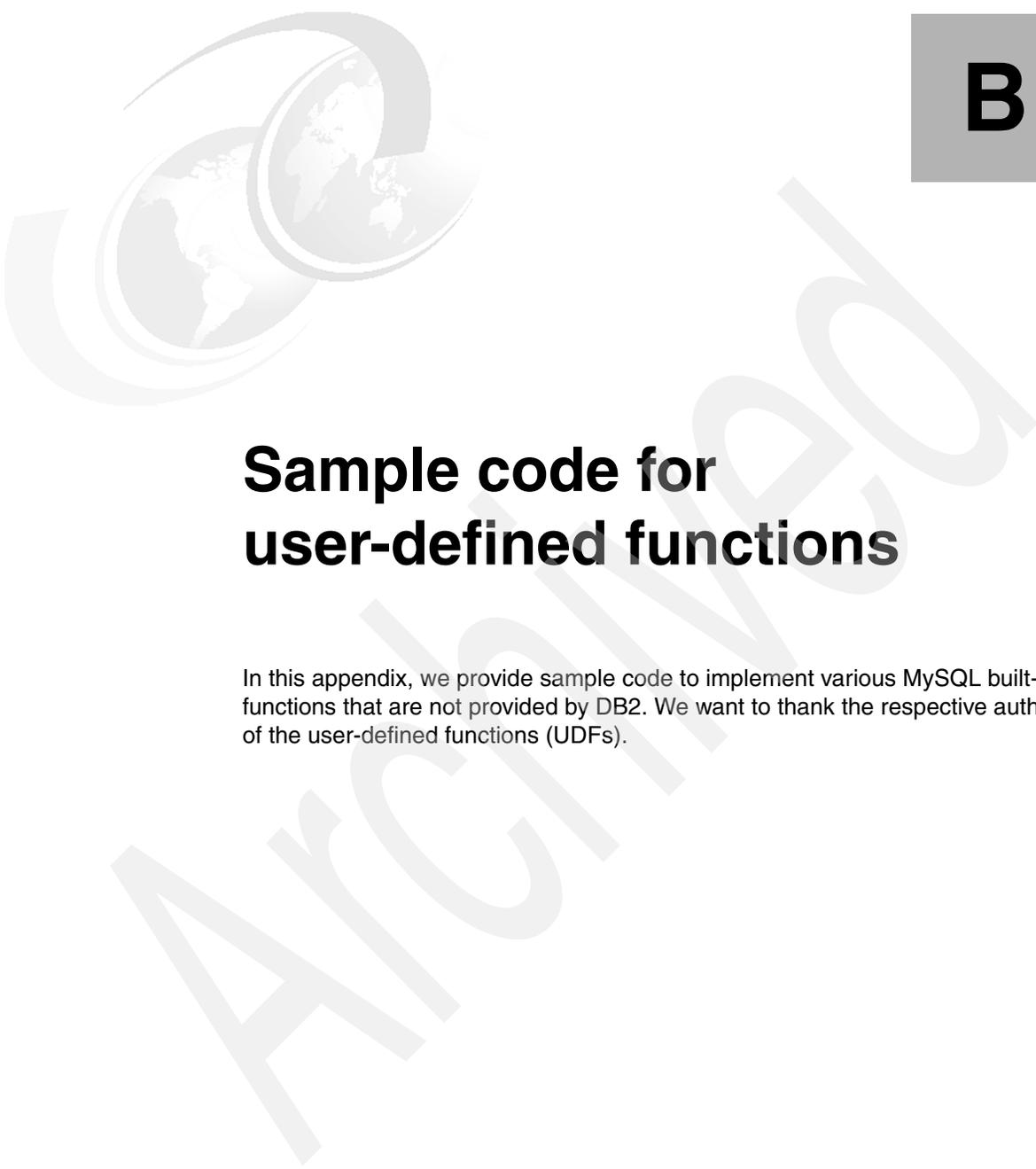
Table A-5 shows MySQL and DB2 operators.

Table A-5 MySQL and DB2 operator comparison

MySQL	DB2	Comments
logical NOT as '!' in SELECT list	VALUES CASE WHEN 1!=1 THEN 0 ELSE 1 END	Implement using CASE expression and VALUES statement
%	MOD	In MySQL, % is a synonym for modulo
& (bitwise and)	Not available. Implement using UDF.	Refer to UDF B.1, "Sample code for BIT_AND" on page 414.
logical AND as '&&' in SELECT list	CASE	Implement using CASE expression and VALUES statement.
not equal, <> or != in SELECT list:select 1<>1	SELECT CASE WHEN 1<> 1 THEN x ELSE y END	Implement using CASE expression.
Function = in SELECT list: select (1=1)	CASE	Implement using CASE expression and VALUES statement.
BETWEEN in SELECT	CASE	Implement using CASE expression and VALUES statement.
<< and >> (bitwise shifts)	No equivalent	Implement using <i>power</i> function: MySQL: SELECT (x>>y) SELECT(x<<y) DB2 : SELECT(x/power(2,y)) SELECT(x*power(2,y)):
BIT_COUNT	No equivalent, implement using UDF	Refer to UDF B.6, "Sample code for BIT_COUNT" on page 431.
ENCRYPT	ENCRYPT	DB2 requires encryption password.
FIELD	CASE	Implement using CASE expression and VALUES statement.
GREATEST	FnGratst	Refer to UDF B.4, "Sample code for GREATEST function" on page 422.
IF	CASE	Implement using CASE expression and VALUES statement.

IN on numbers in SELECT	CASE	Implement using CASE expression and VALUES statement.
IN on strings in SELECT	CASE	Implement using CASE expression and VALUES statement.
LOCATE as INSTR	LOCATE	Arguments are swapped.
INTERVAL	CASE	Implement using CASE expression and VALUES statement.
LAST_INSERT_ID	IDENTITY_VAL_LOCAL	DB2 returns NULL. MySQL returns the most recently assigned value for an identity column.
LEAST	FnLeastN	See UDF example in B.5, "Sample code for LEAST" on page 427.
LIKE in SELECT	CASE with LIKE	Implement using CASE expression and VALUES statement.
LIKE ESCAPE in SELECT	CASE with LIKE and ESCAPE	Implement using CASE expression and VALUES statement.
LOG(m,n)	LOG(m)/LOG(n)	To convert logarithm to an arbitrary base
NOT in SELECT	CASE	Implement using CASE expression and VALUES statement.
NOT BETWEEN in SELECT	CASE	Implement using CASE expression and VALUES statement.
NOT LIKE in SELECT	CASE	Implement using CASE expression and VALUES statement.
PASSWORD	ENCRYPT	For encryption
POW	POWER	Returns value of arg1 to the power of arg2
REGEXP in SELECT	No equivalent	See article on developerWorks® for workaround: http://www-106.ibm.com/developerworks/db2/library/techarticle/0301stolze/0301stolze.html
VERSION	db2level	To retrieve installed version of database management system (DBMS)

Archived



Sample code for user-defined functions

In this appendix, we provide sample code to implement various MySQL built-in functions that are not provided by DB2. We want to thank the respective authors of the user-defined functions (UDFs).

B.1 Sample code for BIT_AND

Example B-1 shows conversion code for the MySQL BIT_AND function.

Example: B-1 User-defined function to map BIT_AND

```
--
-- DB2 UDF(User-Defined Function) Samples for conversion
--
-- 2001/08/29
--
-- Name of UDF: BIT_AND (N1 Integer, N2 Integer)
--
-- Used UDF: None
--
-- Description: Returns bit by bit and of both parameters.
--
-- Author: TOKUNAGA, Takashi
--
-----
CREATE FUNCTION BITAND (N1 Integer, N2 Integer)
  RETURNS Integer
  SPECIFIC BITANDMySQL
  LANGUAGE SQL
  CONTAINS SQL
  NO EXTERNAL ACTION
  DETERMINISTIC
  RETURN
  WITH
  Repeat (S, M1, M2, Ans) AS
  (Values (0, N1, N2, 0)
  Union All
  Select S+1, M1/2, M2/2, Ans+MOD(M1,2)*MOD(M2,2)*power(2,S)
  From Repeat
  Where M1 > 0
  AND M2 > 0
  AND S < 32
  )
  SELECT ANS
  FROM Repeat
  WHERE S = (SELECT MAX(S)
  FROM Repeat)
  ;
```

Example B-2 shows the results of the BITAND user-defined function.

