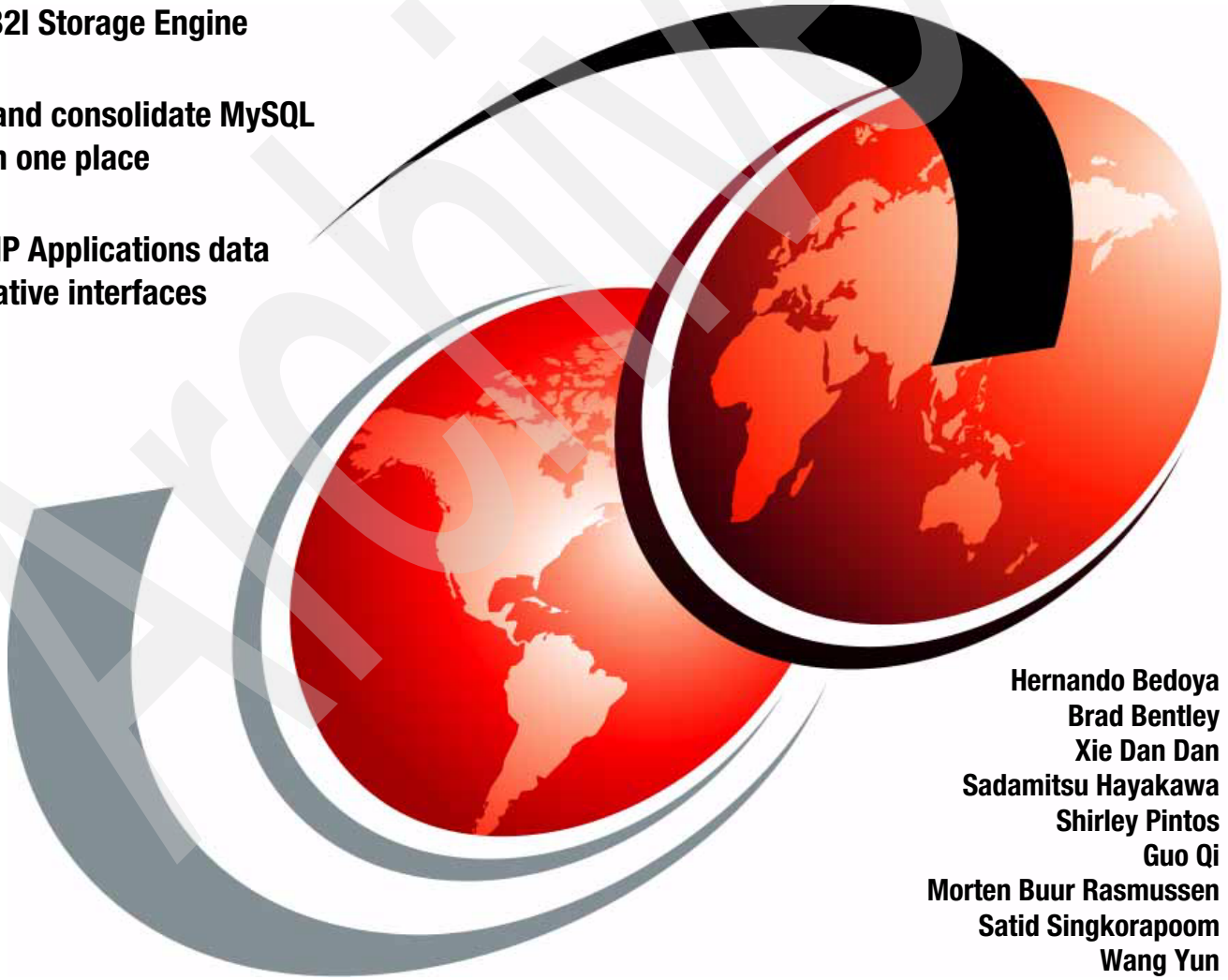
IBM

Using IBM DB2 for i as a Storage Engine of MySQL

- Discover how to configure and manage the IBMDB2I Storage Engine
- Integrate and consolidate MySQL and DB2 in one place
- Access PHP Applications data through native interfaces



Hernando Bedoya
Brad Bentley
Xie Dan Dan
Sadamitsu Hayakawa
Shirley Pintos
Guo Qi
Morten Buur Rasmussen
Satid Singkorapoom
Wang Yun

Redbooks

ibm.com/redbooks

International Technical Support Organization

Using IBM DB2 for i as a Storage Engine of MySQL

March 2009

SG24-7705-00

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

First Edition (March 2009)

This edition applies to IBM i 6.1

Contents

Notices

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which 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 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:

AIX®	Integrated Language Environment®	POWER®
AS/400®	iSeries®	Redbooks®
DB2®	Language Environment®	Redbooks (logo) ®
i5/OS®	OS/400®	System i®
IBM®	Power Systems™	WebSphere®

The following terms are trademarks of other companies:

SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

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

Excel, Internet Explorer, Microsoft, SQL Server, Windows NT, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

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

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

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

Preface

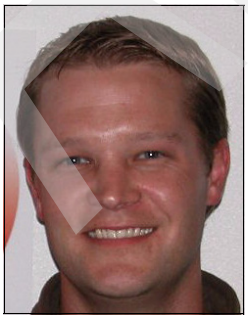
With the Apache, MySQL™, and PHP (AMP) stack, IBM® i has the open source middleware to run thousands of PHP applications and scripts that have been written to the MySQL database. MySQL is a database that is used on millions of Web sites. To support the wide variety of usage, the developers of MySQL has developed an open storage engine architecture for data functionality and storage. Over a dozen storage engines are available for MySQL. IBM and Sun™ Microsystems have worked together to deliver a DB2® for i Storage Engine for MySQL. With this support, PHP applications written to MySQL database can have the data stored in the DB2 for i database. This approach provides management benefits for the IBM i customer because DB2 is integrated into IBM i and customers already know how to manage, back up, and protect DB2 data. In addition, the DB2 for i Storage Engine provides access to the MySQL data from IBM i environments such as RPG, CL, and DB2 Web Query. The DB2 for i Storage Engine offers the management and data access integration that can make IBM i the preferred platform for running open source applications for IBM i customers.

This IBM Redbooks® publication provides broad information to help you understand this storage engine. The book also helps you install, tailor, and configure DB2 for i Storage Engine for MySQL support.

The team that wrote this book

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

Hernando Bedoya is a Senior IT Consultant in STG Lab Services in Rochester, Minnesota. He worked 8 years in ITSO, coordinating IBM Redbooks publications in the area of DB2 for i. He writes extensively and teaches IBM classes worldwide in all areas of DB2 for i. Before joining the ITSO more than seven years ago, he worked for IBM Colombia as an IBM AS/400® IT Specialist doing presales support for the Andean countries. He has 24 years of experience in the computing field and has taught database classes in Colombian universities. He holds a Master's degree in computer science from EAFIT, Colombia. His areas of expertise are database technology, application development, and high availability.



Brad Bentley is a Staff Software Engineer for STG Lab Services in Rochester, Minnesota. He started his career with IBM in January 2004, hiring in as an experienced professional with five years of I/ experience before joining IBM. Since 2006, he has been a principle player on the Lab Services Application Modernization team, focusing on SOA enablement of Legacy Applications. Before joining STG Lab Services, he was a Software Engineer developing and testing DB2 for i. He is a subject matter expert on application development and services enablement in DB2 for i, RPG/Cobol, Java™, Perl, PHP and MySQL on IBM i.

Xie Dan Dan is responsible for regression testing and Functional Verification Test (FVT) in DB2 for i. She is working on extension unit test and test design. She is IBM Certified DB2 Family Application Developer and IBM i Operator.

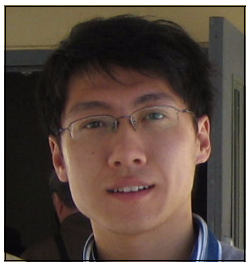
Sadamitsu Hayakawa is an Accredited Senior IT Specialist for Technical Support Service function under GTS in Japan. In this function, he works on skill transfer tasks and customer projects. Formerly, he was a Field IT Specialist and worked with customers on large scaled system building project of AS/400. He has 22 years of experience in the midrange server area, which includes System/38, System/36, AS/400, and now Power Systems™ with IBM i. His areas of expertise include design and tuning of DB2 for IBM i, high availability design and implementation, IBM i performance tuning, and IBM i administration design.

Shirley Pintos is a Software Engineer for IBM in Rochester, Minnesota. She began her career at IBM in December 2000. For the past seven years, she has been part of the Database Team in the Support Center. Her role includes providing second-level support to IBM i customers. Her areas of support include SQL, database, query performance, and journaling. She is the SQL subject matter expert on the database team and has taught new release education. She holds a Master's degree in Computer Science in Information Systems from the University of Phoenix.

Guo Qi is a Software Test Specialist. He leads a team that is responsible for testing related areas of DB2 for i. His current area of focus is Test Design and Project Management. He is an IBM Certified DB2 Family Application Developer and IBM i Operator. He has authored a book related to DB2 for i, and has one filed patent and two published patents.

Morten Buur Rasmussen is a senior IT Specialist at the IBM Systems and Technology Group Lab Services (formerly CTC), in La Gaude, France. He covers Europe and the Middle East in client-facing areas of IBM System i®, database, and WebSphere® performance. Before joining the Lab Services, he worked for IBM Denmark and several European banks. He has 21 years of experience in the computing field. His areas of expertise are database technology, application development, and work management for IBM System i.

Satid Singkorapoom is an IBM Certified Senior IT Specialist for Power Systems Technical Sales unit of IBM Thailand. He has 18 years of experience with Power Systems with IBM i since it was named AS/400. He holds a Master of Computer Engineering degree from the Asian Institute of Technology in Thailand. His areas of expertise include DB2 for i5/OS® technology, SQL and system performance analysis and tuning, logical partitioning design and configuration, SAP® on System i technical infrastructure, Power Systems hardware architecture, and general IBM i technology. He has coauthored six IBM Redbooks and three DB2 for i5/OS training materials from the ITSO Rochester over the past 15 years. He provides education to System i customers on product technology updates, SQL and system performance analysis and tuning, IBM i for SAP deployment, and System i administration.



Wang Yun is responsible for Functional Verification Test (FVT) and tooling in DB2 for i. He is working on test design and automation development. He is an IBM Certified DB2 Administrator, DB2 Family Application Developer and IBM i Operator. He has publishing experience in IBM world wide conference, developerworks and two patents.

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

Jim Cook
Thomas Gray
James Hansen
Joanna Pohl-Miszczyk
Diane Sherman
Scott Vetter
International Technical Support Organization, Rochester Center

Timothy Clark
Jeffrey Huebert
Craig Johnson
Kathryn Steinbrink
Eric Will
IBM Rochester Development

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, 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 in one of the following ways:

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

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

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

1

Overview

In this chapter, we provide an overview of the DB2 for i Storage Engine (IBMDB2I) for the MySQL 5.1 database server product that is running on the IBM i operating system. This support was available in late 2008 for IBM i 5.4 and 6.1 as an enhancement to the original MySQL support on i5/OS V5R4 in August 2007.

This chapter contains the following topics:

To get the most recent information about MySQL on IBM i and IBMDB2I, see the following Web site, which links to technotes and changes:

http://www.ibm.com/systems/i/software/mysql/index.html

1.1 MySQL on IBM i

MySQL is one of the most popular open source database management products in the market today. It has gained popularity in the Web application world and is used in most of the leading PHP-based applications.

In the world of open-source and Web-based applications, the LAMP stack is well-recognized and widely used. The LAMP acronym refers to Linux®, Apache, MySQL, and PHP. The LAMP technologies are claimed to be the most popular components used in the vast majority of Web applications.

The IBM i operating system for IBM Power Systems has been designed with flexibility in mind. This principle extends support to PHP as a viable choice for Web application development and deployment on IBM Power Systems or System i with IBM i V5R4 and later for i5/OS V5R3. Most PHP-based applications also run with MySQL as their popular database server components. So, it is natural that IBM i also extends its support for MySQL as another choice for the database management system apart from DB2 for i. MySQL combined with IBM i support for the open source Apache server helps enable simple and effective portability of LAMP stack to iAMP stack, with “i” being the IBM i operating system. See Figure 1-1.

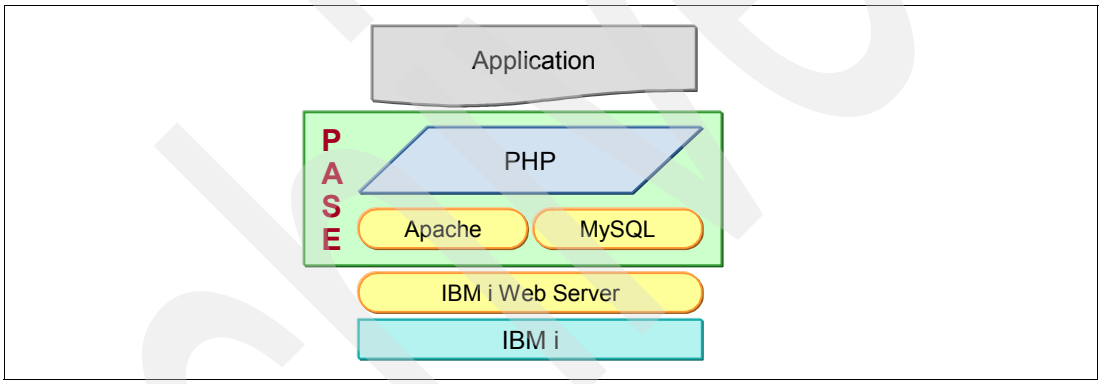


Figure 1-1 iAMP (IBM i, Apache, MySQL, PHP) Web Application Stack - PASE (Portable Application Solutions Environment)

The iAMP Web application stack allows for development and deployment of Web-based applications integrated with the MySQL database and is well suited for those customer environments that want to leverage existing open source applications that are based on PHP and MySQL.

The MySQL Enterprise database server product for IBM i was available for IBM i V5R4 in August 2007. This enabled System i customers to run many PHP-based Web applications on IBM i with minimal porting and modification effort. The initial supported version of MySQL Enterprise for IBM i was version 5.0.

With the initial availability of MySQL for IBM i, MySQL stored table and index data as stream files in the integrated file system (IFS) rather than as DB2 for i objects. The stream file data was stored in MySQL-specific formats, and thus it was difficult for applications other than MySQL to access the data.

For more information about how to use MySQL in IBM i or i5/OS, see:

- *Discovering MySQL in IBM i5/OS*, SG24-7398
- http://www.ibm.com/systems/i/software/mysql/index.html

1.2 MySQL pluggable storage engine

Within the MySQL architectural design, there is a unique component called *pluggable storage engine* (which used to be called table type) that provides many implementation types for table objects that are created in a MySQL database. When a MySQL table is created, it must be created with a specified storage engine type, otherwise MySQL assumes a default engine type for it. The default engine type can also be changed but, in order for it to take effect, it must have MySQL to restart. Different storage engines can be used for different tables that are created within the same MySQL database. See Figure 1-2.

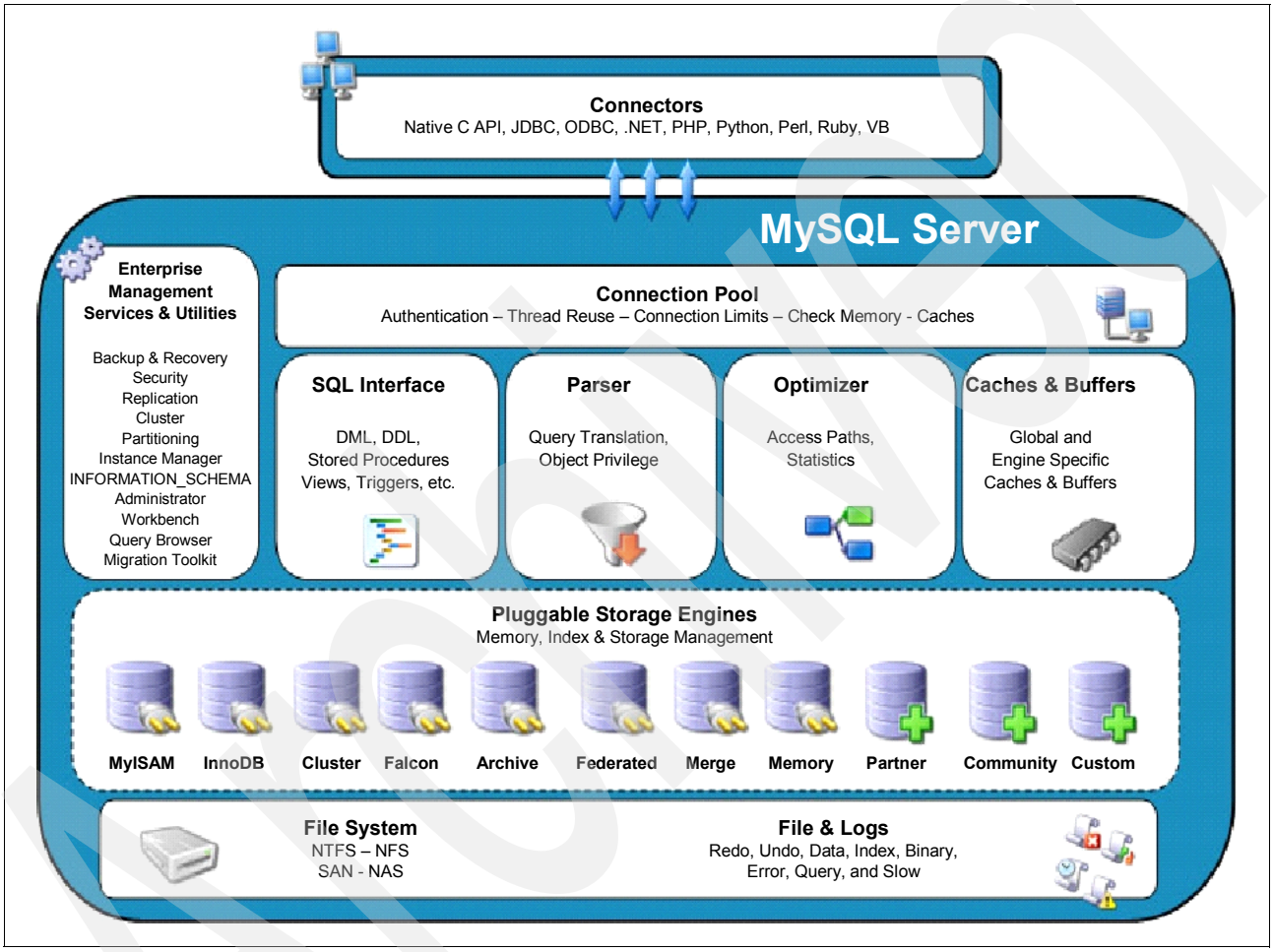


Figure 1-2 MySQL 5.0 Pluggable Storage Engine Architecture

Prior to the availability of the DB2 for i Storage Engine for MySQL (known as the IBMDB2I Storage Engine) in late 2008, MySQL Enterprise for i5/OS supported the following storage engines:

- The MyISAM engine manages nontransactional tables. It offers high-speed storage and retrieval as well as full text searching capabilities but is not crash-safe. MyISAM is supported in all MySQL configurations and it is the default storage engine for any newly created tables unless you explicitly specify a different engine to be used before you create a table.
- The InnoDB engine delivers transaction support with COMMIT and ROLLBACK with full ACID[1] compliance. It also provides auto-recovery after a crash, row-level locking with Multi-Versioning Concurrency Control (MVCC) and non-locking read.

- The MEMORY engine delivers in-memory data storage implementation which is generally beneficial to temporary tables. It provides very fast storage of temporary result sets but supports fixed-size rows only.
- The MERGE engine enables a collection of identical MyISAM tables to be handled as a single table. Like MyISAM, the MEMORY and MERGE engines handle nontransactional tables. Both are also included in MySQL by default.
- The EXAMPLE engine is a *stub* engine that does nothing. You can create tables with this engine, but no data can be stored in them or retrieved from them. This engine illustrates how to begin writing new storage engines. As such, it is of interest primarily to developers.
- The ARCHIVE engine stores large amounts of data, without indexes, with a very fast and efficient compressed storage implementation that can take considerably less disk storage space than MyISAM. Because it does not support indexes, table scan is the only supported operation for data retrieval. You use only non-blocking inserts to ARCHIVE tables.
- The CSV engine stores data in text files using comma-separated values format.

A table in MySQL database is implemented by an available storage engine to exploit a specific benefit that the storage engine is designed to deliver. For example, a temporary table that is used only during the run time of a procedure can be created with the MEMORY engine type, which places its data only in system memory rather than on the disk storage. This provides fast access time for the temporary table manipulation as opposed to relatively slower access time when it is stored on the disk. Because the temporary table does not require persistency for its data after the procedure execution reaches its end, creating it with the MEMORY engine delivers fast access benefit that a temporary table which is created on the disk cannot deliver.

The storage engine associated with a table can be switched at any time by using the ALTER TABLE statement with ENGINE specified. For example, the following statement moves the specified table from its existing storage implementation to the IBMDB2I Storage Engine:

```
ALTER TABLE myisamtable ENGINE=IBMDB2I;
```

This statement can simplify the process of moving existing MySQL Web application's data into another storage engine.

You can select a specific storage engine to be the default for new MySQL tables by setting the `default_storage_engine` configuration option. This can be done by modifying the MySQL startup option file (often located at `/etc/my.cnf`) or by using the command line parameter when starting the MySQL server. If this option is not explicitly set, then MyISAM is the default storage engine. For example, you may add the following line to your `my.cnf` file:

```
default_storage_engine = <a storage engine name>
```

After you restart MySQL, all tables created thereafter will be implemented with the specified storage engine. For more information, see the following resources:

- Chapter 2 of *Discovering MySQL in IBM i5/OS*, SG24-7398 discusses various MySQL storage engines in more detail.
- Information about MySQL Pluggable Storage Engine Architecture:

  http://solutions.mysql.com/engines.html
- The latest information MySQL on IBM i and the IBMDB2I Storage Engine:

  http://www.ibm.com/systems/i/software/mysql/index.html

[1] Atomicity, consistency, isolation, durability (ACID)

1.3 IBM DB2 for i Storage Engine for MySQL on IBM i

In the first quarter of 2009, an additional pluggable storage engine option specific to MySQL running in IBM i, V5R4 and V6R1, was introduced and given the name of IBMDB2I Storage Engine.

This IBMDB2I Storage Engine was delivered for MySQL version 5.1. It is a fully featured storage engine that supports persistent data store, table scanning, insert, update, delete, indexes, constraints and transactions, with certain occasional restrictions applied. It is primarily created to meet the requirements of a selected group of MySQL applications that are known to run on IBM i. See Figure 1-3.

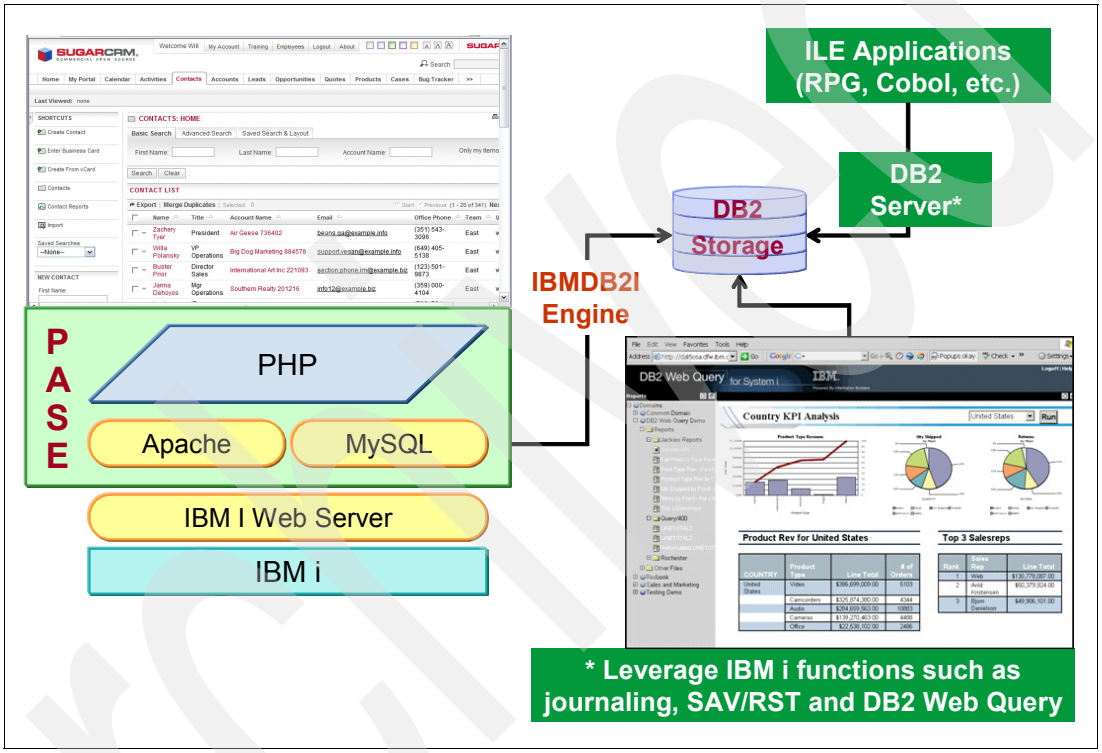


Figure 1-3 IBM i applications can access MySQL tables created with the IBMDB2I Storage Engine

With the support of IBMDB2I Storage Engine for MySQL on IBM i, applications that use MySQL as the database server can store their data in table objects, which are created in the QSYS file system as the native format of DB2 for i physical file objects. The contents of these MySQL tables can be manipulated from both MySQL and DB2 for i (both SQL and native I/O) perspectives. The latter approach carries some restrictions. See Table 2-1 on page 15 for examples of considerations.

Storing the data in DB2 for i more conveniently allows other IBM i applications to access and exchange data with MySQL tables in a simple manner through DB2 for i native I/O and SQL interfaces. See Figure 1-4 on page 6.

	MyISAM	InnoDB	IBMDB2I
Usage	Fastest for read heavy applications	Fully ACID compliant transactions	**Fully ACID compliant; data visible externally**
Locking	Large -grain table locks, no non - locking reads	Multi - versioning, row - level locking	**Row - level locking**
Durability	Table recovery	Durability recovery	**Durability recovery**
Supports Transactions	NO	YES	**YES**
Supports foreign keys	NO	YES	**YES**
Allows access through DB2 interfaces	NO	NO	**YES**

Figure 1-4 Comparison between popular MySQL storage engines and IBMDB2i

The IBMDB2I Storage Engine for MySQL opens an opportunity for better interoperability between MySQL-based applications running in IBM i and other native IBM i applications. For example, you can run a Web merchandise store application with PHP and MySQL to receive purchase orders from your customers and let another IBM i native application read the order entries from the order entry table, which is created and maintained by MySQL, with IBMDB2I Storage Engine for sales and distribution processing. After the order is processed for delivery by IBM i sales and distribution application, it updates the MySQL order entry table to indicate completion of the order fulfillment and the MySQL Web store application can use this information to send notification e-mail to the customers.

Currently, no IBMDB2I Storage Engine support is available that enables MySQL to interact with DB2 for i tables (created from a DB2 for i interface). These tables cannot be accessed from within MySQL environment.

You can make table-level changes to DB2 for i tables (created by MySQL) from a DB2 for i interface. However, several of the changes can be incompatible with MySQL and produce undesired effects. For example, you must not delete the MySQL tables that are created through IBMDB2I Storage Engine from DB2 for i environment because MySQL would not be aware of this event and an error could occur when MySQL tries to access the deleted tables. A careful object-based authority assignment to such table objects helps to prevent this undesired event.

Regarding tables created with the IBMDB2I Storage Engine, several table or column-level operations are interchangeable between the DB2 for i environment and MySQL. For example, referential and check constraints that are defined on the IBMDB2I tables are enforced by DB2 for i on database operations from both MySQL and DB2 for i (native and SQL).

Although certain operations are compatible between environments, some table or column-level changes to the MySQL tables made by DB2 for i are valid only within DB2 for i environment and MySQL has no acknowledgement of the changes. This same principle applies to certain table-level changes made by MySQL. In some cases DB2 for i has no awareness of the changes made by MySQL. We discuss the interoperability considerations between MySQL and DB2 for i in details starting from 2.5, “Usage notes for the IBMDB2I Storage Engine” on page 14.

2

Architecture and functional support

In this chapter, we discuss the architectural and usage aspects of the IBMDB2I Storage Engine for MySQL database server running in IBM i operating system 5.4 and 6.1. We also provide some detailed usage information of the IBMDB2I Storage Engine.

This chapter contains the following topics:

2.1 Architecture introduction

The IBMDB2I pluggable storage engine for MySQL is a custom open-source engine designed and created to comply with MySQL standards. It can be used only with MySQL database server that runs in the Portable Application Solutions Environment (PASE) of IBM i 5.4 or 6.1.

MySQL architectural design is implemented in a two-layer approach. The upper SQL layer includes the SQL engine with parser, optimizer, and global and engine-specific caches and buffers; the lower storage layer comprises a set of various pluggable storage engines (see Figure 1-2 on page 3 and Figure 2-1). The pluggable storage engine layer is designed to be transparent to the upper SQL layer. So, the SQL layer is free of dependencies on which the storage engine manages any table that is being accessed. You do not have to be concerned about which engines are involved in processing SQL statements for their results.

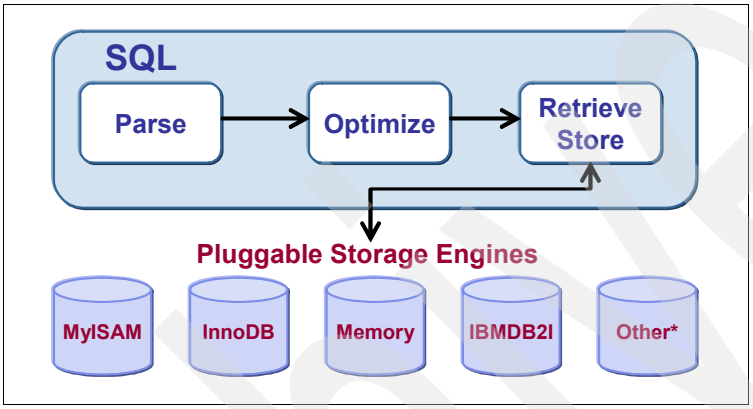


Figure 2-1 MySQL two-layer architecture

With MySQL 5.1, the makers of MySQL introduced a new architecture that allows storage engines to be dynamically loaded into and unloaded from a running MySQL server. Hence the term *pluggable storage engine*. Sun Microsystems actively cultivates both external and internal storage engine developments, which means many storage engines are available now and more will be available in the future. MySQL partners (including IBM) continue to develop many new specific-purpose engines, which can be open or non-open source.

Storage engines manage data storage for tables that are created by MySQL. The most basic storage engines implement read-only table scanning. More advanced storage engines can implement indexing, transactions and other modern-day database management features.

The MySQL server communicates with the storage engine through a set of defined application programming interfaces (APIs). Several APIs pertain to the storage engine as a whole, and others are specific to a particular type of engine.

In terms of low-level implementation, each storage engine is a class, with each instance of the class communicating with MySQL server through a special handler interface. Handlers are instanced on the basis of one handler for each thread that has to work with a specific table. For example, if three connections all start working with the same table, three handler instances will have to be created. After a handler instance is created, MySQL server issues commands to the handler to perform data storage and retrieval tasks such as opening a table, manipulating rows, and managing indexes.

2.2 DB2 for i SQL Server Mode

MySQL server runs in PASE of IBM i. PASE is an integrated runtime environment for AIX®-based executable codes of applications running on IBM i. It is not an operating system.

When the MySQL server starts a thread for each connection, it calls the IBMDB2I Storage Engine from a connection thread within PASE. Then the storage engine makes use of DB2 for i SQL Server® Mode job (QSQSRVR) running in the Integrated Language Environment® (ILE) to complete the operation on behalf of the MySQL connection. The DB2 for i SQL Server Mode jobs are the primary instrument used by the IBMDB2I Storage Engine in creating and maintaining the tables with SQL Data Definition Language (DDL) and manipulating the table data records with native record level IOs. Any of the MySQL statements that is reconstructed for DB2 for i execution and any committable transactions runs in a DB2 for i SQL Server Mode job.

SQL Server Mode offers the following benefits:

- Transaction management

  Because each connection uses its own QSQSRVR job, transactions can be committed or rolled back without affecting other connections. Non-server mode jobs are permitted to have only one active transaction for each activation group, which provides relatively less transaction flexibility.

- Performance

  Applications that manage multiple connections can easily achieve parallel processing through SQL Server Mode jobs when different threads are working at servicing unique connections.

- Connection management

  Because connections to the database manager establish the QSQSRVR job or jobs, the application can establish multiple connections to the database. Non-server mode jobs are permitted to have only one connection to the database.

- SQL Server Mode

  This provides isolation of job log messages and is also more convenient for debugging.

The operational flow is shown in Figure 2-2 on page 10 and is described briefly as follows:

1. An SQL statement on an IBMDB2I table is sent to the MySQL server.
2. MySQL server parses and optimizes the statement. No DB2 for i optimization is involved here.
3. The IBMDB2I Storage Engine is called by MySQL server to perform operations associated with the statement.
4. IBMDB2I passes the operation to the QSQSRVR job associated with the MySQL application connection, as follows:
   - Data Definition Language (DDL) operations (such as CREATE TABLE, ADD INDEX, and others) are reconstructed as DB2 for i SQL statements and executed.
   - Data Manipulation Language (DML), such as read, insert, delete, and update, is translated to record level access native I/O operations.
5. Results are returned to the QSQSRVR server and then to MySQL.

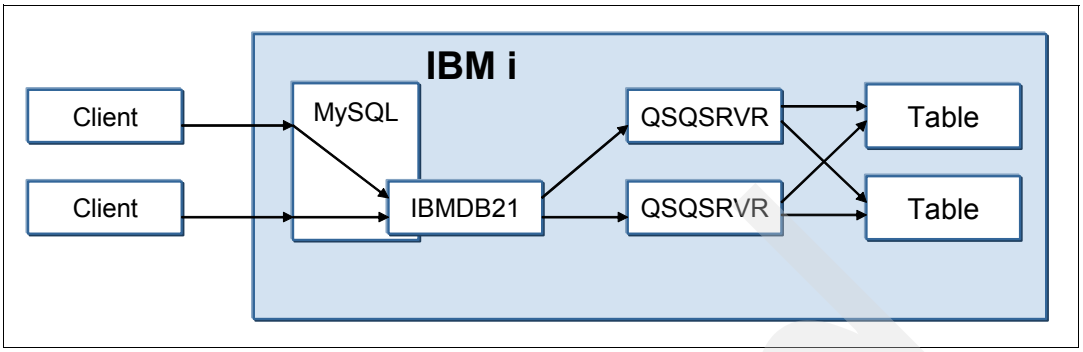


Figure 2-2 MySQL and DB2 for i SQL Server Mode

For more information about DB2 for i SQL Server Mode, see *DB2 for i5/OS: SQL Server Mode Primer*, TIPS0658:

http://www.redbooks.ibm.com/abstracts/tips0658.html?Open

2.3 Using the IBMDB2I Storage Engine

The IBMDB2I Storage Engine for MySQL is designed for ease of use. MySQL also provides a flexible environment for you to use the engine.

2.3.1 Plugging in the storage engine

Before a storage engine can be used, the storage engine plug-in shared library must be loaded into MySQL by using the INSTALL PLUGIN statement. For example, the plug-in and shared library have the following names:

ibmdb2i This is the DB2 for i engine plug-in.

ha_ibmdb2i.so This is the shared library.

Load the plug-in with the following MySQL statement:

```
mysql> INSTALL PLUGIN ibmdb2i SONAME "ha_ibmdb2i.so";
```

To install a pluggable storage engine, the plug-in file must be located in the MySQL plug-in directory, and the user issuing the INSTALL PLUGIN statement must have INSERT privileges for the mysql.plugin table.

The shared library must be located in the MySQL server plug-in directory, the location of which is given by the plugin_dir system variable.

2.3.2 Unplugging the storage engine

To unplug a storage engine, use the UNINSTALL PLUGIN statement:

```
mysql> UNINSTALL PLUGIN <storage engine name>;
```

If you unplug a storage engine that is needed by existing tables, those tables become inaccessible but will still be present on disk (where applicable). You should ensure that no tables are using a storage engine before you unplug that storage engine.

2.3.3 Setting the storage engine

When you create a new table in MySQL, you can specify which storage engine to use by adding an ENGINE table option to the CREATE TABLE statement, for example:

```
CREATE TABLE t1(id INT) ENGINE = IBMDB2I;
```

If you omit the ENGINE (or TYPE) option in the CREATE TABLE statement, the default storage engine is used. Normally, this is MyISAM, but you may change the default engine by using the server startup option when you start MySQL:

```
mysqld_safe --default-storage-engine=IBMDB2I &
```

Or, you can specify the parameter named default_storage_engine in the MySQL startup option file (default location at `/etc/my.cnf`), which takes effect when MySQL is restarted.

You may also set the default storage engine to be used during your current MySQL session by setting the storage_engine variable:

```
SET storage_engine = IBMDB2I;
```

To convert a table from one storage engine to another, use an ALTER TABLE statement that indicates the new engine:

```
ALTER TABLE t1 ENGINE = IBMDB2I;
```

If you try to use a storage engine that is not available in your environment, MySQL instead creates a table by using the default storage engine, which usually is MyISAM. This behavior is convenient when you want to copy tables between MySQL servers that support different storage engines. For example, in a replication setup, perhaps your master server supports transactional storage engines for increased safety, but the subordinate servers use only non-transactional storage engines for greater speed. This automatic substitution of the default storage engine for unavailable engines can be confusing for new MySQL users. A warning is generated when a storage engine is automatically changed.

When new tables are created, MySQL always creates files (of type .frm) with corresponding names in the following MySQL data directory default path to store metadata of the table and column definitions:

`/data/`*`<MySQL database name>`*

The table's index and data can be stored in one or more other files, depending on the storage engine. The server creates the .frm file above the storage engine level. Individual storage engines create any additional files required for the tables that they manage.

A database may contain tables of different types. That is, all tables in the same MySQL database instance do not have to be created with the same storage engine.

2.3.4 MySQL metadata files when using IBMDB2I

When a MySQL database entity is created with the IBMDB2I Storage Engine as a schema object in QSYS file system, MySQL also creates the metadata of its database in a directory in an IBM i IFS directory. By default, the directory created to store the metadata is the following path:

`/data/`*`<MySQL database name>`*

In this directory and for each table object created in the schema, MySQL creates and stores the metadata in two stream files with the same name as the table, one with the .frm extension and the other with the .FID extension.

Figure 2-3 shows the content of a sample directory that MySQL uses to store its metadata information for its SUGARDB2 database.

```
                            Work with Object Links

Directory  . . . . :   /data/SUGARDB2

Type options, press Enter.
  2=Edit   3=Copy   4=Remove   5=Display   7=Rename   8=Display attributes
  11=Change current directory ...

Opt   Object link            Type     Attribute    Text
      accounts.frm           STMF
      accounts.FID           STMF
      accounts_audit.frm     STMF
      accounts_audit.FID     STMF
      accounts_bugs.frm      STMF
      accounts_bugs.FID      STMF
      accounts_cases.frm     STMF
      accounts_cases.FID     STMF
      accounts_contacts. >   STMF
                                                                      More...
Parameters or command
===>
F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F12=Cancel   F17=Position to
F22=Display entire field           F23=More options
```

Figure 2-3 A sample directory for MySQL metadata of SUGARDB2 database

This metadata directory path is specified in the startup option file `/ect/my.cnf` (`/etc` is the default location but it can be in any directory) with the following parameter in the [mysqld] section:

```
datadir = <directory path>
```

As shown in the preceding figure, the default value for the IBMDB2I Storage Engine is:

```
datadir = /data
```

You may change this datadir path to another directory path, but you should keep the directory under QOpenSys file system for a valid support of mapped case-sensitive MySQL database object names when they are created in DB2 for i.

The .frm file is a MySQL data dictionary information file that stores the definition of each of the corresponding MySQL table created with IBMDB2I. The .FID file is created and used by IBMDB2I to help the MySQL server make sure that no incompatible or problematic changes occur to the tables without the awareness from the MySQL perspective. For example, if an eligible DB2 for i user alters a table (by proper access permission) by deleting a column or changing the column attributes, MySQL has no awareness of this action and unpredictable results could occur when the table is accessed from MySQL environment.

The .FID file stores the last known file-level identifier (not the record format-level identifier) of the corresponding DB2 for i table object. IBMDB2I compares the file-level ID of the DB2 for i

table object being accessed against the value stored in the .FID file when that table is accessed from MySQL, as follows:

- If the values are the same, it means the table definition is not changed and thus can be accessible.
- If the values are different, IBMDB2I interprets that the table definition is modified from its last-known state and thus should not be used, to avoid an unexpected result. However, if you change the table definition in such a way that you know it is a safe change because the structure of the table is still maintained, you can delete the corresponding .FID file after the change is made and before the changed table is accessed. You should then execute the MySQL FLUSH TABLE command against the changed table. IBMDB2I will recreate the corresponding .FID file with the most current file-level ID the moment the modified table is accessed.

2.4 Comparison of MySQL and DB2 for i

Many architectural, syntactical, and feature differences exist between MySQL and DB2 for i. One important difference is that the MySQL database server uses a dedicated thread-based server architecture. A MySQL server process can create a number of threads. A global thread is responsible for creating and managing user connections. A thread is created to handle each new user connection, and authentication and queries are executed in the connection thread. The MySQL server process creates other threads for various database management tasks. Although thread-safe for most operations, DB2 for i is designed based on a process-based architecture and uses a server job model for complete thread-safety of SQL.

Another major architectural difference is that MySQL database uses a pluggable storage engine architecture. Each storage engine has different characteristics that you may choose to use to implement on a table by table basis. Alternatively, DB2 for i uses its own integrated storage engine.

Although DB2 for i has true schema support, MySQL does not provide such a support. Each database in MySQL can be thought of as an equivalent to a schema in DB2 for i and it is implemented as such by the IBMDB2I Storage Engine. A single instance of MySQL with many databases can be visualized as a single database in DB2 for i with each MySQL database implemented as a schema in QSYS file system.

MySQL has several nonstandard SQL features and syntax. Although the IBMDB2I Storage Engine accommodates much of this, good practice is to use standard SQL for compatibility with DB2 for i.

MySQL modes define what SQL syntax should be supported and what kind of data validation checks should be performed. This makes using MySQL easier in different environments and using MySQL with other database servers. Although any of the available modes may be used with the IBMDB2I Storage Engine, certain behaviors for non-strict modes, such as allowing `'0000-00-00'` as a valid date, are not allowed by DB2 for i and will fail. For more information, read about Server SQL Modes, in the *MySQL 5.1 Reference Manual* from Sun Microsystems, Inc. at:

`http://dev.mysql.com/doc/refman/5.1/en/`

MySQL and DB2 for i has different sets of supported data types. When MySQL creates its tables with IBMDB2I, data type mapping occurs for the ones that DB2 for i does not support. MySQL to DB2 for i data type mapping information is discussed in 2.8, “Data type mapping from MySQL to IBMDB2I Storage Engine” on page 28.

MySQL uses case-sensitive names for its database, tables, and columns. DB2 for i stores its object names in uppercase by default. However, DB2 for i object names can be stored in mixed or lowercase if you enclose the names in a pair of double quotes in CREATE SCHEMA and CREATE TABLE statements. For example:

```
CREATE SCHEMA “SugarDB2”
CREATE TABLE “SugarDB2”.“customers”...
```

2.5 Usage notes for the IBMDB2I Storage Engine

The IBMDB2I Storage Engine for MySQL primarily provides a way in which DB2 for i can implement persistent storage mechanism for database objects create by MySQL running in IBM i PASE. The engine also provides a limited set of database management features, some of which are accessible only from jobs running in IBM i. In this section, we discuss the details of such supports.

2.5.1 Supports available from IBMDB2I for MySQL

The IBMDB2I Storage Engine provides the API methods that are defined for the table handler. They are called on a per-table, per-thread basis. Examples of basic API methods include methods to create and drop a table, and to scan, insert, update, and delete rows in the table. The more advanced API methods provide functions for indexing and transactions.

One function that must be performed in the create table method for the IBMDB2I Storage Engine is to map data types from MySQL to DB2 for i, because of differences in the data types they support. For DDL operations in general, the storage engine is designed to be tolerant of syntactical differences between MySQL and DB2 for i, and of proprietary, nonstandard behavior of MySQL.

Another functional necessity of the storage engine is to convert record formats. When the IBMDB2I Storage Engine returns a row to MySQL, the data is converted from the DB2 for i record format to the MySQL internal record format. Inversely, when writing or updating a row from MySQL, the data is converted from the MySQL internal format to that of DB2 for i. The engine also performs conversions between the MySQL character sets and the DB2 for i CCSIDs

One other conversion that the IBMDB2I Storage Engine handles is that of error codes. DB2 for i error codes are mapped to MySQL error codes when possible.

Although DB2 tables created by MySQL can be available for access by DB2 applications (with proper authority assignments according to IBM i security implementation), MySQL does not have the capability to read or access the DB2 native tables that are created by DB2 for i in the QSYS file system.

The MySQL tables that are implemented with the IBMDB2I Storage Engine are created as the DB2 for i physical file objects in QSYS file system and applications running in IBM i can also access and manipulate their data. But these tables might not make use of all the available features that a native DB2 for i table can. You should keep in mind not to use DB2 for i operation over a table created by MySQL with the IBMDB2I Storage Engine that would cause any conflict of usage from MySQL perspective. Also be aware that most of MySQL database features defined from MySQL perspective (for example triggers, stored procedures, views, and so on) are not observed from DB2 for i perspective. The converse is also true. See Table 2-1 on page 15 for more information. This restriction is based on the fact that the tables maintained by MySQL have to conform to MySQL’s support features without any conflict. For

example, you should not delete tables that are created by MySQL using the IBM i command DLTF or the DB2 for i SQL statement DROP TABLE because MySQL would not be aware of this operation. In summary, it is prudent to assign proper IBM i object authority for those MySQL tables to prevent unqualified IBM i users from creating such an undesired incident. The architecture is illustrated in Figure 2-4.

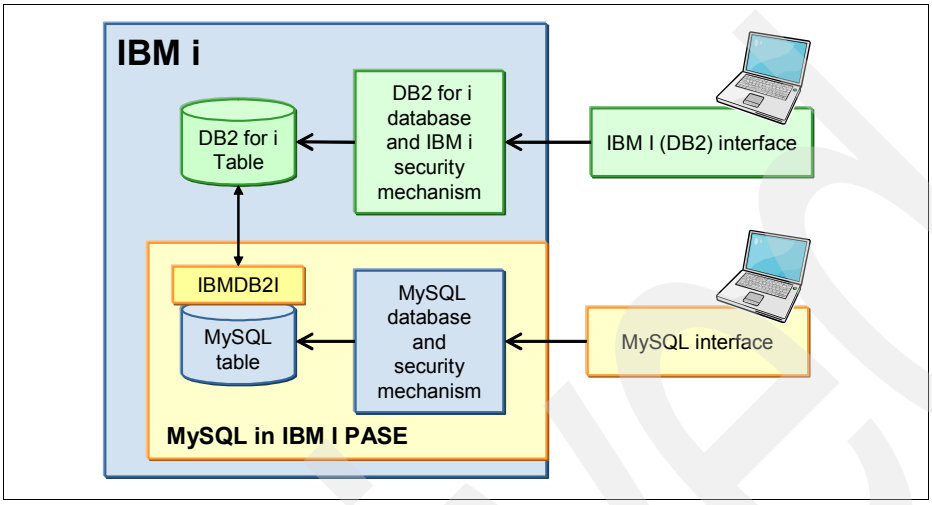


Figure 2-4 MySQL and DB2 for i environments are separated but linked by IBMDB2I Storage Engine

You should also be aware of the information in Table 2-1.

Table 2-1 The IBMDB2I Storage Engine supported functions from different perspectives

Database functions	IBMDB2I supports	IBMDB2I considerations
Data access for MySQL-based and DB2-based applications	IBMDB2I enables you to store MySQL data into the DB2 for i database so that other IBM i applications and tools can access data for interoperability.	Other contemporary database features (such as triggers, stored procedures, views, and others) are valid only within the environment in which they are created. You can find more details about this in the remaining portion of this chapter.
MySQL Data Definition Language (DDL) execution	IBMDB2I supports a limited set of MySQL DDL: ► Creating and dropping MySQL database ► Creating, renaming, dropping tables (and their primary and unique key constraints) ► Creating and dropping indexes	Other MySQL DDL statements run only within MySQL environment. DB2 for i has no awareness of the MySQL DDL statements unsupported by the IBMDB2I Storage Engine. You can also find more details about this in the remaining sections of this chapter.
MySQL Data Manipulation Language (DML) execution	IBMDB2I supports MySQL DML statements for scanning (SELECT), inserting, updating, and deleting rows.	The supported MySQL DML operations are executed in DB2 for i as native I/O operations, not as SQL statements.
MySQL Data Manipulation Language (DML) performance optimization	MySQL DML are executed as DB2 for i native I/O operations.	Tuning of MySQL DML must be done totally within the MySQL environment.

Database functions	IBMDB2I supports	IBMDB2I considerations
Indexes	MySQL indexes are created by the MySQL CREATE TABLE, CREATE INDEX, or ALTER TABLE statements.	DB2 for i can also create indexes for MySQL-IBMDB2I tables but these indexes are used exclusively within the DB2 for i environment.
Authentication and authorizations	MySQL and DB2 for i have separate security schemes that must be separately administered.	IBM i user profiles are not known to MySQL and MySQL users are not known to IBM i. Authentication and authorizations, including GRANT and REVOKE statements, are handled separately in their respective environments.
Transaction support	IBMDB2I supports commitment control for transactions.	IBMDB2I does not support XA transactions.
Long DB2 for i schema name	IBMDB2I supports long schema names (MySQL database name) up to 30 characters in length for IBM i 6.1 and up to 10 characters for IBM i 5.4.	See Table 2-2 on page 18 for more information.
Large object (LOB) data types support	IBMDB2I supports LOB data types in DB2 for i 5.4 and 6.1.	None
Database trigger	Triggers created on a table using the MySQL CREATE TRIGGER statement are fired only by operations within MySQL environment. DB2 for i is not aware of triggers created in MySQL environment.	DB2 for i can create triggers over MySQL tables created by IBMDB2I. Although these triggers are unknown to MySQL, they are fired by operations originated from both DB2 for i and MySQL environments.
Database primary, unique, and foreign key constraints	Primary key, unique key, and foreign key constraints are supported by IBMDB2I.	DB2 for i can also create constraints over MySQL tables with the ALTER TABLE statement. These constraints apply to operations performed by both MySQL and DB2 for i. However, a violation of such a constraint might not be reported clearly to the MySQL environment. Instead, it will likely be reported as a form of generic errors, and then you have to browse the joblog to learn more about them.
Database view	Views can be created over a table using the MySQL CREATE VIEW statement but they are used only within MySQL environment.	DB2 for i can also create views over MySQL-IBMDB2I tables but they are exclusively used within DB2 for i environment. DB2 for i does not support MySQL views; MySQL does not support DB2 for i views.

Database functions	IBMDB2I supports	IBMDB2I considerations
SQL function and procedure	Functions and procedures created by MySQL can be used only within MySQL environment.	DB2 for i does not support MySQL functions and procedures. MySQL does not support DB2 functions and procedures.
Partitioned table	Partitioned tables are supported through the MySQL partition engine. Separate tables are created for the partitions.	IBMDB2I supports tables partitioned through MySQL with some limitations.

2.5.2 Case-sensitive name mapping support for MySQL

The MySQL case-sensitive name mapping support by the IBMDB2I Storage Engine is determined by the IBM i file system that contains the MySQL data directory (datadir). Because QOpenSys is the primary file system on IBM i that supports case-sensitive file names, you should use the QOpenSys file system for the MySQL data directory if you want to have the letter case of MySQL schema and table names preserved when they are mapped to DB2 for i object names. The installer that is shipped with MySQL uses `/data` as the default location for MySQL datadir. User-defined file systems (UDFS) can also be created as case-sensitive, but this is generally an appropriate solution only when an auxiliary storage pool (ASP) is being used to manage the DB2 data.

You can place the MySQL data directory in a non-case-sensitive file system, but the letter case of MySQL schema and table names might not be preserved when they are created as DB2 for i object names by the IBMDB2I Storage Engine.