Example: B-2 Results for UDF BITAND

```
SQL0347W The recursive common table expression "DB2ADMIN.REPEAT"
may contain an infinite loop.  SQLSTATE=01605
```

```
-----
values bitand(10,8);
-----
```

```
1
```

```
-----
      8
```

```
-----
values bitand(14,3);
-----
```

```

1
-----
      2

-----
values bitand(1038,78);
-----

1
-----
      14

```

B.2 Sample code for FORMAT function

This section provides the UDF for FORMAT function. Example B-3 shows the code for a user-defined function emulating FORMAT.

Example: B-3 FORMAT user-defined function

```

--
-- DB2 UDF(User-Defined Function) Samples for conversion
--
-- Created: 2004/02/29
--
-- Name of UDF: FORMAT (X Decimal(31,10), D Integer)
--
-- Used UDF: None
--
-- Description: Returns truncated to the precision specified by D and a "," for each 3 digits as a
-- separator.
--
-- Author: TOKUNAGA, Takashi
--

----- Command Entered -----
CREATE FUNCTION FORMAT (X Decimal(31,10), D Integer)
RETURNS VARCHAR(50)
LANGUAGE SQL
SPECIFIC FORMAT_MySQL
CONTAINS SQL
NO EXTERNAL ACTION
DETERMINISTIC
BEGIN ATOMIC
DECLARE XN    DECIMAL(21,0);
DECLARE RetVal VARCHAR(50);
SET RetVal = SUBSTR(CHAR(MOD(ABS(X), 1)), 22, D+1);
SET XN = ABS(X);

Main_Loop:
WHILE XN > 0 DO
    SET RetVal = SUBSTR(CHAR(MOD(XN,1000)),19,3) || RetVal;
    SET XN = XN/1000;
    IF XN > 0 THEN
        SET RetVal = ',' || RetVal;
    ELSE
        LEAVE Main_Loop;
    END IF;
END WHILE;

RETURN CASE WHEN X < 0 THEN '-' ELSE '' END

```

```

|| TRANSLATE(LTRIM(TRANSLATE(RetVal, ' ', '0')), '0', ' ');
END
!

```

Example B-4 shows the results of the converted FORMAT.

Example: B-4 Converted FORMAT UDF result

```

----- Command Entered -----
SELECT N
      , FORMAT(N, 2)
      , FORMAT(N, 0)
FROM (VALUES (12.34567, -12.34567, 120034.567, 123400123456789.) S(N))
-----
--Return result

```

N	2	3
12.34567	12.34	12.
-12.34567	-12.34	-12.
120034.56700	120,034.56	120,034.
23400123456789.00000	123,400,123,456,789.00	123,400,123,456,789.

4 record(s) selected.

B.3 Sample code for RPAD and LPAD functions

This section provides the UDFs for the LPAD and RPAD functions. Example B-5 shows code for a user-defined function emulating RPAD.

Example: B-5 CREATE FUNCTION RPAD and sample usage

```

-- DB2 UDF(User-Defined Function) Samples for conversion
--
-- 2001/08/27, 09/27, 11/06
--
-- Name of UDFs: RPAD (C1 VarChar(4000), N integer, C2 VarChar(4000))
--              RPAD (I1 Integer, N integer, C2 Varchar(4000))
--              LPAD (C1 VarChar(4000), N integer, C2 Varchar(4000))
--              LPAD (I1 Integer, N integer, C2 Varchar(4000))
--
-- Used UDF: None
--
-- Description: Add repeatedly C2 to the right(RPAD) or left(LPAD) of parameter 1 (C1 or I1)
--              and return N byte.
--
-- Author: TOKUNAGA, Takashi
--
-----
CREATE FUNCTION RPAD (C1 VarChar(4000), N integer, C2 VarChar(4000))
RETURNS VARCHAR(4000)
LANGUAGE SQL
SPECIFIC RPADBase
DETERMINISTIC
CONTAINS SQL
NO EXTERNAL ACTION
RETURN

```

```

substr(C1 ||
repeat(C2, ((sign(N-length(C1)+1)/2)*(N-length(C1)+length(C2))/(length(C2)+1-sign(length(C2))))),1,N)
;

```

Example B-6 shows the results of the converted RPAD function.

Example: B-6 Usage of UDF RPAD

```

SELECT char(rpad('ABCDE',12,'*'),20) FROM SYSIBM.SYSDUMMY1;

```

```

-----
1
-----
ABCDE*.*.*

```

1 record(s) selected.

```

SELECT char(rpad('ABCDE',3,'*'),20) FROM SYSIBM.SYSDUMMY1;

```

```

-----
1
-----
ABC

```

1 record(s) selected.

```

SELECT char(rpad('ABCDE',20,'') || 'X',50) FROM SYSIBM.SYSDUMMY1;

```

```

-----
1
-----
ABCDE X

```

1 record(s) selected.

UDF RPAD with the third parameter omitted is shown in Example B-7.

Example: B-7 RPAD omitting the third parameter

```

CREATE FUNCTION RPAD (C1 VarChar(4000), N integer)
RETURNS VARCHAR(4000)
LANGUAGE SQL
SPECIFIC RPADVarCharParm2
DETERMINISTIC
CONTAINS SQL
NO EXTERNAL ACTION
RETURN
RPAD(C1,N,' ');
;

```

Running the RPAD function gives you the results that are shown in Example B-8.

Example: B-8 Results of RPAD omitting the third parameter

```

SELECT char(rpad('ABCDE',15) || 'X',50) FROM SYSIBM.SYSDUMMY1;

```

```

-----
1
-----
ABCDE X

```

1 record(s) selected.

```
-----  
SELECT char(rpad('ABCDE',3) || 'X',50) FROM SYSIBM.SYSDUMMY1;  
-----
```

1

ABCX

1 record(s) selected.

Function RPAD allows a set of different input arguments. Example B-9 shows two more RPAD UDFs.

Example: B-9 RPAD with first parameter as integer, 2, and 3 parameters

```
CREATE FUNCTION RPAD (I1 Integer, N integer, C2 Varchar(4000))  
  RETURNS VARCHAR(4000)  
  LANGUAGE SQL  
  SPECIFIC RPADIntParm3  
  DETERMINISTIC  
  CONTAINS SQL  
  NO EXTERNAL ACTION  
  RETURN  
  RPAD(rtrim(char(I1)),N,C2)  
;
```

```
CREATE FUNCTION RPAD (I1 Integer, N integer)  
  RETURNS VARCHAR(4000)  
  LANGUAGE SQL  
  SPECIFIC RPADIntParm2  
  DETERMINISTIC  
  CONTAINS SQL  
  NO EXTERNAL ACTION  
  RETURN  
  RPAD(rtrim(char(I1)),N,' ')  
;
```

And, Example B-10 shows the results of the previous UDFs.

Example: B-10 Results of RPAD with first parameter as integer, 2, and 3 parameters

```
SELECT char(rpad(927,12,'*'),50) FROM SYSIBM.SYSDUMMY1;
```

1

927*.*.*.*

1 record(s) selected.

```
-----  
SELECT char(rpad(927,12,' ') || 'X',50) FROM SYSIBM.SYSDUMMY1;  
-----
```

```

1
-----
927      X
        1 record(s) selected.

```

```

-----
SELECT char(rpad(9021,3),20) FROM SYSIBM.SYSDUMMY1;
-----

```

```

1
-----
902
        1 record(s) selected.

```

The counterpart for RPAD are the LPAD functions, which are shown in Example B-11.

Example: B-11 LPAD: CREATE FUNCTION and sample usage

```

CREATE FUNCTION LPAD (C1 VarChar(4000), N integer, C2 VarChar(4000))
  RETURNS VARCHAR(4000)
  LANGUAGE SQL
  SPECIFIC LPADBase
  DETERMINISTIC
  CONTAINS SQL
  NO EXTERNAL ACTION
  RETURN
  CASE
    WHEN N > length(C1) THEN
      substr(repeat(C2, (N-length(C1)+length(C2))/(length(C2)+1-sign(length(C2))))),1,N-length(C1)) || C1
    ELSE substr(C1,1,N)
  END
;

```

The results of LPAD look like Example B-12.

Example: B-12 Results of LPAD: CREATE FUNCTION and sample usage

```

SELECT char(lpad('ABCDE',15,'*'),50) FROM SYSIBM.SYSDUMMY1;
-----
1
-----
*.*.*.*.*ABCDE
        1 record(s) selected.

```

```

-----
SELECT char(lpad('ABCDE',3,'*'),50) FROM SYSIBM.SYSDUMMY1;
-----

```

```

1
-----
ABC

```

1 record(s) selected.

```
-----  
SELECT char(lpad('ABCDE',15,' ') || 'X',50) FROM SYSIBM.SYSDUMMY1;  
-----
```

```
1  
-----  
          ABCDEX
```

1 record(s) selected.

Because RPAD allows LPAD a different number and data type for input arguments, Example B-13 shows LPAD without the third parameter.

Example: B-13 LPAD: Omitting the third parameter

```
CREATE FUNCTION LPAD (C1 VarChar(4000), N integer)  
  RETURNS VARCHAR(4000)  
  LANGUAGE SQL  
  SPECIFIC LPADParm2  
  DETERMINISTIC  
  CONTAINS SQL  
  NO EXTERNAL ACTION  
  RETURN  
  LPAD(C1,N,' ' )  
;
```

The results of Example B-13 must look like those results in Example B-14.

Example: B-14 Result of LPAD: Omitting the third parameter

```
SELECT char(lpad('ABCDE',15),20) FROM SYSIBM.SYSDUMMY1;  
-----
```

```
1  
-----  
          ABCDE
```

1 record(s) selected.

```
-----  
SELECT char(lpad('ABCDE',3),20) FROM SYSIBM.SYSDUMMY1;  
-----
```

```
1  
-----  
          ABC
```

1 record(s) selected.

Two more LPAD UDFs with different characteristics are shown in Example B-15.

Example: B-15 LPAD: The first parameter is integer, 2, and 3 parameters

```
CREATE FUNCTION LPAD (I1 Integer, N integer, C2 Varchar(4000))
```

```

RETURNS VARCHAR(4000)
LANGUAGE SQL
SPECIFIC LPADIntParm3
DETERMINISTIC
CONTAINS SQL
NO EXTERNAL ACTION
RETURN
LPAD(rtrim(char(I1)),N,C2)
;

```

```

-----
CREATE FUNCTION LPAD (I1 Integer, N integer)
RETURNS VARCHAR(4000)
LANGUAGE SQL
SPECIFIC LPADIntParm2
DETERMINISTIC
CONTAINS SQL
NO EXTERNAL ACTION
RETURN
LPAD(rtrim(char(I1)),N,' ')
;

```

The results are shown in Example B-16.

Example: B-16 Results of LPAD: The first parameter is integer, 2, and 3 parameter

```

-----
SELECT char(lpad(9021,15,'*.'),50) FROM SYSIBM.SYSDUMMY1;

```

```

-----
1
-----
*.*.*.*.*9021
1 record(s) selected.

```

```

-----
SELECT char(lpad(9021,15,''),50) FROM SYSIBM.SYSDUMMY1;

```

```

-----
1
-----
9021
1 record(s) selected.

```

```

-----
SELECT char(lpad(9021,3),20) FROM SYSIBM.SYSDUMMY1;

```

```

-----
1
-----
902
1 record(s) selected.
-----

```

B.4 Sample code for GREATEST function

Example B-17 is a set of UDF examples emulating the behavior of the MySQL GREATEST function. The various UDFs accept input parameters in *varchar* and from 2 - 10 input parameters.

Example: B-17 User-defined functions to map GREATEST

```
--
-- DB2 UDF(User-Defined Function) Samples for conversion
--
-- 2001/08/28, 08/29
-- Name of UDF: GREATEST (P1 VarChar(254), P2 VarChar(254), ...)
--
--
-- Used UDF: None
--
-- Description: Returns greatest value of list of data.
--
-- Author: TOKUNAGA, Takashi
--
-----
CREATE FUNCTION GREATEST (P1 VarChar(254), P2 VarChar(254))
  RETURNS VarChar(254)
  LANGUAGE SQL
  SPECIFIC GREATEST0rac1e2
  DETERMINISTIC
  CONTAINS SQL
  NO EXTERNAL ACTION
  RETURN
CASE
  WHEN P1 >= P2 THEN P1
  ELSE P2
END
;
-----
--
-- GREATEST function with three parameters
--
-----
CREATE FUNCTION GREATEST (P1 VarChar(254), P2 VarChar(254), P3 VarChar(254))
  RETURNS VarChar(254)
  LANGUAGE SQL
  SPECIFIC GREATEST0rac1e3
  DETERMINISTIC
  CONTAINS SQL
  NO EXTERNAL ACTION
  RETURN
CASE
  WHEN P1 >= P2
  THEN CASE
    WHEN P1 >= P3 THEN P1
    ELSE P3
  END
  ELSE CASE
    WHEN P2 >= P3 THEN P2
    ELSE P3
  END
END
;
-----
--
-- GREATEST function with four parameters
```

```

--
-----
CREATE FUNCTION GREATEST (P1 VarChar(254), P2 VarChar(254), P3 VarChar(254), P4 VarChar(254))
RETURNS VarChar(254)
LANGUAGE SQL
SPECIFIC GREATESTOracle4
DETERMINISTIC
CONTAINS SQL
NO EXTERNAL ACTION
RETURN
CASE
WHEN P1 >= P2
THEN CASE
WHEN P1 >= P3
THEN CASE
WHEN P1 >= P4 THEN P1
ELSE P4
END
ELSE CASE
WHEN P3 >= P4 THEN P3
ELSE P4
END
END
ELSE CASE
WHEN P2 >= P3
THEN CASE
WHEN P2 >= P4 THEN P2
ELSE P4
END
ELSE CASE
WHEN P3 >= P4 THEN P3
ELSE P4
END
END
END
END
;
-----
--
-- GREATEST function with five parameters
--
-----
CREATE FUNCTION GREATEST (P1 VarChar(254), P2 VarChar(254), P3 VarChar(254), P4 VarChar(254), P5
VarChar(254))
RETURNS VarChar(254)
LANGUAGE SQL
SPECIFIC GREATESTOracle5
DETERMINISTIC
CONTAINS SQL
NO EXTERNAL ACTION
RETURN
CASE
WHEN P1 >= P2
THEN CASE
WHEN P1 >= P3
THEN CASE
WHEN P1 >= P4
THEN CASE
WHEN P1 >= P5 THEN P1
ELSE P5
END
ELSE CASE
WHEN P4 >= P5 THEN P4
ELSE P5
END
END
ELSE CASE
WHEN P3 >= P4

```

```

        THEN CASE
            WHEN P3 >= P5 THEN P3
            ELSE P5
            END
        ELSE CASE
            WHEN P4 >= P5 THEN P4
            ELSE P5
            END
        END
    END
ELSE CASE
    WHEN P2 >= P3
    THEN CASE
        WHEN P2 >= P4
        THEN CASE
            WHEN P2 >= P5 THEN P2
            ELSE P5
            END
        ELSE CASE
            WHEN P4 >= P5 THEN P4
            ELSE P5
            END
        END
    ELSE CASE
        WHEN P3 >= P4
        THEN CASE
            WHEN P3 >= P5 THEN P3
            ELSE P5
            END
        ELSE CASE
            WHEN P4 >= P5 THEN P4
            ELSE P5
            END
        END
    END
END
END
;
-----
-- GREATEST function with six parameters
--
-----
CREATE FUNCTION GREATEST (P1 VarChar(254), P2 VarChar(254), P3 VarChar(254), P4 VarChar(254)
, P5 VarChar(254), P6 VarChar(254))
RETURNS VarChar(254)
LANGUAGE SQL
SPECIFIC GREATESTOracle6
DETERMINISTIC
CONTAINS SQL
NO EXTERNAL ACTION
RETURN
GREATEST(GREATEST(P1,P2,P3),GREATEST(P4,P5,P6))
;
-----
-- GREATEST function with seven parameters
--
-----
CREATE FUNCTION GREATEST (P1 VarChar(254), P2 VarChar(254), P3 VarChar(254), P4 VarChar(254)
, P5 VarChar(254), P6 VarChar(254), P7 VarChar(254))
RETURNS VarChar(254)
LANGUAGE SQL
SPECIFIC GREATESTOracle7
DETERMINISTIC
CONTAINS SQL
NO EXTERNAL ACTION
RETURN