For more information about case-sensitivity support for file names in the IBM i IFS, see:

`http://publib.boulder.ibm.com/infocenter/iseries/v5r4/topic/rzahl/rzahlnscase.htm`

2.5.3 Object access control between IBMDB2I and native IBM i jobs

A table or index is opened by the IBMDB2I Storage Engine on an as-need basis to provide for flexible access control on the database objects created by IBMDB2I. The in-use indicator of the opened object affects this and other open tables under the same instance in other jobs. If a native IBM i job requests an exclusive lock to a table or index that is already opened by the IBMDB2I Storage Engine, DB2 for i attempts to close that file to accommodate the requesting IBM i job. If IBMDB2I already finishes its operation on that object, DB2 for i is able to close the file and let the requesting IBM i job have the exclusive lock. However, if IBMDB2I operation is still in progress on the object, the requesting IBM i job waits until it gets a timeout or gets the lock. Later, if MySQL requires access again, the IBMDB2I Storage Engine automatically reopens the file if the file is not under an exclusive lock.

Be aware that when MySQL issues the LOCK TABLES statement for an exclusive lock to its IBMDB2I table. If that intended table is already in used by other IBM i or MySQL jobs, the MySQL job that issues the exclusive lock request waits, without getting a timeout signal until the intended table is free to get the exclusive lock. This behavior might be a cause of a dead-lock within that MySQL job in certain situations. However, this behavior is in accordance with the MySQL design.

For more information:

- About the syntax, see:

  http://dev.mysql.com/doc/refman/5.1/en/lock-tables.html

- About transaction management, see Chapter 6, “Transaction management and locking considerations” on page 93.

2.5.4 Using System i Navigator database function with MySQL schemas

When you create MySQL database objects with mixed-case names longer than 8 characters or with all-uppercase names longer than 10 characters, the IBM i *system names* of these objects are changed and enclosed in a pair of double quotation marks. Table 2-2 demonstrates this name mapping scheme.

Table 2-2 MySQL database object name mapping to DB2 for i system name

MySQL database object name	Mapped DB2 for i object system name
SUGARDB2 This is all uppercase and not longer than 10 characters.	SUGARDB2
SUGARCRMDB2 This is all uppercase and not longer than 10 characters. It is not supported in DB2 for i 5.4.	SUGARnnnnn The nnnnn starts from 00001.
Sugardb2 This can be mixed case and not longer than 8 characters.	“Sugardb2”
Sugarcrmdb2 This can be mixed case and not longer than 8 characters.	“Sugannnn” The nnnn starts from 0001.
sugardb2 This is all lowercase and not longer than 8 characters.	“sugardb2”
sugarcrmdb2 This is all lowercase and not longer than 8 characters.	“sugannnn” The nnnn starts from 0001.

The original MySQL database object names in the table can also be referred to by SQL statements executed in both MySQL and DB2 for i environments. However, in the DB2 for i environment, you have to put the MySQL name within a pair of double quotation marks. See Figure 2-5 on page 19.

The statement in the following example reads from the table named `leads` that was created in the IBMDB2I schema named SUGARCRMDB2:

```
SELECT * FROM “SUGARCRMDB2”.”leads”
```

Note: If you want to use System i Navigator 5.4 and 6.1 to manage MySQL databases created by the IBMDB2I Storage Engine, note the following information about System i Navigator:

If you create MySQL database names as mixed-case and longer than 8 characters, or all uppercase and longer than 10 characters, you can see the schemas from System i Navigator in their *system names* format but cannot see any database objects created in those schemas.

You may use the System i Navigator 5.4 and 6.1 tool to work with all the database objects in a schema if the schema is created with a name that is:

- All uppercase and not longer than 10 characters
- Mixed-case or lowercase and not longer than 8 characters

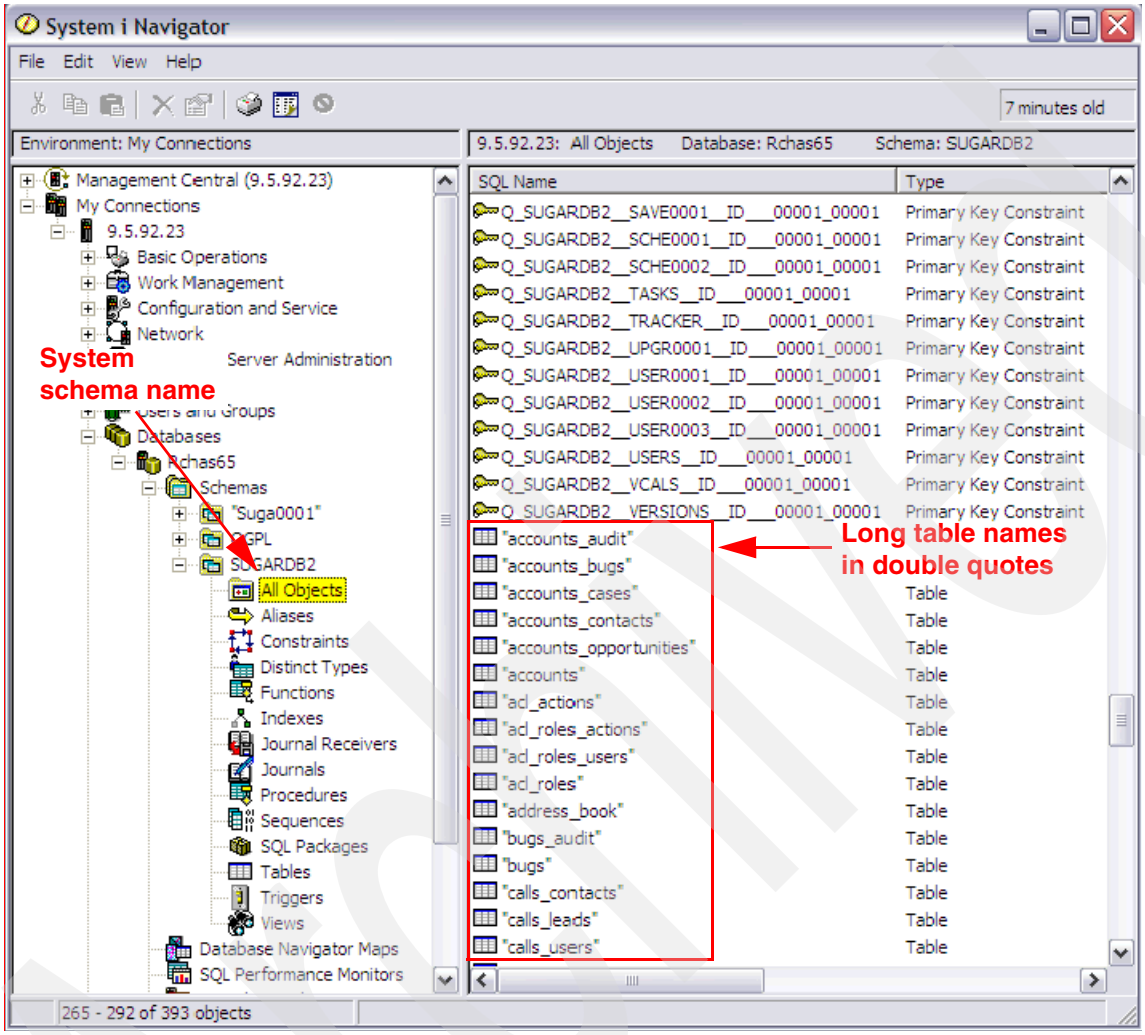


Figure 2-5 Working with the MySQL database created with IBMDB2I from System i Navigator

You may use the System i Navigator to manage the IBMDB2I schemas and tables in various ways, such as defining trigger and Check and Referential constraints, assigning access permission to IBM i user or group profiles, reorganizing the table, and so on. Be careful when using table-level operations that affect the file-level ID because the change is not known to MySQL and can cause difficulties from MySQL perspective. The remainder of this chapter contains more information.

2.6 IBMDB2I support for MySQL DDL and DML statements

The IBMDB2I Storage Engine provides various ways of handling SQL Data Definition Language (DDL) and Data Manipulation Language (DML) from MySQL to DB2 for i. In this section, we list all the MySQL statements (as documented in the *MySQL 5.1 Reference Manual*) and their mapping to DB2 for i by the IBMDB2I Storage Engine.

2.6.1 MySQL statements

Table 2-3 lists the supported MySQL statements, the function and description of each statement, the IBMDB2 engine support and mapping to DB2 for i, and other considerations.

Table 2-3 MySQL statements, function, description, and IBMDB2I Storage Engine support and mapping

MySQL statement	Intended MySQL function and description	IBMDB2I Storage Engine support and mapping
ALTER TABLE	Changes the structure of an existing MySQL table. In most (not all) cases, ALTER TABLE works by making a temporary copy of the original table, altering the copy, and then deleting the original table and renaming the new one.	IBMDB2I is notified of ALTER TABLE statements that create or drop indexes. These are mapped to a DB2 for i CREATE INDEX or DROP INDEX statement. Altering the auto_increment value column is mapped to a DB2 for i ALTER TABLE statement and does not result in a recreation of the table. The MySQL ALTER TABLE statement in most other cases is mapped to the DB2 for i CREATE TABLE, DROP TABLE, and RENAME TABLE statements. IBMDB2I also supports foreign key constraints. All reference_options are supported except ON UPDATE CASCADE. **Warning**: Re-creating the DB2 for i table causes it to lose any specific table attributes, dependent files, or granted authorities.
ANALYZE TABLE	Analyzes and stores the key distribution for a table. MySQL uses the stored key distribution to optimize the order in which tables should be joined when performing a join. If some join is not optimized in the right way, you can try using ANALYZE TABLE.	DB2 for i gathers the updated statistics about the table for use by the MySQL optimizer. It is useful when MySQL queries run slowly after a large update to a table.
BACKUP TABLE	Copies, to the backup directory, the .frm format file and .MYD data file needed to restore the table. **Note:** This statement works only for MyISAM tables.	This statement is deprecated in MySQL 5.1 and is not supported by IBMDB2I.
CHECKSUM TABLE	Reports a table checksum.	Supported by IBMDB2I
COMMIT	Commits the current MySQL transaction. **Note:** By default, MySQL runs with autocommit mode enabled.	Supported by IBMDB2I
CREATE DATABASE	Creates a MySQL database with the given name. CREATE SCHEMA is a synonym for CREATE DATABASE. The CHARACTER SET clause specifies the default database character set. The COLLATE clause specifies the default database collation.	A DB2 for i schema is not immediately created with this statement. It will be created at the moment a subsequent CREATE TABLE is executed. When creating a table, if the schema does not already exist, DB2 for i creates the schema using the CREATE SCHEMA statement. If a schema with the specified name already exists, only the table is created.
CREATE INDEX	Creates an index for a MySQL table.	Mapped to the DB2 for i CREATE INDEX statement.

MySQL statement	Intended MySQL function and description	IBMDB2I Storage Engine support and mapping
CREATE TABLE	Creates a MySQL table.	Mapped to a DB2 CREATE TABLE statement and, when applicable, one or more CREATE INDEX statements. If the schema (MySQL database) does not already exist, IBMDB2I creates the schema before creating the table. When creating a temporary table, the IBMDB2I Storage Engine maps this to a global temporary table. If the MySQL temporary table has a primary key, then IBMDB2I maps this to a DB2 unique index. DB2 for i does not allow primary keys on a global temporary table. IBMDB2I also supports foreign key constraints. All reference_options are supported except ON UPDATE CASCADE.
DELETE	Deletes rows from one or more MySQL tables.	Mapped to DB2 for i native I/O DELETE function.
DROP DATABASE	Drops all tables in the database and delete the MySQL database. DROP SCHEMA is a synonym for DROP DATABASE.	Mapped to the DB2 for i DROP SCHEMA statement. Be aware that all objects in the dropped schema are deleted regardless of their origins.
DROP INDEX	Drops an existing MySQL index from a MySQL-IBMDB2I table.	Mapped to the DB2 for i DROP INDEX statement.
DROP TABLE	Removes one or more MySQL tables. **Notes:** ► When a table is dropped, MySQL user privileges on the table are not automatically dropped. ► RESTRICT and CASCADE are allowed to make porting easier. In MySQL 5.1, most engines ignore these clauses.	Mapped to the DB2 for i DROP TABLE statement. **Note:** The IBMDB2I Storage Engine honors the RESTRICT and CASCADE keywords as specified in MySQL statement.
FLUSH	Clears or reloads various internal caches used by MySQL.	The IBMDB2I Storage Engine does not restrict this function, but it is not recommended. Deadlocks can occur in situations where multiple connections are accessing the table.
INSERT	Inserts new rows into an existing MySQL table.	Mapped to DB2 for i native I/O PUT function.
KILL	Each connection to mysqld runs in a separate thread. KILL allows the optional CONNECTION or QUERY modifier: ► CONNECTION (default) terminates the connection associated with a given thread ID. ► QUERY terminates the statement that the connection is currently executing, but leaves the connection itself intact.	Supported by IBMDB2I

MySQL statement	Intended MySQL function and description	IBMDB2I Storage Engine support and mapping
LOAD DATA INFILE	Reads rows from a text file into a MySQL table, at a very high speed.	Supported by IBMDB2I
LOCK TABLES	Locks base tables (but not views) for the current thread.	For committable transactions, the process scoped locks are retained in the recovery object until commit or rollback. **Note:** The MySQL lock types are READ and WRITE. The READ lock is mapped to the DB2 for i lock type “shared read only” to allow other reads and prevent other updates. But the WRITE lock is mapped to an “exclusive no read” lock.
OPTIMIZE TABLE	Reclaims unused space and defragments the data file. This statement works only for MyISAM, InnoDB, and ARCHIVE tables. (For other storage engines, MySQL can map this statement to ALTER TABLE.) Use this statement if you delete a large number of rows in a table or if you make many changes to a table with variable-length rows (tables that have VARCHAR, VARBINARY, BLOB, or TEXT columns).	Mapped to the RGZPFM command with default values for its parameters.
RELEASE SAVEPOINT	Removes the named savepoint from the set of savepoints of the current MySQL transaction.	Supported by IBMDB2I.
RENAME TABLE	Renames one or more tables. If the statement renames more than one table, renaming operations are done from left to right. As long as two MySQL databases are on the same file system, you can use RENAME TABLE to move a table from one database to another, for example: `RENAME TABLE current_db.tbl_name TO other_db.tbl_name.` RENAME TABLE also works for views, as long as you do not try to rename a view into a different database. Any MySQL privileges granted specifically for the renamed table or view are not migrated to the new name. They must be changed manually. If MySQL encounters any error in a multiple-table rename, it reverses the rename for all renamed tables to return everything to its original state.	The rename function is mapped in DB2 for i to the SQL RENAME operation but the table move function can fail. **Note:** ► Unlike MySQL, for the rename, any DB2 for i privileges granted specifically for a renamed table are preserved. ► Any MySQL indexes created over the table will also be renamed in DB2 for i to ensure unique index names. MySQL associates an index with a table and ensures unique index names within the scope of that table. DB2 for i accommodates this aspect by associating an index with a table and creating an object into the schema, so the index name must be unique within the schema.

MySQL statement	Intended MySQL function and description	IBMDB2I Storage Engine support and mapping
REPLACE	Works exactly like INSERT, except that if an old row in the table has the same value as a new row for a PRIMARY KEY or a UNIQUE index, the old row is deleted before the new row is inserted. (It either inserts, or deletes and inserts.)	Mapped to DB2 for i DELETE+INSERT or UPDATE
RESTORE TABLE	Restores the table or table from a backup that was made with BACKUP TABLE. **Note:** This statement currently works only for MyISAM tables.	This statement is deprecated in MySQL 5.1 and is not supported by IBMDB2I.
ROLLBACK	Rolls back the current MySQL transaction.	Supported by IBMDB2I
ROLLBACK TO SAVEPOINT	Rolls back a MySQL transaction to the named savepoint.	Supported by IBMDB2I
SAVEPOINT	Sets a named MySQL transaction savepoint.	Supported by IBMDB2I
SELECT	Retrieves rows selected from one or more MySQL tables.	Mapped to DB2 for i native I/O GET function.
SET TRANSACTION	Sets the transaction isolation level for the next transaction, globally, or for the current session.	Supported by IBMDB2I
START TRANSACTION	Begins a new MySQL transaction. Autocommit is disabled until the transaction is ended with COMMIT or ROLLBACK, at which time autocommit reverts back to its previous state.	Starts the transaction with DB2 for i SQL Server Mode job.
TRUNCATE	Empties a MySQL table completely. For storage engines other than InnoDB, truncate operations drop and recreate the table, rather than deleting rows one by one. For InnoDB, if the table has foreign key constraints, then a DELETE is done for all the rows. Regardless of how the table is emptied, the auto increment value is reset. This statement is not transaction safe.	Mapped to a DB2 for i DELETE FROM and ALTER TABLE statement.
UNLOCK TABLES	Releases, explicitly, any table locks held by the current thread.	Supported by IBMDB2I
UPDATE	Updates columns of existing rows in a MySQL table with new values.	Mapped to DB2 for i native I/O UPDATE function.

The following MySQL statements are handled only within the MySQL environment with no interface to, and no regard for, any particular storage engine:

- ALTER DATABASE
- CREATE, ALTER, CALL, DROP PROCEDURE
- CREATE, ALTER, DROP EVENT
- CREATE, ALTER, DROP FUNCTION
- CREATE, ALTER, DROP LOGFILE GROUP
- CREATE, ALTER, DROP VIEW
- CREATE, ALTER, DROP SERVER
- CREATE, ALTER, DROP TABLESPACE
- CREATE, DROP TRIGGER
- CREATE, RENAME, DROP USER
- DESCRIBE
- DO
- GRANT, REVOKE
- HELP
- RESET
- REPAIR TABLE
- SET PASSWORD
- USE

The following MySQL statements are not supported by the IBMDB2I Storage Engine.

- HANDLER
- CHECK TABLE
- REPAIR TABLE
- CACHE INDEX
- LOAD INDEX INTO CACHE

2.6.2 Column DEFAULT values

If default values or timestamp behaviors are defined by MySQL on an IBMDB2I table, the storage engine attempts to apply these attributes to the DB2 table. This ensures that data inserted both from MySQL and from traditional DB2 interfaces receive the same defaults. However, in certain cases described later, these attributes cannot be preserved, and a warning number 2528 is displayed following a CREATE TABLE statement. This warning indicates that data inserted from traditional DB2 interfaces might not receive default values in the manner defined by MySQL. However, any data inserted by MySQL continues to receive the correct values.

By default, MySQL defines TIMESTAMP columns such that they receive the current time both when inserted and when updated. Version 5.4 of DB2 for i only supports this behavior when a row is inserted. Because 5.4 does not support the default MySQL behavior, you might see frequent 2528 warnings when creating IBMDB2I tables with TIMESTAMP columns. DB2 for i 6.1 does support both insert and update behaviors through the ROW CHANGE TIMESTAMP syntax. It does not, however, allow the behavior to be limited to the update operation only, as MySQL allows.

Other scenarios that can cause a 2528 include a default value of 0 for a DATE or TIMESTAMP column and any strings that cannot be converted to the appropriate CCSID.

2.7 Other factors in DB2 for i interoperability

IBM i users can issue many IBM i commands or DB2 for i SQL statements against MySQL tables. In this section, we discuss considerations on the effects of these actions. The tables in this section show that certain IBM i or DB2 for i operations are valid only within IBM i environment, others are also valid for operations from MySQL environment, and a few others can negatively interfere with MySQL operations and should not be allowed.

2.7.1 Effect of the IBM i commands on the MySQL tables

Table 2-4 lists the effects or considerations of various IBM i command language (CL) commands running against the MySQL tables created with IBMDB2I Storage Engine.

Table 2-4 IBM i CL commands and their effects on MySQL tables

IBM i command	Effect or consideration on MySQL-IBMDB2I tables
Add Physical File Member (ADDPFM)	An SQL table is created in DB2 for i, with the maximum number of members set to 1. Therefore, you cannot add a member to a non-partitioned SQL table in DB2 for i.
Add Physical File Trigger (ADDPFTRG)	A DB2 for i trigger fires when the trigger event is initiated either through MySQL or DB2 for i. However, MySQL has no support for a DB2 for i trigger. A doubly defined trigger by both MySQL and DB2 for i will be doubly fired within their respective environments.
Change Physical File Trigger (CHGPFTRG)	The enable or disable state of a DB2 for i trigger is changed. The command has no effect on a MySQL trigger.
Remove Physical File Trigger (RMVPFTRG)	One or more DB2 for i triggers is removed. This command has no effect on MySQL triggers.
Add Physical File Constraint (ADDPFCST)	A constraint added to a DB2 for i table is enforced for operations initiated by both MySQL and DB2 for i. MySQL does not support the constraint in its table definition.
Change Physical File Constraint (CHGPFCST)	The enable or disable state of a DB2 for i constraint is changed. The command has no effect on the constraint definition defined in MySQL environment. However, it can enable or disable a DB2 for i constraint that is created by the IBMDB2I Storage Engine.
Remove Physical File Constraint (RMVPFCST)	The constraints created by DB2 for i or the IBMDB2I Storage Engine are removed. Constraints created through MySQL remain intact in MySQL environment. MySQL has no awareness of constraints defined by DB2 for i.
Change Physical File (CHGPF)	MySQL tables created with the IBMDB2I Storage Engine can be altered. However, be aware that if an incompatible change is made, MySQL has no awareness of the change and unpredictable results can occur. See 2.3.4, “MySQL metadata files when using IBMDB2I” on page 11 for more information.
Clear Physical File Member (CLRPFM)	All the data (including deleted records) from the specified member of a physical file are removed. The file is locked and MySQL has no immediate awareness of the command being executed.

IBM i command	Effect or consideration on MySQL-IBMDB2I tables
Delete File (DLTF) and Delete Library (DLTLIB)	A DB2 for i file or schema is deleted. A MySQL table or schema created by the IBMDB2I Storage Engine is deleted by this command but its definition still exists in MySQL environment. An error occurs when MySQL accesses the deleted table or schema. You should assign proper access authorities to MySQL tables to prevent such operations from IBM i environment while still allowing data manipulation from DB2 for i.
Rename Object (RNMOBJ)	An IBM i object is renamed. A MySQL table created by the IBMDB2I Storage Engine is renamed by this command but its original definition still exists in MySQL environment. An error occurs when MySQL accesses the renamed table. Similarly, a schema that is renamed cannot be accessed by MySQL in IBM i. A schema with long names in IBM i 6.1 cannot be renamed.
Move Object (MOVOBJ)	A table is moved from one schema to another. The effect is similar to DLTF or RNMOBJ.
Save Object (SAVOBJ) and Save Library (SAVLIB)	You may use these commands with schemas and tables created by MySQL with IBMDB2I. These commands are preferred over the backup utilities in MySQL or PHP because they preserve authority assignments and other DB2-specified attributes (triggers, dependent views, and others) of the saved objects. See Chapter 7, “Backup and restore considerations of the MySQL databases” on page 103.
Restore Object (RSTOBJ) and Restore Library (RSTLIB)	You may use these commands with schemas and tables created by MySQL with IBMDB2I. However, be aware that if a down-level version of table attributes are restored, MySQL has no awareness of this and unpredictable results could occur. For example, if the table had been saved prior to a MySQL ALTER TABLE to drop or change a field in that table.
Grant Object Authority (GRTOBJAUT)	You may use this command with schemas and tables created by MySQL with IBMDB2I. Its effect is valid within IBM i environment only.
Revoke Object Authority (RVKOBJAUT)	You may use this command with schemas and tables created by MySQL with IBMDB2I. Its effect is valid within IBM i environment only.
Delete User Profile (DLTUSRPRF)	No special considerations exist for this command.

2.7.2 Effect of DB2 for i SQL statements on MySQL tables

Any MySQL statements or sequence of statements that drop and re-create the table causes the loss of any DB2-specific attributes on that table. DB2-specific attributes include, but are not exclusive to, triggers, constraints, authorities, dependent files (views, logical files, and MQTs). Although these operations are supported by IBMDB2I, you should ensure that IBMDB2I tables are not unknowingly dropped. This consideration also applies when you decide whether to use MySQL backup features or IBM i save and restore commands. the reason is that the MySQL backup and restore tools do not preserve the DB2-specific attributes on the tables.

Table 2-5 on page 27 lists the effects or considerations of various DB2 for i SQL statements running against the MySQL tables created with the IBMDB2I Storage Engine.

Table 2-5 DB2 for i SQL statements and their effects on MySQL tables

DB2 for i SQL statement	Effect or consideration over MySQL-IBMDB2I tables
ALTER TABLE	This command has the same effect as the CHGPF command in Table 2-4 on page 25.
COMMENT (table, view, alias, index, column)	No special considerations. MySQL is not aware of this statement.
CREATE ALIAS	No special considerations. MySQL is not aware of this statement.
CREATE INDEX	The indexes created on MySQL tables by DB2 for i are exclusively used by SQL statements running in DB2 for i.
CREATE TABLE	No special considerations. MySQL is not aware of this statement.
CREATE SCHEMA	No special considerations. MySQL is not aware of this statement.
CREATE TRIGGER	The same as ADDPFTRG command in Table 2-4.
CREATE VIEW	No special considerations. MySQL is not aware of this statement.
DELETE	You may use this statement against the data of tables created by MySQL with theIBMDB2I Storage Engine and the result is reflected into MySQL environment.
DROP TABLE	This command has the same effect as the DLTF command in Table 2-4.
DROP SCHEMA	This command has the same effect as the DLTLIB command in Table 2-4.
DROP TRIGGER	This command has the same effect as the RMVPFTRG command in Table 2-4.
DROP VIEW	No special considerations. MySQL is not aware of this statement.
DROP INDEX	No special considerations. MySQL is not aware of this statement.
GRANT (table or view privileges)	This command has the same effect as the GRTOBJAUT command in Table 2-4.
INSERT	You can use this statement against the data of tables created by MySQL with the IBMDB2I Storage Engine and the result is reflected into MySQL environment.
LOCK TABLE	No special considerations.
OPEN	No special considerations.
REFRESH TABLE	No special considerations. MySQL is not aware of this statement which is used only against DB2 for i Materialized Query Table.
RENAME	This command has the same effect as the RNMOBJ command in Table 2-4.
REVOKE	This command has the same effect as the RVKOBJAUT command in Table 2-4.
UPDATE	You can use this statement against the data of tables created by MySQL with IBMDB2I Storage Engine and the result is reflected into MySQL environment.

2.8 Data type mapping from MySQL to IBMDB2I Storage Engine

In this section, we discuss MySQL data type mapping and handling supported by the IBMDB2I Storage Engine.

2.8.1 Data type mapping table

DB2 for i does not support all of the MySQL data types when you create a MySQL table with the IBMDB2I Storage Engine. Some incompatible data types are converted to their best match of DB2 for i data types. Table 2-6 lists the data type mappings.

Note: The IBMDB2I Storage Engine does not support spatial data.

Table 2-6 Data type mapping from MySQL to tDB2 for i by the IBMDB2I Storage Engine

MySQL data type	Mapped to DB2 for i data type	Remark, if applicable
BIGINT	BIGINT	None
BIGINT UNSIGNED	DECIMAL(20, 0)	None
BINARY	BINARY	None
BIT(x)	BINARY((x-1)/8+1)	None
BLOB	BLOB(64K)	If a length is specified for MySQL BLOB column and it is less than or equal to 255, for example, MySQL BLOB(250) will be mapped to DB2 VARBINARY(255).
BOOLEAN	SMALLINT	None
CHAR(n) or CHARACTER(n)	CHAR(n)[a]	The maximum size of n in MySQL is 255. CHAR(0) in MySQL is mapped to CHAR(1) in DB2 for i.
DATE	DATE	MySQL supports date value of 0000-00-00; DB2 for i does not.
DATETIME	TIMESTAMP	None
DECIMAL(p, s) or NUMERIC(p, s)	DECIMAL(p,s) or NUMERIC(p,s)	If p is greater than 63 and s is greater than (p-63), the field definition is truncated to DECIMAL(63, s-(p-63)) for DB2 for i. If p is greater than 63 and s is less than or equal to (p-63), the definition is not supported.
DOUBLE REAL	DOUBLE	None
ENUM	BIGINT	None
FLOAT	REAL	None
INTEGER	INTEGER	None
INTEGER UNSIGNED	BIGINT	None
LONGBLOB	BLOB(2G)	The maximum length of DB2 BLOB data type supported is 2 GB.
LONGTEXT	CLOB(2G)[a]	The maximum size for DB2 for i is 2 GB.
MEDIUMBLOB	BLOB(16M)	None

MySQL data type	Mapped to DB2 for i data type	Remark, if applicable
MEDIUMINT or MEDIUMINT UNSIGNED	INTEGER	If the table is also updated by IBM i applications, you can add a DB2 for i CHECK constraint to prevent the insertion or update of values that are invalid to MySQL.
MEDIUMTEXT	CLOB(16M)[a]	None
NCHAR(n)	CHAR(n) CCSID(UTF8)	If collation utf8_general_ci is specified, then it is mapped to CCSID UTF16.
SET	BIGINT	None
SMALLINT	SMALLINT	None
SMALLINT UNSIGNED	INTEGER	None
TEXT	CLOB(64K)[a] or LONG VARCHAR/LONG VARGRAPHIC	For more information, see the IBMDB2I option in 5.3.9, “ibmdb2i_compat_opt_blob_cols” on page 89.
TIME	TIME	Can be INTEGER if the system value ibmdb2i_compat_opt_time_as_duration is set to 1. See 5.3.3, “ibmdb2i_compat_opt_time_as_duration” on page 86.
TIMESTAMP	TIMESTAMP	None
TINYBLOB	VARBINARY(255)	None
TINYINT or TINYINT UNSIGNED	SMALLINT	If the table is also updated by IBM i applications, you may add a DB2 for i CHECK constraint to prevent the insertion or update of values that are invalid to MySQL.
TINYTEXT	VARCHAR(255)[a]	None
VARBINARY	VARBINARY	If the size is larger than 32KB, it is mapped to a BLOB instead.
VARCHAR(n) or CHARACTER VARYING(n)	VARCHAR(n)[a]	Use CLOB if n is more than 32KB. VARCHAR(0) in MySQL is mapped to VARCHAR(1) in DB2 for i.
YEAR	CHAR(4) CCSID(1208)	Can be SMALLINT if the system value ibmdb2i_compat_opt_year_as_int is set to 1. See 5.3.14, “ibmdb2i_compat_opt_year_as_int” on page 92. If the table is also updated by IBM i applications, you can add a DB2 for i CHECK constraint to prevent the insertion/update of values that are invalid to MySQL.

a. All character fields defined by MySQL (CHAR, VARCHAR, CLOB) which are represented in a double-byte DB2 for i CCSID (UCS2 or UTF16) are created as DB2 for i graphic fields (GRAPHIC, VARGRAPHIC, DBCLOB).

2.8.2 IBMDB2I support for the MySQL UTF8 data

MySQL uses a triple (3x) factor when determining the length of a CHAR, NCHAR, or CLOB column with the character set of type UTF8. For example, a CHAR(10) CHARSET(UTF8) column is created by IBMDB2I with a length of 30 bytes. Although DB2 for i normally uses a 1x factor, the factor that the IBMDB2I Storage Engine uses for converting MySQL UTF8 columns to DB2 for i is three times as much (3x). In the example, the MySQL CHAR(10) CHARSET(UTF8) column is then converted to DB2 CHAR(30).

2.8.3 A usage note on invalid data handling of MySQL column

For the MySQL columns of type YEAR, TINYINT, MEDIUMINT (as shown in Table 2-6 on page 28) of the table created by IBMDB2I, if these columns are also updated by IBM i applications, you can add a DB2 for i CHECK constraint to prevent the insertion or update of values that are invalid to MySQL. However, if you do not declare such a DB2 for i CHECK constraint to help prevent invalid data in these columns, then it is possible for an incompatible value to be stored by IBM i applications into the column. For example, a value greater than 255 may be inserted in a TINYINT column (which is mapped to SMALLINT in DB2 for i). A warning is issued during a SELECT operation issued by MySQL, but be aware that the data will be set to the maximum value for that MySQL column. See Example 2-1.

Example 2-1 MySQL Handling of invalid data sample scenario

```
1) create table t2 (f1 tinyint); (by MySQL)
2) insert a value of 500 to t2 (by IBM i applications)
3) select * from t2; (by MySQL)
   +------+
   |  f1  |
   +------+
   |  127 | (Max value of TINYINT for MySQL)
   +------+
   1 row in set, 1 warning (6.60 sec)
4) show warnings;
   +---------+------+-----------------------------------------------------------------------+
   | Level   | Code | Message                                                               |
   +---------+------+-----------------------------------------------------------------------+
   | Warning | 2521 | Some data returned by DB2 for table t2 could not be converted for MySQL|
   +---------+------+-----------------------------------------------------------------------+
   1 row in set (0.00 sec)
```

2.9 MySQL auto_increment column attribute

This section discusses the auto-increment feature that is available for DB2 for i tables.

The MySQL auto_increment column attribute can be used to generate a unique identity for new rows. The IBMDB2I Storage Engine maps the MySQL auto_increment attribute to the DB2 for i identity attribute. The mapping occurs on the DB2 CREATE TABLE statement using the GENERATE BY DEFAULT AS IDENTITY clause. MySQL allows a starting value for the auto_increment column to be specified and, if so, it is mapped to the DB2 for i START WITH clause.

A storage engine generates a value for an auto_increment column when a row is inserted if a value is not specified for the column. Most MySQL storage engines generate an auto_increment value by adding one to the maximum value stored in the table for the column. For the IBMDB2I Storage Engine, DB2 for i generates the identity value by adding one to the last generated value. The following sample MySQL statements illustrate the point:

```
create table t1 (a int auto increment, primary key(a)) engine = ibmdb2i;
insert into t1 values(3);
insert into t1 values(null),(null);
```

For the first INSERT statement, an explicit value of 3 is specified for the auto_increment column, so the value 3 is stored in the inserted row. For the second INSERT statement null is specified, so generated values 1 and 2 will be stored in the rows.

Duplicate key failures can occur in DB2 for i if MySQL applications or DB2 applications mix explicit auto_increment values with generated values within a table. For the previous MySQL statements sample, if one more record is inserted into the table for which an auto_increment value is generated, a duplicate key error will occur because the value 3 (that is, the last generated value plus one) already exists in the table. To effect the MySQL behavior for auto_increment columns, the IBMDB2I Storage Engine will detect a duplicate key error, alter the restart value for the DB2 identity column to the maximum value plus one, and retry the failed insert, but only if the following conditions are true:

- The duplicate key error occurred on an index for which the auto_increment column is a key field.
- An exclusive (LENR) lock can be acquired on the table.
- The error occurred on the first or only row of the INSERT statement.

The IBMDB2I Storage Engine does not support the following usage of auto_increment columns:

- Any MySQL global or session variable that affects the start, increment, or offset for generated auto_increment values
- Any MySQL feature that returns the *next value to be used* for an auto_increment column
- An auto_increment column on a MySQL partitioned table

2.10 National language support in IBMDB2I

National language support in the IBMDB2I Storage Engine bridges ASCII and Unicode-based character sets and collations in MySQL to EBCDIC and UNICODE based CCSIDs and sort sequences in DB2. Whenever a table is created against the IBMDB2I Storage Engine, equivalent DB2 CCSIDs must be found for each character-based column. This is true regardless if an explicit 'CHARACTER SET' clause is specified for a character-based column or not. If no CCSID can be found, an unsupported error is returned. With exceptions in the utf8 and ucs2 character sets, translations have to occur when data is passed between MySQL and DB2. Table 2-7 shows the mapping used by the storage engine. If no CCSID is listed, the character set is not supported by IBMDB2I.

Table 2-7 MySQL character set -DB2 CCSID (coded character set identifier) mapping

MySQL character set	DB2 column CCSID
armscii8	-
ascii	500
big5	1200
cp1250	1153
cp1251	1025
cp1256	420
cp1257	1156
cp850	500

MySQL character set	DB2 column CCSID
cp852	870
cp866	880
cp932	1200
dec8	-
eucjpms	-
euckr	1200
gb2312	1200
gbk	1200
geostd8	-
greek	875
hebrew	424
hp8	-
keybcs2	-
koi8r	-
latin1	1148
latin2	870
latin5	1026
latin7	1112
macce	870
macroman	-
sjis	1200
swe7	-
tis620	838
ucs2	13488
ujis	1200
utf8 (Except for utf8_general_ci)	1208
utf8 (utf8_general_ci)	1200

When a character-based primary key, index, or foreign key constraint is being created, the storage engine must associate a DB2 sort sequence with it. This is true regardless of whether an explicit 'COLLATE' clause is specified for a character-based column or not. DB2 only supports sort sequences at the table or index level, so all character-based columns in the primary key, index, or constraint must use the same collation. If multiple columns with differing collations are included in a single primary key, index, or constraint, the primary key, index, or constraint will not be created and an error will be returned. Similarly, if both a primary key and a foreign key constraint are created on a table, all of the participating character-based columns must use the same collation.If no DB2 sort sequence can be found to match the MySQL collation, an error is returned.

Note that MySQL collates NULL values first but DB2 collates NULL values last. When returning data, the IBMDB2I Storage Engine collates the NULL value first. The mapping used by the storage engine is shown in Table 2-8. If no sort sequence is listed, the collation is not supported in any release of IBM i.

Table 2-8 MySQL collation - DB2 sort sequence mapping

MySQL collation	DB2 sort sequence
armscii8_general_ci	-
armscii8_bin	-
ascii_general_ci	QALA101F4S
ascii_bin	QBLA101F4U
big5_chinese_ci	QACHT04B0S
big5_bin	QBCHT04B0U
cp1250_croatian_ci	QALA20481S
cp1250_czech_cs	QBLA20481U
cp1250_general_ci	QCLA20481S
cp1250_polish_ci	QDLA20481S
cp1250_bin	QELA20481U
cp1251_bulgarian_ci	QACYR0401S
cp1251_general_ci	QBCYR0401S
cp1251_general_cs	QBCYR0401U
cp1251_ukrainian_ci	-
cp1251_bin	QCCYR0401U
cp1256_general_ci	QAARA01A4S
cp1256_bin	QBARA01A4U
cp1257_general_ci	-
cp1257_lithuanian_ci	-
cp1257_bin	-
cp850_general_ci	QCLA101F4S
cp850_bin	QDLA101F4U
cp852_general_ci	QALA20366S
cp852_bin	QBLA20366U
cp866_general_ci	-
cp866_bin	-
cp932_japanese_ci	QAJPN04B0S
cp932_bin	QBJPN04B0U
dec8_swedish_ci	-

MySQL collation	DB2 sort sequence
dec8_bin	-
eucjpms_japanese_ci	-
eucjpms_bin	-
euckr_korean_ci	QAKOR04B0S
euckr_bin	QBKOR04B0U
gb2312_chinese_ci	QACHS04B0S
gb2312_bin	QBCHS04B0U
gbk_chinese_ci	QCCHS04B0S
gbk_bin	QDCHS04B0U
geostd8_general_ci	-
geostd8_bin	-
greek_general_ci	QAELL036BS
greek_bin	QBELL036BU
hebrew_general_ci	QAHEB01A8S
hebrew_bin	QBHEB01A8U
hp8_english_ci	-
hp8_bin	-
keybcs2_general_ci	-
keybcs2_bin	-
koi8r_general_ci	-
koi8u_general_ci	-
koi8u_bin	-
latin1_danish_ci	QALA1047CS
latin1_general_ci	QBLA1047CS
latin1_general_cs	QBLA1047CU
latin1_german1_ci	QCLA1047CS
latin1_german2_ci	-
latin1_spanish_ci	QDLA1047CS
latin1_swedish_ci	QELA1047CS
latin1_bin	QFLA1047CU
latin2_croatian_ci	QCLA20366S
latin2_czech_cs	QDLA20366U
latin2_general_ci	QELA20366S
latin2_hungarian_ci	QFLA20366S

MySQL collation	DB2 sort sequence
latin2_bin	QGLA20366U
latin5_turkish_ci	QATRK0402S
latin5_bin	QBTRK0402U
latin7_estonian_cs	-
latin7_general_ci	-
latin7_general_cs	-
latin7_bin	-
macce_general_ci	QHLA20366S
macce_bin	QILA20366U
macroman_general_ci	-
macroman_bin	-
sjis_japanese_ci	QCJPN04B0S
sjis_bin	QDJPN04B0U
swe7_swedish_ci	-
swe7_bin	-
tis620_thai_ci	QATHA0346S
tis620_bin	QBTHA0346U
ucs2_czech_ci	ACS
ucs2_danish_ci	ADA
ucs2_esperanto_ci	AEO
ucs2_estonian_ci	AET
ucs2_general_ci	QAUCS04B0S
ucs2_hungarian_ci	AHU
ucs2_icelandic_ci	AIS
ucs2_latvian_ci	ALV
ucs2_lithuanian_ci	ALT
ucs2_persian_ci	AFA
ucs2_polish_ci	APL
ucs2_roman_ci	-
ucs2_romanian_ci	ARO
ucs2_slovak_ci	ASK
ucs2_slovenian_ci	ASL
ucs2_spanish_ci	AES
ucs2_spanish2_ci	AES__TRADIT

MySQL collation	DB2 sort sequence
ucs2_turkish_ci	ATR
ucs2_unicode_ci	AEN
ucs2_bin	*HEX
ujis_japanese_ci	QEJPN04B0S
ujis_bin	QFJPN04B0U
utf8_czech_ci	ACS
utf8_danish_ci	ADA
utf8_esperanto_ci	AEO
utf8_estonian_ci	AET
utf8_general_ci	QAUCS04B0S
utf8_hungarian_ci	AHU
utf8_icelandic_ci	AIS
utf8_latvian_ci	ALV
utf8_lithuanian_ci	ALT
utf8_persian_ci	AFA
utf8_polish_ci	APL
utf8_roman_ci	-
utf8_romanian_ci	ARO
utf8_slovak_ci	ASK
utf8_slovenian_ci	ASL
utf8_spanish_ci	AES
utf8_spanish2_ci	AES__TRADIT
utf8_turkish_ci	ATR
utf8_unicode_ci	AEN
utf8_bin	*HEX

Table 2-9 shows the supported collations (based upon character sets) in the currently released DB2 versions. In the table, supported is indicated by x.

Table 2-9 MySQL character set and collation support In DB2 releases

MySQL collation	Supported in V5R4	Supported in V6R1
armscii8_general_ci	-	-
armscii8_bin	-	-
ascii_general_ci	-	x
ascii_bin	-	x
big5_chinese_ci	x	x

MySQL collation	Supported in V5R4	Supported in V6R1
big5_bin	x	x
cp1250_croatian_ci	-	x
cp1250_czech_cs	-	x
cp1250_general_ci	-	x
cp1250_polish_ci	-	x
cp1250_bin	-	x
cp1251_bulgarian_ci	-	x
cp1251_general_ci	-	x
cp1251_general_cs	-	x
cp1251_ukrainian_ci	-	-
cp1251_bin	-	x
cp1256_general_ci	-	x
cp1256_bin	-	x
cp1257_general_ci	-	-
cp1257_lithuanian_ci	-	-
cp1257_bin	-	-
cp850_general_ci	x	x
cp850_bin	x	x
cp852_general_ci	-	x
cp852_bin	-	x
cp866_general_ci	-	-
cp866_bin	-	-
cp932_japanese_ci	x	x
cp932_bin	x	x
dec8_swedish_ci	-	-
dec8_bin	-	-
eucjpms_japanese_ci	-	-
eucjpms_bin	-	-
euckr_korean_ci	x	x
euckr_bin	x	x
gb2312_chinese_ci	x	x
gb2312_bin	x	x
gbk_chinese_ci	x	x
gbk_bin	x	x

MySQL collation	Supported in V5R4	Supported in V6R1
geostd8_general_ci	-	-
geostd8_bin	-	-
greek_general_ci	x	x
greek_bin	x	x
hebrew_general_ci	x	x
hebrew_bin	x	x
hp8_english_ci	-	-
hp8_bin	-	-
keybcs2_general_ci	-	-
keybcs2_bin	-	-
koi8r_general_ci	-	-
koi8r_bin	-	-
koi8u_general_ci	-	-
koi8u_bin	-	-
latin1_danish_ci	x	x
latin1_general_ci	x	x
latin1_general_cs	x	x
latin1_german1_ci	x	x
latin1_german2_ci	-	-
latin1_spanish_ci	x	x
latin1_swedish_ci	x	x
latin1_bin	x	x
latin2_croatian_ci	x	x
latin2_czech_cs	x	x
latin2_general_ci	x	x
latin2_hungarian_ci	x	x
latin2_bin	x	x
latin5_turkish_ci	x	x
latin5_bin	x	x
latin7_estonian_cs	-	-
latin7_general_ci	-	-
latin7_general_cs	-	-
latin7_bin	-	-
macce_general_ci	-	x

MySQL collation	Supported in V5R4	Supported in V6R1
macce_bin	-	x
macroman_general_ci	-	-
macroman_bin	-	-
sjis_japanese_ci	x	x
sjis_bin	x	x
swe7_swedish_ci	-	-
swe7_bin	-	-
tis620_thai_ci	x	x
tis620_bin	x	x
ucs2_czech_ci	-	x
ucs2_danish_ci	-	x
ucs2_esperanto_ci	-	x
ucs2_estonian_ci	-	x
ucs2_general_ci	x	x
ucs2_hungarian_ci	-	x
ucs2_icelandic_ci	-	x
ucs2_latvian_ci	-	x
ucs2_lithuanian_ci	-	x
ucs2_persian_ci	-	x
ucs2_polish_ci	-	x
ucs2_roman_ci	-	-
ucs2_romanian_ci	-	x
ucs2_slovak_ci	-	x
ucs2_slovenian_ci	-	x
ucs2_spanish_ci	-	x
ucs2_spanish2_ci	-	x
ucs2_turkish_ci	-	x
ucs2_unicode_ci	x	x
ucs2_bin	x	x
ujis_japanese_ci	x	x
ujis_bin	x	x
utf8_czech_ci	-	x
utf8_danish_ci	-	x
utf8_esperanto_ci	-	x

MySQL collation	Supported in V5R4	Supported in V6R1
utf8_estonian_ci	-	x
utf8_general_ci	x	x
utf8_hungarian_ci	-	x
utf8_icelandic_ci	-	x
utf8_latvian_ci	-	x
utf8_lithuanian_ci	-	x
utf8_persian_ci	-	x
utf8_polish_ci	-	x
utf8_roman_ci	-	-
utf8_romanian_ci	-	x
utf8_slovak_ci	-	x
utf8_slovenian_ci	-	x
utf8_spanish_ci	-	x
utf8_spanish2_ci	-	x
utf8_turkish_ci	-	x
utf8_unicode_ci	-	x
utf8_bin	x	x

3

Installing and configuring MySQL V5.1 Server on IBM i

In this chapter, we explain how to install and configure the MySQL V5.1 database server on IBM i. We also discuss the IBM i Portable Application Solutions Environment (PASE) runtime environment on which the MySQL database server runs.

This chapter contains the following topics:

Note: To get the most recent information about MySQL on IBM i and the IBMDB2I Storage Engine, see the following Web site, which provides links to technotes and changes:

`http://www.ibm.com/systems/i/software/mysql/index.html`

3.1 Packaging

The Version 5.1 package of the MySQL Database Server on IBM i was created by Sun Microsystems in cooperation with IBM. The MySQL Database Server runs within the IBM i Portable Application Solutions Environment (PASE) on iSeries®, System i, or Power System hardware and provides database services for MySQL running on IBM i. Basically, MySQL provides an open source database that is installed on the IBM i integrated file system. With the addition of the DB2 for i Storage Engine for MySQL 5.1 (specific name is IBMDB2I) in the first quarter of 2009, you have a choice to create the MySQL database with the new IBMDB2I Storage Engine that stores the MySQL database in DB2 for i environment with library, physical file, and logical file objects.

Current release: The Version 5.1 package of the MySQL Database Server on IBM i is the current stable (production-quality) release.

3.2 Product structure

When you install the MySQL Database Server on IBM i, the product uses the following objects:

- Library
- User profile
- Directories
- Files

We describe each of these objects in this section.

Note: the installer that ships with new MySQL distributions uses different default installation and data directories than were used by installers that accompanied previous versions.

Library

The MYSQLINST library contains specific IBM i code for installing, configuring, and starting the product environment.

User profile

The user profile in Table 3-1 is created by default when you run the command to install MySQL (using the INSMYSQL command). This profile is for the MySQL administrative user. The profile is used for specific tasks such as to start or end subsystem jobs and internal tasks for IBM i PASE and IBM i. It also owns the files installed by MySQL.

Table 3-1 MySQL 5.1 user profile

Parameter	Definition
User profile	MySQL
User class	*USER
Special authorities	*NONE
Group profile	*NONE

Important: The MySQL user profile is created without a password. For this reason, you cannot use the user profile to sign on to the system.

Version 5 of the MySQL Database Server on IBM i uses the QSECOFR (or *SECOFR user profile) system-supplied user profile for the entire installation process including the MYSQL user-profile creation.

Directories

By default, the installer places all of the MySQL Database Server files in the directory `/QOpenSys/usr/local/mysql`.

The default data directory is `/QOpenSys/usr/local/mysql/data` and the default directory for executables has a form such as (see Figure 3-1):

`/QOpenSys/usr/local/mysql/mysql-5.1.33-i5os-power-64bit`

For ease of reference, a symbolic link named `mysql` is in the installation directory and points to the full name of the executables directory.

Important: The mysql directory in the data directory contains the MySQL system schema and other important data. This directory should not be deleted.

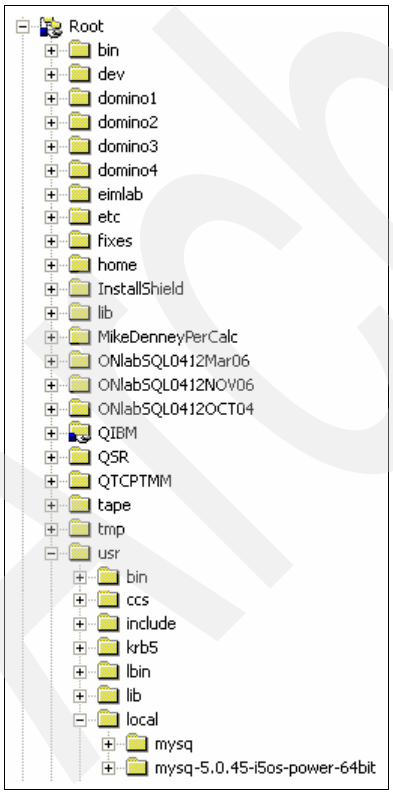


Figure 3-1 Structure of the MySQL product in the integrated file system

Files

One available *startup configuration file* (named `my.cnf`) for the MySQL Database Server on IBM i is located in the `/etc` folder by default. Figure 3-2 shows the contents of this file, which is generated during the installation process.

```
 Edit File: /etc/my.cnf
 Record :       1   of       4 by  10                    Column :    1    522 by 126
 Control :

CMD ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+....0....+.
     ************Beginning of data**************
    # Created at installation.
    [mysqld]
    datadir = usr/local/data
    user = MYSQL
     ************End of Data********************



F2=Save  F3=Save/Exit   F12=Exit   F15=Services   F16=Repeat find   F17=Repeat change   F19=Left   F20=Right
```

Figure 3-2 The my.cnf file

Two of the rows in the `my.cnf` file are of particular importance:

- The database directory, indicated as:

  ```
  datadir = usr/local/data
  ```

- The MySQL user profile, indicated as:

  ```
  user = MYSQL
  ```

These parameters are provided during the installation process in which the INSMYSQL command was issued. We explain how to use the INSMYSQL command in 3.4.2, “Installing and configuring the MySQL Database Server on IBM i” on page 50.

Important: The MySQL startup option file (`my.cnf`) must not have public write permission (called *world-writable* by MySQL). Otherwise, MySQL ignores the options in the file and generates a warning message similar to the following message:

```
Warning: World-writable config file '/etc/my.cnf' is ignored
```

3.3 IBM i PASE, runtime environment

MySQL Database Server on IBM i runs in an IBM i PASE (runtime environment). This environment consists of an interface between IBM i and the AIX run-time libraries for C/C++. It is not a complete shell, but it is suitable for running general purpose AIX executable codes in IBM i.

IBM i PASE is designed to expand the solutions portfolio of the System i platform by allowing customers and software vendors to port existing AIX applications to IBM i with minimal effort. IBM i PASE is an integrated runtime environment for AIX or other UNIX® types of applications that run on IBM i. It provides a broad subset of the application binary interface (ABI) of AIX. As a runtime environment, IBM i PASE does not experience the drawbacks of an emulation environment. However, IBM i PASE is not a UNIX operating system on IBM i; it is also not a Linux operating system on IBM i. IBM i PASE is designed to accept direct ports from AIX.

Ports from any other UNIX-based environment might require an initial port to AIX as the first step toward compatibility.

3.3.1 File systems

All file systems that are available in the IBM i integrated file system are available within IBM i PASE. Table 3-2 lists the file systems that are available to IBM i PASE.