```

```

GREATEST(GREATEST(P1,P2,P3,P4),GREATEST(P5,P6,P7))
;
-----
--
-- GREATEST function with eight parameters
--
-----
CREATE FUNCTION GREATEST (P1 VarChar(254), P2 VarChar(254), P3 VarChar(254), P4 VarChar(254)
, P5 VarChar(254), P6 VarChar(254), P7 VarChar(254), P8 VarChar(254))
RETURNS VarChar(254)
LANGUAGE SQL
SPECIFIC GREATESTOracle8
DETERMINISTIC
CONTAINS SQL
NO EXTERNAL ACTION
RETURN
GREATEST(GREATEST(P1,P2,P3,P4),GREATEST(P5,P6,P7,P8))
;
-----
--
-- GREATEST function with nine parameters
--
-----
CREATE FUNCTION GREATEST (P1 VarChar(254), P2 VarChar(254), P3 VarChar(254), P4 VarChar(254)
, P5 VarChar(254), P6 VarChar(254), P7 VarChar(254), P8 VarChar(254)
, P9 VarChar(254))
RETURNS VarChar(254)
LANGUAGE SQL
SPECIFIC GREATESTOracle9
DETERMINISTIC
CONTAINS SQL
NO EXTERNAL ACTION
RETURN
GREATEST(GREATEST(P1,P2,P3,P4,P5),GREATEST(P6,P7,P8,P9))
;
-----
--
-- GREATEST function with ten parameters
--
-----
CREATE FUNCTION GREATEST (P1 VarChar(254), P2 VarChar(254), P3 VarChar(254), P4 VarChar(254)
, P5 VarChar(254), P6 VarChar(254), P7 VarChar(254), P8 VarChar(254)
, P9 VarChar(254), P10 VarChar(254))
RETURNS VarChar(254)
LANGUAGE SQL
SPECIFIC GREATESTOracle10
DETERMINISTIC
CONTAINS SQL
NO EXTERNAL ACTION
RETURN
GREATEST(GREATEST(P1,P2,P3,P4,P5),GREATEST(P6,P7,P8,P9,P10))
;
-----

```

Example B-18 shows the results of UDFs GREATEST.

Example: B-18 Result of UDFs mapping GREATEST

```

SELECT char(greatest('abcdefg','abcfgh'),20) FROM sysibm.sysdummy1;
-----
1
-----
abcfgh

```

1 record(s) selected.

```
-----  
SELECT char(greatest('abcdefg','defgh','abcfgh'),20) FROM sysibm.sysdummy1;  
-----
```

1

defgh

1 record(s) selected.

```
-----  
SELECT char(greatest('abcdefg','defgh','abcfgh','endof...'),20) FROM sysibm.sysdummy1;  
-----
```

1

endof...

1 record(s) selected.

```
-----  
SELECT char(greatest('abcdefg','defgh','abcfgh','endof...','add on'),20) FROM sysibm.sysdummy1;  
-----
```

1

endof...

1 record(s) selected.

```
-----  
SELECT char(greatest('abcdefg','defgh','abcfgh','endof...','add on','extra'),20) FROM  
sysibm.sysdummy1;  
-----
```

1

extra

1 record(s) selected.

```
-----  
SELECT char(greatest('abcdefg','defgh','abcfgh','endof...','add on','extra','a bit of'),20) FROM  
sysibm.sysdummy1;  
-----
```

1

extra

1 record(s) selected.

```
-----  
SELECT char(greatest('abcdefg','defgh','abcfgh','endof...','add on','extra','a bit of','more'),20)  
FROM sysibm.sysdummy1;  
-----
```

1

more

1 record(s) selected.

```

SELECT char(greatest('abcdefg','defgh','abcfgh','endof...','add on','extra','a bit of','more','more
and '),20) FROM sysibm.sysdummy1;
-----
1
-----
more and

      1 record(s) selected.

-----
SELECT char(greatest('abcdefg','defgh','abcfgh','endof...','add on','extra','a bit of','more','more
and ',' something'),20) FROM sysibm.sysdummy1;
-----
1
-----
more and

      1 record(s) selected.

```

B.5 Sample code for LEAST

Example B-19 is a set of UDF examples emulating the behavior of the MySQL LEAST function. The various UDFs accept input parameters in *varchar* and from 2 - 10 input parameters.

Example: B-19 User-defined functions to map LEAST

```

-- DB2 UDF(User-Defined Function) Samples for conversion
--
-- 2001/08/28, 08/29
--
-- Name of UDF: LEAST (P1 VarChar(254), P2 VarChar(254))
--
-- Used UDF: None
--
-- Description: Returns least value of list of data.
--
-- Author: TOKUNAGA, Takashi
--
-----
CREATE FUNCTION LEAST (P1 VarChar(254), P2 VarChar(254))
RETURNS VarChar(254)
LANGUAGE SQL
SPECIFIC LEAST0racle2
DETERMINISTIC
CONTAINS SQL
NO EXTERNAL ACTION
RETURN
CASE
WHEN P1 <= P2 THEN P1
ELSE          P2
END
;
-----
--
-- LEAST function with three parameters
--

```

```

-----
CREATE FUNCTION LEAST (P1 VarChar(254), P2 VarChar(254), P3 VarChar(254))
RETURNS VarChar(254)
LANGUAGE SQL
SPECIFIC LEASTOracle3
DETERMINISTIC
CONTAINS SQL
NO EXTERNAL ACTION
RETURN
CASE
WHEN P1 <= P2
THEN CASE
    WHEN P1 <= P3 THEN P1
    ELSE P3
    END
ELSE CASE
    WHEN P2 <= P3 THEN P2
    ELSE P3
    END
END
;
-----
--
-- LEAST function with four parameters
--
-----
CREATE FUNCTION LEAST (P1 VarChar(254), P2 VarChar(254), P3 VarChar(254), P4 VarChar(254))
RETURNS VarChar(254)
LANGUAGE SQL
SPECIFIC LEASTOracle4
DETERMINISTIC
CONTAINS SQL
NO EXTERNAL ACTION
RETURN
CASE
WHEN P1 <= P2
THEN CASE
    WHEN P1 <= P3
    THEN CASE
        WHEN P1 <= P4 THEN P1
        ELSE P4
        END
    ELSE CASE
        WHEN P3 <= P4 THEN P3
        ELSE P4
        END
    END
ELSE CASE
    WHEN P2 <= P3
    THEN CASE
        WHEN P2 <= P4 THEN P2
        ELSE P4
        END
    ELSE CASE
        WHEN P3 <= P4 THEN P3
        ELSE P4
        END
    END
END
;
-----
--
-- LEAST function with five parameters
--
-----
CREATE FUNCTION LEAST (P1 VarChar(254), P2 VarChar(254), P3 VarChar(254), P4 VarChar(254)
, P5 VarChar(254))

```

```

RETURNS VarChar(254)
LANGUAGE SQL
SPECIFIC LEASTOracle5
DETERMINISTIC
CONTAINS SQL
NO EXTERNAL ACTION
RETURN
CASE
WHEN P1 <= P2
THEN CASE
  WHEN P1 <= P3
  THEN CASE
    WHEN P1 <= P4
    THEN CASE
      WHEN P1 <= P5 THEN P1
      ELSE P5
      END
    ELSE CASE
      WHEN P4 <= P5 THEN P4
      ELSE P5
      END
    END
  ELSE CASE
    WHEN P3 <= P4
    THEN CASE
      WHEN P3 <= P5 THEN P3
      ELSE P5
      END
    ELSE CASE
      WHEN P4 <= P5 THEN P4
      ELSE P5
      END
    END
  END
ELSE CASE
  WHEN P2 <= P3
  THEN CASE
    WHEN P2 <= P4
    THEN CASE
      WHEN P2 <= P5 THEN P2
      ELSE P5
      END
    ELSE CASE
      WHEN P4 <= P5 THEN P4
      ELSE P5
      END
    END
  ELSE CASE
    WHEN P3 <= P4
    THEN CASE
      WHEN P3 <= P5 THEN P3
      ELSE P5
      END
    ELSE CASE
      WHEN P4 <= P5 THEN P4
      ELSE P5
      END
    END
  END
END
END
;
-----
--
-- LEAST function with six parameters
--
-----
CREATE FUNCTION LEAST (P1 VarChar(254), P2 VarChar(254), P3 VarChar(254), P4 VarChar(254)

```

```

        , P5 VarChar(254), P6 VarChar(254))
    RETURNS VarChar(254)
    LANGUAGE SQL
    SPECIFIC LEASTOracle6
    DETERMINISTIC
    CONTAINS SQL
    NO EXTERNAL ACTION
    RETURN
    LEAST(LEAST(P1,P2,P3),LEAST(P4,P5,P6))
;
-----
--
-- LEAST function with seven parameters
--
-----
CREATE FUNCTION LEAST (P1 VarChar(254), P2 VarChar(254), P3 VarChar(254), P4 VarChar(254)
    , P5 VarChar(254), P6 VarChar(254), P7 VarChar(254))
    RETURNS VarChar(254)
    LANGUAGE SQL
    SPECIFIC LEASTOracle7
    DETERMINISTIC
    CONTAINS SQL
    NO EXTERNAL ACTION
    RETURN
    LEAST(LEAST(P1,P2,P3,P4),LEAST(P5,P6,P7))
;
-----
--
-- LEAST function with eight parameters
--
-----
CREATE FUNCTION LEAST (P1 VarChar(254), P2 VarChar(254), P3 VarChar(254), P4 VarChar(254)
    , P5 VarChar(254), P6 VarChar(254), P7 VarChar(254), P8 VarChar(254))
    RETURNS VarChar(254)
    LANGUAGE SQL
    SPECIFIC LEASTOracle8
    DETERMINISTIC
    CONTAINS SQL
    NO EXTERNAL ACTION
    RETURN
    LEAST(LEAST(P1,P2,P3,P4),LEAST(P5,P6,P7,P8))
;
-----
--
-- LEAST function with nine parameters
--
-----
CREATE FUNCTION LEAST (P1 VarChar(254), P2 VarChar(254), P3 VarChar(254), P4 VarChar(254)
    , P5 VarChar(254), P6 VarChar(254), P7 VarChar(254), P8 VarChar(254)
    , P9 VarChar(254))
    RETURNS VarChar(254)
    LANGUAGE SQL
    SPECIFIC LEASTOracle9
    DETERMINISTIC
    CONTAINS SQL
    NO EXTERNAL ACTION
    RETURN
    LEAST(LEAST(P1,P2,P3,P4,P5),LEAST(P6,P7,P8,P9))
;
-----
--
-- LEAST function with ten parameters
--
-----
CREATE FUNCTION LEAST (P1 VarChar(254), P2 VarChar(254), P3 VarChar(254), P4 VarChar(254)
    , P5 VarChar(254), P6 VarChar(254), P7 VarChar(254), P8 VarChar(254)
    , P9 VarChar(254),P10 VarChar(254))

```

```

RETURNS VarChar(254)
LANGUAGE SQL
SPECIFIC LEASTOracle10
DETERMINISTIC
CONTAINS SQL
NO EXTERNAL ACTION
RETURN
LEAST(LEAST(P1,P2,P3,P4,P5), LEAST(P6,P7,P8,P9,P10))
;

```

Example B-20 shows the results of UDF LEAST.

Example: B-20 Results of UDFs mapping LEAST

```

SELECT least('HARRY','HARRIOT','HAROLD') FROM sysibm.sysdummy1;
-----
1
-----
HAROLD

1 record(s) selected

```

B.6 Sample code for BIT_COUNT

Example B-21 is a UDF example emulating the behavior of MySQL's BIT_COUNT function. It returns the number of set bits in the parameter (the number of 1 in the binary value of the parameter) assuming that the parameter is a 32-bit INTEGER.

Example: B-21 User-defined function to map BIT_COUNT

```

CREATE FUNCTION BIT_CNT (N1 Integer)
RETURNS Integer
SPECIFIC BITCNTMySQL
LANGUAGE SQL
CONTAINS SQL
NO EXTERNAL ACTION
DETERMINISTIC
RETURN
WITH
Repeat (S, M1, Ans) AS
(Values (0, N1, 0)
Union All
Select S+1, M1/2, Ans+MOD(M1,2)
From Repeat
Where M1 <> 0
AND S < 32
)
SELECT case when ANS > 0 then ANS else 32 + ANS end
FROM Repeat
WHERE S = (SELECT MAX(S)
FROM Repeat)
;

```

Example B-22 shows the sample output.

Example: B-22 Sample output of BIT_CNT

```
db2> values bit_cnt(64)
```

```
1
-----
1
```

```
1 record(s) selected.
```

```
db2> values bit_cnt(63)
```

```
1
-----
6
```

```
1 record(s) selected.
```

```
db2> values bit_cnt(-7)
```

```
1
-----
29
```

```
1 record(s) selected.
```

B.7 Sample code for SUBSTRING_INDEX

Example B-23 is a UDF example emulating the behavior of MySQL's SUBSTRING_INDEX function. It returns the substring from the input string before counting occurrences of the delimiter.

Example: B-23 User-defined function to map SUBSTRING_INDEX

```
create function SUBSTRING_INDEX(In varchar(2000),delimiter varchar(200), n Int)
returns varchar(2000)
deterministic no external action contains sql
begin atomic
  declare out varchar(2000);
  declare dem varchar(2000);
  declare num int;
  declare pos int;
  declare temp varchar(2000);
  set dem=delimiter;
  set temp=In;
  set num=n;
  set pos=1;
  if(num<0) then
    while(locate(delimit,temp)!=0) do
      set temp=substr(temp,locate(delimit,temp)+1);
      set num=num+1;
    end while;
    set num=num+1;
    set temp=In;
  end if;
  while (num>0) do
    set pos=pos+locate(delimit,temp)-1;
```

```
        set temp=substr(temp,locate(delimit,temp)+1);
        set num=num-1;
    end while;
    if(n>0) then
        return substr(In,1,pos);
    else
        return substr(In,pos+1);
    end if;
end
```

B.8 Sample code for UNIX_TIMESTAMP

Example B-24 shows UDF examples emulating the behavior of MySQL's UNIX_TIMESTAMP function.

Example: B-24 UNIX_TIMESTAMP

```
CREATE FUNCTION UNIX_TIMESTAMP ()
RETURNS INTEGER
RETURN
SELECT TIMESTAMPDIFF(2,CHAR(CURRENT_TIMESTAMP - TIMESTAMP('1970-01-01-00.00.000000')))
FROM SYSIBM.SYSDUMMY1;
```

```
CREATE FUNCTION UNIX_TIMESTAMP (ts VARCHAR(255))
RETURNS INTEGER
RETURN
SELECT TIMESTAMPDIFF(2,CHAR(CAST(ts AS TIMESTAMP) - TIMESTAMP('1970-01-01-00.00.000000')))
FROM SYSIBM.SYSDUMMY1;
```

Archived

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics that are covered in this book.

IBM Redbooks publications

For information about ordering these publications, see “How to get IBM Redbooks publications” on page 439. Note that several of the documents referenced here might be available in softcopy only.