Table 3-2 File systems available to IBM i PASE

File system	Description
/	Root file system
QOpenSys	Case sensitive, hierarchical file system; designed to support POSIX standards
QSYS.LIB	Library file system, library/file.member (database storage)
QOPT	Optical file system, CD-ROM access
QNTC	Microsoft® Windows® NT servers using SMB, the Microsoft file serving protocol
QFileSvr.400	IBM OS/400® File Server, access to remote IBM AS/400 systems
QDLS	Document Library Services, folder and document library objects; these were used by OV/400, the AS/400 office support product
/dev/QASPxx	User-defined file system, created in the auxiliary storage pool

3.3.2 Shells and utilities

The default IBM i PASE shell `/QOpenSys/usr/bin/sh` is the Korn shell. The Bourne and C shells are also available. IBM i does not currently provide support for teletypewriter (TTY) devices or Berkeley job control. Therefore, the shell functions that depend on these elements are not supported by the IBM i PASE shells.

The IBM i PASE shells and utilities run in ASCII and do no conversion between ASCII/EBCDIC bytestream file data. Users can run the iconv utility to do conversions as necessary.

The IBM i PASE shells and utilities listed alphabetically in Table 3-3 on page 46 are shipped with IBM i Option 33 as symbolic links in the `/QOpenSys/usr/bin` directory. The AIX information that follows Table 3-3 on page 46 describes the syntax and behavior of all the utilities except for the utility system that is unique to IBM i utility system and that provides an interface for invoking Control Language (CL) commands or programs from the IBM i PASE terminal.

Table 3-3 IBM i PASE-supplied AIX utilities

alias	compress	expr	ksh	ps	time
apply	cp	false	ln	psh	touch
ar	cpio	fc	locale	pwd	tr
awk	csh	fg	logname	read	true
banner	csplit	fgrep	ls	rev	type
basename	cut	file	mkdir	rm	ulimit
bc	date	find	mv	rmdir	umask
bdiff	dbx	fold	nawk	sed	unalias
bfs	dc	getconf	newform	sh	uname
bg	dd	getopt	nl	sleep	uncompress
bsh	diff	getopts	nm	sort	unexpand
cat	diff3	grep	od	split	uniq
cd	dircmp	hash	pack	strings	unpack
chgrp	dirname	head	pagesize	strip	untab
chmod	dspcat	hostname	paste	sum	wait
chown	dspmsg	iconv	patch	system	wc
chroot	du	id	pax	tab	what
cksum	dump	install	pcat	tail	which
cmp	echo	jobs	pr	tar	xargs
colrm	egrep	join	printenv	tee	yes
comm	env	kill	printf	test	zcat
command	expand				

The system utility

The system utility is a unique IBM i command that runs a CL command that was introduced in V4R5. The system utility manages ASCII/EBCDIC conversions for stdin, stdout, and stderr so that any Integrated Language Environment (ILE) code that is run by the CL command uses EBCDIC data, while the IBM i PASE shell and utilities detect ASCII data.

Syntax

The system utility runs a CL command. You might have to quote the CL command to avoid IBM i PASE shell processing for special characters in the command string. The command has the following syntax:

```
system [-b] [-h] [-i] [-k] [-K] [-n] [-q] [-s] [-v] CL-command
```

Flags

The flags for the **system** command are:

-b Forces binary mode processing for the stdin, stdout, or stderr files used by the CL command. When -b is not specified, the **system** command converts any data that is read from stdin from the IBM i (ASCII) PASE CCSID to the (EBCDIC) job default CCSID, and any data written to stdout or stderr from EBCDIC to ASCII.

This option only controls processing for stream data that is read and written by the CL command processing program. It does not affect the encoding of text lines that are written to stdout and stderr for messages and spooled output file data, which is always converted to ASCII.

-h Writes a brief description of allowable syntax for the **system** command to stdout.

-i Runs the CL command in the same process (IBM i job) where the **system** utility runs. Many CL commands are not supported in a multithreaded process. The **system** utility creates multiple threads to handle CCSID conversion for stdin, stdout, and stderr, so that it defaults to running any CL command in a separate IBM i job with only a single thread. Using this option can improve performance for CL commands that can tolerate operation in a multithreaded job.

-k Keeps spooled output files after they are processed by writing the data to stdout. The **system** utility defaults to removing spooled output files that are produced by the CL command after it writes the data to stdout. This option retains the spooled output files.

-K Generates a job log for the process where the CL command runs. In most cases, the **system** utility does not force a job log even if the CL command ends in error. This option can help problem determination when a CL command does not work as expected.

-n Does not include IBM i message identifiers in any text line written to stdout or stderr for a message that is sent by the CL command. The default format for any text lines written for IBM i messages is `XXX1234: message text`, where `XXX1234` is the IBM i message identifier. This option suppresses the message identifier, so that only the first-level message text is written to the stream.

-q Prevents writing any text lines to stdout or stderr for any IBM i messages that are sent by the CL command.

-s Does not process spooled output files that are produced by the CL command. Spooled data is not written to stdout, and spooled output files are not deleted.

-v Writes the CL command invocation string to stdout before running the CL command.

Exit status

The **system** command reports either of the following results for exit status:

0 The CL command completed successfully.

>0 An error occurred.

3.3.3 Additional commands

By using the commands listed in Table 3-4, you can obtain a secure connection, a secure copy, and a secure transfer. These commands require the installation of the 5733-SC1 LPO.

Table 3-4 Additional commands

Command	Description
ssh	A secure Telnet replacement that allows an IBM i user to connect as a client to a server running the sshd daemon. An ssh client can also be used to connect to the Hardware Management Console (HMC) on IBM i models.
scp	A secure FTP replacement. As with all implementations of sftp on other platforms, scp can only transfer data in binary format.
sftp	A secure FTP replacement. As with all implementations of sftp on other platforms, sftp can only transfer data in binary format. Note that sftp also does not provide the enhanced functions that are available.
ssh-keygen	A public or private key generation and management tool. SSH allows users to authenticate using these public and private keys as an alternative to using their operating system sign-on password.
ssh-agent	An authentication agent that can store private keys. ssh-agent allows a user to load their public/private key passphrase into memory to avoid retyping the passphrase each time an SSH connection is started.
sshd	The daemon that handles incoming ssh connections. The sshd daemon utility allows users to connect to IBM i through an ssh client.

3.3.4 Additional information and links

For additional information about the IBM i PASE runtime environment, refer to *Porting UNIX Applications Using AS/400 PASE*, SG24-5970.

You might also consider referring to the following Web pages for more information:

- Recommended IBM i fixes (including database)

  http://www-912.ibm.com/s_dir/slkbase.nsf/recommendedfixes
- Current IBM i PASE PTFs by IBM i release

  http://www.ibm.com/servers/enable/site/porting/iseries/pase/misc.html
- MySQL official Web site downloads

  http://dev.mysql.com/downloads/
- IBM Redbooks Web site

  http://www.redbooks.ibm.com
- IBM i Domain Redbooks publications

  http://www.redbooks.ibm.com/portals/systemi

3.4 Installation and configuration of the MySQL Database Server on IBM i

In this section, we explain the tasks to install the MySQL Database Server on IBM i and to perform the basic configuration.

3.4.1 Checking the prerequisites

By taking the time to check the items on your system as presented in this section, you can avoid common installation problems. A suggestion is to perform the following activities to ensure that your system is ready for installing the MySQL Database Server on IBM i.

Schema and database: Inside the MySQL Database Server environment, the terms *schema* and *database* both refer to a collection of database objects, such as tables, indexes, views, and so on. A MySQL database is created as a schema by the IBMDB2I Storage Engine.

Hardware prerequisites

At this time, no formal hardware requirements exist for running the MySQL Database Server on IBM i. For the MySQL product itself (not including software prerequisites) a consideration is to have at least 165 MB of free hard disk space.

The MySQL Database Server on IBM i environment that is provided by the MySQL Community Server for IBM i itself is not highly processing-intensive or heavily constrained by system resources.

The hardware resource requirements depend on your answers to the following questions:

- How many PHP, C, MySQL Query Browser, and similar-type applications are you planning to run? How large and complex are they?
- How many users are you planning to support? How intensive do you anticipate their usage to be, for example, light or heavy?
- How processing-intensive are your PHP, C, MySQL Query Browser, and similar-type applications? Is there a high degree or low degree of dynamic content?
- How much database or system object access do your PHP, C, MySQL Query Browser, and similar-type applications perform?

The higher the amount of applications or files, users, processing, and resource access, the more hardware resources you need.

Software prerequisites

Before you install the MySQL Database Server on IBM i, make sure all prerequisite software and fixes have been installed.

Checking the licensed programs

Ensure that your server is at IBM i V5R4 (required) and then perform the following steps to verify that all prerequisite licensed programs are installed on your system:

1. Sign on to IBM i and run the GO LICPGM command.
2. On the Work with Licensed Programs display, type option 10 (Display installed licensed programs).
3. Press F11 twice to display the product options.
4. Ensure that the software listed in Table 3-5 is installed.

Table 3-5 Software prerequisites

Licensed program	Option	Description text
5722SS1	*BASE	IBM i Version 5 Release 4 (V5R4)
5722SS1	30	Qshell
5722SC1	*BASE	IBM Portable Utilities for IBM i
5722SS1	13	System Openness Includes[a]
5722SS1	33	Portable Application Solutions Environment[b]
5799PTL	*BASE	iSeries Tools for Developers[c]

a. System Openness Includes are not necessary but may be useful. This licensed program provides all source includes for APIs that ship with IBM i.
b. IBM Portable Utilities for IBM i is not necessary but might be useful. For more information, refer to 3.3, “IBM i PASE, runtime environment” on page 44.
c. iSeries Tools for Developers is not necessary but is required for the Perl compiler in an AIX environment running under IBM i PASE. Some MySQL scripts may be compiled before running the script, such as **`mysqlhotcopy`**.

5. Press F3 twice to return to the main menu.

Checking the IBM i software PTF fixes

Make sure that you have the latest individual and group fixes for your system. *A group fix* is a collection of fixes that pertain to a specific product.

To enable the IBMDB2I Storage Engine for MySQL on IBM i, you have to apply the following program temporary fixes (PTFs) that deliver this support on IBM i 5.4 and 6.1. These lists PTFs must be applied before you can start installing the IBMDB2I Storage Engine:

- IBM i fixes (including database)

  http://www-912.ibm.com/s_dir/slkbase.nsf/recommendedfixes

- IBM i PASE fixes

  http://www.ibm.com/servers/enable/site/porting/iseries/pase/misc.html

- Storage engine enablement PTFs (INFO APAR II14442)

  http://www-912.ibm.com/n_dir/nas4apar.nsf/c79815e083182fec862564c00079d117/67d12878076e4827862574e2003c6d4a?OpenDocument

Use the Display PTF (DSPPTF) command (for individual fixes) and the Work with PTF Groups (WRKPTFGRP) command (for group fixes) to check which fixes have been applied to your system. Be sure to order and install any missing fixes prior to installing the MySQL Database Server on IBM i.

User profile authorities

For the installation process (and other administrative activities), you must use a user profile of the *SECOFR user class (with all special authorities). Use the Work with User Profiles (WRKUSRPRF) command to check your user profile.

TCP/IP configuration

Web technologies rely heavily on TCP/IP. Before you install the MySQL Database Server, ensure that TCP/IP is appropriately configured. In particular, check the following configuration settings:

1. Ensure that a host name is defined for the system. Run the Configure TCP/IP (CFGTCP) command and select option 12 (Change TCP/IP domain information) to display this setting. Make sure that a value is listed in the Host name field.
2. Make sure that the loopback entry, which represents localhost or 127.0.0.1, is configured in the TCP/IP host table. Run the Configure TCP/IP (CFGTCP) command and select option 10 (Work with TCP/IP host table entries) to display the host table. Ensure that an entry for IP address 127.0.0.1 exists and is mapped to the host names LOOPBACK and LOCALHOST.

   In addition, on this display, check that the IP address of the IBM i machine is mapped to its host name, such as SYSTEMA, and fully qualified host name, such as SYSTEMA.MYCOMPANY.COM. You should be able to successfully ping both the loopback address and the fully qualified host name of your system from IBM i, and the fully qualified host name from any browser that will be used to access PHP scripts.

3.4.2 Installing and configuring the MySQL Database Server on IBM i

After you have verified and set up the prerequisites, you are ready to install the MySQL Database Server on IBM i product. The MySQL Database Server on IBM i is provided as a save file (.savf) package that you can download directly without performing any additional steps.

Tar file procedure: Alternatively, you can download a compressed tar file (.tar) package to install the MySQL Database Server on IBM i. The installation procedure of the tar file package, which is not explained in this book, preceded the method of using the save file package. For more information about the tar file method, see the following Web address:

`http://dev.mysql.com/doc/refman/5.1/en/installing-binary.html`

The following instructions describe how to download and install the free Community edition of the MySQL Database Server. Users who require a more comprehensive level of support may wish to install the Enterprise edition, a paid offering. See the MySQL Web site for more information about the Enterprise edition.

To install the MySQL Database Server on IBM i:

1. Open the MySQL 5.1 Downloads page (see Figure 3-3 on page 52):

   `http://dev.mysql.com/downloads/`

2. If you have *already registered*:
   a. Click the **Download** button under MySQL Community Server.
   b. Go to step 4 on page 52.
3. If you have *not registered*:
   a. Click **Register**. (Registration is not mandatory, but it is useful for future references.)
   b. On the Register for a MySQL.com Account page, complete the form and click **Submit** to download the MySQL Package.
   c. Go to step 4 on page 52.

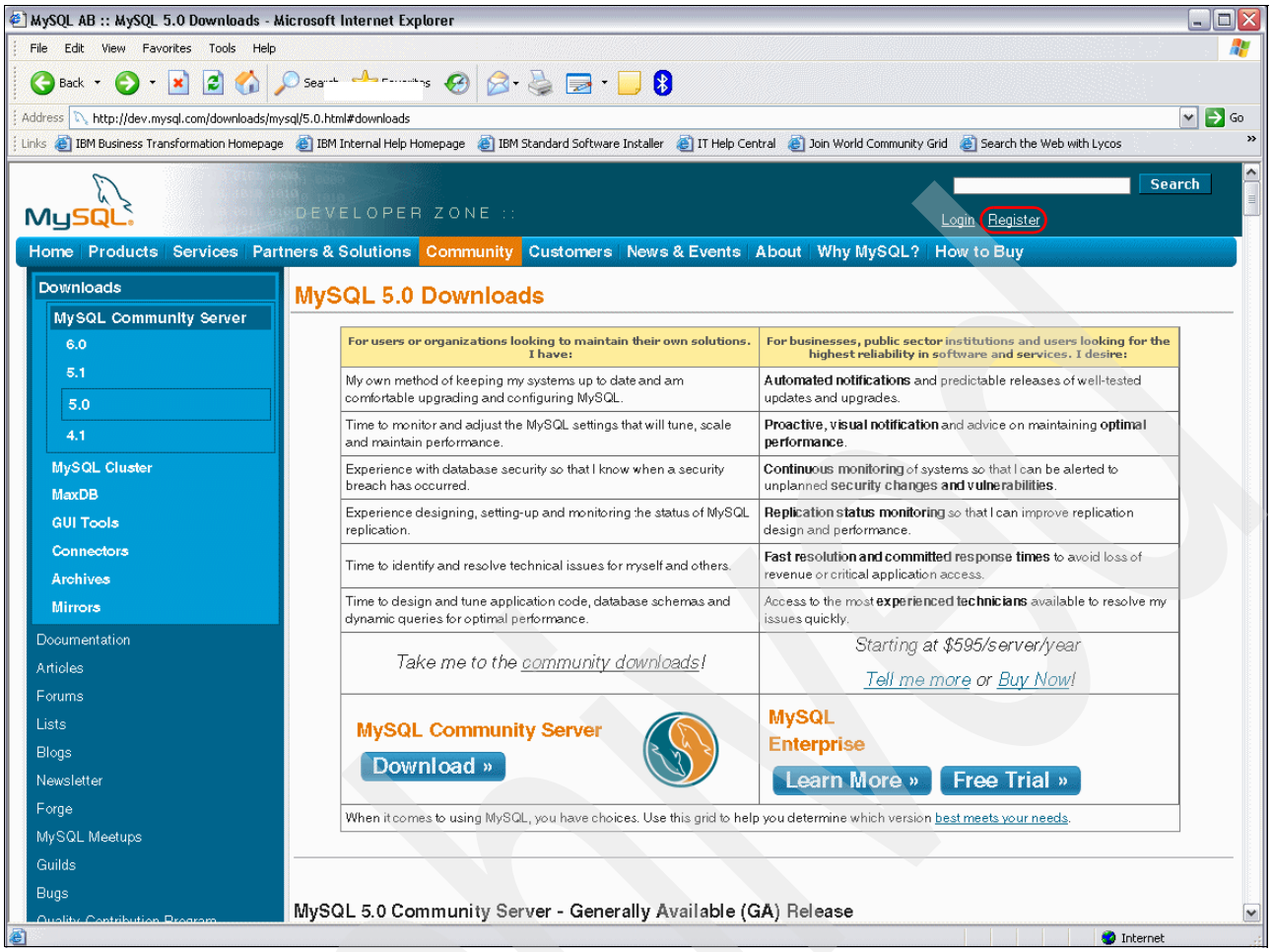


Figure 3-3 MySQL 5.0 Downloads page

4. On the next page (shown in Figure 3-4), scroll down until you find the IBM i5/OS SAFV or TAR package downloads. Click either **Download** or **Pick a mirror** link to download the file.

> **Important:** You must select the **IBM i5/OS (POWER®, 64-bit)** option regardless of whether you choose the SAVF or TAR download package.

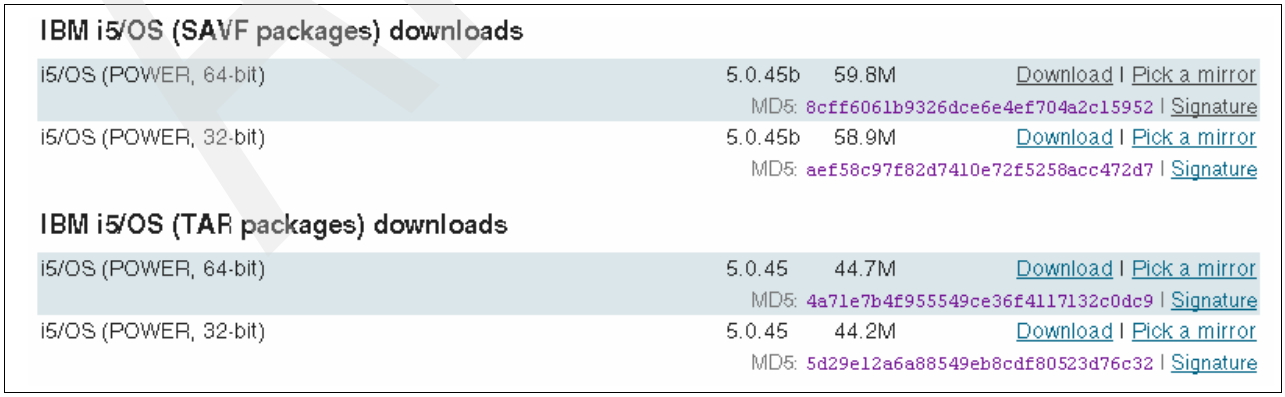


Figure 3-4 Downloading packages for the MySQL Database Server on IBM i

Tip: The **Pick a mirror** option is useful for downloading packages from a different site. Use the MD5 checksum and GnuPG signatures to verify the integrity of the packages that you download.

5. In the File Download window (Figure 3-5) that opens, click **Save** to save the package file to your workstation.

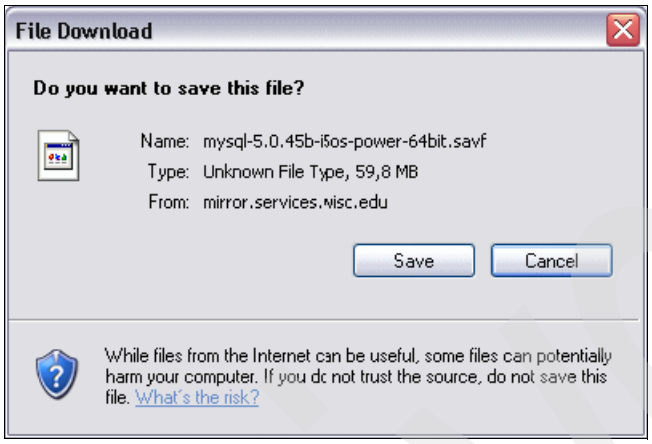


Figure 3-5 Save file dialog box

6. Log on to IBM i with a user profile that has a user class of *SECOFR with all special authorities (QSECOFR if available).
7. Create a save file by using the following command:

   ```
   CRTSAVF FILE(QGPL/MYSQLINST) TEXT('MySQL 5.1 save file')
   ```

8. Verify that FTP is running on your IBM i system:

   ```
   NETSTAT *CNN
   ```

   Look for `ftp-con` (or port `21` by pressing F14) in the Local Port column as shown in Figure 3-6 on page 54.

   If FTP is not running on your system, enter the following command:

   ```
   STRTCPSVR *FTP
   ```

```
                      Work with TCP/IP Connection Status
                                                                 System:   RCHASM27
Type options, press Enter.
  3=Enable debug   4=End   5=Display details   6=Disable debug
  8=Display jobs

     Remote           Remote     Local
Opt  Address          Port       Port       Idle Time  State
     *                *          ftp-con >  067:43:58  Listen
     *                *          telnet     001:01:32  Listen
     *                *          www-http   000:00:33  Listen
     *                *          ntp        000:42:59  *UDP
     *                *          netbios >  067:43:02  Listen
     *                *          netbios >  000:00:15  *UDP
     *                *          netbios >  000:00:14  *UDP
     *                *          netbios >  067:42:57  Listen
     *                *          ldap       067:42:43  Listen
     *                *          cifs       067:37:22  Listen
     *                *          drda       067:44:04  Listen
     *                *          ddm        067:44:04  Listen
                                                                        More...
F3=Exit   F5=Refresh   F9=Command line   F11=Display byte counts   F12=Cancel
F20=Work with IPv6 connections   F22=Display entire field   F24=More keys
```

Figure 3-6 FTP ports view

9. On your workstation, open a command prompt and transfer the MySQL save file to IBM i by using FTP:
   a. Change the directory to the one that contains the files that you downloaded from the MySQL Web site, for example:

      ```
      cd /temp
      ```

   b. Run the **ftp** command and specify the name of your IBM i system, for example:

      ```
      ftp systema
      ```

   c. If requested, enter a valid user profile and password.
   d. Enter the **bin** command to specify a binary transfer.

      > **Tip:** You can see the entire ftp transaction process by typing the **hash** command. When you do this, you see a progress bar that uses the `#####...` characters.

   e. Transfer the save file to IBM i by entering the following command, for example:

      ```
      put mysql-5.1.33-i5os-power-64bit.savf mysqlinst.savf
      ```

      > **Tip:** The IBM i naming convention does not support long names that have more than ten characters, nor does it support special characters.

   f. When the transfer has completed, enter the **quit** command.
10. Return to the 5250 session and run the Display Saved Objects (DSPSAVF) command:

    ```
    DSPSAVF FILE(MYSQLINST)
    ```

11. In the Display Saved Objects panel (Figure 3-7), verify the contents of the save file that you uploaded before. Then, Press F3 to return to the main menu.

```
                          Display Saved Objects

Library saved . . . . . :   MYSQLINST

Type Options, press Enter.
  5=Display

Opt  Object      Type      Attribute   Owner          Size (K)  Data
     MYSQLINST   *LIB      PROD        MYSQLDEV            84  YES
     INSMYSQL    *PGM      CLP         MYSQLDEV            68  YES
     TARF        *FILE     SAVF        MYSQLDEV         59416  YES
     INSMYSQL    *CMD                  MYSQLDEV             8  YES




                                                                Bottom
F3=Exit   F11=Alternate view   F12=Cancel   F16=Display header
```

Figure 3-7 Display Saved Objects panel

12. Restore the MYSQLINST library that is compressed into the save file that you uploaded to IBM i by using the **ftp** command.
13. Restore the library by using the Restore Library (RSTLIB) command:

```
RSTLIB SAVLIB(MYSQLINST) DEV(*SAVF) SAVF(MYSQLINST) MBROPT(*ALL)
ALWOBJDIF(*ALL)
```

> **Security changes message:** Ignore the security changes-type messages at the bottom of your panel. The messages, which are normally displayed, are in regard to the objects that you just restored.

14. When you finish restoring the MYSQLINST library, check that all necessary objects for installation are on the system by using the Display Library (DSPLIB) command:

```
DSPLIB LIB(MYSQLINST)
```

15. Review the information about the Display Library panel (Figure 3-8 on page 56). Then, press F3 to return to the main menu.

```
                              Display Library

Library  . . . . . :   MYSQLINST       Number of objects  . :   3
Type . . . . . :   PROD           Library ASP number . :   1
Create authority . . :   *SYSVAL        Library ASP device . :   *SYSBAS
                                       Library ASP group  . :   *SYSBAS

Type options, press Enter.
  5=Display full attributes   8=Display service attributes

Opt  Object      Type      Attribute              Size  Text
     INSMYSQL    *PGM      CLP                   69632  Install MySQL
     TARF        *FILE     SAVF                60841984
     INSMYSQL    *CMD                             8192  Install MySQL




                                                                  Bottom
F3=Exit   F12=Cancel   F17=Top   F18=Bottom
```

Figure 3-8 Display MYSQLINST library

If you are installing on DBCS systems:

On DBCS systems, a problem has been identified with the installation process. On these systems, you must change your job's coded character set identifier (CSSID) to 37 (EBCDIC) before you run the INSMYSQL installation command:

a. Determine your existing CSSID by using the DSPJOB command and selecting option 2.

b. Enter the following command:

```
CHGJOB CSSID(37)
```

c. Run the INSMYSQL command to install MySQL.

d. Run the CHGJOB command again with your original CSSID.

16. Enter the INSMYSQL command:

```
MYSQLINST/INSMYSQL
```

17. On the Install MySQL (INMYSQL) panel (Figure 3-9 on page 57), you see the following installation parameters:

– DIR('/QOpenSys/usr/local/mysql')

This parameter identifies the installation location for the MySQL files. The directory is created if it does not exist.

Note: The MySQL Database Server on IBM i can be installed anywhere. For this example, we assume that the MySQL Database Server will be installed into the `/QOpenSys/usr/local/mysql` folder in the integrated file system.

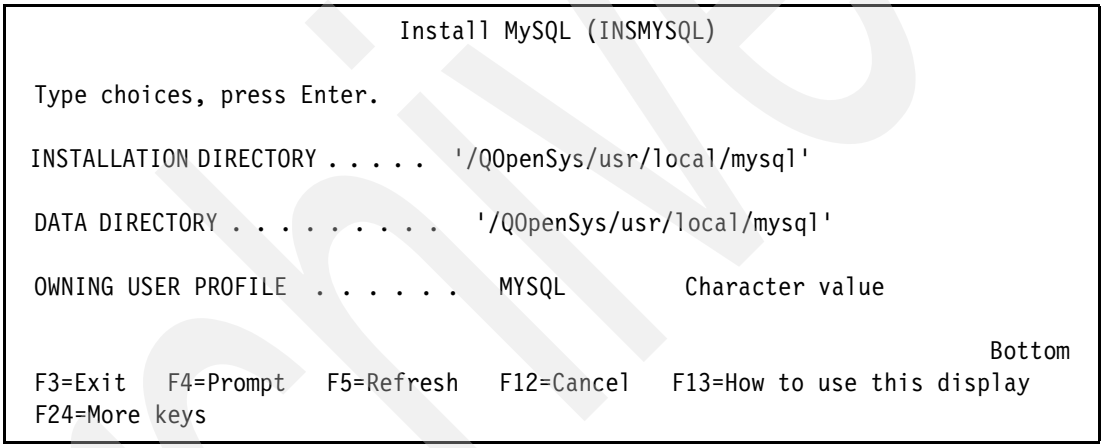
– DATADIR('/QOpenSys/usr/local/mysql/data')

  This parameter defines the location of the directory that will be used to store the database files and binary logs. This is the default value.

– USRPRF(MYSQL)

  This parameter defines the user profile that will own the files that are installed.

> **Note:** Selecting the appropriate USRPRF is an important step in securing and maintaining your MySQL Database Server installation. You should log in with this profile whenever you subsequently start the server. When the specified profile does not exist, the installer will create it but leave it disabled. Therefore, you should enable this profile after completing the installation. If you choose to use a profile other than the default profile, be sure that it has an appropriate level of authority. Keep in mind that when the IBMDB2I Storage Engine is installed, MySQL users with appropriate MySQL authorities will be able to perform operations on DB2 schemas and tables with all the authority of the user profile that the MySQL Database Server is running under. For more information, see Chapter 8, "Security" on page 137.


```
                          Install MySQL (INSMYSQL)

Type choices, press Enter.

INSTALLATION DIRECTORY . . . . .  '/QOpenSys/usr/local/mysql'

DATA DIRECTORY . . . . . .  '/QOpenSys/usr/local/mysql'

OWNING USER PROFILE  . . . . . .  MYSQL         Character value

                                                                      Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 3-9 INSMYSQL panel

> **Note:** For the case-sensitivity support of MySQL names mapping to DB2 for i object names, MySQL data directory (datadir) must reside in /QOpenSys file system. You can create and use MySQL data directory outside of QOpenSys file system but you cannot expect to have all MySQL schema and table names with preserved case when they are created as DB2 for i object names by the IBMDB2I Storage Engine.

18. Press F4 to start the MySQL installation on the IBM i server.

After successful completion of the INSMYSQL command, you see a message to the example in Figure 3-10 on page 58. You have to verify the installation, described in the next section.

```
PLEASE REMEMBER TO SET A PASSWORD FOR THE MySQL root USER ]
To do so, start the server, then issue the following commands:
./bin/mysqladmin -u root password 'new-password'
./bin/mysqladmin -u root -h RCHASM27.RCHLAND.IBM.COM password 'new-password'
See the manual for more instructions.
You can start the MySQL deamon with:
cd . ; ./bin/mysqld_safe &

You can test the MySQL deamon with mysql-test-run.pl
cd mysql-test ; perl mysql-test-run.pl

Please report any problems with the ./bin/mysqlbug script]

The latest information about MySQL is available on the Web at
http://www.mysql.com
Support MySQL by buying support/licenses at http://shop.mysql.com
Press ENTER to end terminal session.

===>

F3=Exit F4=End of File F6=Print F9=Retrieve F17=Top
F18=Bottom F19=Left F20=Right F21=User Window
```

Figure 3-10 INSMYSQL message

3.4.3 Verifying the installation

After you install the MySQL Database Server on IBM i, you must ensure that the installation process was successful:

1. Press Enter or F3 to exit the INSMYSQL message panel (Figure 3-10 on page 58).
2. Run the Display Job Log (DSPJOBLOG) command and press F10 to check the previous command execution. The Display All Message panel (Figure 3-11) opens.

   The words `exit status 0` indicate that the installation has completed successfully.

Messages to ignore: Ignore the `Security changes` and `Object not found` messages because they occur normally.

```
                              Display All Messages
                                                                System:   RCHASM27
Job . . :   QPADEV0001 User . . :   JAVIER       Number . . . :   013126

    Special authorities granted *NONE.
    User profile MYSQL created.
    Command ended normally with exit status 0.
    Command ended normally with exit status 0.
    Owner changed for object /tmp/mysql_i5os_install.tar.
    Security changes ocurred for 1 objects.
    1 object restored. 0 objects not restored.
    Current directory changed.
    Command ended normally with exit status 0.
    Object not found. Object is mysql.
    Command ended normally with exit status 0.
    Command ended normally with exit status 0.
    Command ended normally with exit status 0.
    Link removed.
                                                                          More...
Press Enter to continue.

F3=Exit   F5=Refresh   F12=Cancel   F17=Top   F18=Bottom
```

Figure 3-11 Display All Messages panel

3.4.4 Post installation tasks

In this section, we explain the additional steps that are necessary to complete the MySQL Database Server configuration so that you can access MySQL by using either a command line or MySQL Administrator on Linux, or Windows NT®, 2000, or XP.

To complete the configuration by using the command line:

1. Sign on to IBM i by using the profile specified on the INSMYSQL command (MYSQL by default), and execute the QP2TERM program to start IBM i PASE:

   ```
   CALL QP2TERM
   ```

   Note: To make your work easier, change the user profile to your home directory by using the following command:

   ```
   CHGPRF HOMEDIR('/QOpenSys/usr/local/mysql/mysql/bin')
   ```

   You must sign off and then sign on again for the change to take effect. By using this command, every time you start the IBM i PASE environment by using CALL QP2TERM, you will always be in the `/QOpenSys/usr/local/mysql/mysql/bin` folder, enabling you to more easily invoke the MySQL commands.

2. On the IBM i PASE command line in the terminal window that opens (Figure 3-12), enter the following command to change to the MySQL commands directory:

   ```
   cd /QOpenSys/usr/local/mysql/mysql/bin
   ```

   ```
                                /QOpenSys/usr/bin/-sh

     $

   ===>

    F3=Exit  F6=Print F9=Retrieve F11=Truncate/Wrap
     F13=Clear   F17=Top    F18=Bottom    F21=CL command entry
   ```

   Figure 3-12 CALL QP2TERM (terminal console)

3. Check that you are in the correct directory by entering the following command:

   ```
   pwd
   ```

4. Before creating a MySQL user profile, verify whether the MySQL Database Server is started by using the following steps:

   a. Verify whether the MySQL Database Server is started by typing one of the following commands:

      ```
      mysqladmin -u root status
      mysqladmin -u root ping
      ```

      If the server is started, you see a message like the one shown in Figure 3-13 on page 61.

```
> mysqladmin -u root status
   Uptime: 80618  Threads: 1  Questions: 254  Slow queries: 0  Opens: 32  Flush tables: 2  Open tables:
19  Queries per second avg:
  0.003
  $
> mysqladmin -u root ping
  mysqld is alive
   $
```

Figure 3-13 MySQL Database Server status

b. If the MySQL Database Server is not started, enter the following command:

```
mysqld_safe &
```

Note: The command ends with the ampersand (&) character, which indicates that it should be run in the background.

You now see a message like the one shown in Figure 3-14.

```
> mysqld_safe &
  [1] 182
   $ Starting mysqld daemon with databases from /QOpenSys/usr/local/mysql/data
```

Figure 3-14 MySQL starting server

Verify that the MySQL Database Server has started:

```
ps -ef | grep mysqld
```

A panel like the one in Figure 3-15 opens, indicating that the MySQL Database Server has started.

```
                                                    /QOpenSys/usr/bin/-sh

   $
> cd  /QOpenSys/usr/local/mysql/mysql/bin
   $
> pwd
  /QOpenSys/usr/local/mysql/mysql/bin
   $
> mysqld_safe &
   ^1!     182
  $ Starting mysqld daemon with databases from /QOpenSys/usr/local/mysql/data
> ps -ef | grep mysqld
     javier 182 181   0 10:36:55      -  0:00 /bin/sh mysqld_safe
     javier 202 182   0 10:37:04      -  0:00 /usr/local/mysql/bin/mysqld --basedir=/QOpenSys/usr/local/mysql/mysql
--datadir=/QOpenSys/usr/local/mysql/data
   --user=MYSQL --pid-file=/QOpenSys/mysql/data/RCHASM27.RCHLAND.IBM.COM.pid -u root
   $


 ===>

 F3=Exit     F6=Print   F9=Retrieve   F11=Truncate/Wrap
 F13=Clear   F17=Top    F18=Bottom    F21=CL command entry
```

Figure 3-15 MySQL Database Server status

5. Create an administrative user profile by adding this user to the *user* table into the *mysql* schema with the following command. In this example, we use *itso* for the administrative user profile:

   ```
   mysql -u root mysql -e “insert into user (host, user, password) values (‘%’, ‘itso’, ‘itso’)”
   ```

6. Grant administrative privileges to the user *itso* and encrypt the password that was generated before by entering the following commands (also shown in Figure 3-16):

   ```
   mysql -u root mysql -e “grant all privileges on *.* to ‘itso’@’%’ identified by ‘itso’ with grant option”
   mysql -u root mysql -e “flush privileges”
   ```

```
 > mysql -u root mysql -e "insert into user (host, user, password) values ('%', 'itso', 'itso')"
   $
 > mysql -u root mysql -e “grant all privileges on *.* to ‘itso’@’%’ identified by ‘itso’ with grant
option”
   $
 > mysql -u root mysql -e “flush privileges”
   $
```

Figure 3-16 Creating the MySQL administrative user profile with the grant option

> **Messages:** Notice that no messages are displayed in this step, unless an error occurs when you enter the command.

7. Check the user profile you created before. Enter the following command to log in to the MySQL Database Server:

   ```
   mysql -u root
   ```

8. Select the mysql schema:

   ```
   use mysql;
   ```

9. Execute a query over the table *user*:

   ```
   select user, password from user;
   ```

   Figure 3-17 on page 63 shows the results of running the command.

```
> mysql -u root
  Welcome to the MySQL monitor.  Commands end with ; or \g.
  Your MySQL connection id is 2
  Server version: 5.0.45 MySQL Community Server (GPL)

  Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

  mysql>
> select user, password from user;
  +--------+-------------------------------------------+
  | user   | password                                  |
  +--------+-------------------------------------------+
  | root   |                                           |
  | root   |                                           |
  | root   |                                           |
  | javier | *B174F2517BA8F7BD62D6AF171D91AB2F537BCB94 |
  | itso   | *63C5A3E03987225C0620E974CD173F0A77FF888D |
  +--------+-------------------------------------------+
  5 rows in set (0.01 sec)
```

Figure 3-17 MySQL query

10. While you are still signed on to MySQL, continue with the steps in 3.4.5, "Installing the IBMDB2I Storage Engine plug-in component for MySQL" on page 63

3.4.5 Installing the IBMDB2I Storage Engine plug-in component for MySQL

While you are still signed on to MySQL, install the IBMDB2I Storage Engine plug-in component:

1. Run the following MySQL SQL command:

   ```
   mysql> install plugin ibmdb2i soname "ha_ibmdb2i.so";
   ```

   The following message is displayed. It indicates a successful installation of the IBMDB2I plugin component.

   ```
   Query OK, 0 rows affected (0.16 sec)
   ```

2. From this point on, you can use the following statement to create MySQL database in DB2 for i with the IBMDB2I Storage Engine without having to restart your MySQL server:

   ```
   create database mydb1;
   create table mydb1.mytable1 ..... engine = ibmdb2i;
   ```

3. At this time, you may also modify the `/etc/my.cnf` file so that the IBMDB2I can be a global engine for your instance, as follows:

   a. Add the following line to the `/etc/my.cnf` file:

      ```
      default_storage_engine=ibmdb2i
      ```

   b. Restart the MySQL server so that the change can take effect. You may instead restart it later at your convenience.

4. Type `quit` and press Enter to log out of MySQL Database Server.

3.4.6 Common installation and restoration errors

Installation failures are usually caused by one or more of the following conditions:

- Your user profile does not have sufficient authority.
- You entered the wrong folder in the command. The folder must be `/QOpenSys/usr/local/mysql/mysql/bin`.
- Structures from a previous installation are found by the installer.
- The library list does not contain the QGPL or QTEMP libraries.
- Prerequisite software products or fixes are missing.
- A file named `/etc/my.cnf` already exists. Rename this file and re-try the installation.

If any of these conditions exist, correct the problem, and then remove any product files that were created during the failed installation. To find certain installation failures, run the Display Job Log (DSPJOBLOG) command. Press F10 and check for messages in the job log. You can also check for suitable logs in the integrated file system.

Use care when restoring objects to the IBM i. The most common problems during object restoration are related to object authorities or nonexistent users in the system. You should use valid users with enough authority to restore the MYSQLINST library. Otherwise, you will not be able to restore all the objects.

3.4.7 Uninstalling the MySQL Database Server on IBM i

Before you uninstall the MySQL Database Server on IBM i, verify that you no longer need the MySQL database. To make a copy of your database before you delete the product, see Chapter 7, “Backup and restore considerations of the MySQL databases” on page 103.

When you are sure that you want to delete the MySQL Database Server, follow these steps:

1. Sign on to a 5250 session on your IBM i with a user profile that has a user class of *SECOFR with all special authorities (QSECOFR if available).
2. Connect IBM i with the IBM i PASE interface:

   ```
   CALL QP2TERM
   ```

3. Stop the MySQL Database Server to avoid lock problems during file deletion in the integrated file system:

   ```
   cd /QOpenSys/usr/local/mysql/mysql/bin
   mysqladmin -u root shutdown &
   ```

4. Ensure that the server is shut down by running the following command. You might have to wait for a while.

   ```
   ps -ef | grep mysqld
   ```

   If no rows are shown, the server is shut down.

5. Delete completely the following folders:

   Attention: Make a backup of your folders before you start deleting them. The integrated file system has no way to recover deleted folders if you delete the wrong one.

   - /QOpenSys/usr/local/mysql/data
   - /QOpenSys/usr/local/mysql/mysql
   - /QOpenSys/usr/local/mysql/mysql/-1.33-i5os-power-64bit

To delete the folders:

a. Enter the Work With Link (WRKLNK) command.

b. Navigate to the correct folder.

c. Select option `2` (Edit) in the parent directory in order to delete all files and folders that are contained in the specific folder to be deleted.

d. Select option `9` (Delete recursively).

6. Navigate to `/etc` folder and delete it to remove the `/etc/my.cnf` file.

7. Optionally, delete the user profile. If you wish to delete all of the DB2 schemas and tables created through the IBMDB2I Storage Engine, you may add the OWNOBJOPT(*DLT) parameter to the DLTUSRPRF command:

```
DLTUSRPRF USRPRF(MYSQL) OWNOBJOPT(*DLT)
```

You have now uninstalled the MySQL Database Server.

3.5 Running additional same-release MySQL instances

After you have the first instance of MySQL server running in IBM i, you might want to run additional instances of MySQL server, which can serve such purposes as:

- Security

  You might want to maintain multiple MySQL databases that are totally independent from each other in all aspects. Although you might be able to do this with one MySQL server instance, using separated MySQL server instances can be a more robust way.

- Catering for different globally scoped MySQL environments

  You might encounter a situation where you cannot or should not mix multiple MySQL databases within one MySQL server instance because of conflicting requirements, such as:

  – Using different IBM i independent ASP (IASP)

    For example, if you want to create and maintain two MySQL databases in two different IASPs (which is controlled by the IBMDB2I system variable named ibmdb2i_rdb_name), you can run two instances of MySQL server, with each using a different IASP.

  – Using different table creation options

    For example, because you can use two different time-of-day formats (as controlled by the IBMDB2I system variable named ibmdb2i_compat_opt_time_as_duration), if you want to run different MySQL databases that implement those different time-of-day formats but you do not want to mix them, you may run each MySQL database in a separate MySQL server instance. Using different table creation options also applies to the options:

    - ibmdb2i_compat_opt_blob_cols
    - ibmdb2i_create_index_option
    - ibmdb2i_compat_opt_year_as_int
    - ibmdb2i_compat_opt_allow_zero_date_vals
    - ibmdb2i_propagate_default_col_vals

To start and run an additional instance of the MySQL server, you can use the existing MySQL base directory (basedir) that was created when you installed the very first MySQL server instance in your IBM i partition (or system with only one partition). However, you have to

create and use the following new items to successfully start and run an additional instance of MySQL server:

- A new MySQL data directory (datadir) path
- A new TCP/IP port number
- A new TCP/IP socket file name
- A new directory path for a new `my.cnf` startup option file
- A new IBM i user profile (For security reasons, you might require this to run the new instance)

To run additional same-release MySQL server instances using the common MySQL base directory in the same IBM i partition:

1. Log on to an IBM i 5250 session. You might want a new IBM i user profile to do this.
2. Create a new MySQL data directory for the new MySQL server instance, for example (from IBM i command line):

   ```
   md '/QOpenSys/mysql/instance2'
   md '/QOpenSys/mysql/instance2/data'
   ```

   Note: For the case-sensitivity support of MySQL names mapping to DB2 for i object names, MySQL data directory (datadir) must reside in the `/QOpenSys` file system. You can create and use MySQL data directory outside of QOpenSys file system but you cannot expect to have all MySQL schema and table names with preserved case when they are created as DB2 for i object names by the IBMDB2I Storage Engine.

3. Create a new directory and a new MySQL startup option file, for example:

   ```
   md '/etc/instance2'
   CPY OBJ('/etc/my.cnf') TODIR('/etc/instance2') DTAFMT(*BINARY)
   ```

4. Modify the `my.cnf` file:
   a. Run the following command:

      ```
      EDTF STMF('/etc/instance2/my.cnf')
      ```

   b. Modify the new `my.cnf` file with the lines in Example 3-1. Ensure that the MySQL system variables `datadir`, `port`, and `socket` contain different values from those used by existing MySQL server instances.

   Example 3-1 Modifying the new my.cnf file

   ```
   ************Beginning of data**************
   # Created at installation.
   [mysqld]
   datadir=/QOpenSys/mysql/instance2/data
   user=MYSQL
   basedir=/QOpenSys/usr/local/mysql/mysql
   port=3305
   socket=/tmp/mysql-instance2.sock
   # default_storage_engine=ibmdb2i (delete the comment sign after
   # new instance IBMDB2I plugin is installed)
   ************End of Data********************
   ```

Note: In the `my.cnf` file, the line `user=MYSQL` is used only when you sign on to a 5250 session as QSECOFR user profile to start the MySQL server. Starting the MySQL server job switches to run under the IBM i user profile specified by this line. MYSQL is the default IBM i user profile created by the MySQL installation process.

If you sign on to the 5250 session with an IBM i user profile other than QSECOFR, you can omit the following line from `my.cnf` file because the MySQL start up process ignores this line in such a case:

```
user=<an IBM i user profile>
```

5. Connect to PASE interface of IBM i and change to the existing MySQL base directory:

```
CALL QP2TERM
cd /QOpenSys/usr/local/mysql/mysql
```

6. Run the installation script in the new MySQL data directory to install MySQL system tables for the new instance, shown in Example 3-2. The command option `--defaults-file` must be the first parameter in the command line.

Example 3-2 Run the installation script

```
bin/mysql_install_db --defaults-file=/etc/instance2/my.cnf
--basedir=/QOpenSys/usr/local/mysql/mysql
--datadir=/QOpenSys/mysql/instance2/data
```

The following message is displayed:

```
Installing MySQL system tables...
OK
Filling help tables...
OK
```

7. Start the new MySQL instance. In doing this, you also need to specify *a new available TCP/IP port number and a new directory path for socket file* for the new instance to start successfully (which you have done in the preceding step by modifying the `/instance2/etc/my.cnf` file in the previous step), for example:

```
bin/mysqld_safe --defaults-file=/etc/instance2/my.cnf &
```

The messages in Example 3-3 are displayed.

Example 3-3 Sample messages

```
YYMMDD HH:MM:SS mysqld_safe Logging to '/QOpenSys/mysql/instance2/data/<Host
Name>.<Domain Name>.err'.

YYMMDD HH:MM:SS mysqld_safe Starting mysqld daemon with databases from
/QOpenSys/mysql/instance2/data
```

8. At this point, MySQL server should be started and active. Press **Enter** key once to get to QShell command line prompt.
9. Check the MySQL server status by using one of the following commands:
   - The **`ping`** command:

     ```
     bin/mysqladmin -u root --socket=/tmp/mysql-instance2.sock ping
     ```

     The following message is displayed:

     ```
     mysqld is alive
     ```

– The **status** command:

```
bin/mysqladmin -u root --socket=/tmp/mysql-instance2.sock status
```

A message similar to the following message is displayed:

```
Uptime: 215  Threads: 1  Questions: 2  Slow queries: 0  Opens: 15  Flush
tables: 1  Open tables: 8  Queries per second avg: 0.9
```

– The **shutdown** command, from the MySQL base directory:

```
mysqladmin -u root --socket=/tmp/mysql-instance2.sock shutdown
```

> **Note:** To shutdown a specific instance of an active MySQL server, use the command from the MySQL base directory.

The important point is to specify the correct path name of the specific socket file used by the instance against which you want to run the command **mysqladmin**.

10. You may now connect from an IBM i QShell session to the new MySQL server instance by using the following command:

```
bin/mysql -u root --socket=/tmp/mysql-instance2.sock
```

> **Note:** For an external MySQL client connection, when you use an external client to connect to an instance of MySQL server through a TCP/IP communication link (for example, to perform the preceding step 9 and 10), you must specify the following additional parameters for the command used in the external client:
>
> - --host = <host name> or <IP address>
> - --port = <TCP/IP port number of the server instance>
> - --socket parameter is not required for external client connection
>
> For more information about connecting to the MySQL Server, see:
>
> http://dev.mysql.com/doc/refman/5.1/en/connecting.html

11. Install the IBMDB2I Storage Engine plugin for the new instance:

```
mysql> install plugin ibmdb2i soname "ha_ibmdb2i.so";
```

The following message is displayed to indicate a successful installation of the IBMDB2I plugin component:

```
Query OK, 0 rows affected (0.16 sec)
```

12. From this point on, you can use the following commands to create to create MySQL database in DB2 for i with the IBMDB2I Storage Engine without having to restart your MySQL server:

```
create database mydb1 ;
create table mydb1.mytable1 ..... engine = ibmdb2i ;
```

13. If you want, you may edit the file `/instance2/etc/my.cnf` to delete the comment symbol (#) from the following line:

```
# default_storage_engine=ibmdb2i
```

Removing the comment sign causes the IBMDB2I Storage Engine to be a global one for your instance. You must then, restart the MySQL server to make this change take effect. Or you can do this later on when it is convenient to restart the MySQL server.

For more information about running multiple MySQL servers on the same machine for UNIX and Windows environments, and principles common to running in IBM i PASE, see:

http://dev.mysql.com/doc/refman/5.1/en/multiple-servers.html

3.6 Installing additional MySQL instances of different releases

After you have the first instance of MySQL server running in IBM i, for a testing purpose, you might want to install and run an additional instance of MySQL server, which has a different release from existing instances. Because the installation script of MySQL is not designed for this purpose, you have to take further actions to make this work. You may also require a new IBM i user profile to install and run the new instance.

To install and run a different release of MySQL server instance:

1. Download the TAR package file of the MySQL server product version you want to install.
2. Choose or create an IBM i IFS directory that you want to use as the installation directory (for example, /QOpenSys/usr/local/mysql).
3. Send the MySQL package files to that installation directory by using **ftp** in binary mode, Netserver, or others.)
4. Untar the TAR file into the installation directory. One way to do this is to:

   a. Start a QShell session.
   b. Use the command TAR with the option -tvf first to check the content of the TAR file
   c. Use the option -xvf to extract the files.

   This creates a subdirectory to the installation directory (for example, `/QOpenSys/usr/local/mysql/mysql-5.1.33`. This path is now the MySQL base directory (basedir) for all subsequent steps.
5. You may optionally change the owner of the MySQL base directory and its subtree to MYSQL or another IBM i user profile that you want to use to install and run the MySQL server. For example:

   ```
   chown -R /QOpenSys/usr/local/mysql/mysql-5.1.29-rc mysql
   ```

   Optionally changing the owner is what the provided installation script does. Therefore, you determine whether to have the same action applied to the additional installation.
6. Begin with the first step in 3.5, “Running additional same-release MySQL instances” on page 65, making sure to specify a new base directory and another new directory for the MySQL startup option file (`my.cnf`) where appropriate on the command line and in the new `my.cnf` file.

4

Implementation

This chapter provides information about how to find and access the MySQL data in the DB2 database, how the objects are represented in DB2, and what to be aware of when you update the MySQL database from DB2.

This chapter contains the following topics:

4.1 Finding objects in DB2

In this section, we discuss how to find the objects in DB2 when you have created them in MySQL. We look at the ways to find the objects through character-based interface and through System i Navigator. We provide a small PHP script that can extract System i system names from your SQL. This script is shown in Appendix A, “Tool to look up DB2 SQL and system names” on page 177.

Note: In the System i Navigator, you are not able to see tables and indexes when the schema name is longer than 10 characters in uppercase or longer than 8 characters in lowercase.

4.1.1 Libraries

When you have created a database in MySQL, nothing is created in DB2 for i until the first IBMDB2I table is created in the database. The creation of the first IBMDB2I table in a schema causes the corresponding DB2 schema to be created, if it does not already exist.

A schema (that is created in MySQL with CREATE DATABASE command) can be accessed from a variety of DB2 for i interfaces such as interactive SQL, native OS commands, Web Query, System i Navigator, high-level language programs, embedded SQL, and others. Depending on what DB2 interface you are using, there might be certain differences in the way the schema names are referenced.

If you have to access the schema through a DB2 for i interface, differences exist in how you reference the objects if you create schema names that are longer than 10 characters in length and are in uppercase, or are longer than 8 characters and in lowercase. Use of schema names longer than 10 characters for IBMDB2I tables requires the use of IBM i 6.1. Let us examine how a table is created in a particular schema and how you can locate the object:

1. Sign on to i5/OS and execute the QP2TERM program to start the i5/OS PASE environment as explained in Chapter 3, “Installing and configuring MySQL V5.1 Server on IBM i” on page 41.
2. Change to the directory where the MySQL tools are installed:

   ```
   cd /QOpenSys/usr/local/mysql/mysql/bin
   ```

3. Connect to the MySQL Database Server:

   ```
   mysql -u root
   ```

4. Create a new schema:

   ```
   CREATE DATABASE myschema;
   ```

5. Select the schema as the default schema to work with:

   ```
   USE myschema;
   ```

6. Create a dummy table in the schema:

   ```
   CREATE TABLE TABLE1 (NUMBER INT) ;
   ```

   You can now use another i5/OS session to find the SQL names and system names. If you use the command STRSQL or System i Navigator, then you have to use double quotation marks for the schema names when they are longer than 10 characters in uppercase or longer than 8 characters in lowercase.

7. Repeat these steps for a number of different schema names. The result is placed in the Table 4-1 on page 74.

From any SQL interface to the DB2 you can query the systems tables to extract the system names. You can use the STRSQL green screen CL command or you can use Run SQL Script in System i Navigator similar to in Figure 4-1.

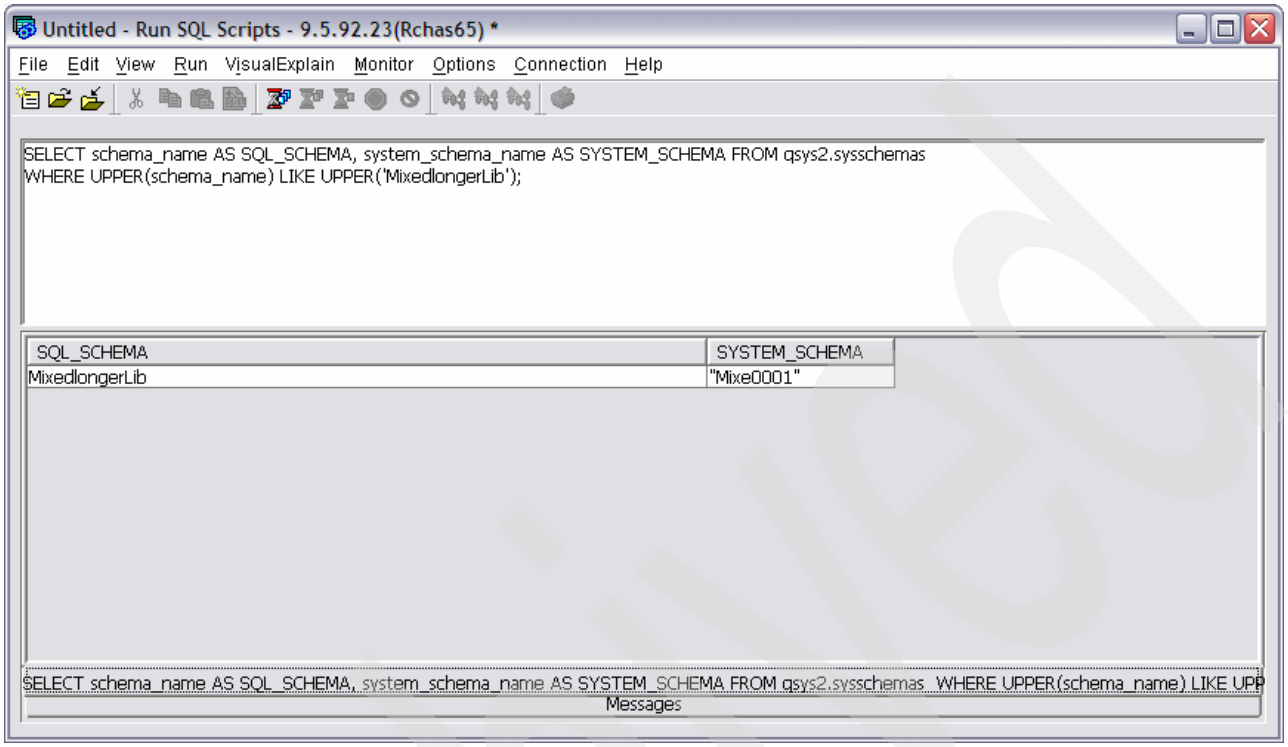


Figure 4-1 Use Run SQL script to find system name for SQL schema

When querying the system tables, you may use the following SQL request:

```
select SCHEMA_NAME, SYSTEM_SCHEMA_NAME from qsys2.sysschemas where
upper(SCHEMA_NAME) = upper('MixedlongerLib');
```

You may also use the tool described in Appendix A, “Tool to look up DB2 SQL and system names” on page 177. Then you can perform a similar lookup similar to Figure 4-2 on page 74 in which we look up the system name for the library named MixedlongerLib.

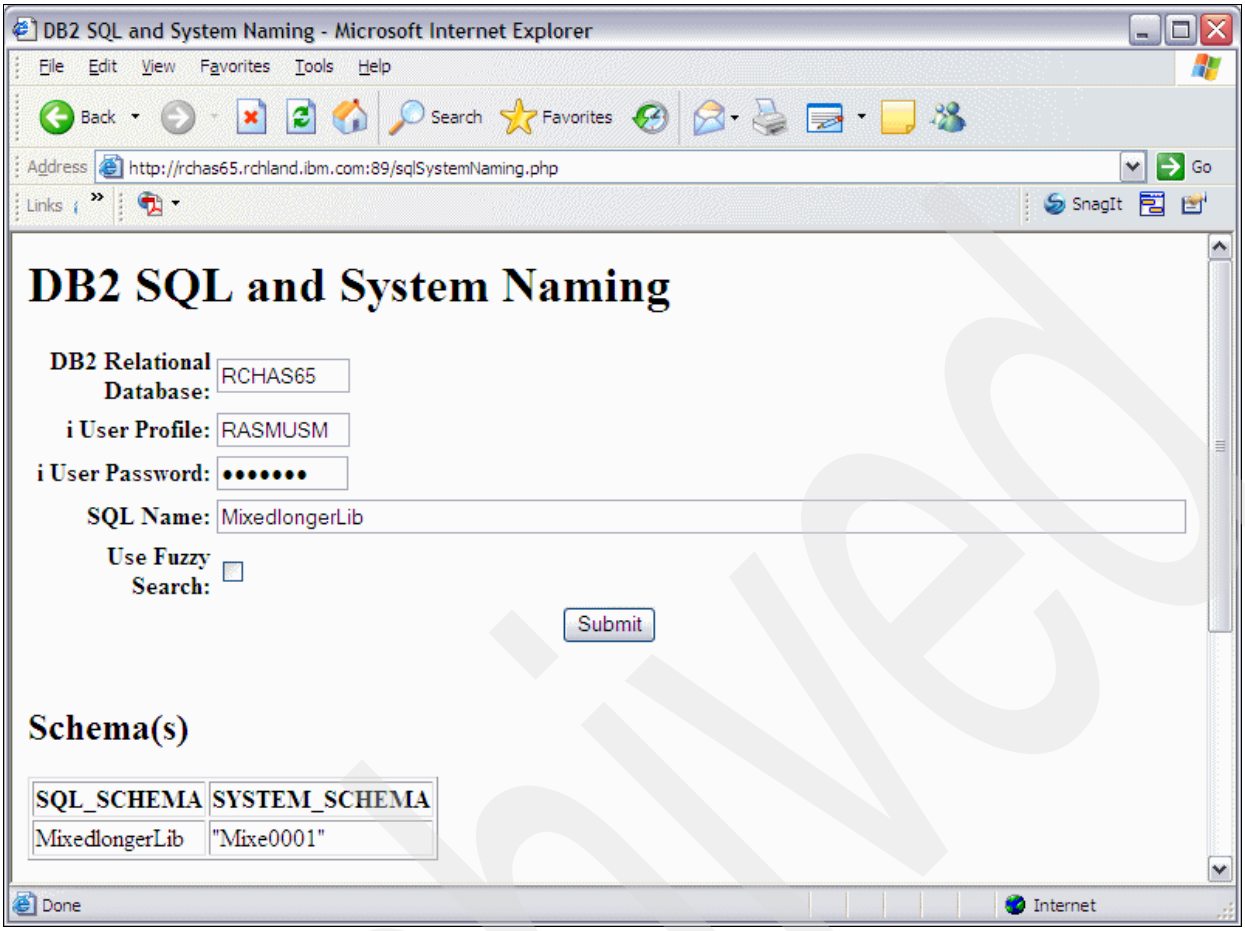


Figure 4-2 Find the library system name for long SQL name

When we do the same for a number of different library names, you can see the names from MySQL, the DB2 names, the way to reference the schema from DB2, and the system name shown in Table 4-1.

Table 4-1 Example of schema names created by MySQL

MySQL name	DB2 name	DB2 reference name	System name
myschema	myschema	“myschema”	“myschema”
mylongerschema	mylongerschema	“mylongerschema”	“mylo0001”
UPPERSCHEM	UPPERSCHEM	UPPERSCHEM	UPPERSCHEM
UPPERLONGSCHEMA	UPPERLONGSCHEMA	UPPERLONGSCHEMA	UPPER00001
MixedLib	MixedLib	“MixedLib”	“MixedLib”
MixedlongerLib	MixedlongerLib	“MixedlongerLib”	“Mixe0001”

As you can see in Table 4-1, the names vary regarding upper and lowercase and the length. Any time you use names with lowercase you must have the double quotation marks shown in column labeled DB2 reference name. The reason is because the system always uses uppercase names, so you have to indicate that it is actually the lowercase name you want to use.

In System i Navigator, you cannot see tables in libraries, where the library names are longer than longer than 10 characters in uppercase or 8 characters in lowercase. A suggestion is to use library names in uppercase and that the names do not exceed 10 characters. This convention enables you to more easily find your libraries by using the traditional tools on the System i without having to do any lookups in the system tables.

4.1.2 Tables

Similar to the schema names, you have to take care of the system names and the DB2 reference names when you use uppercase names longer than 10 characters and when you use lower or mixed case, and especially when you have names longer than 8 characters. Table 4-2 lists examples of the names.

Table 4-2 Example of table names created by MySQL

MySQL name	DB2 name	DB2 reference name	System name
mytable	mytable	“mytable”	“mytable”
mylongertable	mylongertable	“mylongertable”	“mylo0001”
UPPERTABLE	UPPERTABLE	UPPERTABLE	UPPERTABLE
UPPERLONGTABLE	UPPERLONGTABLE	UPPERLONGTABLE	UPPER00001
MixedTab	MixedTab	“MixedTab”	“MixedTab”
MixedlongerTab	MixedlongerTab	“MixedlongerTab”	“Mixe0001”

As you can see, names in upper or lowercase and length of the table name do matter.

Altering a table

Any changes to a table should come from MySQL. If you add a column from DB2, a MySQL error can occur next time you try to access the table data from MySQL. To prevent this situation, drop the new column and delete the table's associated FID file, as described in 2.3.4, “MySQL metadata files when using IBMDB2I” on page 11.

When you alter a table from MySQL to perform any action other than adding or dropping an index, the table is dropped, along with any associated indexes and constraints. The table is re-created with the new description, the data is copied form the original table, and the indexes and constraints are re-created. If you have any indexes over the table that are created from DB2 they are dropped and not re-created in this process. Similarly, any other objects created from DB2 that are related to the table, such as triggers, will be dropped and not re-created.

4.1.3 Indexes

When indexes are created in MySQL, they have a special name format related to the table name. The format is the index name specified with the SQL CREATE INDEX command followed by three underscores and then the table name, as follows:

```
<index name>___<table name>
```

If you have activated the ibmdb2i_create_index_option in the configuration options for IBMDB2I, then additional indexes are created. They have the following format:

```
<index name>__H_<table name>
```

Because the index name depends on the table name, be careful when you rename your table, because you are also renaming the index name. The indexes created by MySQL generally have an ASCII-based sorting sequence and not the default for the System i, which is *HEX sorting sequence. Figure 4-3 shows an example of the indexes for the account table from the SugarCRM application.

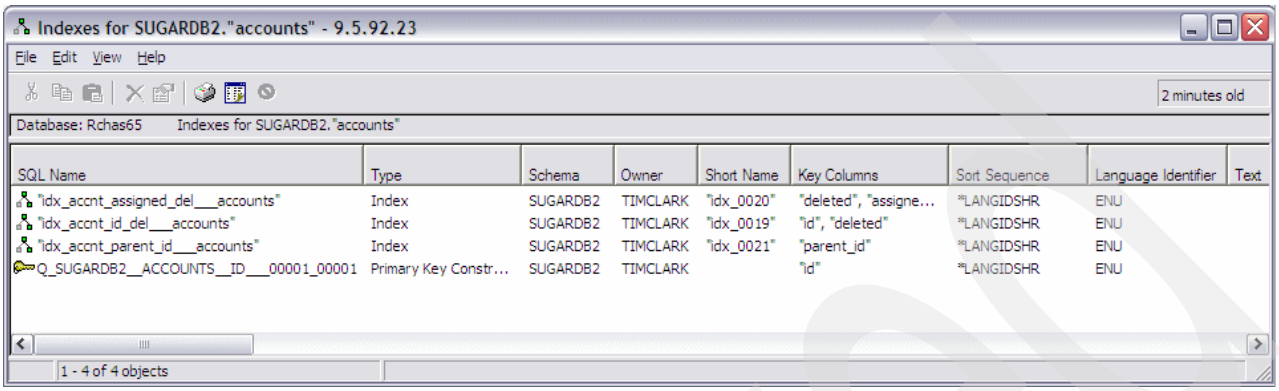


Figure 4-3 Indexes over the accounts table in SugarCRM

You should mark the Sort Sequence column that is not *HEX; you might need *HEX indexes if you want to access the database through DB2 SQL. You can automatically get the missing indexes created by activating the ibmdb2i_create_index_option in the configuration options for IBMDB2I. The default value is zero (0) for not creating additional indexes. If you change this value to one (1), then additional indexes will be created. Those indexes have the *HEX sorting sequence and can help the optimizer and the database engine analyze and execute the queries from DB2. If you have activated the creation of additional indexes, the SugarCRM application can look similar to Figure 4-4.

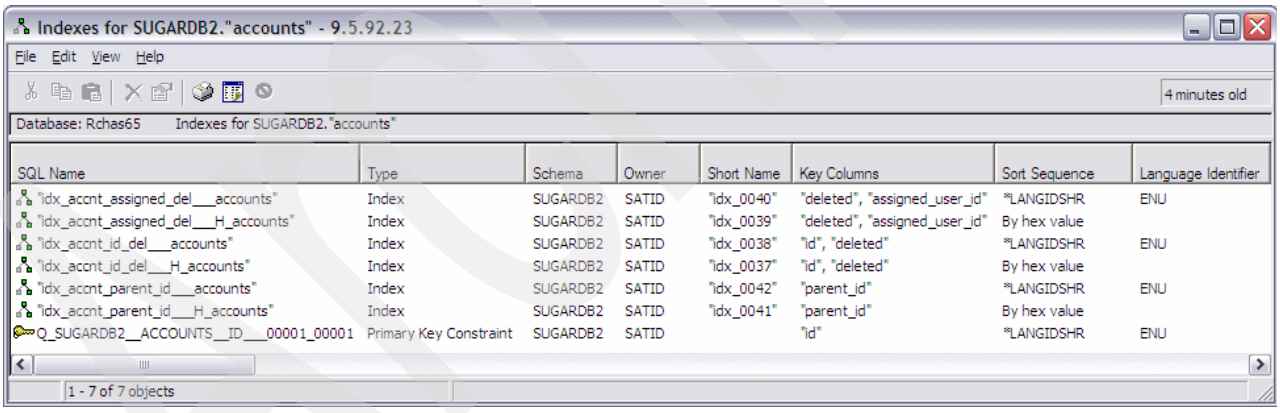


Figure 4-4 Indexes over the accounts table in SugarCRM, including the additional hex indexes

If you create indexes outside MySQL, those indexes will not be known by MySQL, so they cannot be used by the MySQL optimizer and will be deleted when you are altering a table or you are restoring the MySQL database.

4.1.4 Views

When SQL views are created in MySQL they are not populated to the DB2 database. The description is kept in the IFS and it will only be used by MySQL.

4.1.5 Journal and journal receivers

When the first IBMDB2I table is created in a MySQL schema, a corresponding SQL CREATE SCHEMA is issued in DB2 if the schema does not already exist. The CREATE SCHEMA creates the journal receiver and journal for the library.

If the journal name is QSQJRN, the first journal receiver is QSQJRN followed by a four-digit sequence number beginning with 0001, so QSQJRN0001 is the journal receiver name.

4.2 Accessing MySQL data

You can access the MySQL data in DB2 by using a variety of interfaces. This section discusses the various access methods.

4.2.1 Accessing MySQL data with DB2 tools

Generally, you can access the MySQL data with all known DB2 tools or programs on the System i. In certain situations, you have to implement workarounds primarily because of the upper and lowercase differences, which we described previously in this chapter.

Examples of accessing the MySQL data from several common DB2 methods are in:

- 4.2.2, “Accessing MySQL data from RPG using embedded SQL” on page 77
- 4.2.3, “Accessing MySQL data from RPG with native access” on page 78
- 4.2.4, “Accessing MySQL data from Query/400” on page 79

4.2.2 Accessing MySQL data from RPG using embedded SQL

You may use embedded SQL to access data from RPG programs. This approach is direct and has no workarounds. We show both how to use embedded SQL. You may also use native access, described in 4.2.3, “Accessing MySQL data from RPG with native access” on page 78.

Example 4-1 uses embedded SQL (which allows double quotation marks in file names) and is a suggested method.

Example 4-1 Using embedded SQL in RPG.

```
d recordCount     s              9
     c*
     c                   exsr      countRecords
     c                   dsply                   recordCount
     c                   eval      *inlr = *on
     c                   return
     c*---------------------------------------------------------------------
     c     countRecords  begsr
     c/exec sql
     c+ select count(*) into :recordCount from "accounts"
     c/end-exec
     c                   endsr
```

To compile use the following syntax:

```
CRTSQLRPGI OBJ(RPGCRMSQL) SRCMBR(RPGCRMSQL)
```

4.2.3 Accessing MySQL data from RPG with native access

When accessing data from any kind of RPG program, you may use native data access. In this case, use physical or logical files in your program. If you are using indexes generated by the MySQL engine, be aware of the possibility for name changes as a result of the way MySQL works.

Remember that most of the PHP application likely uses certain kinds of large objects such as LOB, CLOB, or GRAPHICS, which means that the table cannot be accessed through native access, but only through embedded SQL in the RPG programs.

To read a file in the IBMDB2I Storage Engine by using native file access:

1. Duplicate the double-quotation version of the file to a temporary file. RPG does not support double quotation marks of file names in the f-specs. In the temporary file, remove the double quotation marks:

   ```
   CRTDUPOBJ OBJ(mySQLFile) FROMLIB(mySQLSchema) OBJTYPE(*FILE) NEWOBJ(ovrDbFile)
   ```

2. Compile the RPG program using the duplicated object as the file template:

   ```
   CRTBNDRPG PGM(rpgPgmName) SRCMBR(rpgSrcMemberName)
   ```

3. After the RPG program is successfully compiled, you may permanently remove the file template used for compilation purposes:

   ```
   DLTF FILE(ovrDbFile)
   ```

4. Prior to running the RPG program, an OVRDBF command must be executed in the session or job to point to the mySQL file template that the RPG was compiled against.

   ```
   OVRDBF FILE(ovrDbFile) TOFILE(mySQLFile)
   ```

5. Execute the RPG program to open the MySQL file rather than the file template. This file is required because RPG is not able to accept file names that have quotation marks in the file specifications.

   ```
   CALL PGM(rpgPgmName)
   ```

6. Remove the override database file after programming execution:

   ```
   DLTOVR FILE(ovrDbFile)
   ```

Example 4-2 on page 79 shows the F and C spec of the RPG program.

Example 4-2 R and C spec of RPG program.

```
fovrDbFile if   e             disk
     f
     d recordCount     s              9  0
      *
     c                   exsr      countRecords
     C                   dsply                   recordCount
     c                   eval      *inlr = *on
     c                   return
      *
      *----------------------------------------------------------------------
     c     countRecords  begsr
      *
     c                   dow       not %eof(ovrDbFile)
     c                   read      ovrDbFile
     c                   if        not %eof(ovrDbFile)
     c                   eval      recordCount = recordCount + 1
     c                   endif
     c                   enddo
      *
     c                   endsr
```

Although this method is straightforward for the RPG programmer, is not as clean as the use of embedded SQL.

4.2.4 Accessing MySQL data from Query/400

You may use Query/400 to access the MySQL data. Always use the system names and be aware that certain data types are not supported in the Query/400.

Figure 4-5 on page 80 shows an example from Query/400 that uses the tables from MySQL. Although warning messages do occur because of unsupported supported character sets, you can still access the data.

```
Specify File Selections

Type choices, press Enter.  Press F9 to specify an additional
  file selection.

  File . . . . . .   "accounts"     Name, F4 for list
    Library  . . . . .    SUGARDB2     Name, *LIBL, F4 for list
  Member . . . . . .   *FIRST         Name, *FIRST, F4 for list
  Format . . . . . .   "accounts"     Name, *FIRST, F4 for list

F3=Exit           F4=Prompt         F5=Report           F9=Add file
F12=Cancel        F13=Layout        F24=More keys
File "accounts" in SUGARDB2 may have DBCS data or text.
```

Figure 4-5 File selection in Query/400

Figure 4-6 shows the Field selection in Query/400. Although using this method is possible, it is not as easy to read because of the Field column's long names in the table.

```
Select and Sequence Fields

Type sequence number (0-9999) for the names of up to 500 fields to
  appear in the report, press Enter.

Seq   Field           Text                                    Len  Dec
      ID___00001                                             36   G
      NAME_00001                                             150  GV
      DATE_00001                                             26   Z
      DATE_00002                                             26   Z
      MODIF00001                                             36   G
      CREAT00001                                             36   G
      DESCR00001                                             *****  3L
      DELET00001                                             4    0
      ASSIG00001                                             36   G
      ACCOU00001                                             50   GV
      INDUS00001                                             50   GV
      ANNUA00001                                             25   GV
```

Figure 4-6 Field selection in Query/400

Any use of Query/400 can be easy if you control the naming of the tables in the MySQL database by using only uppercase names that do not exceed 10 characters.

4.2.5 Use of Copy File

Copy File (CPYF) cannot be used when copying to the same library if the system name is not equal to the SQL name because CPYF only changes the system name. You cannot have two tables with the same SQL name in a library.

4.2.6 Updating MySQL data from CL commands

By using CL commands, you may also update the DB2 data. By running the Clear Physical File Member (CLRPFM) CL command, all the rows in a table are deleted. This deletion is fine and does not negatively affect the MySQL database structure.

4.3 DB2 updates of objects

Data changes to the MySQL database can be done from the traditional applications and tools running on the System i. Be aware that changes to the objects such as tables and indexes are not replicated back to the MySQL database.

4.3.1 Renaming tables

You should not rename tables created by MySQL through DB2 because the MySQL description will not be updated. If the tables are renamed they cannot be accessed by MySQL anymore. If you rename the table back to the original name, MySQL can access the data again.

4.3.2 Altering tables

You should not use DB2 interfaces to alter tables created by MySQL because the MySQL description does not reflect the change. If a table is altered in this way, it may become unusable by MySQL. If a table becomes unusable, it can be reversed by undoing the alteration and then deleting the corresponding FID file, as described in 2.3.4, “MySQL metadata files when using IBMDB2I” on page 11.

4.3.3 Deleting tables

Similar to altering tables, deleting a DB2 table results in discrepancies between the MySQL description and the actual tables in DB2. After you drop a table in DB2, you should also drop the table in MySQL to avoid having definitions in MySQL of tables that do not exist.

4.3.4 Indexes

MySQL accepts indexes you create in DB2. However, remember that MySQL might delete them if you alter a table. If you require additional indexes, create them in MySQL by using the option ibmdb2i_create_index_option in the configuration options for activated IBMDB2I so that you have the correct sorting sequence for your DB2 indexes.

Keep track of any other indexes you create so they can be re-created when you change any of the table objects, or when you restore the MySQL database.

4.3.5 Constraints

Any constraints you add through DB2 are enforced. MySQL is not able to violate the constraints, but does return a MySQL error message when it happens.

Similar as the indexes the constraints are dropped if the MySQL table is altered.

4.3.6 Triggers

You can add triggers to the DB2 tables, and the trigger programs will run when the conditions are fulfilled.

Again you should take care of any modification of the table object from MySQL because the triggers are dropped during the process.

5

Configuration options and variables

In this chapter, we discuss the use startup and system options (variables) that are available specific to the IBMDB2I Storage Engine. This information can help you determine which options are useful to you. This chapter also discusses usage details of these options.

This chapter contains the following topics:

5.1 IBMDB2I Storage Engine startup options and system variables

The configuration of the MySQL server and the interaction of the MySQL server with the selected storage engine are controlled by many system variables created by the owners of the storage engines. The default values for these system variables are set in the startup option file `my.cnf` (located in the `/etc` directory path by default), but can be overridden at server startup by using command line options, or (in most cases) can be changed dynamically while the server is running by using the MySQL SET statement. In addition to the global scope of these options and variables, many can be modified at a session level at any time, while the session is active.

With the addition of the IBMDB2I Storage Engine, several options and variables are created for you to use to regulate the interaction between the IBMDB2I Storage Engine and the DB2 for i relational database itself. These options and variables do not control the communication between the MySQL server and the IBMDB2I Storage Engine. Instead, they control the interaction between the IBMDB2I Storage Engine and the DB2 for i relational database.

Before exploring the system variables that are available with the IBMDB2I Storage Engine, an existing important startup option is the default_storage_engine property. MySQL can use the IBMDB2I Storage Engine as its default storage engine if you set the default_storage_engine option to the value of ibmdb2i in the `my.cnf` MySQL startup option file, or it can be overridden each time the MySQL server starts up by using the command line overrides:

```
mysqld_safe --default-storage-engine=ibmdb2i &
```

5.2 Summary of options

Table 5-1 lists descriptions of the IBMDB2I Storage Engine specific options. For details, see 5.3, “Details of options” on page 85.

Table 5-1 Summary of IBMDB2I Storage Engine options

Options	Description
ibmdb2i_assume_exclusive_use	Informs MySQL that external interfaces (such as DB2 for i) could alter the underlying data within the IBMDB2I Storage Engine
ibmdb2i_async_enabled	Controls the buffering performed by the QSQSRVR SQL Server Mode jobs when retrieving or modifying data from DB2 for i at the request of the IBMDB2I Storage Engine
ibmdb2i_compat_opt_time_as_duration	Controls the format in which MySQL TIME data types are stored in DB2 for i when IBMDB2I Storage Engine is used
ibmdb2i_rdb_name	Represents name of a local DB2 for i relational database that acts as a container for the content that is maintained by the IBMDB2I Storage Engine handler, which includes the option of using independent auxiliary storage pool (ASP)

Options	Description
ibmdb2i_lob_alloc_size	Represents initial size in memory allocated by IBMDB2I Storage Engine before working with any LOB column
ibmdb2i_max_read_buffer_size	Represents the maximum read buffer blocking size that is used in communication between the IBMDB2I Storage Engine and DB2 for i
ibmdb2i_max_write_buffer_size	Represents the maximum write buffer blocking size that is used in communication between the IBMDB2I Storage Engine and DB2 for i
ibmdb2i_transaction_unsafe	Controls the transaction isolation level that is used between the IBMDB2I Storage Engine and DB2 for i
ibmdb2i_compat_opt_blob_cols	Controls how a MySQL TEXT column is mapped to DB2 for i data type CLOB, DBCLOB, and VARCHAR
ibmdb2i_create_index_option	Controls whether additional indexes with *HEX sorting are created with the MySQL ASCII indexes
ibmdb2i_system_trace_level	Controls the level of debugging information to be gathered for QSQSRVR jobs servicing new MySQL connections
ibmdb2i_compat_opt_allow_zero_date_vals	Controls whether the IBMDB2I Storage Engine uses substitute values to support 0000-00-00 date components of DATE, DATETIME, and TIMESTAMP columns
ibmdb2i_propagate_default_col_vals	Controls whether DEFAULT value associated with each column should be propagated to the DB2 definition of the table
ibmdb2i_compat_opt_year_as_int	Controls the format in which MySQL YEAR data types are stored in DB2 for i when the IBMDB2I Storage Engine is used

5.3 Details of options

This section provides detailed descriptions of the IBMDB2I Storage Engine options.

5.3.1 ibmdb2i_assume_exclusive_use

Details include:

- Default value: 0
- Allowable values: 0 = No; 1= Yes
- Scope of option: GLOBAL

The ibmdb2i_assume_exclusive_use option informs MySQL that external interfaces (such as DB2 for i) could alter the underlying data within the IBMDB2I Storage Engine. When the value is set to 1, MySQL assumes no other interface can modify the data and reduces the overhead required to obtain optimization statistics, which could result in better MySQL performance. When the value is set to 0 (the default setting), MySQL is aware that other interfaces can alter the underlying data. The option is controlled at the global level only, but unlike other options

that are only globally scoped, ibmdb2i_assume_exclusive_use can be altered at any time while the server is up by using the following command:

```
mysql> SET GLOBAL ibmdb2i_assume_exclusive_use = <a value>;
```

5.3.2 ibmdb2i_async_enabled

Details include:

- Default value: 1
- Allowable values: 0 = No; 1 = Yes
- Scope of option: GLOBAL or SESSION

The ibmdb2i_async_enabled option controls the buffering performed by the QSQSRVR SQL Server Mode jobs when retrieving or modifying data from DB2 for i at the request of the IBMDB2I Storage Engine. When the option is set to 1 (the default setting), the QSQSRVR jobs asynchronously buffer the rows, which typically increases performance. The attributes that govern buffering (for both asynchronous and synchronous operations) can be tuned by adjusting the ibmdb2i_max_read_buffer_size and ibmdb2i_max_write_buffer_size system options. When the option is set to 0, the QSQSRVR jobs buffer the rows synchronously.

At any time, this option can be controlled at the global level or overridden at a session level by using one of the following commands:

```
mysql> SET GLOBAL ibmdb2i_async_enabled = <a value>;
mysql> SET SESSION ibmdb2i_async_enabled = <a value>;
```

5.3.3 ibmdb2i_compat_opt_time_as_duration

Details include:

- Default value: 0
- Allowable values: 0 = DB2 for i INTEGER data type; 1 = DB2 for i TIME data type
- Scope of option: GLOBAL or SESSION

The ibmdb2i_compat_opt_time_as_duration option controls the data type format in which the MySQL TIME data types are stored in DB2 for i when you use the IBMDB2I Storage Engine. The TIME data type in DB2 for i is stored as a three-part value (hour, minute, second) designating a time of day by using a 24-hour clock. MySQL stores this value as a duration in an integer format. Both DB2 for i and MySQL display this value in the same format, but the difference is in the internal storage of that value. When the option is set to 0 (the default setting), the IBMDB2I Storage Engine stores that value in DB2 for i as a traditional DB2 TIME data type, and makes the conversions necessary for MySQL when requests are made for that column. When the option is set to 1, the IBMDB2I Storage Engine stores MySQL TIME data types in the duration (integer) format.

The most significant difference for you when you use the different values for this option is the range of time duration that you can use. As an integer (value = 0), the TIME data type can represent a duration from -838:59:59 to 838:59:59 but as a traditional DB2 TIME format (value = 1), the range can only be from 00:00:00 to 23:59:59.

When you alter a MySQL table created with IBMDB2I that contains a column of MySQL data type TIME, the same setting for this option should be used as when the table was created. The reason is that the ALTER TABLE operation works by making a new copy of the table, copying the data from the original table to the new table, deleting the original table, and then renaming the new table to the original name. If you do not use the same setting of the option ibmdb2i_compat_opt_time_as_duration as was used when the table was created, data

mapping errors of the TIME column can occur when the data is copied, causing the ALTER TABLE operation to fail.

The ibmdb2i_compat_opt_time_as_duration option is used only when a table is either being created or altered. Ensure that the option has a consistent value through the lifetime of an application.

This option can be controlled at the global level or overridden at a session level at any time with the following commands:

```
mysql> SET GLOBAL ibmdb2i_compat_opt_time_as_duration = <a value>;
mysql> SET SESSION ibmdb2i_compat_opt_time_as_duration = <a value>;
```

5.3.4 ibmdb2i_rdb_name

Details include:

- Default value: None
- Allowable values: Name of any local relational database, including the database name assigned to an independent ASP (IASP)
- Scope of option: GLOBAL

The ibmdb2i_rdb_name option identifies the local DB2 for i relational database instance that acts as a container for the content maintained by the IBMDB2I Storage Engine handler in MySQL. The option can be set to any DB2 for i relational database, including independent auxiliary storage pools (IASP), while it is local to the server hosting the MySQL database.

The default value of none for this option means that IBMDB2I creates all the database objects for MySQL in the system database (often referred to as SYSBAS) in the system ASP (ASP 1).

Note: The ibmdb2i_rdb_name option can only be set at MySQL server startup by using the command line interface, or by changing the default settings in the `my.cnf` file. If the `my.cnf` file is altered, the MySQL server must be restarted for the changes to take effect.

5.3.5 ibmdb2i_lob_alloc_size

Details include:

- Default value: 2 MB
- Minimum values: 64 KB
- Maximum values: 128 MB
- Scope of option: GLOBAL or SESSION

The ibmdb2i_lob_alloc_size option controls the initial size in memory allocated by the IBMNDB2I engine before working with any LOB column. The allocation size value can be used to tune the memory footprint required by the IBMDB2I Storage Engine. Reducing this value lowers the amount of memory initially consumed, but if a LOB column requires additional memory, then the additional space must be re-allocated. If additional space is consistently re-allocated, the performance of the IBMDB2I Storage Engine can suffer.

This option can be controlled at the global level or overridden at a session level at any time with the following commands:

```
mysql> SET GLOBAL ibmdb2i_lob_alloc_size = <a value>;
mysql> SET SESSION ibmdb2i_lob_alloc_size = <a value>;
```

5.3.6 ibmdb2i_max_read_buffer_size

Details include:

- Default value: 1 MB
- Minimum values: 32 KB
- Maximum values: 16 MB
- Scope of option: GLOBAL or SESSION

The ibmdb2i_max_read_buffer_size option represents the maximum read-buffer blocking size that is used in communication between the IBMDB2I Storage Engine and DB2 for i. The read-buffer sizing between MySQL and the IBMDB2I Storage Engine is controlled by a separate standard MySQL variable named read_buffer_size. For optimum performance, ibmdb2i_max_read_buffer_size should be set to a value greater than or equal to read_buffer_size. This option controls buffering for both asynchronous and synchronous operations.

This option can be controlled at the global level or overridden at a session level at any time with the following commands:

```
mysql> SET GLOBAL ibmdb2i_max_read_buffer_size = <a value>;
mysql> SET SESSION ibmdb2i_max_read_buffer_size = <a value>;
```

5.3.7 ibmdb2i_max_write_buffer_size

Details include:

- Default value: 8 MB
- Minimum values: 32 KB
- Maximum values: 64 MB
- Scope of option: GLOBAL or SESSION

The ibmdb2i_max_write_buffer_size option represents the maximum write-buffer blocking size used in communication between the IBNMDB2I engine and DB2 for i. The write-buffer sizing between MySQL and the IBMDB2I Storage Engine is controlled by a separate standard MySQL variable, write_buffer_size. For optimum performance, ibmdb2i_max_write_buffer_size should be set to a value greater than or equal to write_buffer_size. This option controls buffering for both asynchronous and synchronous operations.

The following suggested practices can receive the greatest benefit of using this option:

- When the MySQL option write_buffer_size is increased
- Performing a large number of inserts as part of a single SQL statement. For example:

  ```
  mysql> insert into destinationTable (select * from sourceTable)
  ```

- MySQL Replication

This option can be controlled at the global level or overridden at a session level at any time with the following commands:

```
mysql> SET GLOBAL ibmdb2i_max_write_buffer_size = <a value>; or
mysql> SET SESSION ibmdb2i_max_write_buffer_size = <a value>;
```

5.3.8 ibmdb2i_transaction_unsafe

Details include:

- Default value: 0
- Allowable values: 0 = Use transaction isolation; 1 = No transaction isolation
- Scope of option: GLOBAL or SESSION

The ibmdb2i_transaction_unsafe option controls the transaction isolation level that is used between the IBMDB2I Storage Engine and DB2 for i. When the option is set to 1, the IBMDB2I Storage Engine uses no transaction isolation (much like the MyISAM storage engine) when interacting with DB2 for i, and performance can improve. This setting is ideal in a read-only environment but should be used with caution if updatable statements are processed. Consider that when the value of 1 is used, any isolation level specified for the transaction is ignored, even if you explicitly set the isolation level to a specific value.

When the option is set to 0 (the default setting), IBMDB2I honors the isolation level that you set for your transaction. To preserve transaction integrity for the updatable operations, you should set this option to 0 and specify a proper isolation level for that transaction. Repeatable-read (much like the InnoDB storage engine) is the default isolation level used by MySQL but you can change the transaction isolation level at any time.

In terms of transaction performance, the greatest performance improvement can happen when DB2 for i database journaling is also turned off on tables of your interest. Also consider that with journaling turned off, a connection with ibmdb2i_transaction_unsafe = 0 cannot access those tables.

This option can be controlled at the global level or overridden at a session level at any time with the following commands:

```
mysql> SET GLOBAL ibmdb2i_transaction_unsafe = <a value>;
mysql> SET SESSION ibmdb2i_transaction_unsafe = <a value>;
```

5.3.9 ibmdb2i_compat_opt_blob_cols

Details include:

- Default value: 0
- Allowable values: 0 = Mapped to DB2 for i CLOB/DBCLOB; 1 = Mapped to DB2 for i LONG VARCHAR/LONG VARGRAPHIC
- Scope of option: GLOBAL or SESSION

The ibmdb2i_compat_opt_blob_cols option controls whether a MySQL TEXT or BLOB column is mapped to DB2 for i CLOB/DBCLOB/BLOB (the default mapping) or LONG VARCHAR/LONG VARGRAPHIC/LONG VARBINARY data type when it is created with the IBMDB2I Storage Engine. This option applies to all sizes of MySQL TEXT and BLOB columns. The reason for this option is that MySQL supports an index over a TEXT/BLOB column but DB2 for i does not support an index over the default mapped CLOB/DBCLOB column. During a MySQL application installation by using the IBMDB2I Storage Engine, you might encounter an error similar to the following message:

```
"ERROR 1073 (42000): BLOB column '*unknown*' can't be used in key specification
with the used table type"
```

You may then attempt to retry the operation after you set ibmdb2i_compat_opt_blob_cols to a value of 1, when MySQL TEXT data type is mapped to LONG VARCHAR/LONG VARGRAPHIC, which can be used in an DB2 for i index.

Be aware that all MySQL TEXT columns will be sized to fit within the maximum DB2 for i row length of 32KB. This means that if you really expect anywhere near the full capacity of a MySQL TEXT column (of 64KB size) to be used, the data could be truncated.

The ibmdb2i_compat_opt_blob_cols option is used only when a table is either being created or altered. A suggestion is to ensure it has a consistent value through the lifetime of an application.

This option can be controlled at the global level or overridden at a session level at any time with the following commands:

```
mysql> SET GLOBAL ibmdb2i_compat_opt_blob_cols = <a value>;
mysql> SET SESSION ibmdb2i_compat_opt_blob_cols = <a value>;
```

5.3.10 ibmdb2i_create_index_option

Details include:

- Default value: 0
- Allowable values: 0 = No additional index created; 1 = When the created index is ASCII-based for MySQL, an additional index is created based on EBCDIC hexadecimal sorting for DB2 for i
- Scope of option: GLOBAL or SESSION

The ibmdb2i_create_index_option option controls whether an additional DB2 for i index with *HEX sorting is created for use by a DB2 for i application that also has to access a table created by MYSQL with IBMDB2I. This additional index is never used by MySQL and is only provided for the convenience of IBM i applications such as SQL, DB2 Web Query, native I/O, and others. Note that this additional DB2 for i index does not have a unique key constraints even if the original MySQL ASCII index does.

The additional index is named in DB2 for i in the following format (shown in Figure 5-1):

```
"<Indexname>__H_<tablename>"
```

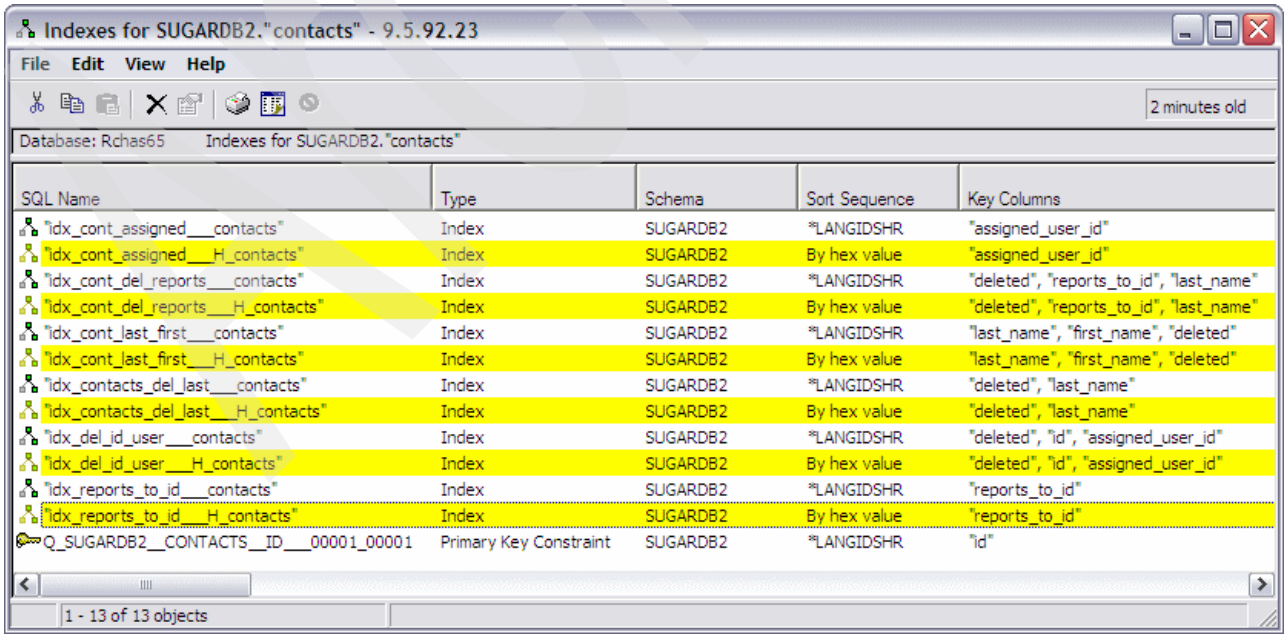


Figure 5-1 Additional DB2 for i indexes created by using the option ibmdb2i_create_index_option = 1

This feature is implemented to address a possibility that the original indexes created by IBMDB2I for use by MySQL can be unusable or inefficient for other IBM i applications that also have to access these MySQL tables. The reason is that the original indexes are in ASCII-based sorting order, which provides different data sorting results for general IBM i applications.

The ibmdb2i_create_index_option option is used only when a table is either being created or altered. Ensure that it has a consistent value throughout the lifetime of an application.

This option can be controlled at the global level or overridden at a session level at any time with the following commands

```
mysql> SET GLOBAL ibmdb2i_create_index_option = <a value>; or
mysql> SET SESSION ibmdb2i_create_index_option = <a value>;
```

5.3.11 ibmdb2i_system_trace_level

Details include:

- Default: 0
- Allowable values: 0-63
- Scope of option: global

This option specifies what kind of debugging information is to be gathered for QSQSRVR jobs that are servicing MySQL connections. Multiple sources of information may be specified by summing the respective values. Changes to this option only affect new connections. Valid values include:

- 0 = No information (Default)
- 2 = STRDBMON
- 4 = STRDBG
- 8 = DSPJOBLOG
- 16 = STRTRC
- 32 = PRTSQLINF

The most useful sources of information are DSPJOBLOG, which captures the job log for each QSQSRVR job in a spoolfile, and STRDBG, which increases the diagnostic information in each job log.

5.3.12 ibmdb2i_compat_opt_allow_zero_date_vals

Details include:

- Default: 0
- Allowable values: 0 = No substitution; 1 = A substitute value of '0001-01-01' is used
- Scope of option: global or session

This options specifies whether the storage engine should allow the '0000-00-00' date in DATETIME, TIMESTAMP, and DATE columns. When the option is 0, an attempt to insert a row containing this zero-date into an IBMDB2I table will fail. Also, a warning is generated when a column is created with this zero value as the default value. When this option is set to 1, the zero value is substituted with '0001-01-01' when stored in DB2, and a '0001-01-01' value is translated to '0000-00-00' when read from DB2. Similarly, when a column with a default zero value is created, the DB2 default value will be '0001-01-01'. Be aware that, when this option is 1, all values of '0001-01-01' in DB2 are interpreted as '0000-00-00'. This option is primarily added for compatibility with applications that rely on the zero date.

Unlike other compatibility options, this option has an effect both when creating and altering tables and when performing DML operations. As a result, the value of this option should remain consistent throughout the creation and usage of any table which uses a DATETIME, TIMESTAMP, or DATE field.

5.3.13 ibmdb2i_propagate_default_col_vals

Details include:

- Default: 1
- Allowable values: 0 = No, 1 = Yes
- Scope of option: global or session

This option controls whether the DEFAULT value associated with each column should be propagated to the DB2 definition of the table when a table is created or altered. This ensures that rows inserted from a standard DB2 interface will use the same default values as when inserted from MySQL.

5.3.14 ibmdb2i_compat_opt_year_as_int

Details include:

- Default: 0
- Allowable values: 0 = CHAR(4) CCSID 1208, 1 = SMALLINT
- Scope of option: global or session

This option controls how YEAR columns are stored in DB2. The default is 0 and causes YEAR columns to be created as CHAR(4) CCSID 1208 columns in DB2. Setting this option to 1 causes the YEAR columns to be created as SMALLINT columns. This provides a slight performance increase and enables indexes that combine a YEAR column with a character column.

The ibmdb2i_compat_opt_year_as_int option is used only when a table is being either created or altered. The option should have a consistent value through the lifetime of an application.

6

Transaction management and locking considerations

This chapter describes transaction management and locking considerations in using the MySQL Database Server for IBM i with the IBMDB2I Storage Engine. The chapter provides an introduction of transaction isolation levels that are supported by the IBMDB2I Storage Engine, settings, locking behavior of each transaction isolation level, and wait lock timeout settings. The also discusses how to start and end, or roll back the transaction.

This chapter contains the following topics:

6.1 MySQL transaction management and IBMDB2I

Transaction management allows data integrity while tables are simultaneously accessed by multiple database connections. MySQL supports transaction management, but not all storage engines implement the underlying support.

The MyIASM storage engine does not support transaction management but instead relies on table-level locking for concurrency management. It locks the entire table when data is being updated, deleted, or inserted. The InnoDB storage engine supports transaction management by using either table-level locking or row-level locking. More information about transaction management and locking mechanism for these storage engines can be found in the MySQL reference manuals. Also see 1.2, “MySQL pluggable storage engine” on page 3.

The IBMDB2I Storage Engine can support transaction management using either table-level locking or row-level locking. Transaction management enables control on concurrency level and granularity of commitment cycle. Transaction support of the IBMDB2I Storage Engine is similar to the transaction support for other DB2 for IBM i operating system interfaces, however there are some differences in implementation. The following sections provide details about the transaction management supported by the IBMDB2I Storage Engine.

6.2 Transaction isolation level and locking

When you develop and run your application using the IBMDB2I Storage Engine with the MySQL Database Server on IBM i, consider what level of transaction management you need. This section discusses levels of transaction, and how to set and use them.

6.2.1 Transaction safe mode set by system variable for IBMDB2I

You can control use of transaction support with the IBMDB2I Storage Engine by the system variable ibmdb2i_transaction_unsafe.

Note the following information about the settings for transaction support:

- When the value is set to 0 (the default) you can use transaction support.
- When this system variable is set to 1, you cannot use transaction support. The variable behaves similarly to an isolation level *NONE on DB2 for i. At this time, each row is updated, inserted, and deleted on a table directly without issuing commit statements and cannot be rolled back. Under this condition, the locking mechanism works as table-level locking. When a row on a table is updated or deleted, the table is locked once and released after the operation. When a row is inserted on a table, it can be done without a table lock.

Note the following information about the settings for isolation level:

- When this system variable is set to 1, the autocommit and isolation level is ineffective.
- When this system variable set to 0 (the default), transaction support is used and you can set proper isolation level.

System variable ibmdb2i_transaction_unsafe can be set to GLOBAL and SESSION by the SET MySQL statement. For the system value, see 5.3.8, “ibmdb2i_transaction_unsafe” on page 89.

6.2.2 Transaction isolation level

The IBMDB2I Storage Engine supports four types of transaction isolation levels:

- SERIALIZABLE
- REPEATABLE READ
- READ COMMITTED
- READ UNCOMMITTED

In addition to these isolation levels, the *transaction unsafe* mode is available as described in 6.2.1, “Transaction safe mode set by system variable for IBMDB2I” on page 94.

Each isolation level is mapped to a DB2 for i isolation level, as listed in Table 6-1.

Table 6-1 Isolation level mapping

MySQL with IBMDB2I	Access intent	DB2 for i equivalent
SERIALIZABLE	Read-only (SELECT)	RR
	Updatable (INSERT,DELETE, UPDATE)	RR
REPEATABLE READ	Read-only (SELECT)	RS
	Updatable (INSERT,DELETE, UPDATE)	RS
READ COMMITTED	Read-only (SELECT)	CS
	Updatable (INSERT,DELETE, UPDATE)	RS
READ UNCOMMITTED	Read-only (SELECT)	UR
	Updatable (INSERT,DELETE, UPDATE)	RS
transaction_unsafe	Read-only (SELECT)	*NONE
	Updatable (INSERT,DELETE, UPDATE)	*NONE

Setting of transaction isolation level

By default, transaction isolation level is set to REPEATABLE READ on the MySQL Database Server on IBM i.

You can set the isolation level in several ways described in this section. It can be set as default isolation level for all connections in starting the MySQL Database Server on IBM i. This default isolation level can be changed dynamically in each connection and each transaction.

Option in a configuration file my.cnf

You can set the option in the [mysql] section in an option file `my.cnf`, for example:

```
transaction-isolation = {READ-UNCOMMITTED | READ-COMMITTED | REPEATABLE-READ |
SERIALIZABLE}
```

When the server is started with this option, the value is set in a system variable tx_isolation as GLOBAL, and used as the default transaction isolation level for all connections.

SET TRANSACTION statement

You can change the isolation level for a single session or for all new incoming connections with the MySQL statement SET TRANSACTION. The syntax is:

```
SET [SESSION | GLOBAL] TRANSACTION ISOLATION LEVEL {READ UNCOMMITTED | READ
COMMITTED | REPEATABLE READ | SERIALIZABLE}
```

If you use the GLOBAL keyword, the default isolation level is set to all new connections created after the point of this statement. If you use the SESSION keyword, the default isolation level is set to the session and used for all new transactions in the session.

Checking current default transaction isolation level

You can check the current global and session default transaction isolation level by showing the value of the tx_isolation system variable by using either of the following methods:

- Using SELECT:

```
SELECT @@global.tx_isolation;
SELECT @@tx_isolation;
```

- Using SHOW:

```
SHOW session variables like 'tx_isolation';
SHOW global variables like 'tx_isolation';
```

6.2.3 Isolation level and behavior of locking

Each isolation level works similarly to the mapped isolation level for other DB2 for IBM i interfaces. However differences exists in behavior that is based on whether an access plan uses an index. For example, when running under READ UNCOMMITTED, an UPDATE statement can only lock a subset of a table's rows if an index is present. In comparison, an UPDATE statement can lock all of the table's rows if no index is present because all rows are scanned for row selection.

SERIALIZABLE

Read-only(SELECT) access works as follows:

- Table is locked until COMMIT or ROLLBACK occurs. No rows can be locked for update by other connections during this transaction. Other connections can read only this table.
- Table cannot be opened when the table was previously opened for update by another connection, until this connection ends.

Updatable(INSERT,DELETE, UPDATE) access works as follows:

- Table is locked until COMMIT or ROLLBACK occurs. No rows can be locked for update by other connections during this transaction. Other connections with a lower isolation level than REPEATABLE READ can only read this table. Other SERIALIZABLE connections have to wait to open the table until the former connection ends.
- Table cannot be opened when the table was previously opened for update by another connection until this connection ends.

In this mode, use of the index for processing the statement makes no difference.

REPEATABLE READ

Read-only(SELECT) access works as follows:

- Selected rows are locked until COMMIT or ROLLBACK occurs. These rows cannot be locked for update by other connections during this transaction. Other connections only can read these rows.
- Any row changed (or a row that is currently locked with an UPDATE row lock) by another connection cannot be read until it is committed.

Updatable(INSERT,DELETE, UPDATE) access works as follows:

- Rows being updated or inserted are locked until COMMIT or ROLLBACK occurs. These rows cannot be locked even for reading by other connections during this transaction. Other connections with a lower isolation level than REPEATABLE READ can only read the other rows in the table. Other SERIALIZABLE connections have to wait to open the table until the former connection ends.
- Rows being deleted cannot be selected by other connections unless the row is rolled back.
- Any row changed (or a row that is currently locked with an UPDATE row lock) by another connection cannot be locked for update until the row change is committed.

If a statement is processed without an index for row selection, all rows in the table are locked and other connections above READ COMMITTED cannot even read any rows until COMMIT or ROLLBACK.

Unlike SERIALIZABLE, REPEATABLE READ gets locks only on rows unless a table scan is performed. As a result, if a second connection inserted rows into the table after the first connection selected a set of rows, the first connection might see these added rows when it issues the next select statement.

READ COMMITTED

Read-only(SELECT) access works as follows:

- Each row is locked when read and released. Because the locks are not kept on a row, other connections can get locks on rows for update.
- Any row changed (or a row that is currently locked with an UPDATE row lock) by another connection cannot be read until it is committed.

Updatable(INSERT,DELETE, UPDATE) access works as follows:

- Rows being updated or inserted are locked until COMMIT or ROLLBACK occurs. These rows cannot be locked even for reading by other connections during this transaction. Other connections below REPEATABLE READ only can read the other rows in the table. Other SERIALIZABLE connections have to wait to open the table until the former connection ends.
- Rows being deleted cannot be selected by other connections unless it is rolled back.
- Any row changed (or a row that is currently locked with an UPDATE row lock) by another connection cannot be locked for update until it is committed.

If a statement is processed without index for row selection, lock behaviors will change. For updatable accesses, all rows in a table are locked and other connections above READ COMMITTED cannot read any rows until COMMIT or ROLLBACK. For read-only access, all rows are locked once and released and they can be locked for update by other connections. However, when a row is changed by another connection, the entire table cannot be read until it is committed.

READ UNCOMMITTED

Read-only(SELECT) access works as follows:

- Rows that are read with the SELECT statement are not locked for read. Other connections can get locks on rows for update.
- Any row changed (or a row that is currently locked with an UPDATE row lock) by another connection can be read even if the change has not been committed.

Updatable(INSERT, DELETE, UPDATE) access works as follows:

- Rows being updated or inserted are locked until COMMIT or ROLLBACK occurs. These rows cannot be locked by other connections during this transaction. Other connections below REPEATABLE READ only can read the other rows in the table. Other SERIALIZABLE connections have to wait to open the table until the former connection ends.
- Rows being deleted cannot be selected by other connections unless the row is rolled back.
- Any row changed (or a row that is currently locked with an UPDATE row lock) by another connection cannot be locked for update until it is committed.

If a statement is processed without index for row selection, lock behaviors will change. For updatable accesses, all rows in a table are locked and other connections above READ COMMITTED cannot read any rows until COMMIT or ROLLBACK. For read-only access, rows are not locked and they can be locked for update by other connections.

Transaction unsafe

Read-only(SELECT) access works as follows:

- Row is not locked for read. Other connections can get locks on rows for update.
- Any row changed (or a row that is currently locked with an UPDATE row lock) by another connection can be read even if the change has not been committed.

Updatable(INSERT,DELETE, UPDATE) access works as follows:

- Table is locked once and released after updatable operation. Other connections below REPEATABLE READ can update any rows in the table. Other SERIALIZABLE connections have to wait to open this table until the end of this transaction_unsafe session.
- Rows being deleted cannot be selected by other connections because they have been deleted already.
- Table cannot be opened when the table was once opened for update by another connection until the connection ends.

Use of index for processing the statement makes no difference under this mode.

Table 6-2 on page 99 is a summary of locking behavior when index is used for row selection.

Table 6-2 Isolation level and lock behavior with index used for row selection

Isolation level	Read only access		Updatable access	
	Lock until commit/rollback	**Access to uncommitted row**	**Lock until commit/rollback**	**Access to uncommitted row**
SERIALIZABLE	Table locked and cannot be locked by others Any row cannot be locked for update	Table cannot be opened until holding session is closed	Table locked and cannot be locked by others Any row cannot be locked for update	Table cannot be opened until holding session is closed
REPEATABLE READ	Row locked for read	Cannot be read until it is committed	Row locked for update	Cannot be locked for read and update until it is committed
READ COMMITTED	Row locked for read and released	Cannot be read until it is committed	Row locked for update	Cannot be locked for read and update until it is committed
READ UNCOMMITTED	No locks	Can be read before it is committed	Row locked for update	Cannot be locked for read and update until it is committed
transaction_unsafe	No locks	Can be read before it is committed	Table locked and released	Table cannot be opened until holding session is closed

Table 6-3 is a summary of locking behavior when index is not used for row selection.

Table 6-3 Isolation level and lock behavior without index for row selection

Isolation level	Read only access		Updatable access	
	Lock until commit/rollback	**Access to uncommitted row**	**Lock until commit/rollback**	**Access to uncommitted row**
SERIALIZABLE	Table locked and cannot be locked by others Any row cannot be locked for update	Table cannot be opened until holding session is closed	Table locked and cannot be locked by others Any row cannot be locked for update	Table cannot be opened until holding session is closed
REPEATABLE READ	All rows locked and cannot be locked for update by others	All rows cannot be locked until it is committed	All rows locked and cannot be locked for update by others	All rows cannot be locked until it is committed
READ COMMITTED	All rows locked and released	All rows cannot be locked until it is committed	All rows locked and cannot be locked for update by others	All rows cannot be locked until it is committed
READ UNCOMMITTED	No locks	Can read before it is committed	All rows locked and cannot be locked for update by others	All rows cannot be locked until it is committed
transaction_unsafe	No locks	Can read before it is committed	Table locked for read and released	Table cannot be opened until holding session is closed

Importance of indexes

As you can see in Table 6-3 on page 99, if an index is not used for row selection, many rows can be locked unnecessarily, which causes high frequency of lock contention among multiple connections. An adequate indexing strategy on DB2 for i is still important even though it is the optimizer of MySQL Database Server that decides to use an index for row selection. This approach can help reduce lock contentions and contribute to better performance.

6.2.4 Lock wait timeout

When a row or table has been locked by any other connection, an attempt to acquire the lock must wait. If the row or table has not been released by the other connection in a certain period of time, the attempt fails. The IBMDB2I Storage Engine also has this timeout setting. However, if you are familiar with the native DB2 for i, you know that several differences exist between the native DB2 for i and IBMDB2I.

Table lock timeout

When table locks conflict by multiple MySQL connections, the default file timeout value is used for wait time expiration. The IBMDB2I Storage Engine also uses the WAITFILE parameter of table object on IBM i. The default file timeout also is applied when a lock holder is native IBM i job and the waiting job is a MySQL connection; the converse is also true.

Row lock timeout

When row locks conflict by multiple MySQL connections, the default row lock timeout is used, for wait time expiration. The IBMDB2I Storage Engine also uses the WAITRCD parameter of table object on IBM i. This also is applied when a lock holder is native IBM i job and a waiting job is a MySQL connection; the converse is also true.

Timeout of LOCK TABLES statement

When you use the LOCK TABLES statement of MySQL to explicitly acquire a lock on a table of the IBMDB2I, there is no timeout. This situation means if the table has been locked by any other connection, the attempt of LOCK TABLES waits until the table is released no matter how long it is. It also means that when a table is locked by one connection that is using this statement, other connections will wait until the holder releases the table with the UNLOCK TABLES statement or ending connection.

When a waiting job for the LOCK TABLES statement of MySQL is a native IBM i job, then timeout occurs based on the WAITFILE parameter or the WAITRCD parameter in the table object. In reverse, when a native IBM i job issued a LOCK TABLE statement on DB2 i, timeout occurs on the MySQL connection side that uses the same parameters.

6.3 Starting transaction, commit, and rollback

This section discusses the support of transactions such as commit and rollback with the IBMDB2I Storage Engine.

6.3.1 Autocommit

By default, MySQL starts new connections with autocommit enabled, and it works with the IBMDB2I Storage Engine. If the autocommit is enabled, each SQL statement works as it does in a single transaction because commit is issued automatically each time. An updated, inserted, or deleted row cannot be rolled back because it committed already. Although this

behavior is similar to when the ibmdb2i_transaction_unsafe option is set to 1, the difference is in whether it issues a commit or not. When ibmdb2i_transaction_unsafe is set to 1, autocommit setting has no effect.

You can disable autocommit by setting AUTOCOMMIT system variable to 0:

```
SET [GLOBAL | SESSION] AUTOCOMMIT = 0
```

When GLOBAL is specified, every new session started after this change has been made will have this value. SESSION makes this change effective only in that session.

When the autocommit is disabled, COMMIT or ROLLBACK has to be issued explicitly. When a session is closed without issuing COMMIT, the last transaction is rolled back.

6.3.2 Start of transaction boundary

When the transaction support is enabled with the IBMDB2I Storage Engine, which means that the system value of ibmdb2i_transaction_unsafe is set to 0, every SQL statement is processed under commitment control of IBM i. Therefore, commitment control is automatically started at the first statement in a session. And it ends at the end of the session automatically. When the autocommit is disabled, COMMIT or ROLLBACK has to be issued explicitly. If a session is closed without issuing COMMIT, the last transaction is rolled back.

Transaction boundary is automatically started by any SQL statement which has to be committed, and it is ended by issuing COMMIT or ROLLBACK statement.

When the autocommit is enabled in a session, transaction ends by implicit COMMIT on each statement. However, you can use START TRANSACTION statement to start a transaction boundary with having autocommit disabled. Even if the autocommit is enabled in a session, START TRANSACTION disables it until next COMMIT or ROLLBACK is issued. After the transaction has been ended by COMMIT or ROLLBACK, autocommit mode reverts to the previous state.

6.3.3 Statements that cause an implicit commit and cannot be rolled back

Certain MySQL statements implicitly issue COMMIT and cannot be rolled back. In general, these include data definition language (DDL) statements, such as those that create or drop databases, and those that create, drop, or alter tables or stored routines. The *MySQL 5.1 Reference Manual* suggests that users should design their transactions not to include such statements. If one of these statements is issued early in a transaction that cannot be rolled back, and then another statement later fails, the full effect of the transaction cannot be rolled back by issuing a ROLLBACK statement. A list of these statements is in the *MySQL 5.1 Reference Manual*:

http://dev.mysql.com/doc/refman/5.1/en/

6.3.4 SAVEPOINT and ROLLBACK TO SAVEPOINT statement

The IBMDB2I Storage Engine supports the SQL statement SAVEPOINT and ROLLBACK TO SAVEPOINT.

The SAVEPOINT statement sets a savepoint in a transaction with a name as an identifier. Changes made to data in a table can be rolled back to the savepoint in the transaction.

The ROLLBACK TO SAVEPOINT statement rolls back a transaction to the savepoint specifying the name of it.

The RELEASE SAVEPOINT statement releases the named savepoint and any subsequently established savepoints in the transaction. When a savepoint is released, ROLLBACK TO SAVEPOINT cannot be performed using that savepoint. Roll back can only be available to savepoints not released.

Refer to the *MySQL 5.1 Reference Manual* for syntax of these statements.

6.3.5 XA transaction

Support for XA transactions is not available on the IBMDB2I Storage Engine at this time.

7

Backup and restore considerations of the MySQL databases

In this chapter, we describe methods that are available for backup and restore of the MySQL databases and special considerations when backing up MySQL databases that are created with the IBMDB2I Storage Engine and shared with native IBM i access. These methods include using a command line and GUI tools. In this chapter, we discuss all of the common tasks that are related to the backup and restore tools.

This chapter contains the following topics:

7.1 Methods for backup and restore

You can perform backup and restore of the MySQL Database Server by using a variety of ways, including the following methods:

- Command line by using the i5/OS PASE runtime environment:
  - mysqldump
  - mysqlhotcopy
  - mysqlimport
  - source
- MySQL Administrator (GUI)

  See *Discovering MySQL in IBM i5/OS*, SG24-7398 for installation and other related tasks.
- phpMyAdmin (GUI)

  Firefox, Netscape, or Microsoft Internet Explorer® is required for this method. See *Discovering MySQL in IBM i5/OS*, SG24-7398 for installation and other related tasks.
- A copy to tape or save file to disk, after a database backup is made to disk

When MySQL uses the IBMDB2I Storage Engine to create its databases as DB2 for i schemas, tables, indexes, and primary key constraints, there is an additional factor for you to consider, which is DB2 for i object-based authority and is a factor of which MySQL environment has no awareness. Proper assignment of the object-based authority is necessary to allow other IBM i applications to access DB2 for i schemas and tables created by MySQL with IBMDB2I for interoperability. Also, you may have to save these assigned authorities so that the restored MySQL data can still be accessible from IBM i applications.

7.2 Making a backup of the MySQL Database Server

In the i5/OS PASE runtime environment, you can use one of several tools described in this section to make a backup. These tools also work with MySQL databases created by the IBMDB2I Storage Engine and used *exclusively* by MySQL environment running in IBM i PASE. But if you share MySQL databases with other native IBM i applications (in a proper manner that would not create incompatible interference to MySQL environment), these tools may not be proper choices because they are not aware of IBM i object-based authorities assigned to MySQL objects after they are created by IBMDB2I. Also, these tools do not actually save the database objects in the same manner as the IBM i SAVXXX commands do and thus would not save the object security information. We discuss this consideration in 7.3, “Saving MySQL databases shared with IBM i applications” on page 121 after the discussion of the following backup tools.

7.2.1 The mysqldump script for backup

The mysqldump client is a backup script. You can use this program to dump a database or a collection of databases for backup or transfer to another SQL server that is not necessarily a MySQL Database Server. The dump typically contains SQL statements to create a table, populate it, or both. The mysqldump program can also be used to generate comma-separated value (CSV) files, other delimited text, or XML format.

If you are doing a backup on the server and your tables all are MyISAM tables, consider using the mysqlhotcopy utility instead because it can accomplish faster backups and restores.

The three general ways to invoke **mysqldump** are:

- For a backup of only one database:

  ```
  mysqldump [options] db_name [tables]
  ```

- For a backup of more than one database:

  ```
  mysqldump [options] --databases db_name1 [db_name2 db_name3...]
  ```

- For a backup of all databases:

  ```
  mysqldump [options] --all-databases
  ```

If you do not name any tables following db_name or if you use either of the following options, entire databases are dumped:

- `--databases`
- `--all-databases`

To obtain a list of the options that your version of mysqldump supports, use:

```
mysqldump --help
```

Certain mysqldump options are shorthand for groups of other options. The options `--opt` and `--compact` fall into this category. For example, use of `--opt` is the same as specifying the following options:

```
--add-drop-table --add-locks --create-options --disable-keys --extended-insert
--lock-tables --quick --set-charset
```

Keep in mind that all of the options that `--opt` stands for, also are on by default because `--opt` is on by default.

To reverse the effect of a group option, use the `--skip-xxx` form:

```
--skip-opt
--skip-compact
```

You may also select only part of the effect of a group option by following it with options that enable or disable specific features. Consider the following examples:

- To select the effect of `--opt` except for some features, use the `--skip` option for each feature. For example, to disable extended inserts and memory buffering, use:

  ```
  --opt --skip-extended-insert --skip-quick
  ```

  As of MySQL 5.0, `--skip-extended-insert --skip-quick` is sufficient because `--opt` is on by default.

- To reverse `--opt` for all features, except index disabling and table locking, use:

  ```
  --skip-opt --disable-keys --lock-tables
  ```

When you selectively enable or disable the effect of a group option, order is important because options are processed first to last. For instance, the following example does not have the intended effect and is the same as `--skip-opt` by itself:

```
--disable-keys --lock-tables --skip-opt
```

The mysqldump script can retrieve and dump table contents row by row, or it can retrieve all the contents from a table and buffer it in memory before dumping it. Buffering in memory can be a problem if you are dumping large tables. To dump tables row by row, use the `--quick` option (or `--opt`, which enables `--quick`). The option `--opt` (and therefore `--quick`) is enabled by default as of MySQL 5.0. To enable memory buffering, use `--skip-quick`.

If you are using a recent version of mysqldump to generate a dump to be reloaded into an old MySQL Database Server, do not use the `--opt` or `--extended-insert` option. Use `--skip-opt` instead.

Table 7-1 lists some of the options that the mysqldump script supports.

Table 7-1 mysqldump options

Option	Description
--add-drop-database	Adds a DROP DATABASE statement before each CREATE DATABASE statement.
--add-drop-table	Adds a DROP TABLE statement before each CREATE TABLE statement.
--all-databases, -A	Dumps all tables in all databases. This is the same as using the `--databases` option and naming all databases on the command line.
--comments, -i	Writes additional information in the dump file such as program version, server version, and host. This option is enabled by default. To suppress this additional information, use `--skip-comments`.
--complete-insert, -c	Uses complete INSERT statements that include column names.
--create-options	Include all MySQL-specific table options in the CREATE TABLE statements.
--databases, -B	Dumps several databases. Normally, mysqldump treats the first name argument on the command line as a database name and following names as table names. With this option, it treats all name arguments as database names. CREATE DATABASE and USE statements are included in the output before each new database.
--disable-keys, -K	For each table, surrounds INSERT statements with /*!40000 ALTER TABLE tbl_name DISABLE KEYS */; and /*!40000 ALTER TABLE tbl_name ENABLE KEYS */; statements. Makes loading the dump file faster because the indexes are created after all rows are inserted. This option is effective only for non-unique indexes of MyISAM tables.
--lock-all-tables, -x	Locks all tables across all databases by acquiring a global read lock for the duration of the whole dump. This option automatically turns off `--single-transaction` and `--lock-tables`.
--lock-tables, -l	Locks all tables before dumping them. The tables are locked with READ LOCAL to allow concurrent inserts in the case of MyISAM tables. For transactional tables, such as InnoDB and BDB, `--single-transaction` is a much better option, because it does not need to lock the tables at all. Note that when dumping multiple databases, `--lock-tables` locks tables for each database separately. Therefore, this option does not guarantee that the tables in the dump file are logically consistent between databases. Tables in different databases may be dumped in completely different states.
--no-create-info, -t	Indicates not to write CREATE TABLE statements that recreate each dumped table.
--no-data, -d	Indicates not to write any table row information. That is, do not dump table contents. This is useful if you want to dump only the CREATE TABLE statement for the table.

Option	Description
--opt	Is a short form for specifying `--add-drop-table --add-locks --create-options --disable-keys --extended-insert --lock-tables --quick --set-charset`. It provides a fast dump operation and produce a dump file that can be reloaded into a MySQL Database Server quickly. The `--opt` option is enabled by default. Use `--skip-opt` to disable it. See the discussion at the beginning of this section for information about selectively enabling or disabling certain options that are affected by `--opt`.
--order-by-primary	Sorts each table's rows by its primary key, or by its first unique index, if such an index exists. This is useful when dumping a MyISAM table to be loaded into an InnoDB table, but will make the dump itself take considerably longer.
--password[=password], -p[password]	Represents the password to use when connecting to the server. If you use the short option form (-p), you cannot have a space between option and password. If you omit the password value following the `--password` or `-p` option on the command line, you are prompted for one.
--result-file=file, -r file	Directs output to a given file. Use this option on Windows to prevent new line '\n' characters from being converted to '\r\n' carriage return or new line sequences. The result file is created and its contents are overwritten, even if an error occurs while generating the dump. The previous contents are lost.
--tab=path, -T path	Produces tab-separated data files. For each dumped table, **mysqldump** creates a tbl_name.sql file that contains the CREATE TABLE statement that creates the table, and a tbl_name.txt file that contains its data. The option value is the directory in which to write the files. By default, the .txt data files are formatted by using tab characters between column values and a new line at the end of each line. The format can be specified explicitly by using the `--fields-xxx` and `--lines-terminated-by` options. Use this option only when **mysqldump** is run on the same machine as the mysqld server. You must have the FILE privilege, and the server must have permission to write files in the directory that you specify.
--tables	Overrides the `--databases` or `-B` option. mysqldump regards all name arguments following the option as table names.
--triggers	Represents dump triggers for each dumped table. This option is enabled by default. You can disable it by using `--skip-triggers`. This option was added in MySQL 5.0.11. Before that, triggers were not dumped.
--user=user_name, -u user_name	Represents the MySQL user name to use when connecting to the server.
--where='where_condition', -w 'where_condition'	Dumps only rows that are selected by the given WHERE condition. Quotation marks around the condition are mandatory if it contains spaces or other characters that are special to your command interpreter.
--xml, -X	Writes dump output as well-formed XML.

To back up a schema or database by using **mysqldump**:

1. Sign on to i5/OS and execute the QP2TERM program to start the i5/OS PASE environment:

   ```
   CALL QP2TERM
   ```

2. In the i5/OS PASE Terminal Console, change to the MySQL commands directory:

   ```
   cd /QOpenSys/usr/local/mysql/mysql/bin
   ```

3. Verify that you are in the correct directory:

   ```
   pwd
   ```

4. Start the MySQL Database Server if it is not started yet:

   ```
   mysqld_safe &
   ```

5. Verify that the MySQL Database Server has started:

   ```
   ps -ef | grep mysqld
   ```

6. Log in to the MySQL Database Server:

   ```
   mysql -u root
   ```

7. Select the schema that you want to work with, which in our case is *world*:

   ```
   use world;
   ```

8. View the tables of the schema. In our example, the world schema contains three tables.

   ```
   show tables;
   ```

   The list of tables is displayed, similar to Figure 7-1.

```
> use world;
  Reading table information for completion of table and column names
  You can turn off this feature to get a quicker startup with -A

  Database changed
  mysql>
> show tables;
  +-----------------+
  | Tables_in_world |
  +-----------------+
  | City            |
  | Country         |
  | CountryLanguage |
  +-----------------+
  3 rows in set (0.00 sec)

  mysql>
```

Figure 7-1 MySQL show tables

9. Enter **quit** and then press Enter to exit the MySQL command line server.

Note: If you want to make a backup of one, many, or all schemas into a folder that does not exist, type the following command, so that you create a backup folder into the integrated file system:

```
mkdir /backup_folder
```

By default, when you run **mysqldump** for first time, the following folder is created:

`/QOpenSys/usr/local/mysql/mysql/bin/backup` folder is created.

10. Make a backup by using **mysqldump** as demonstrated in the following examples:
   – Use **mysqldump** to back up only one schema or database called *world*:

     ```
     mysqldump --user=itso --password=itso world > backup/backup_world.sql
     ```

     Figure 7-2 shows the result.

```
> mysqldump --opt --user=itso --password=itso world > backup/backup_world.sql
  $
> ls -la backup
  total 1704
  drwxrwsrwx   2 javier   0            8192 Aug 25 20:37 .
  drwxrwsrwx  31 qsys     0          602112 Aug 25 18:13 ..
  -rw-rw-rw-   1 javier   0          243219 Aug 25 20:37 backup_world.sql
  $
```

Figure 7-2 mysqldump for one database

Attention: You must add a destination file and folder. In this example, we use `/QOpenSys/usr/local/mysql/mysql/bin/backup/backup_world.sql`. If you do not add a destination file and folder, **mysqldump** command execution is redirected to the panel, and no backup file is created.

   – In this example, we back up two schemas called *world* and *mysql*:

     ```
     mysqldump --opt --user=itso --password=itso --databases world mysql >
     backup/backup_world_mysql.sql
     ```

     Figure 7-3 shows the results.

```
> mysqldump --user=itso --password=itso --databases world mysql > backup/backup_world_mysql.sql
  $
 > ls -la backup
  total 2984
  drwxrwsrwx   2 javier   0            8192 Aug 25 20:46 .
  drwxrwsrwx  31 qsys     0          602112 Aug 25 18:13 ..
  -rw-rw-rw-   1 javier   0          243219 Aug 25 20:45 backup_world.sql
  -rw-rw-rw-   1 javier   0          623081 Aug 25 20:46 backup_world_mysql.sql
  $
```

Figure 7-3 mysqldump for more than one database

– In this example, we use **mysqldump** to back up all schemas into a file:

```
mysqldump --user=itso --password=itso --all-databases >
backup/backup_all_databases.sql
```

Figure 7-4 shows the results.

```
> mysqldump --user=itso --password=itso --all-databases > backup/backup_all_databases.sql
  $
 > ls -la backup
  total 5552
  drwxrwsrwx   2 javier   0             8192 Aug 25 20:53 .
  drwxrwsrwx  31 qsys     0           602112 Aug 25 18:13 ..
  -rw-rw-rw-   1 javier   0           623215 Aug 25 20:54 backup_all_databases.sql
  -rw-rw-rw-   1 javier   0           243219 Aug 25 20:45 backup_world.sql
  -rw-rw-rw-   1 javier   0           623081 Aug 25 20:46 backup_world_mysql.sql
  $
```

Figure 7-4 mysqldump for all databases

In these three examples, notice that not all possibilities of the **mysqldump** command were explained. For customized backups, see Table 7-1 on page 106 for more options.

7.2.2 MySQL Administrator for backup

This section introduces an easy way to backup by using MySQL Administrator. For more information about installing MySQL Tools for 5.0, see *Discovering MySQL in IBM i5/OS*, SG24-7398.

An administrator user must exist before you can connect to MySQL Administrator. In our example, we use the *itso* administrator user profile. See 3.4.4, “Post installation tasks” on page 60. After this program has been installed in your workstation and an administrator user has been created:

1. Go to **Start** → **Programs** → **MySQL** → **MySQL Administrator**.
2. In the login window (Figure 7-5), type the values for Server Host (host name or system IP address), Username, and Password. Then click **OK**.

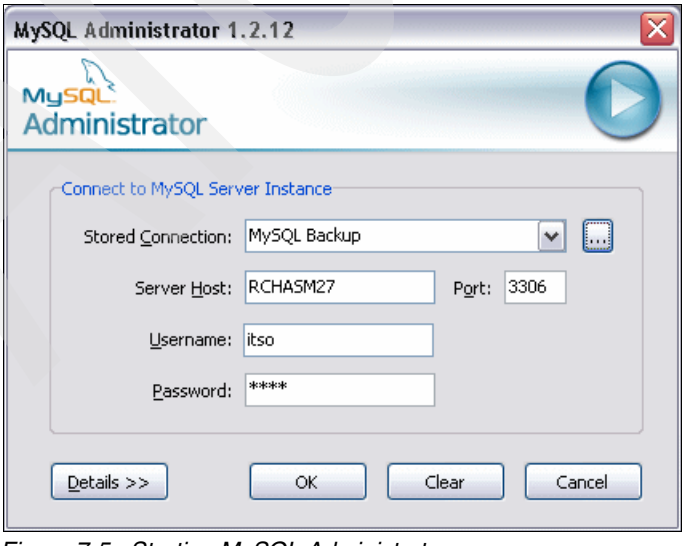


Figure 7-5 Starting MySQL Administrator

Note: In the login window, you may create several connections to select from in order to connect any other MySQL Database Server.

3. In the main window of MySQL Administrator (Figure 7-6) that opens, click **Backup**.

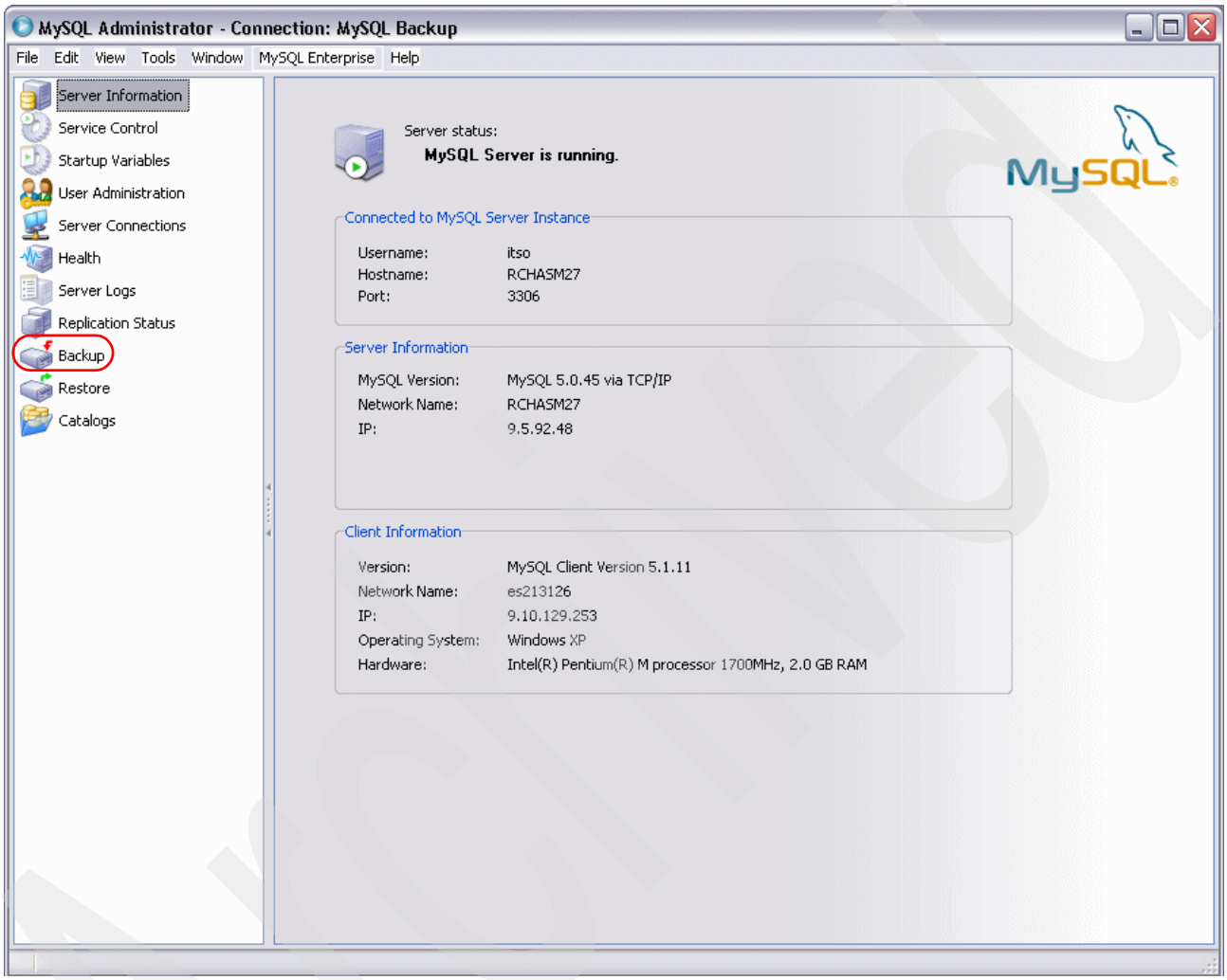


Figure 7-6 MySQL Administrator main window

The right pane of the MySQL Administrator window changes as shown in Figure 7-7 on page 112 to display three tabs.

You can do a backup by using either of the following methods:

- Click the **Backup Project** tab to create a new project and save it.
- Use a stored project, which must have been created previously in order to use it.

Then you can execute a backup by using either of the following methods:

– In an scheduled manner

  In this case, click the **Schedule** tab and select all available options to schedule that project (if it is a new project).

– Immediately by clicking the **Execute Backup Now** button

In addition, you may click the **Advanced Options** tab to specify detailed settings of how your backup should be performed.

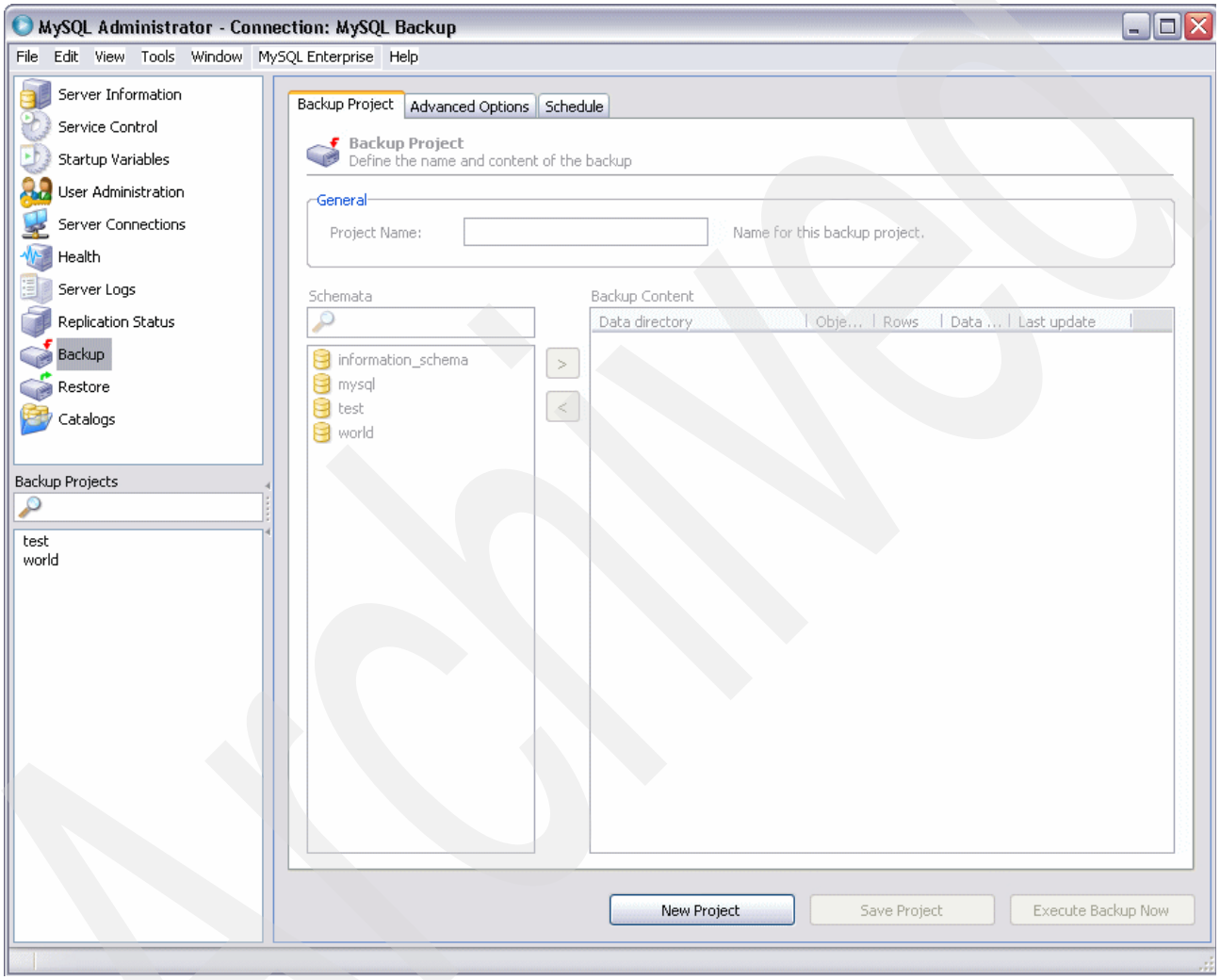


Figure 7-7 MySQL Backup window

If your password is encrypted before you start, you must change the Password storage method to Obscured as shown in Figure 7-8 on page 113:

a. In the MySQL Administrator window, select **Tools** → **Options...**

b. In the left navigation pane under Category, select **General Options**. In the right pane, in the Password Storage section, for Password storage method, select **Obscured**.

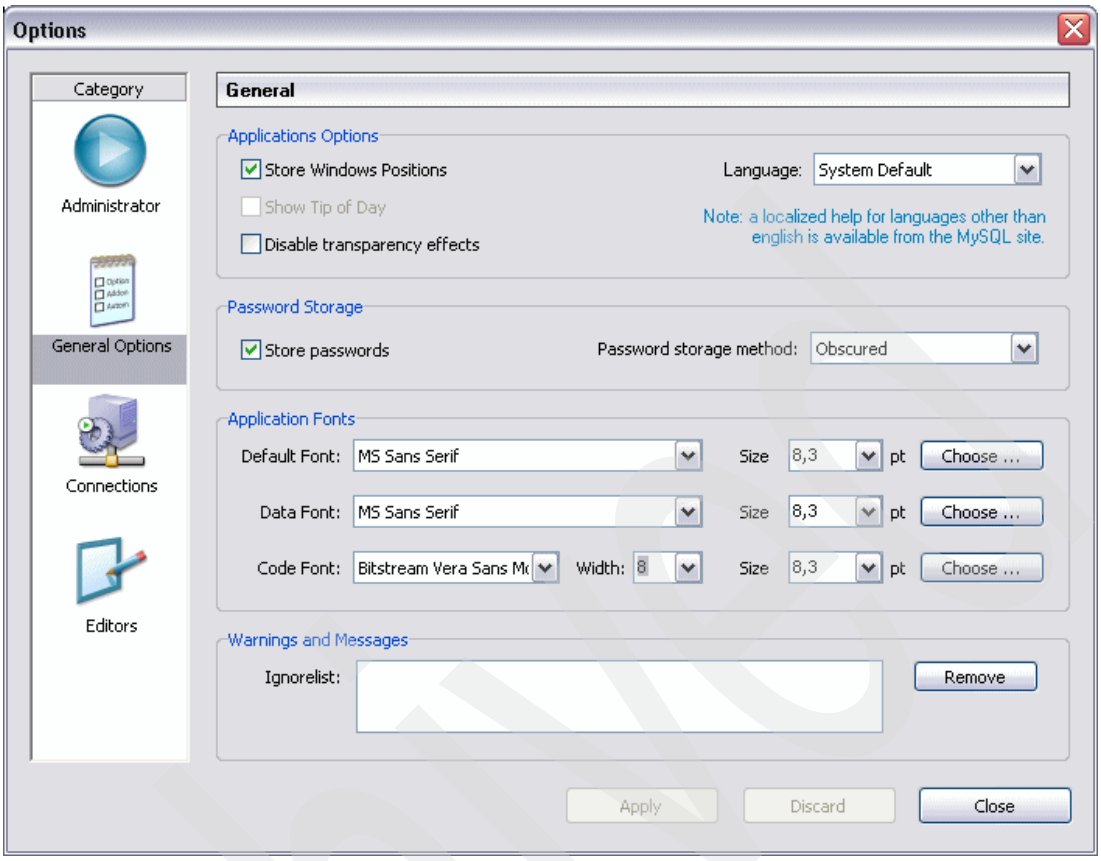


Figure 7-8 MySQL general options window

4. Create a new backup project and save it for later to restore a schema (or schemata). In the MySQL Backup window (Figure 7-9), complete the following tasks:
   a. Click the **New Project** button at the bottom of the window.
   b. Select the **Backup Project** tab and enter a project name (under General).
   c. Under Schemata, select a schema and click the right angle bracket (**>**) button to add this schema to the project. Notice that you can select which tables you want to back up. All tables are selected by default. In this case, we chose to add all tables to the test_backup_project name.

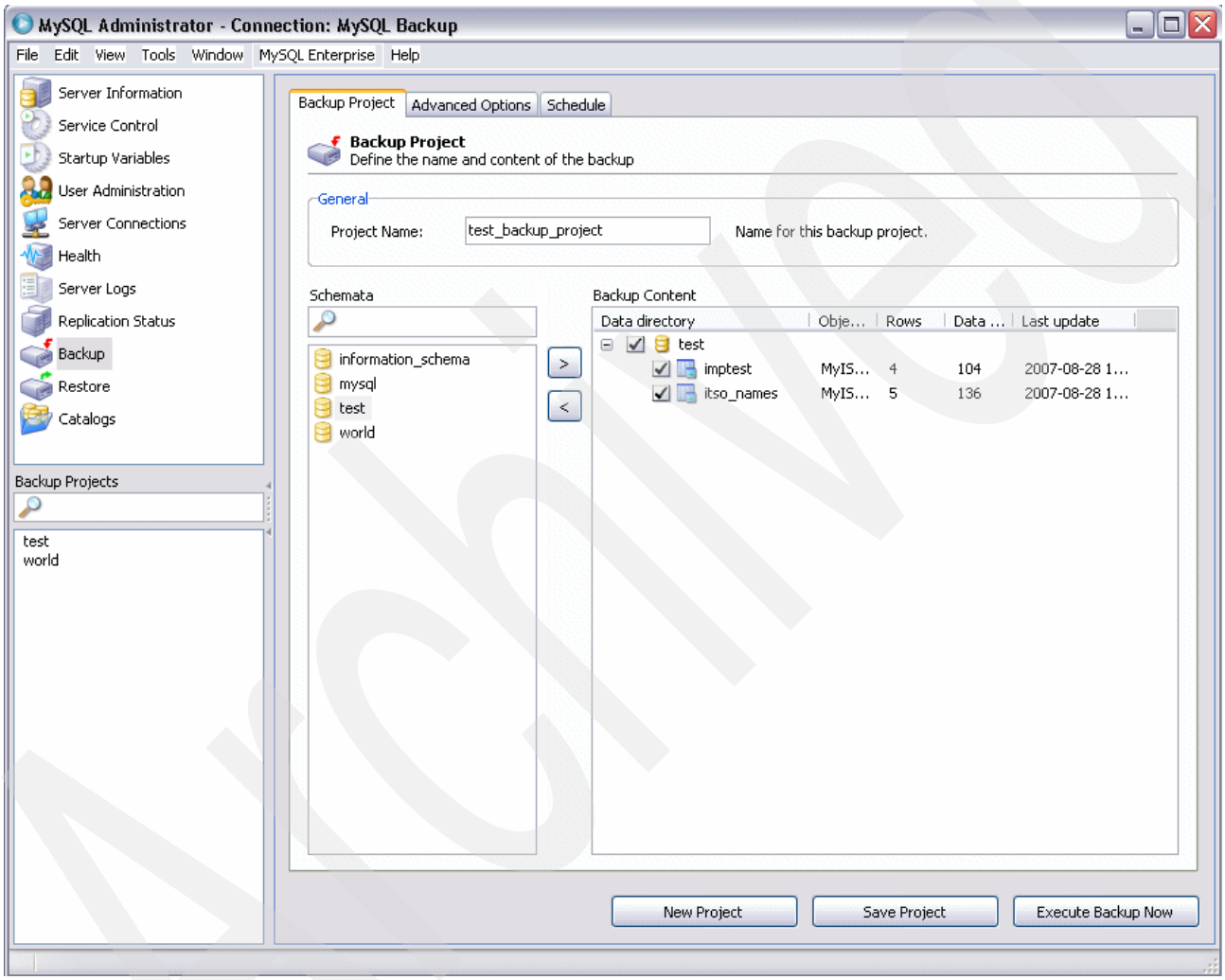


Figure 7-9 MySQL Backup Project tab

d. Click the **Advanced Options** tab and view all available options as shown in Figure 7-10. Keep the default options.

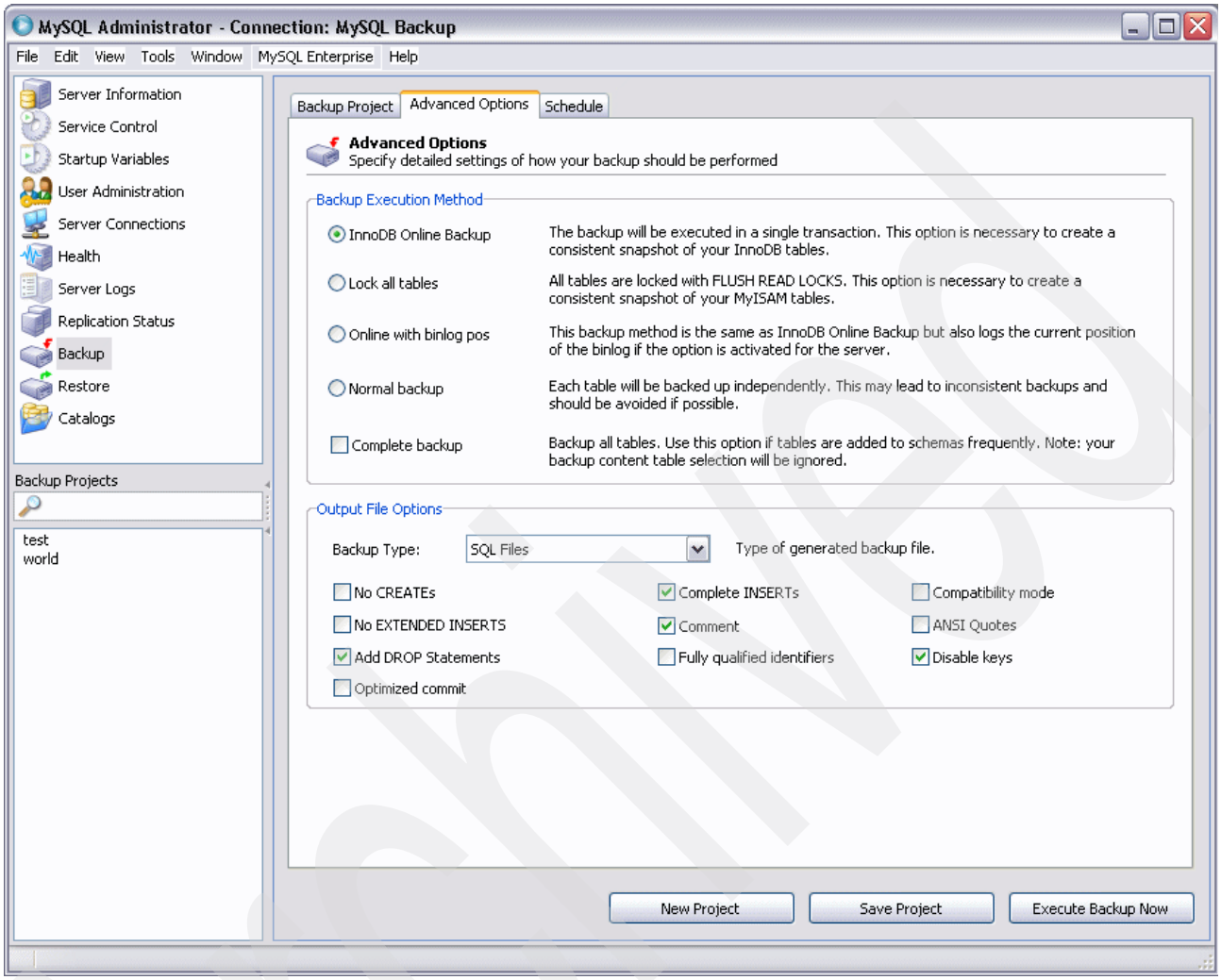


Figure 7-10 MySQL Advanced Options tab

e. Click the **Schedule** tab. On this tab, select the **Schedule this backup project** check box to make available all schedule options. Notice that you can select a Target folder and Filename to add a time stamp to the file name as shown in Figure 7-11. Now clear the **Schedule this backup project** check box.

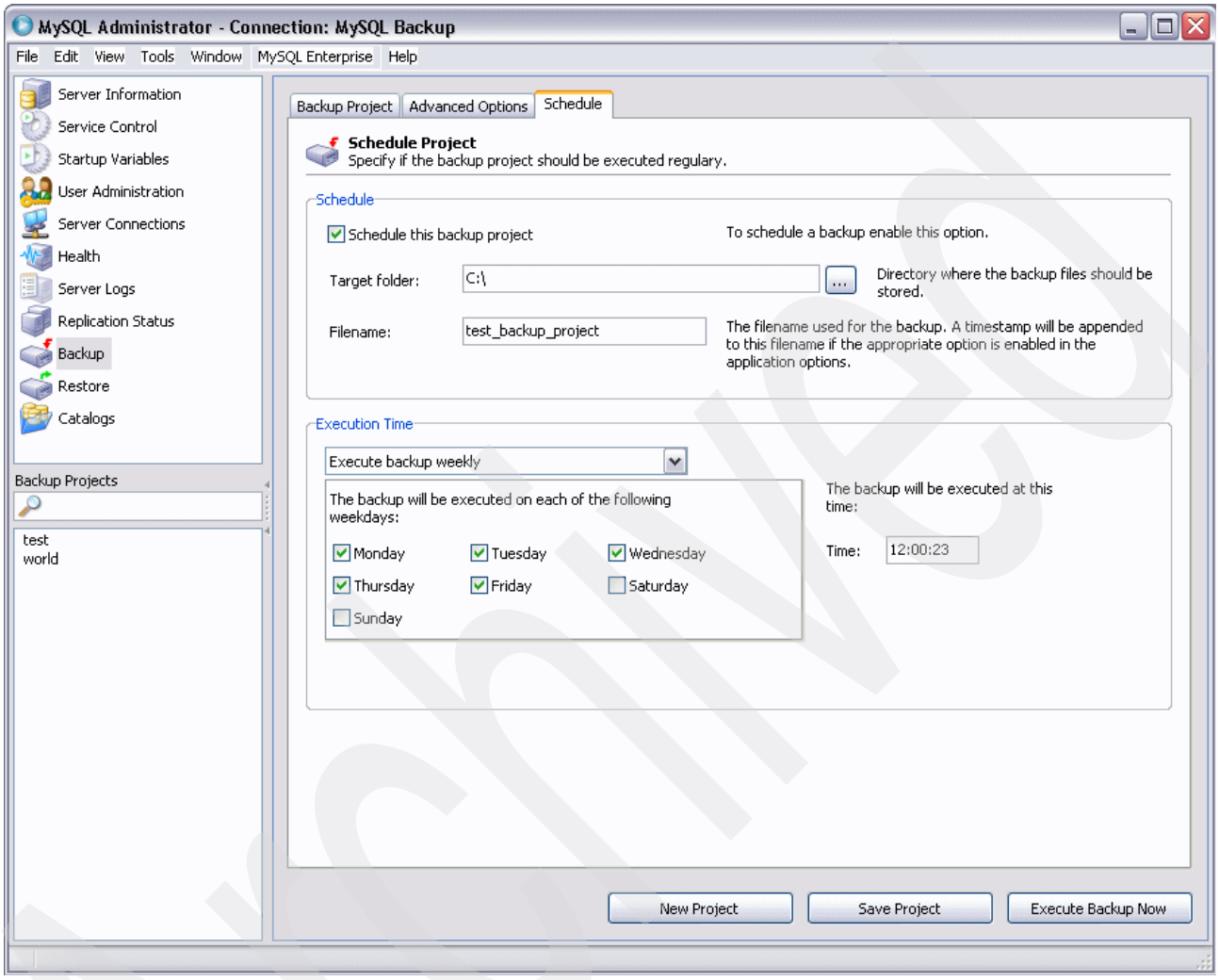


Figure 7-11 MySQL options for the Schedule tab

f. Click the **Save Project** button. In the Save As window (Figure 7-12), save your project.

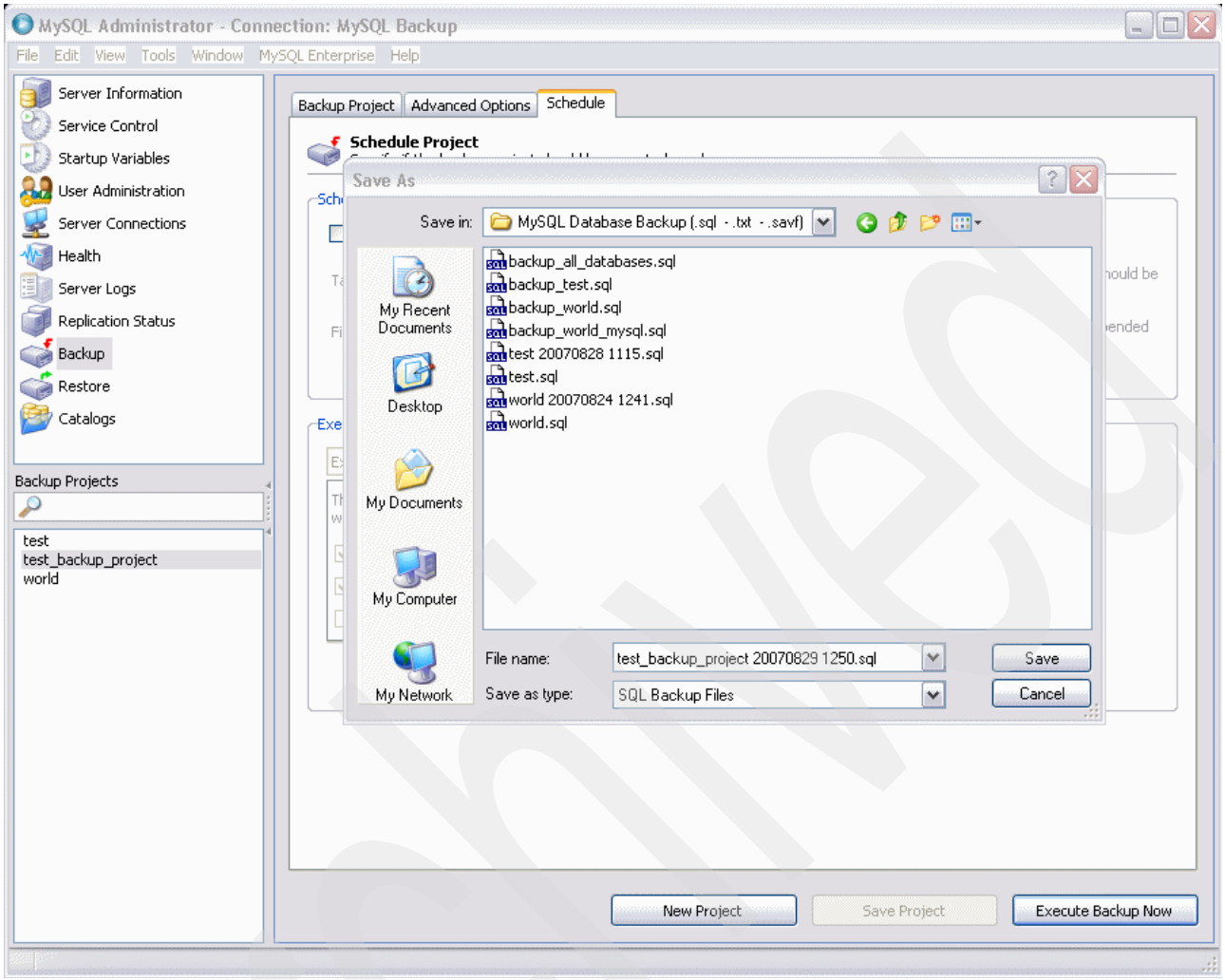


Figure 7-12 MySQL Save As window

g. In the left pane under Backup Projects, verify that this new project has been added.

h. Click the **Execute Backup Now** button.

You have now completed the backup process by using MySQL Administrator.

7.2.3 Using phpMyAdmin for backup

To perform a backup by using phpMyAdmin:

1. Start your browser and go to the following address (indicate the system name):

   `http://system name:89/phpMyAdmin/index.php`

   Port 89: To connect to phpMyAdmin, you must use port 89, by default, directly after the system name or IP address.

2. On the Welcome page for phpMyAdmin (Figure 7-13), type the user name and password. Then click **Go**.

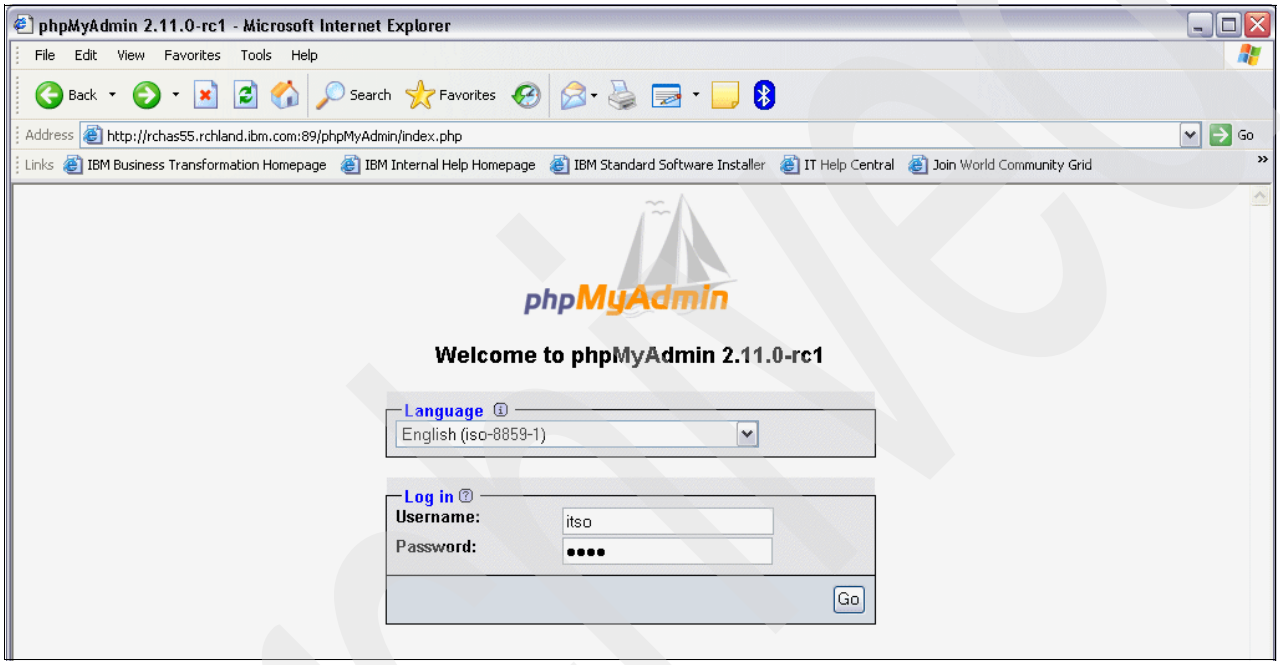


Figure 7-13 phpMyAdmin Welcome page

3. On the next page (Figure 7-14), click the **Export** option.

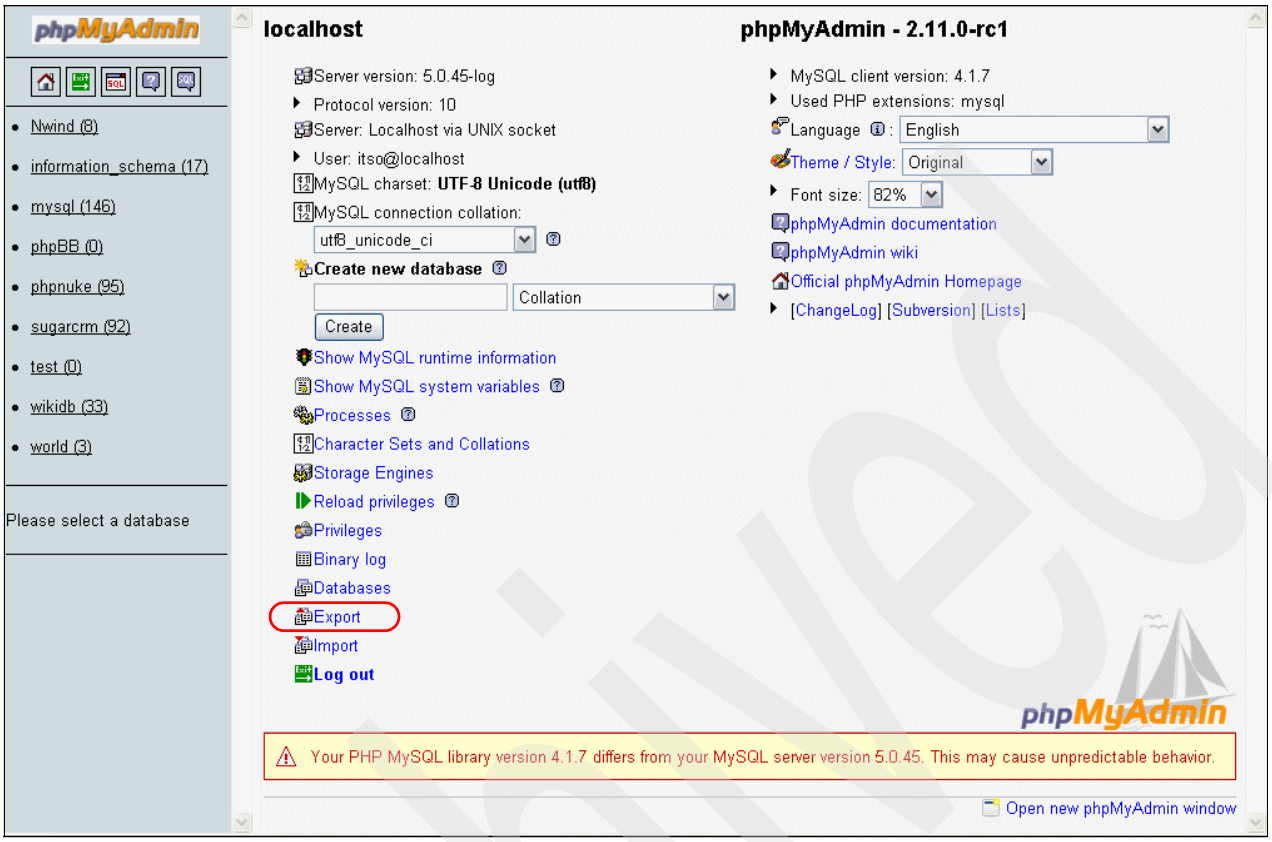


Figure 7-14 phpMyAdmin main page

4. On the next page (Figure 7-15), complete the following tasks:
   a. Export the test schema by using the preferred format. In this example, we export the *test* schema by using SQL. Therefore, in the Export box, select the **test** schema and then below the box, select **SQL**. If you prefer any of the other export formats or additional export options, you can select those instead.
   b. Select the **Save as file** check box. If you do not do this, the execution is redirected to the panel. Then for File name template, type the name.
   c. Click **Go**.

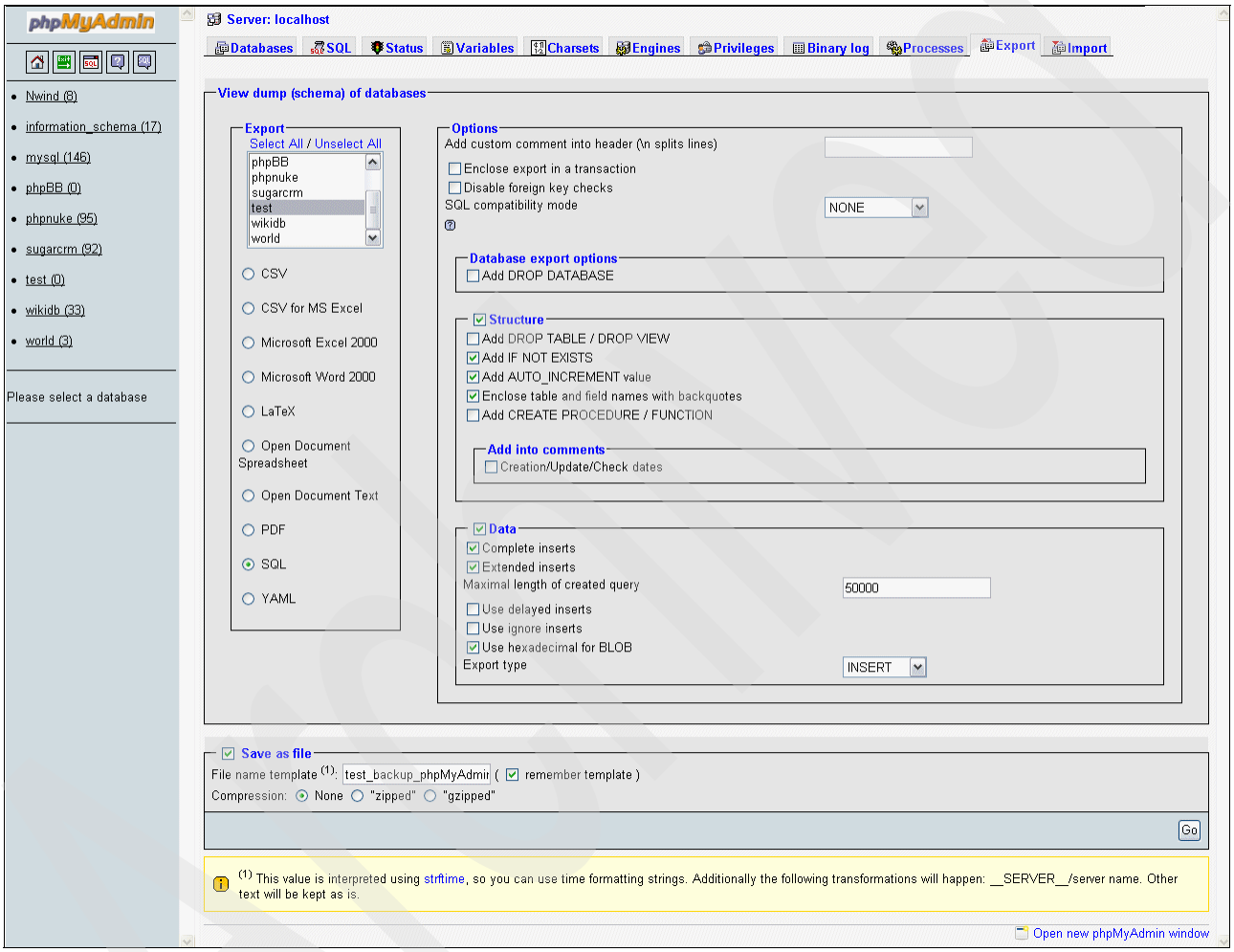


Figure 7-15 phpMyAdmin export main page

5. Save the backup script to the workstation. Choose a folder and name, and save the backup script file.

You have now finalized the backup process by using phpMyAdmin.

7.3 Saving MySQL databases shared with IBM i applications

When using the IBMDB2I Storage Engine, MySQL stores its data in a DB2 for i schema that is created in IBM i QSYS file system. MySQL also stores its metadata of its tables created with IBMDB2I in its data directory within IBM i Integrated File System (IFS) as described in section 2.3.4, “MySQL metadata files when using IBMDB2I” on page 11. Therefore, you have to save both portions of the MySQL database (in addition to the directory path that stores MySQL executable codes) to make sure that the recovery of the MySQL environment can bring back a working environment.

7.3.1 Saving the DB2 for i schema

A DB2 for i schema object is a native IBM i library object created with two additional features:

- Automatic database journaling

  The DB2 for i schema contains a journal object with a default name of QSQJRN and a journal receiver object with the default name of QSQJRNnnnn (with nnnn = running number that starts with 0001). Every newly created table object in the schema is automatically journaled without having to run the command STRJRNPF.

- Local DB2 for i catalog views

  The schema contains a set of DB2 for i catalog view objects that provide local data dictionary information of all database objects in that schema.

When you want to save or restore the DB2 for i tables created by MySQL, the MySQL statements BACKUP TABLE and RESTORE TABLE work only with tables created by the MyISAM engine and thus cannot be used with tables created by the IBMDB2I Storage Engine.

If you use MySQL-based applications with the IBMDB2I Storage Engine as *stand-alone applications* without having to exchange their data with other IBM i applications running in the same system or partition, it is very likely that you would leave all object authority assignment to the IBMDB2I schemas and tables intact at their default values, which are *PUBLIC with *EXCLUDE and the owner with *ALL authority. For such a case, you can use the backup tools described in Chapter 6 of *Discovering MySQL in IBM i5/OS*, SG24-7398 for successful backup and restore of the MySQL databases.

If you use MySQL-based applications with the IBMDB2I Storage Engine to help in the data exchange with other IBM i applications more convenient for you, you likely would make changes to the object authority assignment of those DB2 for i schemas and tables to enable such an interoperability requirement. If you have knowledge and experience with DB2 for i, you are aware that IBM i object-based authority assignment is one important factor to take into consideration when backing up and restoring IBM i objects.

When you create a DB2 for i schema and tables from MySQL environment with CREATE DATABASE and CREATE TABLE statements by using IBMDB2I Storage Engine, the IBM i user profile that *starts* and runs the **mysqld** daemon job is the default owner of the schema, and all objects contained within that schema and *PUBLIC authorities of all these objects are set to *EXCLUDE. This structure is by design for all database objects in DB2 for i that are created by SQL CREATE SCHEMA and CREATE TABLE statements with the SQL naming convention, which is what the IBMDB2I Storage Engine adopts. After the MySQL database objects are created and used, additional changes are possible in the authority assignment of these objects so that additional IBM i group or user profiles can access them for interoperability with other native IBM i applications.

For example, you may use an IBM i user profile named MYSQL1 to install MySQL and start the mysqld daemon job in IBM i PASE. Then, you install the SugarCRM application by using IBMDB2I Storage Engine and create a schema name SugarCRMDB2 with many tables that SugarCRM creates for its use. After the installation finishes, the schema SugarCRMDB2 and all the database objects within that schema have the MYSQL1 user profile as their default owner and their public authorities are set to *EXCLUDE. Then, you want an IBM i native application named ABCSALES to interoperate with SugarCRM. After you study the security scheme involved, you identify that an IBM i group profile named ABCGRP will be created for those user profiles that are running jobs of the ABCSALES application and that have to access and manipulate the data of certain identified tables in the SugarCRMDB2 schema. This means you have to grant the group profile ABCGRP proper access authority to the SugarCRMDB2 schema and those identified tables in the schema.

You may also choose to use authorization list for the IBMDB2I tables instead of private authority. This implementation is beneficial because the authorization list assignment information of a table is saved and restored with the table object. Private authority assignment is not saved with the table object. It remains with the user profile object.

The IBM i security implementation we described is totally outside the MySQL backup and restore environment but maintaining such a valid application interoperation is still important for you to do. This means that when you save a schema created by IBMDB2I, you want all the object-based authority assignments to be saved and restored together with the schema and its database objects. Therefore, when backing up and restoring DB2 for i schemas created by MySQL through the IBMDB2I Storage Engine, consider using such IBM i save and restore commands as SAVLIB/RSTLIB, SAVOBJ/RSTOBJ, and SAV/RST as your choice although other choices also exist from PHP or MySQL perspectives.

7.3.2 Saving the IFS portion of the metadata

MySQL metadata is described in 2.3.4, “MySQL metadata files when using IBMDB2I” on page 11. A good practice is to save the MySQL datadir directory path (the default path is `/QOpenSys/usr/local/mysql/data`) regularly when you know that MySQL database changes occur. The directory is where MySQL saves various kinds of its information, including its table metadata.

Another MySQL file to be saved when changes are made is `my.cnf`, which is the startup option file. By default, this file is created and maintained in `/etc`, which is the IBM i IFS directory path.

Another IFS directory path to be saved for recovery purposes is:

```
/QOpenSys/usr/local/mysql/mysql
```

This path is a default location to store the MySQL PASE executable files and many other system files. This path may not require a regular backup, so be sure to save it when there are changes to MySQL product, for example upgrading a version or applying software fixes.

7.4 Restoring the MySQL databases

This section describes the various ways to restore the MySQL Database Server on i5/OS databases.

7.4.1 The mysqlimport command for restore

The **mysqlimport** command provides a command line interface (CLI) to the LOAD DATA INFILE SQL statement. Most options to **mysqlimport** correspond directly to clauses of the LOAD DATA INFILE syntax. This command is useful to import data from a file or files into a table.

The **mysqlimport** command uses the following syntax:

```
mysqlimport [options] db_name textfile1 [textfile2 ...]
```

For each text file named on the command line, **mysqlimport** strips any extension from the file name and uses the result to determine the name of the table into which to import the file's contents. For example, files called names.txt, names.text, and names are all imported into a table called *names*.

Table 7-2 lists several of the **mysqlimport** options.

Table 7-2 mysqlimport options

Option	Description
--help, -?	Displays a help message and exit.
--columns=column_list, -c column_list	Takes a comma-separated list of column names as the command's value. The order of the column names indicates how to match data file columns with table columns.
--compress, -C	Compresses all information that is sent between the client and the server if both support compression.
--debug[=debug_options], -# [debug_options]	Writes a debugging log. The debug_options string often is 'd:t:o,file_name'.
--delete, -D	Empties the table before importing the text file.
--fields-terminated-by=..., --fields-enclosed-by=..., --fields-optionally-enclosed-by=..., --fields-escaped-by=...	Has the same meaning as the corresponding clauses for LOAD DATA INFILE.
--force, -f	Ignores errors. For example, if a table for a text file does not exist, the command continues processing any remaining files. Without --force, **mysqlimport** exits if a table does not exist.
--host=host_name, -h host_name	Imports data to the MySQL Database Server on the given host. The default host is localhost.
--ignore, -i	See the description for the --replace option.
--lines-terminated-by=...	Has the same meaning as the corresponding clause for LOAD DATA INFILE. For example, to import Windows files that have lines terminated with carriage return/linefeed pairs, use: `--lines-terminated-by="\r\n"` You might have to use double backslashes, depending on the escape conventions of your command interpreter.

Option	Description
--local, -L	Reads input files locally from the client host.
--lock-tables, -l	Locks *all* tables for writing before processing any text files to ensure that all tables are synchronized on the server.
--low-priority	Uses LOW_PRIORITY when loading the table. This affects only storage engines that use only table-level locking (MyISAM, MEMORY, and MERGE).
--password[=password], -p[password]	Represents the password to use when connecting to the server. If you use the short option form (-p), you *cannot* have a space between the option and the password. If you omit the password value following the `--password` or `-p` option on the command line, you are prompted for one. Specifying a password on the command line should is insecure.
--replace, -r	The `--replace` and `--ignore` options control the handling of input rows that duplicate existing rows on unique key values. If you specify `--replace`, new rows replace existing rows that have the same unique key value. If you specify `--ignore`, input rows that duplicate an existing row on a unique key value are skipped. If you do not specify either option, an error occurs when a duplicate key value is found, and the rest of the text file is ignored.
--silent, -s	Represents silent mode. Produces output only when errors occur.
--socket=path, -S path	Represents connections to localhost. On UNIX, it is the socket file to use; on Windows, it is the name of the named pipe to use.
--user=user_name, -u user_name	Represents the MySQL user name to use when connecting to the server.

The following example introduces a common situation of importing data from an Excel® spreadsheet into a MySQL database by using the **`mysqlimport`** command and other tools.

Suppose that you have an Excel spreadsheet with only two columns called *id* and *name*. First, you must save this Excel spreadsheet as a text file by using the Excel Save options. Second, you must upload the file to the System i environment by using FTP or another tool, such as iSeries Navigator.

In our example, we upload a text file named `itso_names` to the following path:

```
/QOpenSys/usr/local/mysql/mysql/bin/backup
```

This file facilitates the data import process, as follows:

1. Sign on to i5/OS and execute the QP2TERM program to start the i5/OS PASE environment:

   ```
   CALL QP2TERM
   ```

2. In the i5/OS PASE Terminal Console, type the following command to run the MySQL commands:

   ```
   cd /QOpenSys/usr/local/mysql/mysql/bin
   ```

3. Verify whether you are in the correct directory:

   ```
   pwd
   ```

4. Start MySQL Database Server if it is not started yet:

   ```
   mysqld_safe &
   ```

5. Verify that the MySQL Database Server has started:

   ```
   ps -ef | grep mysqld
   ```

6. Connect to the MySQL Database Server and select the test schema:

   ```
   mysql -u root
   use test;
   ```

7. Create a TABLE called itso_names and import it into a schema called test:

   ```
   create table itso_names(id int, name varchar(30));
   ```

8. Verify the contents of the *test* schema and the *itso_names* table:

   ```
   show databases;
   select * from itso_names;
   ```

9. After you verify that the itso_names table is created and all data has been uploaded to `/QOpenSys/usr/local/mysql/mysql/bin/backup`, you are ready to import data into this table, as follows:

   a. Exit from the MySQL Database Server command line:

      ```
      quit
      ```

   b. Enter the following **`mysqlimport`** command script to import data into this table:

      ```
      mysqlimport --local --user=itso --password=itso test backup/itso_names.txt
      ```

10. Verify the contents of the table itso_names:

    ```
    mysql -u root -e "select * from itso_names" test -B
    ```

    Figure 7-16 shows the contents of the itso_names table.

```
> mysqlimport --local --user=itso --password=itso test backup/itso_names.txt
   test.itso_names: Records: 5  Deleted: 0  Skipped: 0  Warnings: 0
   $

> mysql -u root -e "select * from itso_names" test -B
   id      name
   1       Hernando Bedoya
   2       Shirley Pintos
   3       Bruno Digiovani
   4       Ervin Earley
   5       Javier Dieguez
   $
```

Figure 7-16 Checking the imported data

-B option: In this sample, we used the `-B` option, which is useful for better data display.

Our example showed how to import a specific schema by using the **`mysqlimport`** command. Notice that we used only a subset of all possible options of the **`mysqlimport`** command.

7.4.2 The source command for restore

The **source** command provides an easy way to restore a selected schema when a backup file is provided. To use this command, you must have a backup copy from the database that you are going to restore and then upload it to the System i environment by using FTP or another tool such as iSeries Navigator.

In this case, we upload the file called test.sql to /QOpenSys/usr/local/mysql/mysql/bin/backup to more easily import data:

1. Sign on to i5/OS and execute the QP2TERM program to start the i5/OS PASE environment:

   ```
   CALL QP2TERM
   ```

2. In the i5/OS PASE Terminal Console, type the following command so that you can run the MySQL commands:

   ```
   cd /QOpenSys/usr/local/mysql/mysql/bin
   ```

3. Verify whether you are in the correct directory:

   ```
   pwd
   ```

4. Start the MySQL Database Server if it is not started yet:

   ```
   mysqld_safe &
   ```

5. Verify that the MySQL Database Server has started:

   ```
   ps -ef | grep mysqld
   ```

6. Connect to the MySQL Database Server:

   ```
   mysql -u root
   ```

7. For demonstration purposes only, drop the test schema in order to restore it later:

   ```
   drop database test;
   ```

8. Verify that all the schemas are available. Notice that the *test* schema is not included because we just deleted it:

   ```
   show databases;
   ```

9. After you verify that the *test* database does not exist, you can restore it by using the **source** command and then use the **show** command to view all databases again:

   ```
   source backup/backup_test.sql;
   show databases;
   ```

The steps are illustrated in Figure 7-17 on page 127.

```
> mysql -u root
 Welcome to the MySQL monitor.  Commands end with ; or \g.
 Your MySQL connection id is 64
 Server version: 5.0.45 MySQL Community Server (GPL)

 Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

 mysql>
> drop database test;
 Query OK, 2rows affected (0.16 sec)
 mysql>
> show databases;
 +--------------------+
 | Database           |
 +--------------------+
 | information_schema |
 | mysql              |
 | world              |
 +--------------------+
 3 rows in set (0.00 sec)
 mysql>

> source backup/backup_test.sql
 Query OK, 0 rows affected (0.00 sec)
 ..........
 ..........
 ..........
 Query OK, 0 rows affected (0.00 sec)

 mysql>
> show databases;
 +--------------------+
 | Database           |
 +--------------------+
 | information_schema |
 | mysql              |
 | test               |
 | world              |
 +--------------------+
 4 rows in set (0.00 sec)

 mysql>
```

Figure 7-17 The source command panel

7.4.3 MySQL Administrator for restore

In this section, we restore a schema, which you would do only in a case of disaster, or a table or tables into a specific schema. We also explain the easiest way to restore a security backup by using MySQL Administrator on a Windows XP workstation.

For information about how to install MySQL Tools for 5.0, see *Discovering MySQL in IBM i5/OS*, SG24-7398 for installation and other related tasks.

To perform these steps, you must have an administrator user profile created before you can connect to MySQL Administrator. In our example, we use *itso* administrator user profile. See 3.4.4, “Post installation tasks” on page 60.

In this case, we restore the complete schema that we saved in 7.2.2, “MySQL Administrator for backup” on page 110. The schema is called *test*. To do this, we need a backup copy from the *test* schema.

1. After this program is installed in your workstation and an administrator user is created, from your desktop, select **Start** → **Programs** → **MySQL** → **MySQL Administrator**.
2. In the login window (Figure 7-5 on page 110), type the values for Server Host (host name or system IP), Username, and Password. Then click **OK**.
3. In the MySQL Administrator window (Figure 7-19 on page 129):
   a. In the left pane, select the **Restore** option.
   b. In the right pane, click the **Open Backup File** button.
   c. On the General tab, for File to restore, select the file. In this case, we select **test_backup_project 20070829 1250.sql**.
   d. For Target schema, select **Another schema** and choose the schema. In this example, we select **test**.

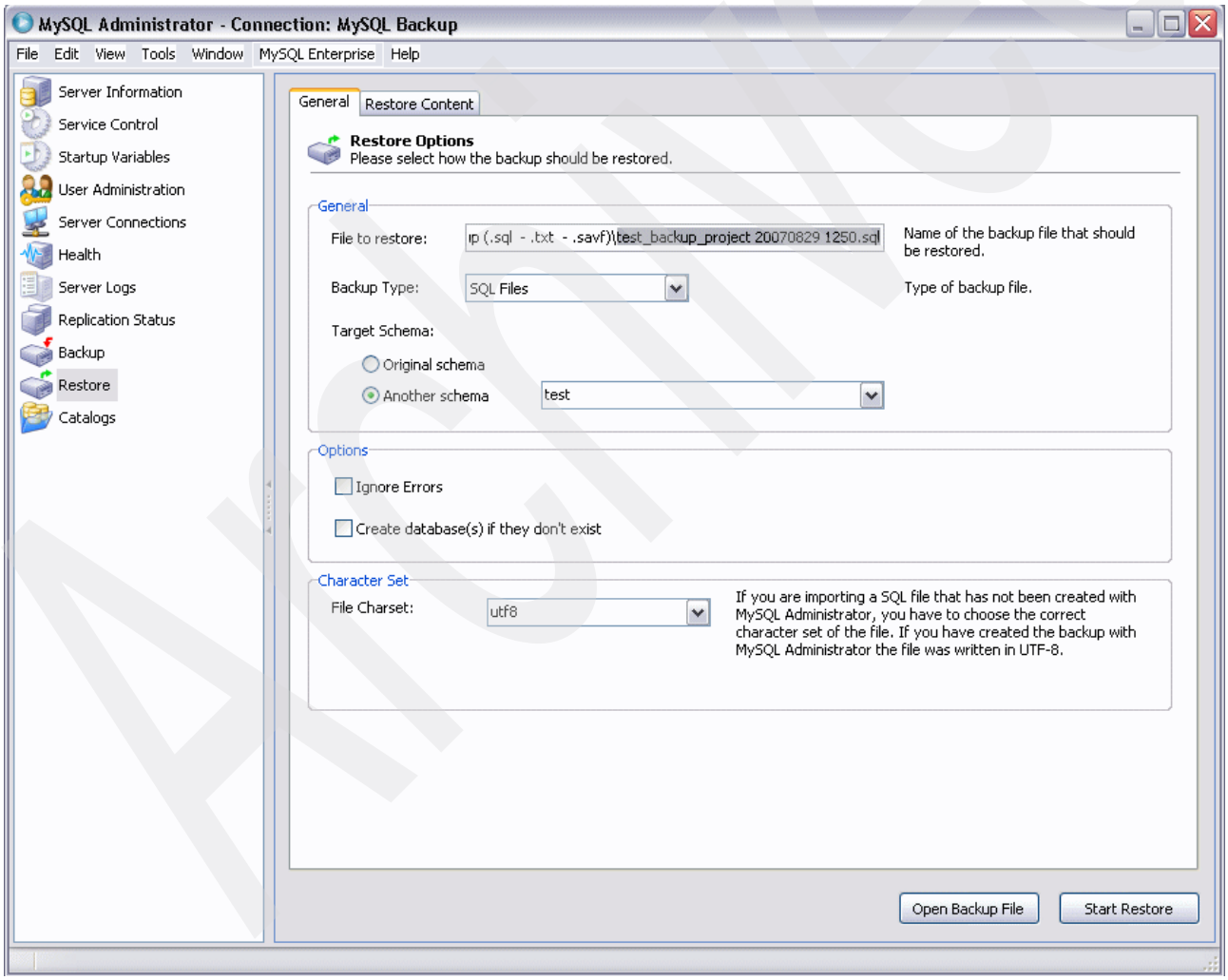


Figure 7-18 MySQL restore window

   e. Click the **Restore Content** tab (Figure 7-19 on page 129). On this tab, you can restore all tables of a desired schema, a subset of the tables, or just one of the tables. Click the

Analyze Backup File Content button and then choose all tables, some tables, or one table to restore them. All tables are selected by default.

f. Click **Start Restore**.

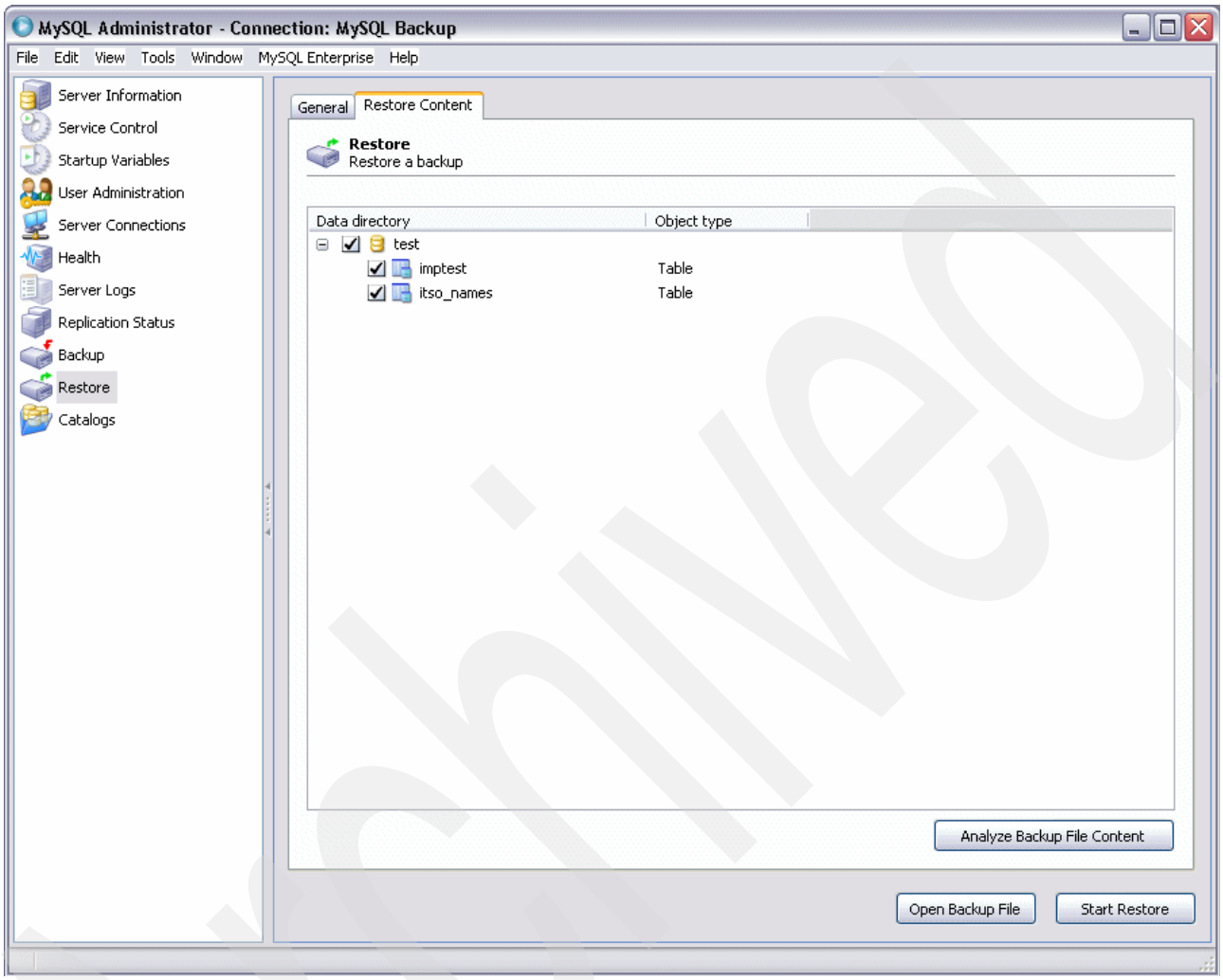


Figure 7-19 MySQL restore content tab window

You have now finalized the restore process by using MySQL Administrator.

7.4.4 Using phpMyAdmin for restore

In this section, we explain how to use phpMyAdmin to restore from a previous backup. This method has additional requirements that you must complete before you begin. See *Discovering MySQL in IBM i5/OS*, SG24-7398 for information about these requirements.

After you install phpMyAdmin:

1. Start your Internet browser and go to the following address (indicate the server name):

   `http://server name:89/phpMyAdmin/index.php`

   Note: To connect to phpMyAdmin, you must use port 89 after the system name or IP address by default.

2. On the Welcome page for phpMyAdmin (Figure 7-13 on page 118), type the user name and password. Then click **Go**.
3. On the phpMyAdmin main page (Figure 7-20), select the **Import** option and click the **Browse** button to find and select a previous backup file. In this case, we select the **backup_phpMyAdmin.sql** file. Click **Go**.

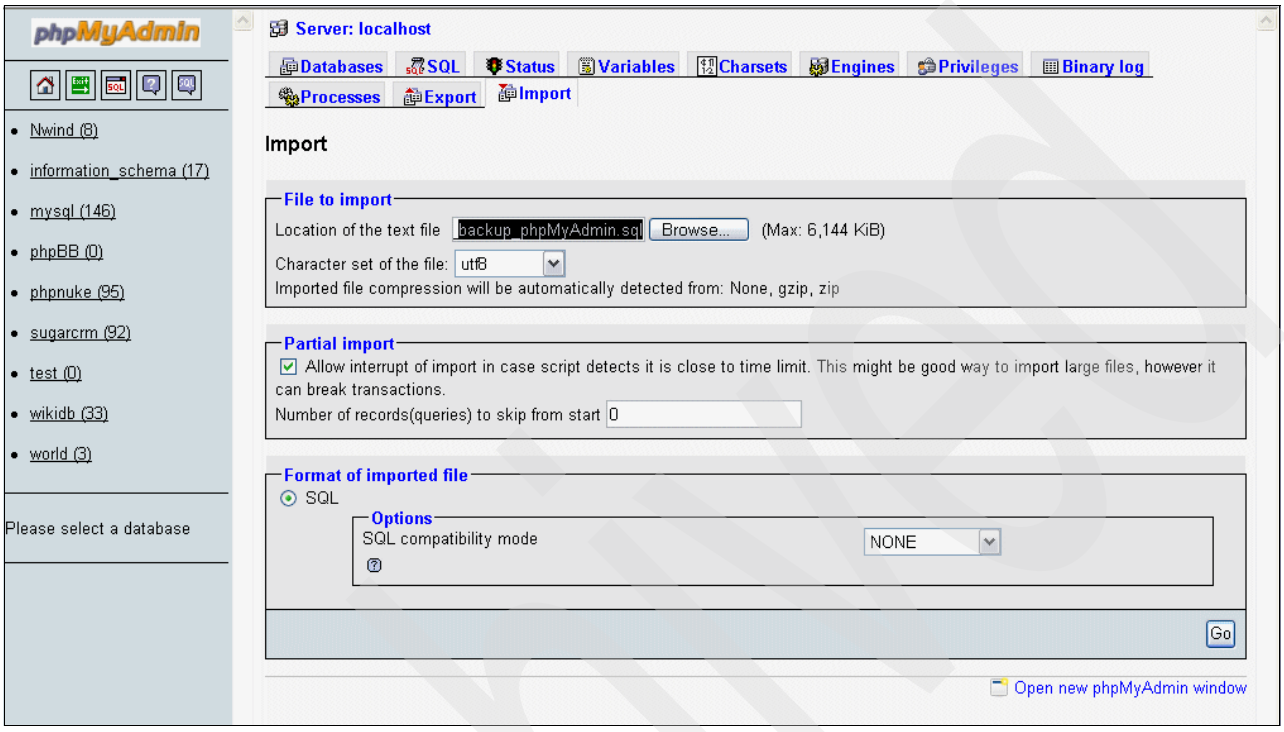


Figure 7-20 phpMyAdmin import main page

Important: You can only use this method when the database does not exist. If you attempt to use this method and the database already exists, you will receive an error message.

You have now completed the restore process by using phpMyAdmin.

7.5 Restoring MySQL databases shared with IBM i applications

When IBM i objects are saved to a data backup media by IBM i SAVXXX commands, the objects' owners and their public authority information are also saved. In IBM i 6.1, there was an enhancement to save and restore commands so that if your IBM i user profile has *SAVSYS or *ALLOBJ special authority, then you can specify to save the private authorities associated with the objects being saved, as follows:

- Specify a value of *YES for a new parameter called *Private authorities* (PVTAUT) of the IBM i commands SAVLIB, SAVOBJ, and SAV.
- If you have *ALLOBJ special authority, the private authorities of the objects can be restored at the same time that you restore the objects by specifying a value of *YES for PVTAUT of the commands RSTLIB, RSTOBJ, and RST.

In IBM i 5.4, the private authorities of the objects that are being saved cannot be optionally saved with the objects. Private authorities remain with IBM i user profiles. If you restore

objects over the same existing objects, there is no change to public and private authority assignments of the existing objects being restored. However, if you restore the objects as new objects, you have to manually assign private authorities to the restored objects. So, it is a good practice to note all existing private authority assignments of objects to be saved and restored to a different machine as new objects, as in the case of a disaster recovery machine.

In IBM i 5.4, you can choose to use authorization list instead of private authority assignment for the IBMDB2I tables because SAVXXX and RSTXXX commands save and restore the assigned authorization list information of the object. You have to also explicitly save the authorization list object. All authorization lists are stored in the QSYS library and can be saves with SAVLIB/SAVOBJ, SAVSYS, and SAVSECDTA commands.

At the schema level, a good practice is to note the IBM i user profile that owns the schema objects created by the IBMDB2I Storage Engine. Noting the information can help when you want to restore your MySQL application to a different machine so that you can check whether a user profile with the same name already exists in the destination machine or not. If not, you should create the required user profile name before restoring the schema to prevent the schema from being owned by QDFTOWN profile when restored. If QDFTOWN becomes the owner of the restored MySQL schema, you might find that MySQL no longer can access the restored data in that schema because the MySQL server job runs under the IBM i user profile who starts the server.

When restoring an IBMDB2I table over an existing table that is corrupted, be aware that if a down-level version of that table object's attributes are restored, MySQL is not aware of them and unpredictable results could occur when it accesses the table. For example, unpredictable results can occur if the table was saved prior to MySQL issuing ALTER TABLE to drop or change a column. The reason is because the RSTLIB, RSTOBJ, and RST commands run within IBM i environment and have no way to communicate with MySQL environment.

7.6 Additional tools for backup and restore

Additional tools are available that can help to fix database problems. In this section, we introduce the following tools:

- Security backup to TAPE
- Security backup to *SAVF
- Restoring from TAPE
- Restoring from *SAVF

Because we do not discuss all of the save and restore utilities for i5/OS to manage common backup and restore errors, refer to *The System Administrator's Companion to AS/400 Availability and Recovery*, SG24-2161, for more information.

7.6.1 Security backup to TAPE

Note: This section is intended only for external backup.

To avoid a loss of data in the event of disk failure, make an external copy of your database. For more information, see *The System Administrator's Companion to AS/400 Availability and Recovery*, SG24-2161.

Because this method is only for external backup, you must first know:

- The name of the tape device
- The full path of folder that contains backup files or full path of backup file (if you want to save only one backup file) on the integrated file system that you will save to tape.

Enter the Save Object (SAV) command and press F4. In this example, as shown in Figure 7-21, we save the `/QOpenSys/usr/local/mysql/mysql/bin/backup` folder that contains all backup files of MySQL Database Server to the TAP01 device.

Note: You may replace the asterisk (*) with the name of the file only if you want to back up a specific file to tape.

```
                                Save Object (SAV)

Type choices, press Enter.

Device . . . . . . . . . . . . . > '/qsys.lib/tap01.devd'

               + for more values

Objects:
  Name . . . . . . . . . . . . . > '/QOpenSys/usr/local/mysql/mysql/bin/backup/*'

  Include or omit  . . . . . . .   *INCLUDE      *INCLUDE, *OMIT
               + for more values
Name pattern:
  Pattern  . . . . . . . . . . .   '*'

  Include or omit  . . . . . . .   *INCLUDE      *INCLUDE, *OMIT
               + for more values
Directory subtree  . . . . . . .   *ALL          *ALL, *DIR, *NONE, *OBJ, *STG
Save active  . . . . . . . . . .   *NO           *NO, *YES, *SYNC
                                                                        More...
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

Figure 7-21 SAV command to tape

7.6.2 Security backup to *SAVF

The easiest and most popular way to make a security backup to disk is to use a compressed save file (a method that is not described in this book). In this case, make a copy to an external device, such as tape, workstation, CD, or DVD.

For more information, see *The System Administrator's Companion to AS/400 Availability and Recovery*, SG24-2161.

To make a security backup:

1. Create a Save File (CRTSAVF) to allow save backup files:

   ```
   CRTSAVF FILE(QGPL/BACKUP)
   ```

2. Run the Save Object (SAV) command and press F4. In the example shown in Figure 7-22, we save the `/QOpenSys/usr/local/mysql/mysql/bin/backup` folder that contains all backup files of the MySQL Database Server to back up the save file device.

   Note: You may replace the asterisk (*) with the name of the file if you only want to back up a specific file to a save file.

```
                                Save Object (SAV)

Type choices, press Enter.

Device . . . . . . . . . . . . . > '/qsys.lib/qgpl.lib/backup.file'

               + for more values

Objects:
 Name . . . . . . . . . . . . . > '/QOpenSys/usr/local/mysql/mysql/bin/backup/*'

  Include or omit  . . . . . . .   *INCLUDE      *INCLUDE, *OMIT
               + for more values
Name pattern:
  Pattern  . . . . . . . . . . .   '*'

  Include or omit  . . . . . . .   *INCLUDE      *INCLUDE, *OMIT
               + for more values
Directory subtree  . . . . . . .   *ALL          *ALL, *DIR, *NONE, *OBJ, *STG
Save active  . . . . . . . . . .   *NO           *NO, *YES, *SYNC
                                                                        More...
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

Figure 7-22 SAV command to save a file

3. Check the contents of the backup save file:

   ```
   DSPSAVF FILE(QGPL/BACKUP)
   ```

   The Display Saved Objects - Save File panel (Figure 7-23) shows objects of the save file.