- ▶ *Developing PHP Applications for IBM Data Servers*, SG24-7218
- ▶ *MySQL to DB2 UDB Conversion Guide*, SG24-7093
- ▶ *Up and Running with DB2 on Linux*, SG24-6899
- ▶ *Oracle to DB2 Conversion Guide for Linux, UNIX, and Windows*, SG24-7048

Other publications

These publications are also relevant as further information sources:

- ▶ *IEEE Standard for Software Test Documentation (829-1998)*, ISBN 0-7381-1444-8
- ▶ *Understanding DB2, Learning Visually with Examples, Second Edition*, ISBN-13:978-0-13-158018-3
- ▶ *Installing IBM Data Server Clients*, GC27-2454-00
- ▶ *Installing DB2 Servers*, GC27-2455-00
- ▶ *Getting Started with DB2 Installation and Administration on Linux and Windows*, G111-9411-00
- ▶ *Database Administration Concepts and Configuration Reference*, SC27-2442-00
- ▶ *Database Monitoring Guide and Reference*, SC27-2458-00
- ▶ *Database Security Guide*, SC27-2443-00
- ▶ *Partitioning and Clustering Guide*, SC27-2453-00
- ▶ *Troubleshooting and Tuning Database Performance*, SC27-2461-00

- ▶ *pureXML Guide*, SC27-2465-00
- ▶ *Data Movement Utilities Guide and Reference*, SC27-2440-00
- ▶ *Data Recovery and High Availability Guide and Reference*, SC27-2441-00
- ▶ *Workload Manager Guide and Reference*, SC27-2464-00
- ▶ *Getting Started with Database Application Development*, GI11-9410-00
- ▶ *Developing ADO.NET and OLE DB Applications*, SC27-2444-00
- ▶ *Developing Embedded SQL Applications*, SC27-2445-00
- ▶ *Developing Java Applications*, SC27-2446-00
- ▶ *Developing Perl, PHP, Python, and Ruby on Rails Applications*, SC27-2447-00
- ▶ *SQL Procedural Languages: Application Enablement and Support*, SC27-2470-00
- ▶ *Administrative API Reference*, SC27-2435-00
- ▶ *Administrative SQL Routines and Views*, SC27-2436-00
- ▶ *Command Reference*, SC27-2439-00
- ▶ *Message Reference Vol.1*, SC27-2450-00
- ▶ *Message Reference Vol.2*, SC27-2451-00
- ▶ *SQL Reference Vol.1*, SC27-2456-00
- ▶ *SQL Reference Vol.2*, SC27-2457-00
- ▶ *XQuery Reference*, SC27-2466-00
- ▶ *Introduction to Replication and Event Publishing*, GC19-1028-02
- ▶ *SQL Replication Guide and Reference*, SC19-1030-02
- ▶ *MySQL 5.0 Certification Study Guide*, ISBN-0-672-32812-7

Online resources

These Web sites are also relevant as further information sources:

DB2

- ▶ Database Management
<http://www.ibm.com/software/data/management/>
- ▶ DB2
<http://www.ibm.com/software/data/db2/>

- ▶ DB2 Express-C
 - <http://www.ibm.com/software/data/db2/express/>
 - <http://www.ibm.com/developerworks/forums/forum.jspa?forumID=805>
- ▶ IBM DB2 Database for Linux, UNIX, and Windows Information Center
 - <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp>
- ▶ DB2 Application Development
 - <http://www.ibm.com/software/data/db2/ad/>
- ▶ DB2 Bootcamp Training
 - <http://www.ibm.com/developerworks/data/bootcamps>
- ▶ DB2 Linux Validation
 - <http://www.ibm.com/software/data/db2/linux/validate/>
- ▶ DB2 9.7 manuals
 - <http://www1.ibm.com/support/docview.wss?rs=71&uid=swg27015148>
- ▶ DB2 9.7 features and benefits
 - <http://www-01.ibm.com/software/data/db2/9/features.html>
- ▶ DB2 Migration Now
 - <http://www.ibm.com/db2/migration>
- ▶ IBM Data Movement Tool
 - <http://www.ibm.com/developerworks/data/library/techarticle/dm-0906datamovement/>
- ▶ DB2 Technical Support
 - http://www.ibm.com/software/data/db2/support/db2_9/
- ▶ Integrated Data Management
 - <http://www.ibm.com/software/data/optim/>
- ▶ IBM developerWorks
 - <http://www.ibm.com/developerworks/>
- ▶ IBM PartnerWorld
 - <http://www.ibm.com/partnerworld>
- ▶ Software Migration Project Office
 - <http://www.ibm.com/software/solutions/softwaremigration/>
- ▶ Leveraging MySQL skills to learn DB2 Express: DB2 versus MySQL administration and basic tasks
 - <http://www.ibm.com/developerworks/db2/library/techarticle/dm-0602tham2/>

- ▶ Leverage MySQL skills to learn DB2 Express: DB2 versus MySQL backup and recovery
<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0606tham/>
- ▶ Leverage MySQL skills to learn DB2 Express, Part 3: DB2 versus MySQL graphical user interface
<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0608tham/>
- ▶ Leverage MySQL skills to learn DB2 Express, Part 4: DB2 versus MySQL data movement
<http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0610tham/>
- ▶ Convert from MySQL or PostgreSQL to DB2 Express-C
<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0606khatri/>
- ▶ DB2 Basics: Fun with Dates and Times
<http://www.ibm.com/developerworks/data/library/techarticle/0211yip/0211yip3.html>

MySQL

- ▶ MySQL home page
<http://www.mysql.com/>
- ▶ MySQL 5.1 Reference Manual
<http://dev.mysql.com/doc/refman/5.1/en/index.html>
- ▶ PHP MyAdmin
http://www.phpmyadmin.net/home_page/index.php

Others

- ▶ VMware
<http://www.vmware.com/>
- ▶ SUSE Linux Enterprise
<http://www.novell.com/linux/>
- ▶ PHP
<http://www.php.net/>
- ▶ PHP PECL extension
<http://pecl.php.net/>
- ▶ PHP Manual - Database extensions
<http://ca2.php.net/manual/en/refs.database.php>

- ▶ APACHE
<http://www.apache.org/>
- ▶ Perl
<http://www.perl.org/>
- ▶ Comprehensive Perl Archive Network
<http://www.cpan.org>
- ▶ Ruby
<http://www.ruby-lang.org/en/>
- ▶ IBM and Ruby
<http://rubyforge.org/projects/rubyibm>
- ▶ MySQL and Ruby
<http://rubyforge.org/projects/mysql-ruby/>
http://tmtm.org/en/ruby/mysql/README_en.html
<http://www.tmtm.org/en/mysql/ruby/>
- ▶ Java.sql Package Documentation
<http://java.sun.com/j2se/1.4.2/docs/api/java/sql/package-summary.html>
- ▶ UnixODBC
<http://www.unixodbc.org/>

How to get IBM Redbooks publications

You can search for, view, or download IBM Redbooks publications, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy IBM Redbooks publications, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Archived

Index

Symbols

.FRM file 124
.MRG file 124
.MYD file 124
.MYI file 124

Numerics

2-tier 25
32-bit 4
64-bit 4

A

access control 178
access package 29
access path 17
access plan 28, 354
access right 167
accessctrl authority 184, 187
accesslist table 268
account detail 79
ActiveX Data Object 32
address space 10
administrative interface 37
administrative views 348
ADO 32
ADO.NET 32
aggregate function 221
all privileges privilege 188
alter privilege 188
alterin privilege 188
AMD workstation 4
ANSI mode 210
ANSI92 standard 217
applaccess table 268
applet 30
application architecture 56, 62
application assessment 62
application client 26
application data 68
application developer 2
application development 92
application environment 56

application flow 76
application function 56
application interface 37, 56
application porting 56
application profile 56
application server 26
application user account 177
archive storage engine 47
ASNCLP command 297
assignment 213
async i/o 14
audit statement 183
authentication 181
autocommit 274
automated backup 6
automatic storage 127
autonomic computing daemon 10

B

background process 10
backup compression 5
bash shell 147
batch application 25
big integer 118
binary data 175
binary large object 175
bind command 28
bind commands 277
bindadd privilege 187
binding 27
blackhole storage engine 47
BLOB 175
block device 14
buffer pool 15, 164
buffers 39
built-in data 116
built-in function 160
business intelligence 7, 57

C

C client library 38
C/C++ 62
cache 39

- cache size 127
- Cartesian product 208
- catalog 18
- catalog command 24
- catalog table space 128
- catalog view 13
- certification 60
- character data type 216
- check constraint 120
- check pending state 327
- client code 10
- client connection 91
- client layer 37
- client library 248
- client support 91
- client-server application 7
- comma separated value 47
- command line interface 66
- command line mode 150
- commercial license 36
- commit 129
- common client 3
- communication 88
- communication encryption 8
- communication protocol 92
- compact installation type 90
- compile time 27
- compressed format 47
- compressed row 16
- compressed table 46
- compression rate 58
- concurrency 71
- concurrent insert 129
- concurrent versions system 23
- configuration file 12, 108
- configuration parameter 127
- configure command 106
- configure script 106
- connect privilege 187
- connection pool 38
- connection protocol 37
- connection statement 223
- connection string 232
- console 144
- console mode 152
- container 14
- context switch 9
- control center 90
- control privilege 186, 188