```
                      Display Saved Objects - Save File

Display level . . . . . :   1
Directory . . . . . . . :   < /local/mysql-5.0.45-i5os-power-64bit//bin/backup


Type options, press Enter.
  5=Display objects in subdirectory   8=Display object specific information

Opt  Object            Type        Owner              Size  Data
     backup_all_dat >  *STMF       JAVIER           655360  Yes
     backup_test.sq >  *STMF       JAVIER             8192  Yes
     backup_world.s >  *STMF       JAVIER           524288  Yes
     backup_world_m >  *STMF       JAVIER           655360  Yes
     imptest.txt       *STMF       JAVIER             8192  Yes
     itso_names.txt    *STMF       JAVIER             8192  Yes

                                                                Bottom
F3=Exit   F11=View 2   F12=Cancel   F16=Display header
F22=Display entire field
6 objects saved on media file.
```

Figure 7-23 DSPSAVF panel

7.6.3 Restoring from TAPE

To restore a file from a tape device into the integrated file system, use the following command:

```
RST DEV('/qsys.lib/tap01.devd') OBJ('/QOpenSys/usr/local/mysql/mysql/bin/backup/*')
```

Note: You may replace asterisk (*) with the name of the file if you only want to restore a specific file from tape.

7.6.4 Restoring from *SAVF

To restore a file from a save file into the integrated file system, use the following command:

```
RST DEV('/qsys.lib/qgpl.lib/backup.file')
OBJ('/QOpenSys/usr/local/mysql/mysql/bin/backup/*')
```

Note: You may replace asterisk (*) with the name of the file if you only want to restore a specific file from a save file.

7.7 Common backup and restore errors

Use care when restoring objects into i5/OS or into the integrated file system. One of the most common problems during the restore process is with objects authorities or nonexistent users in the system. Users should have enough authority to back up and restore the MySQL databases. Verify the logs in the integrated file system.

You can also display i5/OS job logs by running the Display Job Log (DSPJOBLOG) command, pressing F10, and then checking for messages in the job log.

Installation failures are usually caused by one or more of the following conditions:

- Your user profile does not have enough authority.
- The command has the wrong folder. The folder must be:

  `/QOpenSys/usr/local/mysql/mysql/bin`
- When using command line procedure, you do not have an authorized profile.
- Prerequisite software products or fixes are missing.
- The MySQL Database Server is not started yet.

7.7.1 Additional information

For additional information about backup and restore of MySQL databases, consult the following references:

- i5/OS fixes (including database)

  `http://www-912.ibm.com/s_dir/slkbase.nsf/recommendedfixes`
- i5/OS PASE fixes

  `http://www.ibm.com/servers/enable/site/porting/iseries/pase/misc.html`
- IBM Redbooks

  `http://www.redbooks.ibm.com`
- IBM System i Redbooks

  `http://www.redbooks.ibm.com/portals/systemi`
- MySQL Community Server downloads page

  `http://dev.mysql.com/downloads/mysql/5.0.html`
- phpMyAdmin official home Web site and downloads

  `http://phpmyadmin.net`
- The Perl directory

  `http://www.perl.org/`

8

Security

In this chapter we discuss security considerations of the IBMDB2I Storage Engine, how security mechanisms of IBM i and MySQL coexist and share roles, how authorities are set and created. It also discusses reviewing authorities and how to protect them from IBM i users. Finally, it discusses what authority to give objects created on IBM i for MySQL.

This chapter contains the following topics:

8.1 Overview of security in using the IBMDB2I Storage Engine

You might have to manage two different security mechanisms when you use the MySQL Database Server on IBM i, especially with the IBMDB2I as its storage engine:

- Security mechanism of IBM i
- Security mechanism of the MySQL Database Server on IBM i

Which security mechanism you use depends on what you are doing with the MySQL Database Server on IBM i.

We describe the basic concepts of two coexisting security mechanisms.

8.1.1 Introduction to security mechanism coexistence

When the MySQL Database Server on IBM i is active, you are authenticated by the MySQL privilege system to connect to the MySQL Database Server on IBM i. At this time, you do not have to be authenticated by IBM i security using user profile. After connecting to the MySQL Database server on IBM i, you can access any objects to which you have authority given by the MySQL privilege system. Even if you have access to tables that use the IBMDB2I Storage Engine, the actual tables are on IBM i at this time, you need no IBM i authority as long as you only use the MySQL interface.

Because access to DB2 table is made through the IBMDB2I Storage Engine, the user of the MySQL Database Server on IBM i does not require IBM i user profile for authentication and authorization. See Figure 8-1.

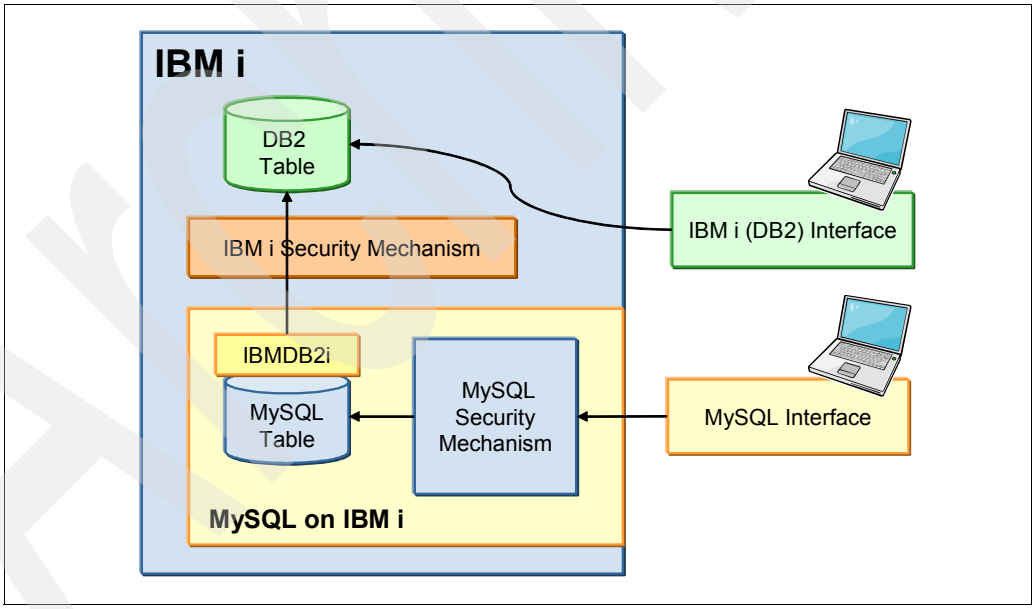


Figure 8-1 Coexistence of security mechanism

8.1.2 Operations that use IBM i security mechanism

Why do we require IBM i user profiles? Basically, the purpose for them is for operations that use IBM i interface, which include:

- Installation and configuration of the MySQL Database Server on IBM i
- Starting and stopping the MySQL Database Server on IBM i
- Administration using command line tools under the IBM i PASE shell
- Accessing objects created through the MySQL interface, using the IBMDB2I as a storage engine
- Accessing files and directories in IFS, which is created for the MySQL Database Server for IBM i (if necessary)

We should give proper authority to users to perform these operations, and protect the operations from other users.

8.2 Authority of IBM i objects through IBMDB2I Storage Engine

This section discusses a job and its user on IBM i, which handles a request from the MySQL Database Server on i. This section can help you understand which user profile is used to process the request.

8.2.1 User profile for starting the MySQL Database Server on IBM i

When you start the MySQL Database Server on IBM i, a new mysqld process is started. Certain conditions determine which user profile is used for this job.

When you are logged in to IBM i by QSECOFR user profile

First, specify a user profile to start this job in a startup option file `my.cnf` as follows:

```
[mysqld]
user = user_profile_name
```

In this option, you have to set an IBM i user profile.

This option is initially set during the installation process performed by INSMYSQL command with the user profile name in the USRPRF parameter. The default user profile is MYSQL.

This option is only used when you have logged in to IBM i as the QSECOFR user profile. When you issue the **`mysqld_safe`** command, this option is always referenced when the mysqld job starts. Then, the user profile name is used as the mysqld job user. This option is not used when you have logged in with other user profiles.

If the user option is not specified in `my.cnf` file, the `--user` option in the mysqld_safe command can provide a user profile name of the mysqld job. This option is specified as:

```
mysqld_safe --user = user_profile_name &
```

If user profile name is not specified in either of `my.cnf` and mysqld_safe command, QSECOFR is used for the mysqld job user.

Note: When the user option is specified in the `my.cnf` file, the `--user` option in the **`mysqld_safe`** command is ignored. Therefore, QSECOFR user cannot override the user of mysqld job dynamically on this condition. Implementation follows a mechanism for the UNIX system. In a UNIX system, using the UNIX root user to run the MySQL server is not recommended (in terms of security). When the root user is used to start the MySQL server, the user is changed with this mechanism. The UNIX root user correspond to QSECOFR on IBM i. See the *MySQL 5.1 Reference Manual* for more detail about the mechanism:

`http://dev.mysql.com/doc/refman/5.1/en/`

When you are logged in to IBM i by other user profiles

When you have logged in to IBM i by any other user profile, you can specify a user profile name for the mysqld job using the `--user` option in the mysqld_safe command. If you do not specify any user profile name, the mysqld job starts with your user profile.

Table 8-1 Decision table of mysqld job user

"user=" in my.cnf	**"--user" in mysqld_safe**	**IBM i login user**	**mysqld job user**
Specified	Ignored when specified	QSECOFR	User profile in my.cnf
		Other user profile	IBM i login user
Not specified	Specified	QSECOFR	User profile in "--user"
		Other user profile	IBM i login user
	Not specified	QSECOFR	QSECOFR
		Other user profile	IBM i login user

8.2.2 User profile of the QSQSRVR job

When a client issues an SQL statement on a MySQL table that uses the IBMDB2I Storage Engine, the request is passed to QSQSRVR job on IBM i. Then, the job performs DDL operations such as CREATE TABLE, CREATE INDEX, and so on. The job also performs I/O operations such as READ, INSERT, UPDATE, and DELETE, all based on the statement issued through MySQL interface. These operations are all processed under a current user of the job QSQSRVR.

As described in Figure 8-2 on page 141, the QSQSRVR job is called by the MySQL server process job (the mysqld job), and it takes over the user from the mysqld job. Therefore, you have to care for the user profile that is used to start the mysqld job so that you can manage ownership and authorities of the IBM i objects that are created and used by the MySQL Database Server on IBM i.

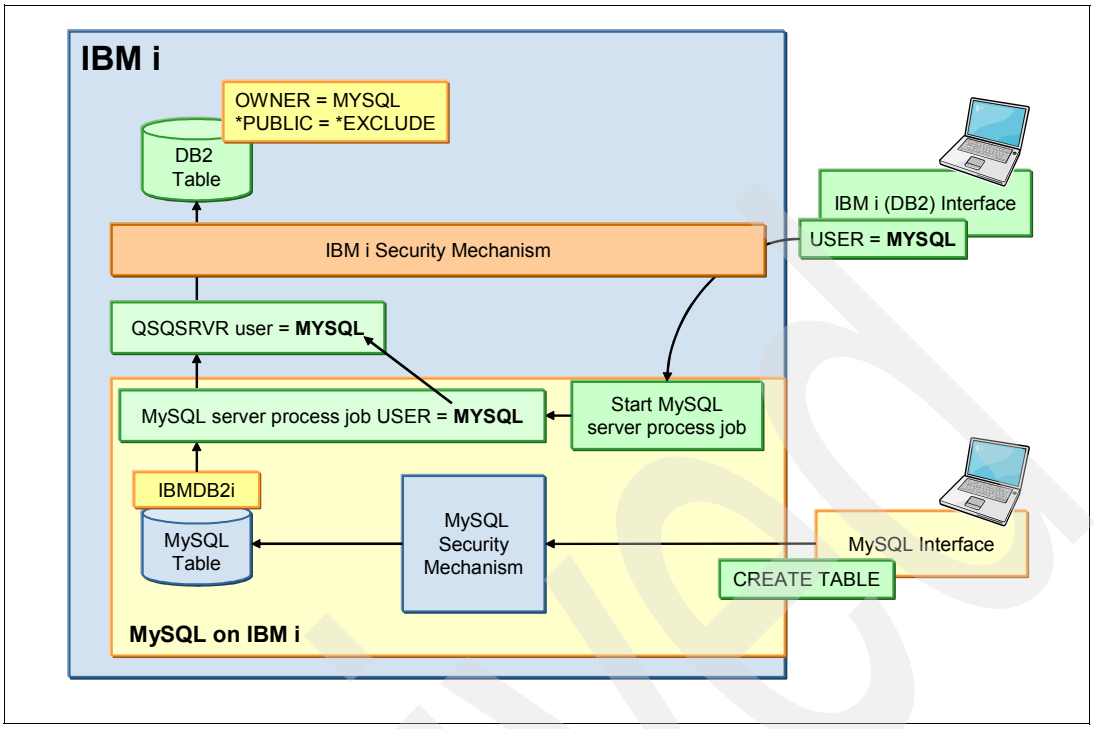


Figure 8-2 User profile of the QSQSRVR job

8.2.3 Consideration on authority of IBM i objects created through IBMDB2I

The owner of all objects, created on IBM i through MySQL interface using the IBMDB2I as a storage engine, is the user who started the MySQL server process job (mysqld job). These objects include library, physical file, logical file, journal, journal receiver. And their public authorities are set to *EXCLUDE at the time of creation. So, these objects cannot be accessed by any other users unless they have *ALLOBJ special authority or they have proper authority granted.

If a different user starts the mysqld job next time, the existing objects will be accessed by the different user, and a new object will also be created and owned by the different user. Therefore, ensure that the same user always starts the mysqld job so that no problems or confusion occur as a result of mixed object owners and authorities.

8.3 Protecting MySQL related objects from IBM i users

Although users and administrators of the MySQL Database Server on IBM i should have proper authorities given by MySQL security mechanism, authorities to IBM i are not always necessary for them.

In using the MySQL Database Server on IBM i, users who require authorities to IBM i include:

- Administrator who performs installation, configuration, and administrative operation to the MySQL Database Server on IBM i
- Application developers who use objects on IBM i, which is generated through the IBMDB2I Storage Engine using MySQL interface

- Users of an application on IBM i, which uses objects generated through the IBMDB2I Storage Engine
- Operators who perform starting and ending the MySQL Database Server on IBM i, backup of objects and directories related to the MySQL Database Server on IBM i, and more.

These users should have proper authorities as required by the current task on objects that reside on IBM i. At the same time, the objects should be protected from other IBM i users that have no need to access the objects.

8.3.1 Summary of default owner and authorities of IBM i objects

As a reference, Table 8-2 lists information of default owner and authorities that are given to directories and objects created for the MySQL Database Server on IBM i.

Table 8-2 Default authorities for MySQL related directories and objects on IBM i

Directory or Object	Specific or default name of the directory or object	Owner	Public authority				
			Data authority	Object authorities			
				Exist	Mgt	Alter	Ref
System directory for MySQL on IBM i	Default: /QOpenSys/usr/local/mysql/mysql-5.1.33-i5os-power-64bit	USRPRF specified during installation	*RX	X	X	X	X
Data directory for MySQL on IBM i	Default: /QOpenSys/usr/local/mysql/data	USRPRF specified during installation	*RWX	X	X	X	X
Directory for "mysql" database schema	Default: /QOpenSys/usr/local/mysql/data/mysql	USRPRF specified during installation	*NONE	X	X	X	X
System files under "mysql" database schema	Default: /QOpenSys/usr/local/mysql/data/mysql/*.*	USRPRF specified during installation	*NONE	X	X	X	X
Directory for user database schema	Default: /QOpenSys/usr/local/mysql/data/<*Database*>	User who *started* the MySQL Database Server on IBM i	*NONE	X	X	X	X
Files in user database schema	Default: /QOpenSys/usr/local/mysql/data/<*Database*>/*.*	User who *started* the MySQL Database Server on IBM i	*NONE	X	X	X	X
Startup option file for MySQL on IBM i	Default: /etc/my.cnf	USRPRF specified during installation	*EXCLUDE	-	-	-	-
Default login user profile ***See note**	MYSQL	User who *installed* the MySQL Database Server on IBM i	*EXCLUDE	-	-	-	-

Directory or Object	Specific or default name of the directory or object	Owner	Public authority				
			Data authority	Object authorities			
				Exist	Mgt	Alter	Ref
Library for user database of MySQL on IBM i	Library name in IBM i naming rule	User who *started* the MySQL Database Server on IBM i	*EXCLUDE	-	-	-	-
File objects in user library for MySQL on IBM i	File object name in IBM i naming rule	User who *started* the MySQL Database Server on IBM i	*EXCLUDE	-	-	-	-
Journal and journal receivers in user library on MySQL on IBM i	The defaults are: Journal: QSQJRN Journal receiver: QSQJRNnnnn	User who *started* the MySQL Database Server on IBM i	*EXCLUDE	-	-	-	-

According to this table, most of the directories and objects are protected from being used by public users by default security settings. One exception is that a public user can change data directory name of the MySQL Database Server on IBM i. Consider which users should be granted authorities on these objects.

Note: This protection is specified on installation, is stored in the startup option file for MySQL, and is used to determine:

- Ownership of installed objects
- The user profile to run MySQL server process job under when logged in as QSECOFR

8.3.2 Scenario of user profiles and authorities on IBM i

Determine which user profile to use for installing the MySQL Database Server on IBM i and which profile to use for performing administration and operations. Also, determine which authorities to assign to application developers and users of IBM i objects, such as tables and indexes, and which authorities to associate with the MySQL Database Server on IBM i.

User profile for administration, and starting and stopping operations

The user profile for administration tasks, such as working on the startup option file, starting and stopping the server, is the default login user that was specified in the USRPRF parameter in INMYSQL command for installation of the MySQL Database Server on IBM i. The default user is MYSQL because all installed objects except the user profile itself are owned by this user. When you use this user profile to start the MySQL Database Server, all IBM i objects created by MySQL SQL statements are owned and used by this user profile. Also, as mentioned in 8.2.3, “Consideration on authority of IBM i objects created through IBMDB2I” on page 141, using a single user profile can avoid problems and confusion caused by having mixed object ownership and authorities.

User profile for developers and users of IBM i application

When you develop an IBM i application by using file objects that are created by the MySQL Database Server on IBM i through the IBMDB2I Storage Engine, it is likely to be an RPG or COBOL application, so you have to grant developers and users proper authorities on the objects they use. The reason is that because these objects have *EXCLUDE as their public authority when created by the MySQL Database Server on IBM i as Table 8-2 on page 142 shows.

You can define private authorities to certain users to certain objects by following the IBM i security mechanisms. This should be used in a limited way. Otherwise, IBM i objects created by the MySQL Database Server on IBM i can be exposed to unexpected changes or deletions, which causes these objects to be invisible or unusable from the MySQL Database Server on IBM i interface.

Typical authority definitions for developers and users include:

- Object authority on a library for:
  - Developers
    - *USE for creating and testing a new program which uses existing objects in the library
    - *CHANGE for creating new objects in the library for new application
  - Users
    - *USE for executing program which uses objects in the library
    - *CHANGE for executing program which creates new objects in the library
- Object authority on objects in a library for:
  - Developers
    - *CHANGE for read, add, update, and delete operations to data in the file
    - *CHANGE and ALTER object authority for creating a new index on existing file created through the MySQL Database Server on IBM i interface
  - Users
    - *CHANGE for executing program which includes read, add, update, and delete operations to data in the file

9

Problem determination and diagnosis

This chapter focuses on identifying and solving problems while you use the IBMDB2I Storage Engine for MySQL.

This chapter contains the following topics:

9.1 Overview

As introduced in Chapter 2, “Architecture and functional support” on page 7, the IBMDB2I Storage Engine is a pluggable storage engine for MySQL server. It handles database requests from MySQL server, passes the requests to underlying DB2 database, and returns the execution result to the requester.

As illustrated in Figure 9-1, MySQL server on IBM i runs in the Portable Application Solutions Environment (PASE); therefore, the IBMDB2I Storage Engine that handles the API calls from MySQL server also runs in PASE. The storage engine translates the API calls to a series of database operations that DB2 for IBM i can process. It then passes the operation requests to the storage engine enablement code in ILE. This ILE part consists of several service programs, which pass the database operation requests to IBM i database, using SQL Server Jobs (QSQSRVR) jobs, and return the results to the IBMDB2I Storage Engine.

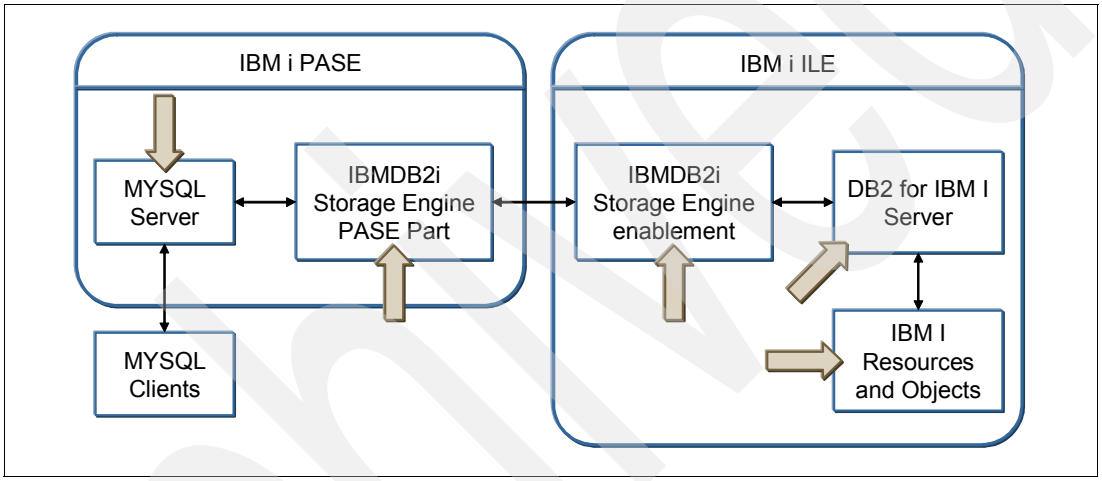


Figure 9-1 Troubleshooting DB2 for i Storage Engine for MySQL

Because of the nature of IBMDB2I Storage Engine's hybrid approach, problems can occur in any component identified in Figure 9-1. When troubleshooting such technology with multiple components, you first have to narrow the problem to a particular component, and then find the root cause. To determine how far down the stack a request has traveled is tricky because most of the problems are reported as application errors. But, you might start your analysis from the beginning, and then drill down to the lower level of the system. The following questions might help you determine where the root of the problem is:

- Is the MySQL server running?
- Is DB2 for IBM i working correctly?
- Is the IBMDB2I Storage Engine working for simple queries?
- Does the query run successfully if you use a different storage engine (MyISAM or InnoDB)?

9.2 Before you start

Common problems can cause the application to fail. In most cases, troubleshooting time can be reduced significantly when these common problems are checked first.

The most common problem is that the MySQL server instance is not started. There are several ways to determine if MySQL server is running or not. For example, you can monitor the status of your MySQL server instance by issuing the command in Figure 9-2.

```
mysqladmin ping -u root
mysqld is alive
$
```

Figure 9-2 Checking the instance status with mysqladmin

Note: For more information about managing MySQL Database Server on IBM i, see *Discovering MySQL in IBM i5/OS*, SG24-7398.

Another common problem is that IBMDB2I Storage Engine for MySQL Database Server is not properly installed. To verify whether IBMDB2I Storage Engine is installed, go to MySQL command line tools and issue the statement in Figure 9-3. You should see the IBMDB2I Storage Engine listed.

```
> show engines;

+------------+---------+------------------------------------------------------------
---------+--------------+-----+------------+
  | Engine     | Support | Comment
| Transactions | XA   | Savepoints |

+------------+---------+------------------------------------------------------------
---------+--------------+-----+------------+
  | InnoDB     | YES     | Supports transactions, row-level locking, and
foreign keys     | YES          | YES | YES        |
  | CSV        | YES     | CSV storage engine
| NO           | NO   | NO         |
  | MEMORY     | YES     | Hash based, stored in memory, useful for temporary
tables       | NO           | NO   | NO         |
  | MyISAM     | YES     | Default engine as of MySQL 3.23 with great
performance          | NO           | NO   | NO         |
  | IBMDB2I    | DEFAULT | IBM DB2 for i Storage Engine
| YES          | NO   | YES        |
  | MRG_MYISAM | YES     | Collection of identical MyISAM tables
| NO           | NO   | NO         |

+------------+---------+------------------------------------------------------------
---------+--------------+-----+------------+
  6 rows in set (0.01 sec)
```

Figure 9-3 Installed storage engine list output

If MySQL database server is running, and IBMDB2I Storage Engine is installed correctly, but your application is still failing, you issue one or two CREATE TABLE and INSERT statements against a table in IBMDB2I Storage Engine from MySQL's command line tools to see if the storage engine is working as expected.

If you still cannot determine whether the problem is caused by IBMDB2I Storage Engine, try to create your application's database tables with another storage engine, for example, MyISAM.

9.3 System jobs related to IBMDB2I Storage Engine

MySQL Database Server on IBM i runs in PASE. As introduced in *Discovering MySQL in IBM i5/OS*, SG24-7398, two methods are available to start a MySQL server:

- Calling mysqld_safe or mysqlmanager in the PASE command line to start the server
- Coding a CL program to submit a batch job to invoke mysqld_safe or mysql-manager in a PASE shell environment

Either way results in a system job running under the same user profile who starts the server, as shown in Figure 9-4.

```
 Work with Active Jobs                    G60B85AE
                                                          08/10/23  21:57:20
CPU %:     9.9     Elapsed time:   00:00:04     Active jobs:   267

Type options, press Enter.
2=Change   3=Hold   4=End   5=Work with   6=Release   7=Display message
8=Work with spooled files   13=Disconnect ...
                   Current
Opt  Subsystem/Job  User        Type  CPU %  Function        Status
       QPADEV0002   LUOWEI      INT      .0  CMD-TELNET       SELW
       QPADEV0007   LUOWEI      INT      .0  CMD-TELNET       SELW
       QPADEV0009   GUOQI        INT      .0  PGM-QP2TERM      DEQW
       QPADEV0009   GUOQI        BCI      .0  PGM-sh           THDW
       QPADEV0009   GUOQI        BCI      .0  PGM-sh           THDW
       QPADEV0009   GUOQI        BCI      .1  PGM-mysqld       SELW
       QPOZSPWT      LUOWEI      BCI      .0  PGM-QZSHCHLD     EVTW
       QPOZSPWT      LUOWEI      BCI     1.2  JVM-com.ibm.es   THDW
       QZSHSH        LUOWEI      BCI      .0  PGM-QZSHSH       EVTW
                                                                  More...
Parameters or command
===>
F3=Exit   F5=Refresh       F7=Find      F10=Restart statistics
F11=Display elapsed data   F12=Cancel   F23=More options   F24=More keys
```

Figure 9-4 MySQL server jobs shown in WRKACTJOB output panel

The job highlighted in Figure 9-4 is the IBM i system job, which runs the mysqld program (MySQL server main program). You can manage, monitor, or diagnose this job by using the normal IBM i job management techniques.

When a user or an application connects to MySQL server and issues SQL statements against tables using IBMDB2I Storage Engine, SQL Server Jobs (QSQSRVR) is used to connect to DB2 system and execute the specified SQL statements. While the connection persists, the

QSQSRVR can be found in subsystem QSYSWRK under job name QSQSRVR, which has a user profile that is same as the user of the MySQL server job. Figure 9-5 shows one of the QSQSRVR jobs that handles the database request from MySQL server.

```
 Work with Active Jobs                       G60B85AE
                                                          08/10/23  23:04:33
CPU %:     4.9     Elapsed time:   01:08:01     Active jobs:   267

Type options, press Enter.
  2=Change   3=Hold   4=End   5=Work with   6=Release   7=Display message
  8=Work with spooled files   13=Disconnect ...
                    Current
Opt  Subsystem/Job  User        Type  CPU %  Function        Status
       QSQSRVR      QSECOFR     PJ       .0                  CNDW
       QSQSRVR      QSECOFR     PJ       .0                  CNDW
       QSQSRVR      QSECOFR     PJ       .0                  CNDW
       QSQSRVR      QDIRSRV     PJ       .0                  CNDW
       QSQSRVR      QSECOFR     PJ       .0                  CNDW
       QSQSRVR      QSECOFR     PJ       .0                  CNDW
       QSQSRVR      GUOQI       PJ       .0                  CNDW
       QSVRMEVJ     QSYS        BCH      .0                  DEQW
       QSVRMSERMD   QSYS        BCI      .0  PGM-QCSTCTEXEC  SELW
                                                                More...
Parameters or command
===>
F3=Exit   F5=Refresh       F7=Find      F10=Restart statistics
F11=Display elapsed data   F12=Cancel   F23=More options   F24=More keys
```

Figure 9-5 QSQSRVR jobs shown in WRKACTJOB output panel

Note that QSQSRVR jobs are pre-start system jobs, thus when the connection between MySQL server and DB2 ends, the servicing QSQSRVR job is recycled (reset to its original state) and is ready to be reused for another connection. Therefore, the WRKACTJOB command might not list an active MySQL server connection's corresponding QSQSRVR job if the connection ends before the user issues the WRKACTJOB command.

9.4 Troubleshooting the MySQL server

The MySQL Database Server has an error log that can be used to determine what is happening inside the server during the running of queries. The MySQL trace file also provides helpful information. This section discusses the MySQL error log and trace file.

For more information about other mysql logs, see information about enabling, maintaining, and querying logs in *Discovering MySQL in IBM i5/OS*, SG24-7398.

9.4.1 Using MySQL Server error log

The MySQL error log file contains information that indicates when the server is started and stopped, and any critical errors that occur while the server is running.

You can specify where mysqld stores the error log file with the following mysqld start-up option:

```
--log-error[=file_name]
```

For example:

```
--log-error=/usr/local/mysql5126data3306/mysqld3306.err
```

When no `file_name` value is given upon starting up, mysqld uses the name `host_name.err` and creates the file in the data directory. If `G60B85AE.CN.IBM.COM` is the server name, the file is set to `G60B85AE.CN.IBM.COM.err`, which is the default name.

Figure 9-6 shows an example of how to use the error log to identify the problem. A user is trying to create a table using IBMDB2I Storage Engine. The table is a partitioned with an auto-increment column. The operation fails and the error is reported.

```
> create table myschema /mytable (
  i1 int NOT NULL auto_increment,
primary key (i1),
  dt1 datetime NOT NULL,
entry_dsc char(100),
f4 int
   )
PARTITION BY HASH (i1) PARTITIONS 8;
  ERROR 1005 (HY000): Can't create table myschema. mytable ' (errno: 1)
  $
```

Figure 9-6 A sample to generate error log

The error message says that MySQL cannot create the table, although the detailed reason of why it fails is not indicated. As this time, you can use the following command to view the MySQL error log (shown in Figure 9-7):

```
tail CSTLDB2B.CN.IBM.COM.err
```

```
> tail CSTLDB2B.CN.IBM.COM.err

081027 10:46:44 mysqld_safe Starting mysqld daemon with databases from
/usr/local/mysql5126data3306
081027 10:46:46 [Warning] Setting lower_case_table_names=2 because file system
for /usr/local/mysql5126data3306/ is case insensitive
081027 10:46:46 [Warning] One can only use the --user switch if running as root

081027 10:46:47 InnoDB: Started; log sequence number 0 25223564
081027 10:46:47 [Note] Event Scheduler: Loaded 0 events
081027 10:46:47 [Note] /mysql/libexec/mysqld: ready for connections.
Version: '5.1.26-rc-debug-log'  socket: '/tmp/mysql.sock'  port: 3306  Source
distribution
ibmdb2i error 2505: Auto_increment is not allowed for a partitioned table
```

Figure 9-7 Display the MySQL error log

The error log indicates that the underlying reason for the problem is:

```
Auto_increment is not allowed for a partitioned table
```

9.4.2 Using the MySQL traces

Generating a trace file requires that you have a debuggable version of MySQL binaries. To determine whether your installation is debuggable, check the mysqld program's version by executing the **mysqld -V** command. Your MySQL server program is capable of producing trace files if the version number ends with -debug, as shown in Figure 9-8.

```
> mysqld -V
  mysqld  Ver 5.1.26-rc-debug for ibm-i5os on power (Source distribu-tion)
```

Figure 9-8 Check mysqld program's version

To start MySQL server in debug mode with a trace file, use the command shown in Figure 9-9.

```
> /bin/mysqld_safe
--debug=d:t:o,/home/guoqi/mysql.trace
```

Figure 9-9 Start MySQL server in debug mode with a trace file

The following option causes the sever to generate a trace file in the /home/guoqi/mysql.trace directory:

```
--debug=d:t:o,/home/guoqi/mysql.trace
```

After MySQL server is started, the server puts all the trace information into the trace file. See Figure 9-10.

```
> tail mysql.trace
  081015 18:17:08 mysqld_safe mysqld from pid file
/usr/local/mysql5126data3306/cstldb2a.cn.ibm.com.pid ended
  $$
  User time 1.40, System time 0.38
  Maximum resident set size 0, Integral resident set size 0
  Non-physical pagefaults 0, Physical pagefaults 118, Swaps 0
  Blocks in 0 out 0, Messages in 0 out 0, Signals 0
  Voluntary context switches 0, Involuntary context switches 0
  | quit: signal_handler: calling my_thread_end()
  | >TERMINATE
  | | safe: Maximum memory usage: 8776368 bytes (8571k)
  | <TERMINATE
```

Figure 9-10 Sample trace file output

The trace file contains very detailed information of all the activities performed by the server and the storage engines, so the file size can grow quickly if the server is very busy. Turn on the trace only when necessary.

9.5 Troubleshooting DB2 for IBM i

Experience shows that problems with DB2 database itself are rare. Most problems that look like a DB2 problem are really functional or syntactical differences between MySQL and DB2.

For example, if your application tries to insert an all-zero date value (`0000-00-00`) into a date column of a table on IBMDB2I Storage Engine, a hard error occurs instead of a warning message. The reason is because the date value format is enforced by DB2 database and the all-zero date value is not a valid date value for DB2 tables.

Several common differences exist between IBMDB2I Storage Engine and other MySQL storage engines:

- DB2 does not accept zero-date and time values.
- DB2 always generates auto-increment values from its internal counter, instead of using the biggest value in auto-inc column plus 1.
- DB2 does not support index prefix.
- DB2 does not support using LOB columns to build indexes.

If a statement fails when you are using IBMDB2I Storage Engine but succeeds when using MyISAM or InnoDB storage engines, check the error log produced by both MySQL and DB2 (through QSQSRVR job log, discussed later in this chapter) to determine whether the error is because of a functional difference between other MySQL storage engines and IBMDB2I Storage Engine, or a real problem.

9.5.1 Database objects consideration

When you create tables and indexes when you are using IBMDB2I Storage Engine, the storage engine creates corresponding SQL tables and indexes in DB2 database system to hold the actual data.

Errors can occur if you try to create a table when you are using the IBMDB2I Storage Engine, while the same object with the same schema name already exists in the DB2 database.

In the IBM i 5.4 (formerly the i5/OS V5R4), the database name length in MySQL is limited to 10 characters (which corresponds to the schema name length limit in DB2).

Avoid using special or ideographic characters as database name or table or index names, because these characters sometimes cause errors when DB2 processing them.

IBMDB2I Storage Engine can be configured to use IASP to store the MySQL schema, table, and index object. And because IBMDB2I Storage Engine uses DB2 schema to store MySQL schema and objects, be careful about the duplicate schemas in system ASP and IASP. IBM i does not allow a schema to be created in IASP while a schema with same name exists in system ASP, and conversely, if a schema with a name exists in system ASP, a schema with the same name cannot be created in IASP.

9.5.2 QSQSRVR server jobs

QSQSRVR job is the SQL Server job that hosts the database connection from IBMDB2I Storage Engine to DB2 database system. An important concept to understand is how to find the correct QSQSRVR that is servicing your request, and how to identify whether you are having problems related to these jobs.

Identify which QSQSRVR is servicing the current MySQL connection

When a user connects to MySQL Database server and uses the tables and indexes in IBMDB2I Storage Engine, the storage engine will use the QSQSRVR server job to connect to DB2 database and process the user's request. Normally, your system will have multiple QSQSRVR jobs running in subsystem QSYSWRK. There are several ways to determine which QSQSRVR is servicing your MySQL request.

One way is to look at the job log of the running MySQL server job by working with the MySQL server job, which shows the QSQSRVR job that is being used. The job log looks similar to Figure 9-11.

```
 Display Job Log
                                                          System:   G60B85AE
Job . . :   MYSQLD3306    User . . :   GUOQI         Number . . . :   190817

    Job 177590/QUSER/QSQSRVR used for SQL server mode processing.
    Job 177808/QUSER/QSQSRVR used for SQL server mode processing.
    Job 177590/QUSER/QSQSRVR used for SQL server mode processing.
    Job 177590/QUSER/QSQSRVR used for SQL server mode processing.
    Job 177590/QUSER/QSQSRVR used for SQL server mode processing.
    Job 177590/QUSER/QSQSRVR used for SQL server mode processing.









                                                                    Bottom
Press Enter to continue.

F3=Exit   F5=Refresh   F10=Display detailed messages   F12=Cancel
F16=Job menu           F24=More keys
```

Figure 9-11 MySQL server job log

As Figure 9-11 shows, when a QSQSRVR job services a MySQL connection, its job number, job user, and job name is output to the job log of MySQL server job. You can then use WRKJOB command to work with them further.

Another way to find the right QSQSRVR job is to use i Navigator. When a connection is active, you can list all QSQSRVR jobs under **My Connection** → **You System** → **Work Management** → **Server Jobs**. The SQL details of each QSQSRVR job can be displayed by right-clicking on the job name and then selecting **Details** → **Last SQL Statement**, as shown in Figure 9-12 on page 154.

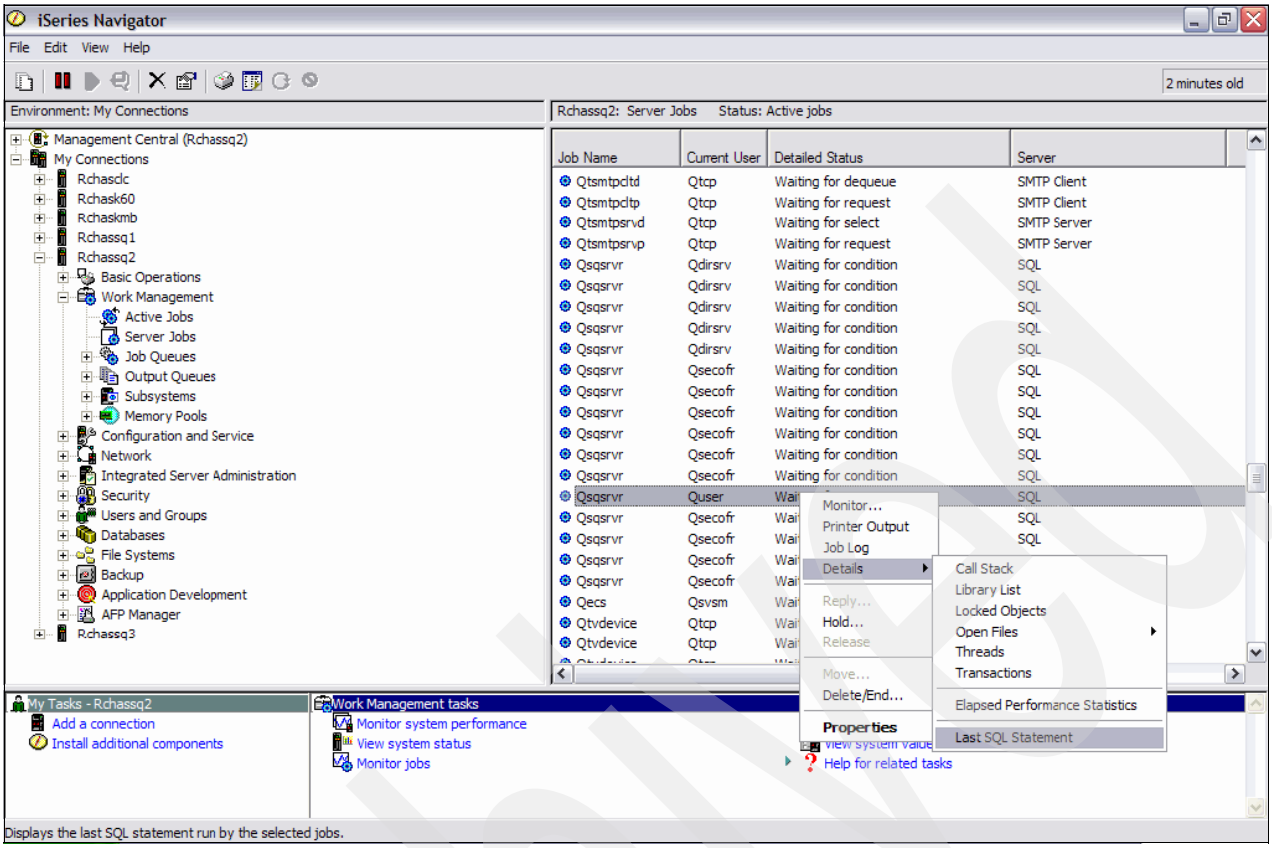


Figure 9-12 Find the last SQL statement run in a QSQSRVR job

Then, you can identify which QSQSRVR job that the server job is servicing your SQL request by comparing the SQL statements executed in MySQL and the last SQL statement run in a given QSQSRVR job.

Tip: All QSQSRVR jobs servicing MySQL connection use the user profile that is running the MySQL server. This information helps to narrow the number of jobs you have to look at.

When you have the correct servicing QSQSRVR job identified, use the IBM i Management commands to look at job logs, locking status, and other information to help you analyze the problem, if one exists.

9.5.3 QSQSRVR job logs and spool files

When an unexpected error occurs in QSQSRVR processing, a spooled job log will output to MySQL server user's output queue. To check this job log, use WRKSPLF SELECT(userid), where the userid is the user who is running the MySQL server job. Browse the spool files to see the error message and take recovery action if necessary.

9.6 Examples of troubleshooting

This section has examples that can help you understand how to troubleshoot IBMDB2I Storage Engine problems. The examples are based on techniques and information that were introduced earlier in this chapter and that can provide you with a general idea of how to debug and analyze problems.

9.6.1 Encountering a message that does not provide enough information

Error messages are always very important in analyzing problems. Usually IBMDB2I Storage Engine returns error or warning messages that have detailed information about debugging SQL statement problems. At other times, IBMDB2I Storage Engine returns general error messages that provide limited information. If this is the case, you must take further steps to obtain more detailed information. For example, suppose you create two tables, as shown in Figure 9-12 on page 154.

```
> create schema mysqltest;
  Query OK, 1 row affected (0.48 sec)
> use mysqltest;
  Database changed
> create table t1 select "02 10" as a, "%d %H" as b;
  Query OK, 1 row affected (14.55 sec)
  Records: 1  Duplicates: 0  Warnings: 0
> create table t2 select str_to_date("02 10","%d %H") from t1;
  ERROR 1030 (HY000): Got error 2021 from storage engine
```

Figure 9-13 Create a table which returns error message

An error returns when you issue the last CREATE TABLE statement. From the error message, the only thing you can find is the error number 2021, which provides an insufficient description of the error. Further investigation is necessary to learn more about the problem.

First, determine the MySQL server job that is using the Work with Active Jobs (WRKACTJOB) command, shown in Figure 9-14.

```
 Work with Active Jobs                       G60B85AE
                                                        08/10/27  17:35:04
CPU %:    13.2     Elapsed time:   00:02:34     Active jobs:   270
                    Current
Opt  Subsystem/Job  User        Type  CPU %  Function        Status
       RNRMGR       QNOTES      BCI      .0  PGM-RNRMGR       SELW
       ROUTER       QNOTES      BCI      .0  PGM-ROUTER       SELW
       SCHED        QNOTES      BCI      .0  PGM-SCHED        SELW
       SERVER       QNOTES      BCI      .1  PGM-SERVER       SELW
       STATS        QNOTES      BCI      .0  PGM-STATS         SELW
       UPDATE       QNOTES      BCI      .0  PGM-UPDATE       SELW
     MYSQLSBS       QSYS        SBS      .0                     DEQW
       MYSQLD3306   WANGYUN     BCH      .2  PGM-sh           THDW
 5     MYSQLD3306   WANGYUN     BCI      .7  PGM-mysqld       SELW
                                                                More...
```

Figure 9-14 Use DSPACTJOB to find the MySQL server job

After locating the correct MySQL server job, work with it by using option 5, as shown in Figure 9-14 on page 155.

Then, use option 10 to view the log of MySQL server job and find the QSQSRVR job name, which services your connection requests, as shown in Figure 9-15.

```
 Display Job Log
                                                          System:   G60B85AE
Job :   MYSQLD3306    User :   WANGYUN       Number :   194644

 Job 192686/QUSER/QSQSRVR used for SQL server mode processing.
```

Figure 9-15 MySQL Server job log

To work with this QSQSRVR job, use the WRKJOB command:

```
WRKJOB 192686/QUSER/QSQSRVR
```

Issuing the command opens the Work With Job menu, shown in Figure 9-16.

```
 Work with Job
                                                          System:   G60B85AE
Job:   QSQSRVR        User:   QUSER          Number:   192686

Select one of the following:

     1. Display job status attributes
     2. Display job definition attributes
     3. Display job run attributes, if active
     4. Work with spooled files

    10. Display job log, if active, on job queue, or pending
    11. Display call stack, if active
    12. Work with locks, if active
    13. Display library list, if active
    14. Display open files, if active
    15. Display file overrides, if active
    16. Display commitment control status, if active
                                More...
```

Figure 9-16 Work with QSQSRVR job

The menu offers many options to work with the job displayed. For example, with option 10, you can view the log of the QSQSRVR job, as shown in Figure 9-17 on page 157.

```
 Display Job Log
                                                              System:   G60B85AE
Job . . :   QSQSRVR       User . . :   QUSER         Number . . . :   192686

    Journal receiver QSQJRN0001 created in library "mysq0001".
    Journal QSQJRN created in library "mysq0001".
    Open of member "t1" was changed to SEQONLY(*NO).
    Data mapping error on member "t2".
    Data mapping error on member "t2".
    Data mapping error on member "t2".
```

Figure 9-17 QSQSRVR job log that holds the connection

From the QSQSRVR job log, you can get further information about why the SQL statement fails. As we can see from Figure 9-15 on page 156, a data mapping error causes the failure. To get the detailed information of the error message, place the cursor in one of the messages and press the F1 key. In this case, as shown in Figure 9-18, the cause of the failure is:

`data in a date, time, or timestamp field that is not valid.`

```
 Additional Message Information

Message ID . . . . . :   CPF5035      Severity . . . . . :   10
Message type . . . . . :   Diagnostic
Date sent  . . . . . :   08/10/27     Time sent  . . . . . :   17:51:06

Message . . . . :   Data mapping error on member "t2".
Cause . . . . . :   A data mapping error occurred on field STR_T00001 in record
number 0, record format "t2", member number 1, in member "t2" file "t2" in
library "mysq0001", because of error code 18. The error codes and their
meanings follow:
    1 -- There is data in a decimal field that is not valid.
    ......
    18 -- There is data in a date, time, or timestamp field that is not valid.
                                                              More...
```

Figure 9-18 The additional message information of QSQSRVR job

9.6.2 Finding locking conflict

Locking happens when table or data changes. For example, when you issue the SQL statements shown in Figure 9-19 on page 158, the table t2 and the row being updated are locked until the transaction is committed or rolled back.

```
Connection 1:
set autocommit=0;
create table t1 (id integer, x integer) engine = ibmdb2i;
create table t2 (b integer, a integer) engine = ibmdb2i;
insert into t1 values(0, 0), (300, 300);
insert into t2 values(0, 0), (1, 20), (2, 30);
commit;
select a,b from t2 UNION SELECT id, x from t1 FOR UPDATE;
Connection 2:
set autocommit=0;
update t2 set a=2 where b = 0; <- This statement will cause the t2 to be locked.
```

Figure 9-19 Queries that lock the object

Note: When two or more connections are trying to lock same resource in MySQL, locking contention might occur.

To determine which object is being locked by your connection, and which object your connection is trying to lock, work with the QSQSRVR job that is servicing your connection and see the status of the locks held and requested by the job. Use option 12 to work with the job locks, as shown in Figure 9-21. Figure 9-20 shows that object t2 in mysq0001 is locked.

```
 Work with Job Locks
                                                        System:   G60B85AE
Job:   QSQSRVR        User:   QUSER          Number:   192686

Job status:   ACTIVE
                              Object                          Member  ASP
Opt  Object    Library      Type         Lock     Status  Locks   Device
      "t2"     "mysq0001"   *FILE-PHY   *SHRRD   HELD    YE
```

Figure 9-20 Work with job locks

You could use option 8 to list all jobs that are locking a given object, as shown in Figure 9-21.

```
 Work with Object Locks
                                                        System:   G60B85AE
Object . . . . :   "t2"                Type . . . . . :   *FILE-PHY
  Library  . . :     "mysqltest"           ASP device . . :   *SYSBAS

Type options, press Enter.
  4=End job   5=Work with job   8=Work with job locks

Opt   Job          User         Lock      Status          Scope     Thread
  5   QSQSRVR      QUSER        *SHRRD     HELD           *JOB
                                  *SHRRD     HELD           *JOB
      QSQSRVR      QUSER        *SHRRD     HELD           *JOB
```

Figure 9-21 Work with object locks

By working with these jobs, you can release locks, see job logs, or get locking status information.

Note: For more information of object and row locking considerations, see Chapter 6, “Transaction management and locking considerations” on page 93.

9.6.3 First Failure Data Capture

Sometimes, a hard error occurs, which creates unexpected function checks on IBM i. IBMDB2I Storage Engine reports such errors as error code 2021. In this case, you can report the error by sending the First Failure Data Capture (FFDC) to your service representative for further problem analysis and reporting.

For illustrative purposes, the following example shows what such a message might look like in the MySQL server log:

```
ibmdb2i error 2021: See message MCH1869 in joblog for job 385613/QUSER/QSQSRVR
```

This example error was encountered on a SELECT statement. The MCH1869 error can be found in the job log for the QSQSRVR job that services the request. See Figure 9-22.

```
Job : QSQSRVR   User : QUSER    Number : 385613
    Job 385613/QUSER/QSQSRVR started on 10/24/08 at 20:25:54 in sub-system
    QSYSWRK in QSYS. Job entered system on 10/24/08 at 20:25:54.
    Open of member "t1" was changed to SEQONLY(*NO).
    Job 385613/QUSER/QSQSRVR held by user KRS with option SPLFILE(*NO).
    Job 385613/QUSER/QSQSRVR released by user KRS.
    Frogger Array Template had an error, the error type is 16.
```

Figure 9-22 QSQSRVR job log

For error 2021, the diagnostic FFDC data is reported in the MySQL server job. For the example error, the MySQL server job log indicates where the FFDC data is dumped. See Figure 9-23.

```
Job : QPADEV0002  User : KRS     Number : 385615
     Job 385613/QUSER/QSQSRVR used for SQL server mode processing.
     Software problem data for QMYSE has been detected.
     Dump output directed to spooled file 3, job 385615/KRS/QPADEV0002 created
on system LPO3UT9 on 10/24/08 20:29:07.
     An internal error was detected while processing a DRDA request.
```

Figure 9-23 MySQL Server job with FFDC data dumped

The Work Job (WRKJOB) command for the MySQL server job 385615/KRS/QPADEV0002, option 4 lists the spooled files that contain the diagnostic data that will be useful to IBM to analyze the problem. See Figure 9-24 on page 160.

```
Job:   QPADEV0002     User:   KRS            Number:   385625

Type options, press Enter.
  1=Send   2=Change   3=Hold   4=Delete   5=Display   6=Release   7=Messages
8=Attributes        9=Work with printing status

                  Device or                          Total   Current
Opt  File         Queue       User Data   Status  Pages     Page   Copies
     QPSRVDMP     QEZDEBUG    FFDC          RDY       3                1
     QPSRVDMP     QEZDEBUG    FFDC          RDY       7                1
    QPSRVDMP   QEZDEBUG   FFDC       RDY      3            1
```

Figure 9-24 Dumps of FFDC in spool files

You may collect these spool files and send them to your service representative.

9.7 Error codes and messages

In general, the IBMDB2I Storage Engine uses error codes in the 2000 range. Errors over 2500 are storage engine errors; errors under 2500 are DB2 for i errors.

Note: The tables in this section use %d and %s to represent numbers and strings, respectively, that are substituted into the message values when they are displayed

Table 9-1 lists codes and messages for errors that occur in DB2 for i that are reported by the IBMDB2I Storage Engine.

Table 9-1 Error codes in DB2 for i that are reported by IBMDB2I Storage Engine

Error code number	Error message text
0	Successful
2016	Thread ID is too long
2017	Error creating a SPACE memory object
2018	Error creating a FILE memory object
2019	Error creating a SPACE synchronization token
2020	Error creating a FILE synchronization token
2021	See message %-.7s in joblog for job %-.6s/%-.10s/%-.10s.
2022	Error unlocking a synchronization token when closing a connection
2023	Invalid action specified for an object lock request
2024	Invalid action specified for a savepoint request
2025	Partial keys are not supported with an ICU sort sequence
2026	Error retrieving an ICU sort key
2027	Error converting single-byte sort sequence to UCS-2

Error code number	Error message text
2028	An unsupported collation was specified
2029	Validation failed for referenced table of foreign key constraint
2030	Error extracting table for constraint information
2031	Error extracting referenced table for constraint information
2032	Invalid action specified for a commitment control request
2033	Invalid commitment control isolation level specified on open request
2034	Invalid file handle
2036	Invalid option specified for returning data on read request
2037	Invalid orientation specified for read request
2038	Invalid option type specified for read request
2039	Invalid isolation level for starting commitment control
2040	Error unlocking a synchronization token in module QMYALC
2041	Length of space for returned format is not long enough
2042	SQL XA transactions are currently unsupported by this interface
2043	The associated QSQSRVR job was killed or ended unexpectedly.
2044	Error unlocking a synchronization token in module QMYSEI
2045	Error unlocking a synchronization token in module QMYSPO
2046	Error converting input CCSID from short form to long form
2048	Error getting associated CCSID for CCSID conversion
2049	Error converting a string from one CCSID to another
2050	Error unlocking a synchronization token
2051	Error destroying a synchronization token
2052	Error locking a synchronization token
2053	Error recreating a synchronization token
2054	A space handle was not specified for a constraint request
2055	An SQL cursor was specified for a delete request
2057	Error on delete request because current UFCB for connection is not open
2058	An SQL cursor was specified for an object initialization request
2059	An SQL cursor was specified for an object override request
2060	A space handle was not specified for an object override request
2061	An SQL cursor was specified for an information request
2062	An SQL cursor was specified for an object lock request
2063	An SQL cursor was specified for an optimize request

Error code number	Error message text
2064	A data handle was not specified for a read request
2065	A row number handle was not specified for a read request
2066	A key handle was not specified for a read request
2067	An SQL cursor was specified for an row estimation request
2068	A space handle was not specified for a row estimation request
2069	An SQL cursor was specified for a release record request
2070	A statement handle was not specified for an execute immediate request
2071	A statement handle was not specified for a prepare open request
2072	An SQL cursor was specified for an update request
2073	The UFCB was not open for read
2074	Error on update request because current UFCB for connection is not open
2075	A data handle was not specified for an update request
2076	An SQL cursor was specified for a write request
2077	A data handle was not specified for a write request
2078	An unknown function was specified on a process request
2079	A share definition was not specified for an allocate share request
2080	A share handle was not specified for an allocate share request
2081	A use count handle was not specified for an allocate share request
2082	A records per key handle was not specified for an information request
2083	Error resolving LOB address
2084	Length of a LOB space is too small
2085	An unknown function was specified for a server request
2086	Object authorization failed. See message %-.7s in joblog for job %-.6s/%-.10s/%-.10s. for more information.
2088	Error locking mutex on server
2089	Error unlocking mutex on server
2090	Error checking for RDB name in RDB Directory
2091	Error creating mutex on server
2094	Error unlocking mutex
2095	Error connecting to server job
2096	Error connecting to server job
2098	Function check occurred while registering parameter spaces. See job log.
2101	End of block

Error code number	Error message text
2102	The file has changed and might not be compatible with the MySQL table definition
2103	Error giving pipe to server job
2104	There are open object locks when attempting to deallocate
2105	There is no open lock
2108	The maximum value for the auto_increment data type was exceeded
2109	Error occurred closing the pipe
2110	Error occurred taking a descriptor for the pipe
2111	Error writing to pipe
2112	Server was interrupted
2113	No pipe descriptor exists for reuse
2114	Error occurred during an SQL prepare statement
2115	Error occurred during an SQL open
2122	An unspecified error was returned from the system.

Table 9-2 lists error codes and messages that occur in the IBMDB2I Storage Engine.

Table 9-2 Error codes in IBMDB2I Storage Engine

Error code number	Error message text
2501	Error opening codeset conversion from %.64s to %.64s (errno = %d
2502	Invalid %-.10s name %-.128s
2503	Unsupported move from %-.128s to %-.128s on RENAME TABLE statement
2504	Unsupported schema %-.128s specified on RENAME TABLE statement
2505	Auto_increment is not allowed for a partitioned table
2506	Character set conversion error because of an unknown encoding scheme %d
2508	Table %-.128s was not found by the storage engine
2509	Could not resolve to %-.128s in library %-.10s type %-.10s (errno = %d)
2510	Error on _PGMCALL for program %-.10s in library %-.10s (error = %d)
2511	Error on _ILECALL for API %.128s (error = %d)
2512	Error in iconv() function during character set conversion (errno = %d)
2513	Error from Get Encoding Scheme (QTQGESP) API: %d, %d, %d
2514	Error from Get Related Default CCSID (QTQGRDC) API: %d, %d, %d
2515	Invalid value %-.128s for column %.192s
2516	Schema name %.128s exceeds maximum length of %d characters
2517	Multiple collations not supported in a single index or constraint

2518	Sort sequence was not found
2519	One or more characters in column %.128s were substituted during conversion
2520	A decimal column exceeded the maximum precision. Data may be truncated.
2521	Some data returned by DB2 for table %s could not be converted for MySQL
2523	Column %.128s contains characters that cannot be converted
2524	An invalid name was specified for ibmdb2i_rdb_name.
2525	A duplicate key was encountered for index %.128s
2528	Certain attributes defined for column %.128s may not be honored by accesses from DB2.

9.8 Resources for troubleshooting

The following resources and materials can help you to debug and analyze problems:

- IBM i Information Center: Troubleshooting

  http://publib.boulder.ibm.com/infocenter/iseries/v5r4/index.jsp?topic=/rzahb/rzahbrtrbshoo1.htm

- IBM i Information Center: Managing jobs

  http://publib.boulder.ibm.com/infocenter/iseries/v5r4/index.jsp?topic=/rzaks/rzaksmanagingjobs.htm

- IBM i Information Center: Job log introduction

  http://publib.boulder.ibm.com/infocenter/iseries/v5r4/index.jsp?topic=/rbam6/jblog.htm

- *Discovering MySQL in IBM i5/OS*, SG24-7398

  http://www.redbooks.ibm.com/abstracts/sg247398.html

10

Performance considerations and settings

In this chapter, we discuss the performance considerations of MySQL and DB2. We also discuss performance settings and tools for MySQL, index considerations, and performance settings for database server jobs.

We will look at the MySQL side of the performance as well as the more traditional DB2 side of the performance.

This chapter contains the following topics:

10.1 MySQL performance considerations and settings

In this section we discuss tools that are available for addressing performance in MySQL.

10.1.1 Logs

Logs that can be helpful in performance analysis are:

- General query log
- Slow query log

The general query log

The general query log is a general record of what mysqld (MySQL server) is doing. The server writes information to this log when clients connect or disconnect, and it logs each SQL statement received from clients. The general query log can be very useful when you suspect an error in a client and want to know exactly what the client sent to mysqld.

The MySQL server (mysqld) writes statements to the query log in the order that it receives them, which might differ from the order in which they are executed. This logging order contrasts to the binary log, to which statements are written after they are executed but before any locks are released. The query log contains all statements, whereas the binary log does not contain statements that only select data.

For more information about the general query log, see:

http://dev.mysql.com/doc/refman/5.1/en/query-log.html

The slow query log

The slow query log consists of all SQL statements that took more than long_query_time seconds to execute and (as of MySQL 5.1.21) required at least min_examined_row_limit rows to be examined. The time to acquire the initial table locks is not counted as execution time. mysqld writes a statement to the slow query log after it has been executed and after all locks have been released, so log order might be different from execution order. The minimum and default values of long_query_time are 1 and 10, respectively. Prior to MySQL 5.1.21, the minimum value is 1, and the value for this variable must be an integer. Beginning with MySQL 5.1.21, the minimum is 0, and a resolution of microseconds is supported when logging to a file. However, the microseconds part is ignored and only integer values are written when logging to tables.

For more information about the slow query log, see:

http://dev.mysql.com/doc/refman/5.1/en/slow-query-log.html

10.1.2 EXPLAIN

The EXPLAIN keyword is also a text tool that can be used to get more information about a SQL request. It is not a very high level tool, but it can help you learn more about your SQL request and how the MySQL is running the query.

In a MySQL session, you place the EXPLAIN keyword in front of the SQL request. For example, to obtain more information about the SQL request that is performing poorly, you might enter something similar to Example 10-1 on page 167.

Example 10-1 Obtain more information about SQL request

```
explain SElect distinct a.Name, b.Name, b.Code, b.LifeExpectancy, c.IsOfficial
from WORLD.City a inner join WORLD.Country b inner join      WORLD.CountryLanguage
c where a.CountryCode = b.Code and a.CountryCode = c.Countrycode and c.IsOfficial
= 2 and b.LifeExpectancy > 70 and            a.Name in (select c.Name c from
WORLD.City c where c.Name = 'Ratingen');
```

This produces information similar to Example 10-2. You might have to use function key F11 to switch between truncating and wrapping the information.

Example 10-2 MySQL EXPLAIN output - left side of view

```
explain SElect distinct a.Name, b.Name, b.Code, b.LifeExpectancy, c.IsOfficial from WORLD.City a inner join WORLD.Country b inner
+----+--------------------+-------+------+-----------------------+------+---------+--------------------+------+-----------------
| id | select_type        | table | type | possible_keys         | key  | key_len | ref                | rows | Extra
+----+--------------------+-------+------+-----------------------+------+---------+--------------------+------+-----------------
|  1 | PRIMARY            | b     | ALL  | PRIMARY,IX02,IX06,IX07 | NULL | NULL    | NULL               |  239 | Using where; Usi
|  1 | PRIMARY            | c     | ref  | PRIMARY,IX01,IX03,IX08 | IX01 | 4       | const,WORLD.b.Code |    2 | Using where; Usi
|  1 | PRIMARY            | a     | ref  | IX04,IX09             | IX04 | 3       | WORLD.b.Code       |   17 | Using where; Usi
|  2 | DEPENDENT SUBQUERY | c     | ref  | IX05,IX10             | IX05 | 35      | const              |    1 | Using where; Usi
+----+--------------------+-------+------+-----------------------+------+---------+--------------------+------+-----------------
4 rows in set (0.01 sec)
```

You might also use function key F20 to move the window to the right to view the remaining information, as shown in Example 10-3.

Example 10-3 MySQL EXPLAIN output - right side of view; using F20 to show more information

```
join      WORLD.CountryLanguage c where a.CountryCode = b.Code and a.CountryCode = c.Countrycode and c.IsOfficial   = 2 and b.Li
-------------+
             |
-------------+
ng temporary |
ng index     |
ng index     |
ng index     |
-------------+
```

The EXPLAIN output shows information about the indexes available for the query and what indexes are used. You can also see in the extra information that the MySQL optimizer decided to create a temporary index to fulfill the request.

The possible_keys column in the output from the EXPLAIN statement lists the indexes MySQL can choose from when fulfilling the query. If the value in this column is NULL, MySQL has no indexes to use for the request or the indexes that are available are not considered useful. Therefore, if the value is NULL, additional indexes will very likely be useful for the query.

10.1.3 EXPLAIN tbl_name

The EXPLAIN tbl_name command is similar to DESCRIBE tbl_name and SHOW COLUMNS FROM tbl_name commands as illustrated in Figure 10-1 on page 168.