- conversion task 143
- Copy index data 173
- core file 335
- crash recovery 38, 129, 291
- create procedure syntax 161
- create trigger statement 160
- create_external_routine privilege 187
- create_not_fenced_routine privilege 187
- createin privilege 188
- createtab privilege 187
- cross join 208
- CSV storage engine 47
- cursor stability 273
- custom installation type 90
- CVS 23
- CVS storage engine 130

D

- dasCRT 103
- dasupdt 101
- data compression 2
- data definition language 67
- data dictionary 41
- data extraction 144
- data file 41
- data manipulation language 71
- data movement utility 279
- data page 14
- data porting 56
- data recovery 279
- data recovery module 57
- data server 2, 91
- data type mapping 115
- data types 175
- data warehouse 58
- dataaccess authority 184, 187
- database 11
- database activity 91
- database administration 38, 103
- database administrator 2, 59
- database architecture 62
- database authority 184
- database backup 72
- database configuration 279
- database configuration parameters
 - database_memory 307
 - locklist 307
 - logarchmeth1 288

- logarchmeth2 288
- maxlocks 307
- pckcachesz 307
- sheapthres_shr 307
- sortheap 307
- trackmod 288
- database design 60
- database discovery 25
- database driver 223, 232
- database element 145
- database feature 2
- database interface 62
- database managed space 13
- database management system 11, 57
- database management utility 39
- database manager 29
- database manager configuration parameter 127
- database modeling tool 147
- database node 125
- database object 13, 18, 123
- database partition 12, 125
- database partition group 12
- database recovery 286
- database replication 279, 296
- database server 38
- database structure 145
- database system monitor 182
- database user account 177
- date and time data type
 - date 118
 - datetime 119
 - time 119
 - timestamp 119
 - year 119
- DB2 commands
 - b2admin 20
 - dasauto 20
 - dascrt 20
 - dasdrop 20
 - dasmigr 20
 - db2acd 10
 - db2ca 104, 283
 - db2cc 20
 - db2dart 20, 334
 - db2icrt 19, 103
 - db2idrop 19
 - db2iilist 19, 281
 - db2imigr 19
 - db2import 301
 - db2isetup 20
 - db2iupd 101
 - db2iupdt 19
 - db2level 20
 - db2load 304
 - db2look 20, 325
 - db2ls 334
 - db2pd 20, 335
 - db2rc 297
 - db2set 281
 - db2setup 20, 101
 - db2sqljcustomize 278
 - db2start 19
 - db2stop 19
 - DB2 dump file 335
 - DB2 privilege level
 - database I 187
 - row or column 188
 - table space 188
 - DB2 system controller 10
 - DB2 watch dog 10
 - db2comm 24
 - db2diag tool 334
 - db2diag.log 335
 - db2fmp 10
 - db2greg tool 334
 - db2isetup command 102
 - db2jcc4.jar 23, 30
 - db2level command 334
 - db2move utility 302
 - db2setup.err 102
 - db2setup.log 102
 - DB2Sqlca 263
 - db2support 335
 - db2sysc 10
 - db2system file 282
 - db2wdog 10
 - DBADM authority 183, 187
 - dbheap 365
 - DBI 33
 - DBI interface 222
 - DBMS 57
 - DDL 67, 122
 - deadlocks 274
 - decompress table 16
 - default password 77
 - delete privilege 188
 - delimited ASCII format file 298
 - design advisor 6

- desktop system 8
- destination server 86
- device name 14
- diagnostic level 282
- direct I/O 14
- directory name 14
- discover_db parameter 25
- discover_inst parameter 25
- disk space 90
- distinctrow keyword 207
- distributed platform 2
- DML 71
- double precision 118
- driver code 3
- dropin privilege 188
- dynamic query 187
- dynamic SQL 187
- dynamic sql statement 28
- dynamic table 46
- dynamic warehouse 7

E

- e-business 57
- EDU 10
- education opportunity 60
- EJB 240
- embedded analytical feature 7
- embedded SQL statement 27
- embedded system 3
- employee inventory 76
- encrypted password 168
- engine dispatchable unit 10
- engine infrastructure 10
- Enterprise JavaBean 240
- execute privilege 184
- explain authority 183, 187
- export mode 302

F

- failover 57
- federated storage engine 47, 130
- file name 14
- file system 90, 123
- fixed term license 5
- floating-point number 118
- foreign key 48, 162
- foreign key constraint 47
- foriegn key 142

- formatted page 173
- frm extension 43
- full table scan 17

G

- global level profile registry 280
- global levels 280
- global variable Privilege 184
- go to label 264
- grant command 179
- grant table buffer 39
- granting privilege 70
- graphical interface 38
- graphical tool 90

H

- hashed index 133
- health and fault monitor 6
- heap size 127
- heap storage engine 130
- heap table 46
- High availability 279
- high availability 4
- host information 25
- host language variable 27
- host name 39

I

- I/O bound workload 16
- ibm_db2 226
- implicit casting 213
- implicit privilege 186
- implicit_schema privilege 187
- import command 168
- import mode 302
- IMS 2
- incremental backup 288
- independent software vendor 5
- index 136
- index blocks 39
- Index privilege 184
- index privilege 188
- indexe 13
- InnoDB engine 162
- InnoDB table 124
- insert privilege 188
- installation method 92

- installation option 101
- installation program 90
- installation status 101
- installation type 90
- installFixPak command 101
- instance 11
- instance level profile registry 280
- instance node level profile registry 280
- instance owner 193
- instance owner information 98
- integrated exchange format 298
- integrity information 18
- interactive deployment mode 153
- interprocess communication 23
- inventory 77
- inventory location 82
- inventory type 82
- IP address 227
- IPC 23
- isolation level 71, 272, 276
- isolation option 277

J

- Java 62
- Java code 31
- Java enabled browser 30
- JavaServer Page 240
- join order 206
- JSP 240

K

- key buffer 39

L

- label based access control 2, 180
- language flag 102
- large object 13
- LBAC 180
- LBAC rule exemption privilege 188
- licensing model 36
- lightweight deployment solution 23
- lightweight security audit mechanism 58
- Linux distribution 88
- list command 141
- ln command 107
- load authority 184
- load mode 302

- load privilege 187
- local connection 23
- local tools catalog 100
- localhost 150
- lock escalation 274
- lock time-out 274
- lock wait 274
- locking 71
- locklist 164
- log file 101, 199
- log file path 127
- log files location 282
- log information 41
- logical model 68
- loopback 24

M

- main process 10
- mainframe 2
- maintenance window 164
- make command 107
- manageability 59
- management tool 59
- materialized query 133
- memory 91
- memory cache 39
- memory model 4
- memory storage engine 46
- memory table 133
- merge storage 46
- merge storage engine 130
- merge table 124
- method privileges 188
- migration assessment 56
- migration project 56
- migration task 56
- mincommit 365
- mobile device 8
- module 184
- module privilege 184
- mounting option 123
- MQT 134
- multi-byte character set 33
- multiple database 140
- multi-user version 5
- mv command 109
- MYD extension 43
- MYI extension 43

- MyISAM table 124
- myisamchk 51
- myisampack 51
- mysql 50
- MySQL Administrator 51
- MySQL index
 - fulltext 136
 - non-unique 136
 - primary key 136
 - spatial 136
 - unique 136
- MySQL Query Browser 51
- MySQL server program
 - comp_err 50
 - innochecksum 51
 - make_binary_distribution 50
 - make_win_bin_dist 50
 - msql2mysql 51
 - my_print_defaults 51
 - myisamlog 51
 - mysql.server 50
 - mysql_config 51
 - mysql_fix_privilege_tables 50
 - mysql_install_db 50
 - mysql_secure_installation 50
 - mysql_tzinfo_to_sql 50
 - mysql_upgrade 50
 - mysqlbug 50
 - mysqld 50
 - mysqld_multi 50
 - mysqld_safe 50
 - mysqld-debug 50
 - mysqld-max 50
 - mysqld-nt 50
 - mysqlmanager 50
 - perror 51
 - replace 51
- mysql.columns_priv table 179
- mysql.procs_priv table 179
- mysql.tables_priv table 179
- mysql.user table 177, 179
- mysql_convert_table_format 51
- mysql_fix_extensions 51
- mysql_setpermissions 51
- mysql_tableinfo 51
- mysql_waitpid 51
- mysql_zap 51
- mysqlaccess 51
- mysqladmin 50
- mysqlbinlog 51
- mysqlcheck 50
- mysqldump 40, 51, 145, 168
- mysqldump options
 - help 169
 - no-create-info 169–171
 - no-data 169
 - password 169
 - tab 170
 - tab= 169
 - user 169
- mysqlhotcopy 40, 51, 172
- mysqlhotcopy script 168
- mysqlimport 51
- mysqlshow 51

N

- named pipe 39
- national language support 71
- natural join 208
- nickname privilege 184
- node 25, 125
- non-recoverable database 287
- non-transaction-safe storage engine 45
- null value 217
- numeric data type 216
 - bigint 118
 - bit 117
 - bool 117
 - boolean 117
 - decimal 118
 - double 118
 - fixed 118
 - float 118
 - int 118
 - integer 118
 - numeric 118
 - real 118
 - smallint 117
 - tinyint 117
- numeric values 215

O

- object privileges 18
- object-oriented extension 107
- ODBC driver manager 29
- OLAP 7, 57
- OLE DB provider 32

- OLTP 57
- on-line analytical processing 57
- online memory tuning 164
- on-line transaction processing 57
- optimizer 39, 206

P

- package 65
- package cache 164
- package cache size 307
- package privilege 184, 188
- parallel database system 11
- parse tree 39
- Parser 38
- partition group 126
- partitioned database environment 125
- password 77
- payload file deployment 92
- PDO 226
- pdo_ibm 33
- performance 9
- Perl 62
- permission 186
- PHP 62
- PHP data object 226
- PHP source 106
- phpinfo() 107
- physical resource 39
- physical storage device 14
- physical structure 12
- plug-in 23
- policy 311
- port address 24
- port number 103
- Porting preparation 56
- precision 117
- precompile option 272
- precompiled package 107
- pre-compiling 27
- PREP 277
- pre-parsed tree 11
- primary key 162
- primary keys 65, 142
- privilege 13, 167, 186
- privilege level
 - column 179
 - database 179
 - global level 179

- routine 179
- table 179
- procedural interface 107
- process identifier 338
- profile 180
- profile registry 282
- profile registry variables 280
- pureXML, autonomics 2

Q

- query cache 39
- query interface 37
- query optimization class 206
- query performance 134
- query syntax 39
- query users 37
- quiesce_connect privilege 187

R

- range-clustered table 135
- raw device 14
- raw disk partitions 47
- RDO 32
- read stability 273
- rebuildconf command 110
- recover history file 287
- recoverable database 288
- recovery log 125
- recovery log file 287
- recovery logs file 286
- recovery method 286
- Redbooks Web site 439
 - Contact us xv
- reference type 398
- REFERENCES privilege 188
- Referential integrity 48
- referential integrity 162
- register variable
 - db2_compatibility_vector 219
- registered user 77
- registry variable 24, 219, 280
- relational modeling 2
- remote connection 23, 36
- remote data object 32
- repeatable read 273
- replication service 4
- reserved words 145
- resource tuning database technology 59

- response file 101
- response file installation 92
- restore utility 294
- result output file 144
- revoke command 179
- roll forward recovery 292
- rollback 129
- rollforward utility 295
- root privilege 180
- root table 135
- root type 398
- routine privilege 184
- routines 184
- row compression 131, 386
- row trigger 137
- runstats command 199
- runtime support 23

S

- schema 43, 128
- Schema level 187
- Schema privilege 184
- secadm authority 183, 187
- security 11, 13
- security label privilege 188
- security model 181
- security system 177
- select privilege 188
- self tuning memory manager 6
- self-tuning memory manager 59, 164–165
- sequence privilege 184, 188
- server code 10
- server privilege 184
- service request 77
- service ticket 76, 81
- service type 82
- servlets 240
- setup wizard 90
- severity level 84
- shared memory networking protocol 39
- shared-nothing architecture 48
- single byte integer 117
- single precision 118
- single-tier 25
- slave database 40
- slave system 40
- small integer 117
- sort heap 164

- sort heap size 307
- sort heap threshold 307
- source code language 62
- source distribution 41
- source table 203
- sources package 109
- special group 186
- SQL 2
- SQL interface 38
- SQL replication 5
- SQL statements 183
- sqladm authority 183, 187
- SQLCODE 259
- SQLj 240
- SQLj applet 31
- SQLj translator 31
- SQLSTATE 259
- standard interface 25
- statement trigger 137
- static sql statement 27
- static tables 46
- statistical data 41
- status 144
- STMM 6
- storage device 126
- storage engine 39–40, 129
- storage engines 45
- storage optimization 4
- stored procedure 22, 84
- straight_join keyword 206, 208
- string and character data type
 - binary 119
 - blob 120
 - char 119
 - enum 120
 - longblob 120
 - longtext 120
 - mediumblob 120
 - mediumtext 120
 - set 120
 - tinyblob 120
 - tinytext 120
 - varbinary 119
 - varchar 119
- string function 211
- strong typing 213
- structured query language 123
- structured type 398
- subquery 209

- subtable 135
- summary table 134
- supertable 135
- symbolic link 123
- synchronous event 14
- synonym 117, 207
- sysadm authority 181
- syscat catalog view 18
- sysctrl authority 181
- sysmaint authority 182
- sysmon authority 182
- sysstat catalog view 18
- system catalog 11
- system catalog table 13, 128
- system directory 11
- system managed space 13
- system monitor 182
- system planning 60
- system requirement 90

T

- table column 11
- table descriptor 39
- table hierarchy 135
- table partitioning 2
- table space change history file 287
- table space privilege 184
- table spaces 12
- table-level encryption 8
- tar file 105
- target system 56, 63
- TCP/IP communication 100
- temporary table 128
- temporary table space 128
- text 145
- threaded engine 2
- threaded model 9
- throttled utility 311
- throttling system 311
- timestamp 215
- trace file 101
- transaction isolation 271
- transaction processing 57
- transaction safe 129
- transaction-safe storage engines 45
- TRG extension 43
- trigger 65, 137
- TRN extension 43

- truncate statement 218
- two-tier architecture 63
- type 4 driver 30
- typed tables 135
- typical installation type 90

U

- uncommitted read 273
- underscore 178
- Unified ODBC 234
- unique key 142, 162
- unit of work 276
- unload script 173
- update privilege 188
- updateable view 175
- use privilege 188
- user account 77, 98
- user account information 39, 177
- user account management 167, 177
- User data 68
- user data 167
- user defined function 10
- user defined table 128
- user privilege 40
- user table space 128
- user-defined data type 116
- user-defined partitioning 124

V

- validation program 88
- value compression 386
- vector I/O 14
- version recovery 291
- view 13

W

- warehouse architect 59
- Web server 26
- Web service 22
- Web-browser 26
- whenever statement 263
- wildcard 178
- wlmadm authority 183, 187
- work sheet format 298
- workload 164
- workload management 2
- workload manager 279

workload privilege 184

X

XML support 71

XSR object privilege 184

Archived



Redbooks

MySQL to DB2 Conversion Guide

(1.0" spine)
0.875" <-> 1.498"
460 <-> 788 pages



MySQL to DB2 Conversion Guide



Guides you through a MySQL database and application conversion to DB2

Enriches applications through advanced DB2 features

Converts an application with detailed examples

Switching database vendors is often considered an exhausting challenge for database administrators and developers. Complexity, total cost, and the risk of downtime are often the reasons that restrain IT decision makers from starting the migration project. The primary goal of this book is to show that, with the proper planning and guidance, converting from MySQL to IBM DB2 is not only feasible but straightforward.

If you picked up this book, you are most likely considering converting to DB2 and are probably aware of several of the advantages of converting to DB2 data server. In this IBM Redbooks publication, we discuss in detail how you can take advantage of this industry leading database server.

This book is an informative guide that describes how to convert the database system from MySQL 5.1 to DB2 V9.7 on Linux and the steps that are involved in enabling the applications to use DB2 instead of MySQL.

This guide also presents the best practices in conversion strategy and planning, conversion tools, porting steps, and practical conversion examples. It is intended for technical staff that is involved in a MySQL to DB2 conversion project.

**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**