```
 /QOpenSys/usr/bin/-sh

explain City;
+-------------+----------+------+-----+---------+----------------+
| Field       | Type     | Null | Key | Default | Extra          |
+-------------+----------+------+-----+---------+----------------+
| ID          | int(11)  | NO   | PRI | NULL    | auto_increment |
| Name        | char(35) | NO   | MUL |         |                |
| CountryCode | char(3)  | NO   | MUL |         |                |
| District    | char(20) | NO   |     |         |                |
| Population  | int(11)  | NO   |     | 0       |                |
+-------------+----------+------+-----+---------+----------------+
5 rows in set (0.01 sec)

describe City;
+-------------+----------+------+-----+---------+----------------+
| Field       | Type     | Null | Key | Default | Extra          |
+-------------+----------+------+-----+---------+----------------+
| ID          | int(11)  | NO   | PRI | NULL    | auto_increment |
| Name        | char(35) | NO   | MUL |         |                |
| CountryCode | char(3)  | NO   | MUL |         |                |
| District    | char(20) | NO   |     |         |                |
| Population  | int(11)  | NO   |     | 0       |                |
+-------------+----------+------+-----+---------+----------------+
5 rows in set (0.01 sec)
SHOW COLUMNS FROM City;
+-------------+----------+------+-----+---------+----------------+
| Field       | Type     | Null | Key | Default | Extra          |
+-------------+----------+------+-----+---------+----------------+
| ID          | int(11)  | NO   | PRI | NULL    | auto_increment |
| Name        | char(35) | NO   | MUL |         |                |
| CountryCode | char(3)  | NO   | MUL |         |                |
| District    | char(20) | NO   |     |         |                |
| Population  | int(11)  | NO   |     | 0       |                |
+-------------+----------+------+-----+---------+----------------+
5 rows in set (0.01 sec)
```

Figure 10-1 Use of Explain, Describe, and Show columns from tbl_name

The commands are simply used to show the definition of the columns in a table.

10.1.4 ANALYZE TABLE

The ANALYZE TABLE command is used to analyze and store the key distribution for a table. Information like cardinality is collected and stored.

The command can be useful when a table has had a large update.

10.1.5 OPTIMIZE TABLE

The OPTIMIZE TABLE command is running the CL command Reorganize Physical File Member (RGZPFM) under the cover, so an exclusive lock is necessary to execute the command.

10.1.6 SHOW INDEX

The SHOW INDEX command shows information about the indexes over a table. For example,Example 10-4 shows the results of issuing the following MySQL command:

```
show index from City;
```

Example 10-4 SHOW INDEX FROM table example

```
show index from City;
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
| City  |          0 | PRIMARY  |            1 | ID          | A         |        4079 |     NULL | NULL   |      | RADIX      |
| City  |          1 | IX04     |            1 | CountryCode | A         |         239 |     NULL | NULL   |      | RADIX      |
| City  |          1 | IX04     |            2 | Name        | A         |        4079 |     NULL | NULL   |      | RADIX      |
| City  |          1 | IX05     |            1 | Name        | A         |        4079 |     NULL | NULL   |      | RADIX      |
| City  |          1 | IX09     |            1 | CountryCode | A         |         239 |     NULL | NULL   |      | RADIX      |
| City  |          1 | IX09     |            2 | Name        | A         |        4079 |     NULL | NULL   |      | RADIX      |
| City  |          1 | IX10     |            1 | Name        | A         |        4079 |     NULL | NULL   |      | RADIX      |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
7 rows in set (0.05 sec)
```

For each key column, you see one line in the output. As you can see, the Cardinality column is a combined value of the columns used in the index.

10.1.7 SHOW VARIABLES

When you investigate issues with the performance, understanding the different system settings is helpful. The SHOW VARIABLES command lists the settings, such as for the IBMDB2I settings shown in Figure 10-2.

```
show variables;
+-----------------------------------+----------------------------------
| Variable_name                     | Value
+-----------------------------------+----------------------------------
| auto_increment_increment          | 1
| auto_increment_offset             | 1
| autocommit                        | ON
| automatic_sp_privileges           | ON
| back_log                          | 50

| ibmdb2i_assume_exclusive_use      | OFF
| ibmdb2i_async_enabled             | ON
| ibmdb2i_create_index_option       | 0
| ibmdb2i_compat_opt_time_as_duration | ON
| ibmdb2i_lob_alloc_size            | 2097152
| ibmdb2i_compat_opt_blob_cols      |0
| ibmdb2i_max_read_buffer_size      | 1048576
| ibmdb2i_max_write_buffer_size     | 8388608
| ibmdb2i_rdb_name                  |
| ibmdb2i_transaction_unsafe        | OFF
```

Figure 10-2 Several of the SHOW VARIABLES output parameters

As the figure shows, the automatic creation of *HEX indexes is set to `OFF`. If you access the MySQL data from DB2, consider setting this to `ON` to get the best performance.

10.1.8 Isolation level

The isolation level has a significant influence on the performance. Less locking results in better performance. The isolation levels are discussed in chapter Chapter 6, “Transaction management and locking considerations” on page 93.

10.1.9 Index hint

In MySQL, you can provide hints to the optimizer about which indexes to use and not to use.

You can specify USE INDEX (<index list>) to tell MySQL to use only one or some of the named indexes to extract rows from the table. You can also use IGNORE INDEX (<index list>) to tell MySQL not to use some indexes.

You can always control how the optimizer will use the indexes by the EXPLAIN command as described in 10.1.2, “EXPLAIN” on page 166. The following example is of a query:

```
explain select * from Country where Continent = 'Asia' and Region like 'S%' order
by population;
```

The output indicates possible indexes and which indexes are in use. See Example 10-5.

Example 10-5 Use of EXPLAIN

```
explain select * from Country where Continent = 'Asia' and Region like 'S%' order by population;
+----+-------------+---------+------+---------------+--------+---------+-------+------+-----------------------------+
| id | select_type | table   | type | possible_keys | key    | key_len | ref   | rows | Extra                       |
+----+-------------+---------+------+---------------+--------+---------+-------+------+-----------------------------+
|  1 | SIMPLE      | Country | ref  | INDEX2,INDEX3 | INDEX3 | 1       | const |   51 | Using where; Using filesort |
+----+-------------+---------+------+---------------+--------+---------+-------+------+-----------------------------+
1 row in set (0.00 sec)
```

The example shows that INDEX3 is used. If we want to force the optimizer to use INDEX2, then we include USE INDEX in the query:

```
explain select * from Country USE INDEX (INDEX2) where Continent = 'Asia' and
Region like 'S%' order by population;
```

We have forced the optimizer to use INDEX2. We run the EXPLAIN command again. Results are in Example 10-6.

Example 10-6 USE INDEX example

```
explain select * from Country USE INDEX (INDEX2) where Continent = 'Asia' and Region like 'S%' order by population;
+----+-------------+---------+------+---------------+------+---------+------+------+-----------------------------+
| id | select_type | table   | type | possible_keys | key  | key_len | ref  | rows | Extra                       |
+----+-------------+---------+------+---------------+------+---------+------+------+-----------------------------+
|  1 | SIMPLE      | Country | ALL  | INDEX2        | NULL | NULL    | NULL |  239 | Using where; Using filesort |
+----+-------------+---------+------+---------------+------+---------+------+------+-----------------------------+
1 row in set (0.01 sec)
```

Specifying an empty index_list for USE INDEX is valid, and means that no indexes are used and a table scan is done. We can try this by running the following EXPLAIN command:

```
explain select * from Country USE INDEX () where Continent = 'Asia' and Region
like 'S%' order by population;
```

In this case, we forced the optimizer not to use indexes. See Example 10-7 on page 171.

Example 10-7 Force optimizer to table scan

```
explain select * from Country USE INDEX () where Continent = 'Asia' and Region like 'S%' order by population;
+----+-------------+---------+------+---------------+------+---------+------+------+-----------------------------+
| id | select_type | table   | type | possible_keys | key  | key_len | ref  | rows | Extra                       |
+----+-------------+---------+------+---------------+------+---------+------+------+-----------------------------+
|  1 | SIMPLE      | Country | ALL  | NULL          | NULL | NULL    | NULL |  239 | Using where; Using filesort |
+----+-------------+---------+------+---------------+------+---------+------+------+-----------------------------+
1 row in set (0.00 sec)
```

You may also tell the optimizer to ignore a specific index by using IGNORE INDEX:

```
explain select * from Country IGNORE INDEX (INDEX3) where Continent = 'Asia' and
Region like 'S%' order by population;
```

Example 10-8 shows that the optimizer is not using INDEX3. Although the optimizer can use INDEX3, it made a table scan as indicated by the NULL value in the key column.

Example 10-8 Force optimizer to ignore index

```
explain select * from Country IGNORE INDEX (INDEX3) where Continent = 'Asia' and Region like 'S%' order by population;
+----+-------------+---------+------+---------------+------+---------+------+------+-----------------------------+
| id | select_type | table   | type | possible_keys | key  | key_len | ref  | rows | Extra                       |
+----+-------------+---------+------+---------------+------+---------+------+------+-----------------------------+
|  1 | SIMPLE      | Country | ALL  | INDEX2        | NULL | NULL    | NULL |  239 | Using where; Using filesort |
+----+-------------+---------+------+---------------+------+---------+------+------+-----------------------------+
1 row in set (0.00 sec)
```

You may specify the scope of a index hint by adding a FOR clause to the hint. This clause provides even more control over the optimizer's selection of an access plan for the different parts of the query. To affect only the indexes used when MySQL decides how to find rows in the table and how to process joins, use FOR JOIN.

In MySQL, you have a number of ways to influence the query optimizer; you have a number of ways to influence queries that are running less efficiently than expected.

For more information, see:

http://dev.mysql.com/doc/refman/5.1/en/index-hints.html

10.1.10 SELECT BENCHMARK

The SELECT BENCHMARK can help you determine the general speed of a partition that is running MySQL at a certain point in time. You can run the following request:

```
SELECT BENCHMARK(1000000,1+1);
```

The result is that MySQL performs one million simple additions, which can help understand the speed of the partition in which the job is running. Figure 10-3 shows that the million additions takes 0.12 seconds.

```
SELECT BENCHMARK(1000000,1+1);
+------------------------+
| BENCHMARK(1000000,1+1) |
+------------------------+
|                      0 |
+------------------------+
1 row in set (0.12 sec)
```

Figure 10-3 Benchmark results

The SELECT BENCHMARK cannot be used to check how many SQL requests that you can execute or similar tests. However, it can indicate that at a certain point in time the partition is slower than another time and might be caused by other jobs running in the partition.

10.2 DB2 performance considerations and settings

When accessing the data from tools other than MySQL, be aware of various tips to give your queries the best performance.

10.2.1 Creating specific indexes for DB2

Create specific indexes because:

- The DB2 optimizer often cannot use the ASCII indexes from MySQL.
- The DB2 optimizer might work differently than the MySQL optimizer, and might suggest other useful indexes.

If you are querying the MySQL data from any DB2 SQL-based tool, you should activate the ibmdb2i_create_index_option in the configuration options for IBMDB2I. The values are:

0 This setting is the default; it does not create additional indexes.
1 This setting creates additional indexes.

Those indexes have the *HEX sorting sequence and can help the optimizer and the database engine analyze and execute the queries coming from DB2. You can see an example of the additional indexes created because of the setting of the option in Figure 10-4.

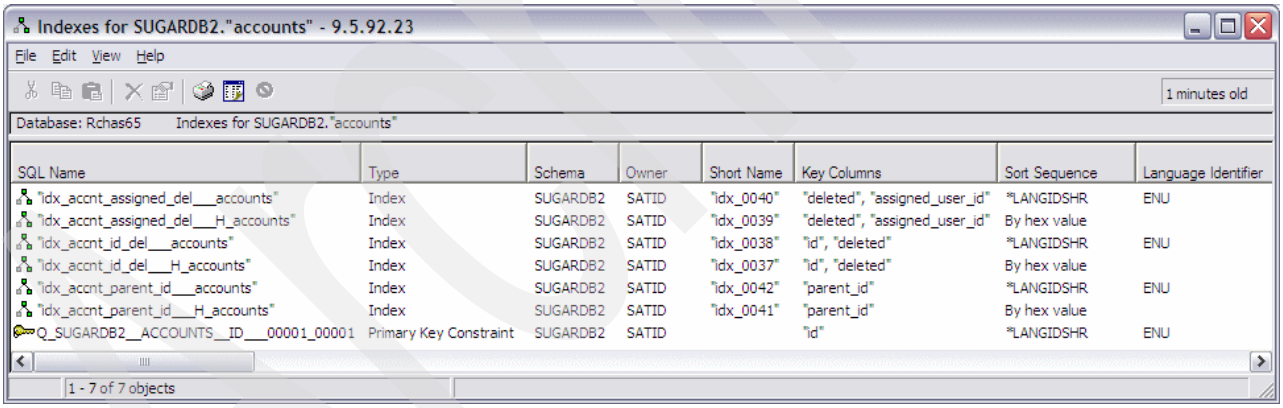


*Figure 10-4 Showing *HEX indexes*

10.2.2 DB2 optimization for tables and indexes created with MySQL

In DB2 for System i we have two types of indexes, which are radix indexes and encoded vector indexes. Because MySQL generates only radix indexes, we should always be aware of which possible encoded vector indexes to create, so that DB2 access to the data can be improved.

For more information about indexing strategy and performance considerations, see *OnDemand SQL Performance Analysis Simplified on DB2 for i5/OS in V5R4*, SG24-7326.

10.2.3 General performance settings for the QSQSRVR jobs

The database server jobs used by the MySQL connections are QSQSRVR jobs. You can optimize the settings for those jobs, but it works for all jobs regardless of where they are from.

To see an overview of the use of the QSQSRVR prestarted jobs with the CL command Display Active Prestart Jobs (DSPACTPJ), issue the following command:

```
DSPACTPJ SBS(QSYSWRK) PGM(QSQSRVR)
```

This command provides information, such as the number of jobs running, as shown in Figure 10-5.

```
Display Active Prestart Jobs RCHAS65
                                                          10/15/08  16:59:21
Subsystem  . . . . . :   QSYSWRK        Reset date . . . . . :   10/08/08
Program  . . . . . :   QSQSRVR          Reset time . . . . . :   09:46:58
  Library  . . . . . :     QSYS         Elapsed time . . . . :   0175:12:23

Prestart jobs:
  Current number . . . . . . . . . . . . . . . . :   29
  Average number . . . . . . . . . . . . . . . . :   27.4
  Peak number  . . . . . . . . . . . . . . . . . :   32

Prestart jobs in use:
  Current number . . . . . . . . . . . . . . . . :   23
  Average number . . . . . . . . . . . . . . . . :   20.7
  Peak number  . . . . . . . . . . . . . . . . . :   29




                                                                      More...
Press Enter to continue.

F3=Exit   F5=Refresh   F12=Cancel   F13=Reset statistics
```

Figure 10-5 DSPACTPJ for the QSQSRVR jobs (first page)

On the second page, shown in Figure 10-6 on page 174, you will get information about jobs that are waiting and how many connections have been made since the last IPL or reset of the statistic, as Figure 10-6 on page 174 shows.

```
Display Active Prestart Jobs                RCHAS65
                                                          10/15/08  16:59:21
Subsystem  . . . . . :   QSYSWRK         Reset date . . . . . :   10/08/08
Program  . . . . . :   QSQSRVR          Reset time . . . . . :   09:46:58
  Library  . . . . . :     QSYS          Elapsed time . . . . :   0175:12:23

Program start requests:
  Current number waiting . . . . . :   0
  Average number waiting . . . . . :   .0
  Peak number waiting  . . . . . :   1
  Average wait time  . . . . . :   00:00:00.0
  Number accepted  . . . . . :   7412
  Number rejected  . . . . . :   0

                                                                   Bottom
Press Enter to continue.

F3=Exit   F5=Refresh   F12=Cancel   F13=Reset statistics
```

Figure 10-6 DSPACTPJ for the QSQSRVR jobs (second page)

Ensure that the peak number of jobs waiting is always 0 by changing the number of jobs to be pre-started, and the threshold. To view the current setting in the subsystem description, use the Display Subsystem Description (DSPSBSD) CL command:

```
DSPSBSD SBSD(QSYSWRK)
```

Use option 10 on the Display Subsystem Description menu to display the prestart job entries; then use option 5 in front of the QSQSRVR program, as shown in Figure 10-7 on page 175.

```
Display Prestart Job Entries
                                                          System:   RCHAS65
Subsystem description:   QSYSWRK        Status:   ACTIVE

Type options, press Enter.
  5=Display details

Opt     Program        Library        User Profile
        QANEAGNT       QSYS            QUSER
        QIWVPPJT       QIWS            QUSER
 5      QSQSRVR        QSYS            QUSER
        QSRRATBL       QSYS            QUSER
        QSRSYNCM       QSYS            QUSER
        QTMMSRVR       QTCP            QTCP
        QTMSCLCP       QTCP            QTCP
        QTMSSRCP       QTCP            QTCP
        Q5BWHSRV       QSYS            QUSER




                                                                      Bottom
F3=Exit   F9=Display all detailed descriptions   F12=Cancel
```

Figure 10-7 Display the prestarted jobs for the QSQSRVR subsystem

The start-up settings are shown in Figure 10-8.

```
Display Prestart Job Entry Detail
                                                          System:   RCHAS65
Subsystem description:   QSYSWRK        Status:   ACTIVE

Program  . . . . . :   QSQSRVR
  Library  . . . . . :     QSYS
User profile . . . . . :   QUSER
Job  . . . . . :   QSQSRVR
Job description  . . . . . :   QDFTSVR
  Library  . . . . . :     QGPL
Start jobs . . . . . :   *NO
Initial number of jobs . . . . . :   5
Threshold  . . . . . :   2
Additional number of jobs  . . . . . :   2
Maximum number of jobs . . . . . :   *NOMAX
Maximum number of uses . . . . . :   200
Wait for job . . . . . :   *YES
Pool identifier  . . . . . :   1

                                                                      More...
Press Enter to continue.

F3=Exit   F12=Cancel   F14=Display previous entry
```

Figure 10-8 Display QSQSRVR prestart job entry

By knowing the number of prestarted jobs that are used, you will have an idea about how many to start up. Do not start a large number of prestarted jobs at the same time; a better

approach is to start them only when they are needed. You can control the number by having a higher threshold than the default. If you set the initial number of jobs to 20, the threshold to 10 and the additional number of jobs to start to 10, you will have a higher buffer than the default settings. You can change the actual settings by using the following command:

```
CHGPJE SBSD(QSYSWRK) PGM(QSQSRVR) INLJOBS(20) THRESHOLD(10) ADLJOBS(10)
```

To provide better conditions for the database server jobs, you may also change the other work management settings for the QSQSRVR jobs.

Run-priority and time slice are kept in the class for the QSQSRVR job. The default class used for the QSQSRVR job is QSYSCLS20. Figure 10-9 shows the class information with the CL command Display Class (DSPCLS).

```
 Display Class Information
                                                          System:   RCHAS65
 Class . . . . . . . . . . . . . . . . . . . . . . . :   QSYSCLS20
   Library . . . . . . . . . . . . . . . . . . . . . :     QSYS
 Run priority  . . . . . . . . . . . . . . . . . . . :   20
 Time slice in milliseconds  . . . . . . . . . . . . :   2000
 Eligible for purge  . . . . . . . . . . . . . . . . :   *YES
 Default wait time in seconds  . . . . . . . . . . . :   30
 Maximum CPU time in milliseconds  . . . . . . . . . :   *NOMAX
 Maximum temporary storage in megabytes  . . . . . . :   *NOMAX
 Maximum threads . . . . . . . . . . . . . . . . . . :   *NOMAX
 Text  . . . . . . . . . . . . . . . . . . . . . . . :   SYSTEM SUBSYSTEM CLASS WITH RUN
PRIORITY 20
```

Figure 10-9 Showing the QSYSCLS20 class

If you want to run with a different priority or run with a different time slice, create a new class that accommodates the needs, by using the following commands:

```
CRTCLS CLS(QGPL/DBCLS19) RUNPTY(19) TIMESLICE(500) TEXT('My class for QSQSRVR
jobs')
```

To activate the class for new QSQSRVR jobs starting up, change the prestart job setting for QSQSRVR by using the following command:

```
CHGPJE SBSD(QSYSWRK) PGM(QSQSRVR) CLS(DBCLS19)
```

To protect the QSQSRVR database server jobs, create a dedicated memory pool for the jobs. Protection is especially important when many other jobs are running in your partition.

First, specify how much memory should be allocated to the memory pool. You can do that by the following command where we allocate 2 GB of memory to the shared memory pool 11:

```
CHGSHRPOOL POOL(*SHRPOOL11) SIZE(2048000) ACTLVL(20) PAGING(*CALC) TEXT('Pool for
QSQSRVR jobs')
```

If you then want the subsystem QSYSWRK to use the shared pool, change that in the subsystem description:

```
CHGSBSD ?*SBSD(QSYS/QSYSWRK) POOLS((2 *SHRPOOL11))
```

Then you have to specify the jobs using that memory pool. If this is only the QSQSRVR jobs, you can simply change the pre started job:

```
CHGPJE SBSD(QSYSWRK) PGM(QSQSRVR) POOLID(2)
```

A

Tool to look up DB2 SQL and system names

This appendix describes a tool you can use to look up the DB2 SQL and system names.

This appendix contains the following topics:

Accessing the tool

You access the tool at the following address (<system name> is the IBM i system name):

```
http://<system name>:89/sqlSystemNaming.php
```

The PHP program opens, as shown in Figure A-1.

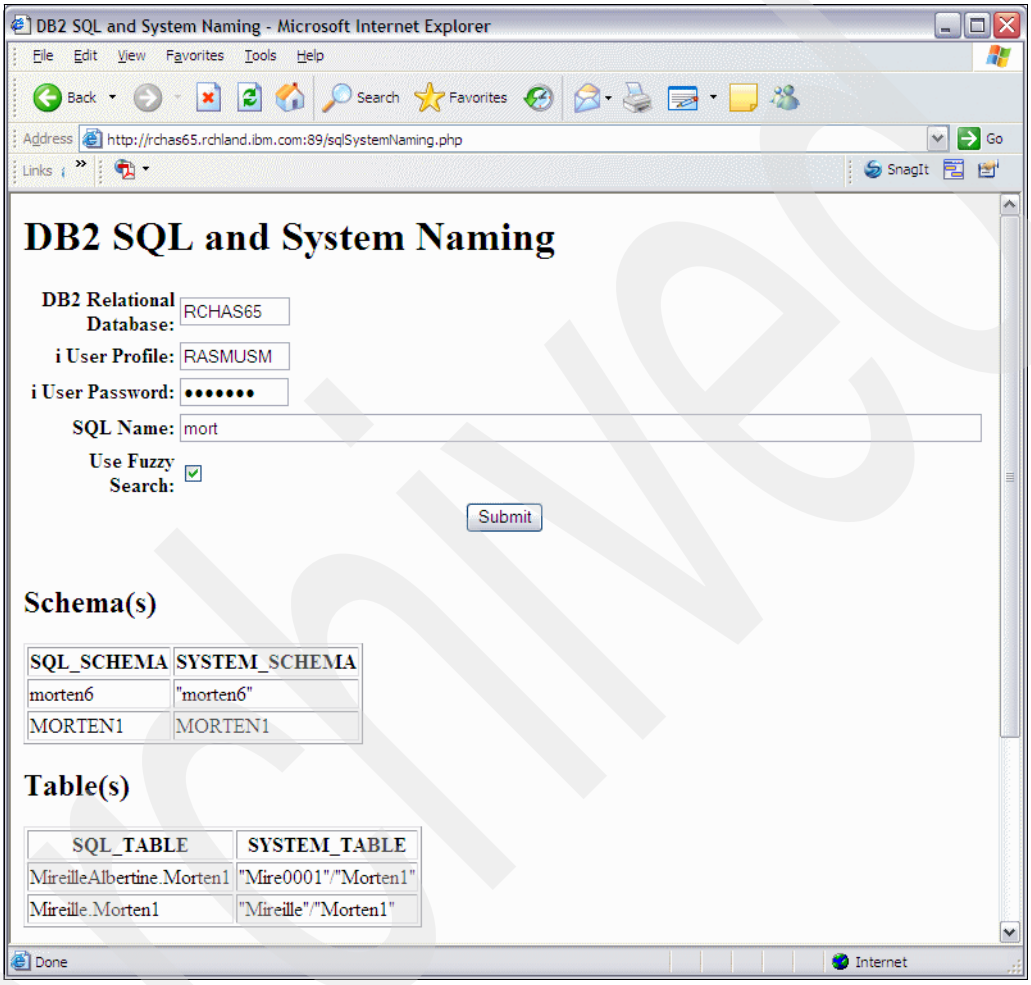


Figure A-1 Running DB2 SQL and System Naming tool

When you check the box Use Fuzzy Search, the SQL performs a LIKE-comparison and probably will return more results back to the requester.

Using the tool by copying the PHP source

To use the tool:

1. Create a text file, named `sqlSystemNaming.php`, in the IFS in a directory under the Apache HTTP server path and that can be placed in the following path:

   `/www/zendcore/htdocs`

2. Copy the source from Example A-1 on page 179, and paste it into the text document.
3. Save the text document.

Example: A-1 Source text for PHP tool to show DB2 SQL and System names

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>DB2 SQL and System Naming</title>
<style>
td.prompt {
   text-align: right;
   font-weight: bold;
}
</style>
</head>
<?php

$rdb = strtoupper ( $_POST ['rdb'] );
$userProfile = strtoupper ( $_POST ['userProfile'] );
$password = strtoupper ( $_POST ['password'] );
$sqlName = $_POST ['sqlName'];
$fuzzySearch = $_POST ['fuzzySearch'];

$EMPTYRESULT = "No results found.\n";

echo "<body>
   <h1>DB2 SQL and System Naming</h1>
   <form method=post action=sqlSystemNaming.php>
      <table>
         <tr><td class=prompt>DB2 Relational Database:</td><td><input name=rdb size=10
value=\"$rdb\"/></td></tr>
         <tr><td class=prompt>i User Profile:</td><td><input name=userProfile size=10
value=\"$userProfile\"/></td></tr>
         <tr><td class=prompt>i User Password:</td><td><input type=password name=password size=10
value=\"$password\"/></td></tr>
         <tr><td class=prompt>SQL Name:</td><td><input name=sqlName size=100
value=\"$sqlName\"/></td></tr>
         <tr><td class=prompt>Use Fuzzy Search:</td><td><input type=checkbox name=fuzzySearch value=\"on\"
" . ( $fuzzySearch ? " checked" : "" ) . " /></td></tr>
         <tr><td colspan=2 align=center><input name=auth type=submit value=Submit></td></tr>
      </table>
   </form>
   <br>";

if ($_POST ['auth']) {

   $i5db2 = db2_connect ( $rdb, $userProfile, $password ) or die ( "Connect error: " . db2_conn_errormsg ()
);

   $whereClause = ( $fuzzySearch ? "LIKE '%" . strtoupper($sqlName) . "%'" : "= UPPER('$sqlName')");

   $sqlQueries = array ( array ("Schema(s)", "SELECT schema_name AS SQL_SCHEMA, system_schema_name AS
SYSTEM_SCHEMA FROM qsys2.sysschemas WHERE UPPER(schema_name) $whereClause;" ),
                        array ("Table(s)", "SELECT table_schema || '.' || table_name AS SQL_TABLE,
system_table_schema || '/' ||
                                           system_table_name AS SYSTEM_TABLE
                                            FROM qsys2.systables
                                            WHERE UPPER(table_type) <> 'V' AND
                                            UPPER(table_name) $whereClause;" ),
                     array ("View(s)", "SELECT table_schema || '.' || table_name AS SQL_VIEW,
                                          system_view_schema || '/' || system_view_name AS SYSTEM_VIEW
                                         FROM qsys2.sysviews
```

```
                                    WHERE UPPER(table_name) $whereClause;" ),
                    array ("Index(es)", "SELECT index_schema || '.' || index_name AS SQL_INDEX,
                                    table_schema || '.' || table_name AS SQL_TABLE,
system_index_schema || '/' || system_index_name
                                    AS SYSTEM_INDEX, system_table_schema || '/' || system_table_name
AS SYSTEM_TABLE
                                    FROM qsys2.sysindexes
                                    WHERE UPPER(index_name) $whereClause;" ) );

   foreach ( $sqlQueries as $sqlQuery ) {
      echo "<h2>$sqlQuery[0]</h2>";

      $stmt = db2_prepare ( $i5db2, $sqlQuery[1] );
      $result = db2_execute ( $stmt );

      if ($row = db2_fetch_assoc ( $stmt )) {
         echo "<table border=1><tr>";
         for($counter = 0; $columnName = db2_field_name ( $stmt, $counter ); $counter ++) {
            print "<th>$columnName</th>";
         }
         echo "</tr>\n";

         do {
            echo "<tr>";
            foreach ( $row as $column ) {
               echo "<td>" . $column . "</td>";
            }
            echo "</tr>\n";
         } while ( $row = db2_fetch_assoc ( $stmt ) );
         echo "</table>\n";
      } else {
         echo "No results found.\n";
      }
   }

   db2_close ( $i5db2 );
}
?>
</body>
</html>
```

B

How to start and stop MySQL server in IBM i

In this appendix, we provide instructions for using the available tools to start, stop, and monitor MySQL server in IBM i.

This information is described in *Discovering MySQL in IBM i5/OS*, SG24-7398.

This appendix contains the following topics:

Starting the MySQL Database Server

In this section, we explain how to start the MySQL Database Server by using **mysqld_safe** and **mysqlmanager** scripts.

Start the server with mysqld_safe

The mysqld_safe script is the MySQL Database Server startup script and is the easiest way to start the MySQL Database Server on IBM i. To start MySQL Database Server on IBM i from a 5250 session command line, enter the following three commands:

```
CALL QP2TERM
cd /QOpenSys/usr/local/mysql/mysql/bin
mysqld_safe &
```

A starting message is displayed, as shown in Figure B-1.

```
/QOpenSys/usr/bin/-sh

> mysqld_safe &
  [2]     11522
  $ Starting mysqld daemon with databases from /QOpenSys/usr/local/mysql/data
```

Figure B-1 The mysqld_safe script starting the server

If the mysqld_safe script fails, even when invoked from the MySQL installation directory, you can specify the `--ledir` and `--datadir` options to indicate the directories in which the server and databases are located on your system.

All options that are specified to the mysqld_safe script on the command line are passed to mysqld daemon. The mysqld_safe script supports many options, of which several of the frequently most used options are listed in Table B-1.

Table B-1 Frequently used options for mysqld

Option	Description
--help	Displays a help message and exit.
--user=user_name	Runs the mysqld server as the user having the name user_name. The occurrence of *user* in this context refers to a system login account, not a MySQL user listed in the grant tables.
--basedir=path	Indicates the path to the MySQL installation directory.
--datadir=path	Indicates the path to the data directory.
--ledir=path	Indicates the path name to the directory where the server is located. Use this option if mysqld_safe cannot find the server
--log-error=file_name	Writes the error log to the given file.
--port=port_num	Indicates the port number that the server should use when listening for TCP/IP connections.
--timezone=timezone	Sets the TZ time zone environment variable to the given option value. Consult your operating system documentation for legal time zone specification formats.

List of options for mysqld_safe

For a list of options that are available for mysqld_safe, see *MySQL 5.1 Reference Manual*, at the following address:

http://dev.mysql.com/doc/refman/5.1/en/mysqld-safe.html

Start the server with mysqlmanager

Another way to start the MySQL Database Server on IBM i is to use mysqlmanager script. This program is the MySQL Instance Manager, with which you can monitor and manage MySQL Database Server instances. MySQL Instance Manager runs on an IBM i PASE environment as a UNIX daemon that listens on a TCP/IP port and a socket file.

MySQL Instance Manager is included in MySQL distributions from version 5.0.3, and can be used in place of the mysqld_safe script to start and stop one or more instances of the MySQL Database Server.

The MySQL Instance Manager offers the following capabilities:

- It can start and stop instances, and report on the status of instances.
- It can treat server instances as guarded or unguarded:
  - When the MySQL Instance Manager starts, it starts each guarded instance. If the instance crashes, the MySQL Instance Manager detects this and restarts it. When the MySQL Instance Manager stops, it stops the instance.
  - An unguarded instance is not started when the MySQL Instance Manager starts or is monitored by it. If the instance crashes after being started, the MySQL Instance Manager does not restart it. When the MySQL Instance Manager exits, it does not stop the instance if it is running.

    Instances are guarded, by default. An instance can be designated as unguarded by including the unguarded option in the configuration file.
- It provides an interactive interface for configuring instances, so that having to edit the configuration file manually is reduced or eliminated.

To create a basic configuration file and start the MySQL Database Server:

1. Use your favorite editor to create the `my.cnf` configuration file with the contents shown in Example B-1 and copy it into the `/etc` directory.

Example: B-1 Sample configuration of my.cnf for mysqlmanager

```
[mysqld]
mysqld-path=/QOpenSys/usr/local/mysql/mysql/bin/mysqld
socket=/tmp/mysql.sock
pid-file = /tmp/hostname.pid1
port=3306
server_id=1

# Log activation statements
log-bin=/QOpenSys/usr/local/mysql/mysql/data/mybinlog
log-error
log=mylog
log-slow-queries
```

Tip: In Example B-1, the occurrence of *mysqld* that is enclosed between brackets is the instance name of the MySQL Database Server. You can use the name of your choice.

2. Create an instance manager password file.

   The MySQL Instance Manager stores its user information in a password file. On IBM i, the default file is `/etc/mysqlmanager.passwd`. If the password file does not exist or contains no password entries, you cannot connect to the instance manager.

   To create a new user and password, run the following statement:

   ```
   mysqlmanager --passwd >> /etc/mysqlmanager.passwd
   ```

   Sometimes this procedure does not work on IBM i because of a problem between the script and the 5250 emulation. If the previous command does not prompt you for a password, enter the following command to generate the correct password file (`/etc/mysqlmanager.passwd`); replace *your_user_name* and *your_password* with your values:

   ```
   mysql -B --skip-column-names  -u root -e 'select
   "your_user_name",password("your_password")' | awk '{print $1":"$2 }' >>
   /etc/mysqlmanager.passwd
   ```

3. Run the **mysqlmanager** program command:

   ```
   mysqlmanager --run-as-service &
   ```

The MySQL Instance Manager supports a number of command options. For a brief listing, invoke **mysqlmanager** with the `--help` option either on the command line or in the MySQL Instance Manager configuration file. In IBM i, the standard file is `/etc/my.cnf`. To specify a different configuration file, start the MySQL Instance Manager with the `--defaults-file` option.

List of options for mysqlmanager

For a list of options available for mysqlmanager, see the *MySQL 5.1 Reference Manual*:

http://dev.mysql.com/doc/refman/5.1/en/instance-manager.html

Start the server with graphical tools

MySQL Administrator and PHPMyAdmin tools require that a MySQL Database Server be running before they can connect. Therefore, we cannot use the MySQL Administrator and PHPMyAdmin tools to control the startup of the MySQL Database Server.

Stopping the MySQL Database Server

In this section, we explain how to stop the MySQL Database Server by using **mysqladmin** and **mysqlmanager** commands.

Stop the server with mysqladmin

The **mysqladmin** command is a client for performing administrative operations. You can use it to stop the server by specifying the user name and password of your installation:

- If your installation has no password, enter the following command:

  ```
  mysqladmin -u root shutdown
  ```

- If your installation is password protected, use the following command:

  ```
  mysqladmin -u root shutdown --password=your_password
  ```

Stop the server with mysqlmanager

To stop your instance by using **mysqlmanager**:

1. Connect to the **mysqlmanager** instance through the mysql CLI using the valid user and password that you created in step 2 on page 184:

   ```
   mysql --port=2273 --host=rchas55 --user="your_user" --password="your_password"
   ```

   You see a panel like the one shown in Figure B-2.

```
> mysql --port=2273 --host=rchas55 --user="bruno" --password="itso"
  Welcome to the MySQL monitor.  Commands end with ; or \g.
  Your MySQL connection id is 1
  Server version: 0.2-alpha

  Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

  mysql>
```

Figure B-2 mysqlmanager instance connection

2. When you are connected, enter the following command:

   ```
   STOP INSTANCE instance_name;
   ```

   Replace *instance_name* with the parameter that is specified in `my.cnf` file as shown in Figure B-3.

```
mysql>
> STOP INSTANCE mysqld;
  Query OK, 0 rows affected (3.16 sec)

  mysql>
> SHOW INSTANCES;
  +---------------+---------+
  | instance_name | status  |
  +---------------+---------+
  | mysqld        | offline |
  +---------------+---------+
  1 row in set (0.00 sec)

  mysql>

===>
```

Figure B-3 Execution of STOP INSTANCE and SHOW INSTANCES

3. When you finish the execution, check the status by entering the following command:

   ```
   SHOW INSTANCES;
   ```

Tip: Remember to exit from the **mysql** command interpreter by issuing the **quit** command before you attempt to run more commands in the IBM i PASE environment.

Checking the status of the MySQL Database Server

There are several ways to check the availability of the MySQL Database Server instances by using graphical and command line tools.

Check the status with mysqladmin

As you have seen before, **mysqladmin** is a client for performing administrative operations. You can monitor the status of your MySQL instances by calling:

```
mysqladmin ping -u root
```

If your MySQL Database Server instance is *alive*, you see a panel similar to the one shown in Figure B-4.

```
mysqladmin ping -u root
mysqld is alive
$
```

Figure B-4 Checking the instance status with mysqladmin

Check the status with mysqlmanager

After you connect to the **mysqlmanager** instance by using the mysql CLI as shown in step 1 on page 185, run the following commands to display details about the status of your instances:

- To learn the status of each instance of the MySQL Database Server:

  ```
  SHOW INSTANCES;
  ```

  The output is similar to Figure B-5.

```
> SHOW INSTANCES;
  +---------------+---------+
  | instance_name | status |
  +---------------+---------+
  | mysqld        | online |
  +---------------+---------+
  1 row in set (0.00 sec)
```

Figure B-5 Sample output for SHOW INSTANCES;

- To see the status and version information of a determined instance (where *instance_name* is the name of your instance):

  ```
  SHOW INSTANCE STATUS instance_name
  ```

  The output is similar to Figure B-6 on page 187.

```
                                    /QOpenSys/usr/bin/-sh

 > SHOW INSTANCE STATUS mysqld;
+---------------+---------+-------------------------------------------------
| instance_name | status | version
+---------------+---------+-------------------------------------------------
| mysqld        | online |  Ver 5.0.45 for ibm-i5os on power (MySQL Community
+---------------+---------+-------------------------------------------------
1 row in set (0.00 sec)

mysql>

===>
```

Figure B-6 Individual instance status

List of options for MySQL Instance Manager commands

For a detailed list of options that are available for MySQL Instance Manager commands, see the *MySQL 5.1 Reference Manual*:

http://dev.mysql.com/doc/refman/5.1/en/instance-manager.html

Check the status with MySQL Administrator

Another way to check whether your MySQL Database Server is active is by trying to connect with MySQL Administrator.

Open MySQL Administrator, and enter the information for your server and your user data as shown in Figure B-7. You can leave the default port if you do not change that setting in your `my.cnf` file or when you call mysqld_safe. Otherwise, you must specify the port that is used.

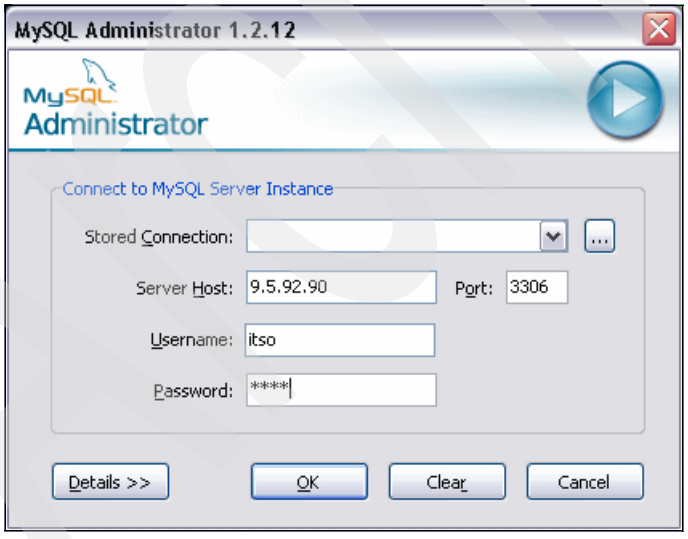


Figure B-7 MySQL Administrator login window

If you cannot connect to the server by using the correct access data, but you can access the server from your workstation by using the **`ping`** command, your MySQL instance might be down.

Password privileges: Before you attempt to connect to MySQL Administrator, you must have a valid password in MySQL Database Server with the appropriate privileges to access from your computer. You cannot access the MySQL Database Server by using an IBM i user profile.

If you connected to your instance, information about your server is displayed by selecting **Server Information,** as shown in Figure B-8.

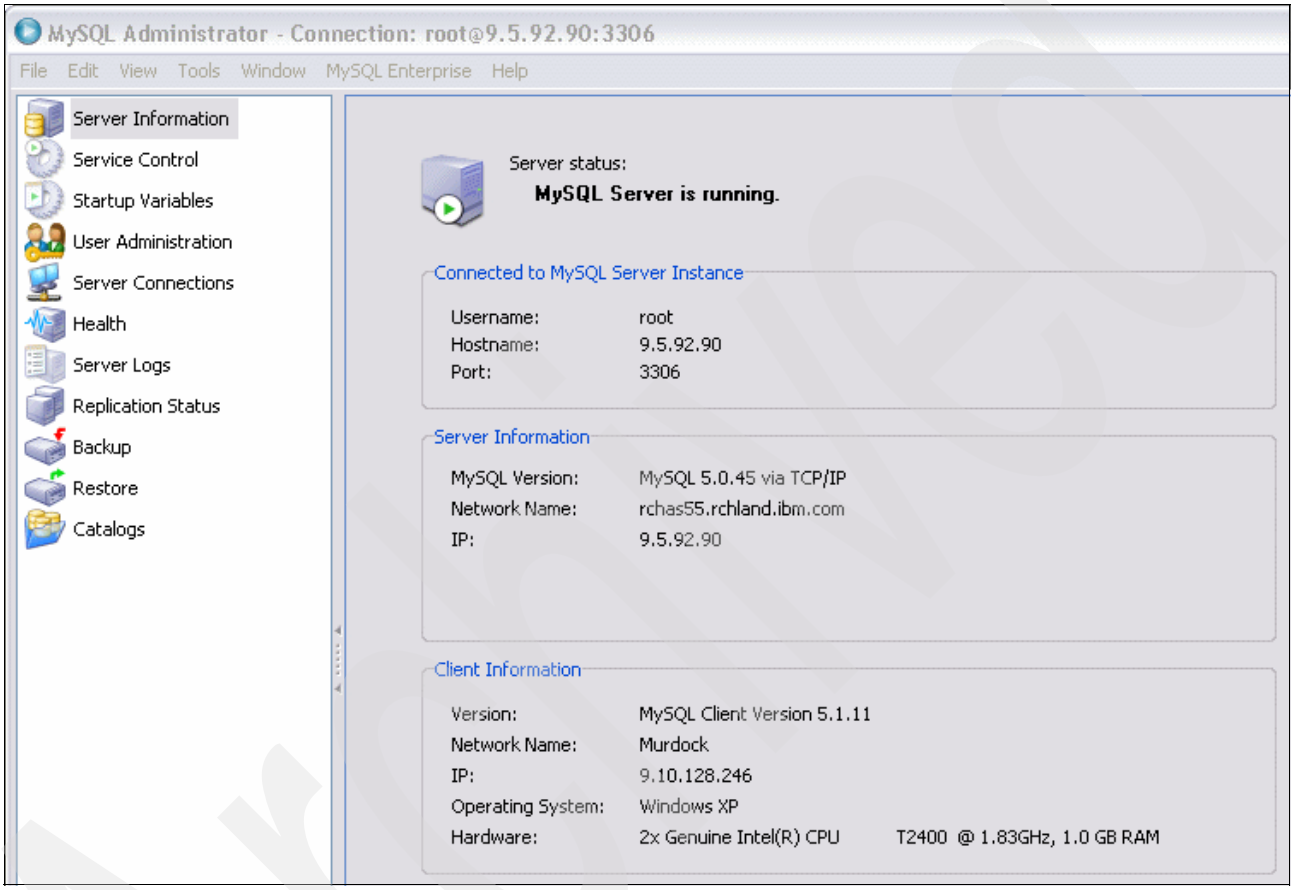


Figure B-8 MySQL Administrator displaying server information

Also, select **Health**, and then click the **System Variables** tab to obtain more information similar to the information shown in Figure B-9.

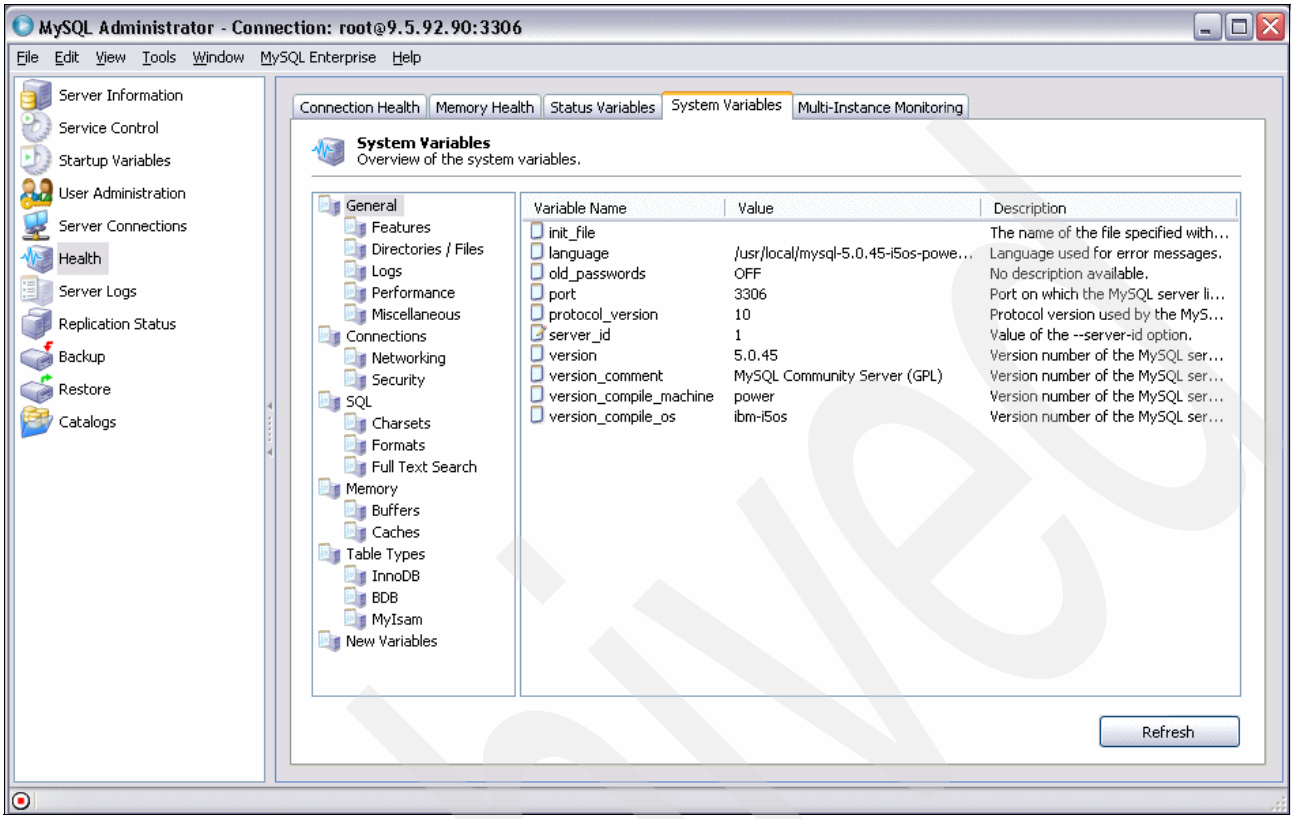


Figure B-9 MySQL Administrator displaying system variables

Check the status with phpMyAdmin

To check your MySQL Database Server by using phpMyAdmin, point your browser to:

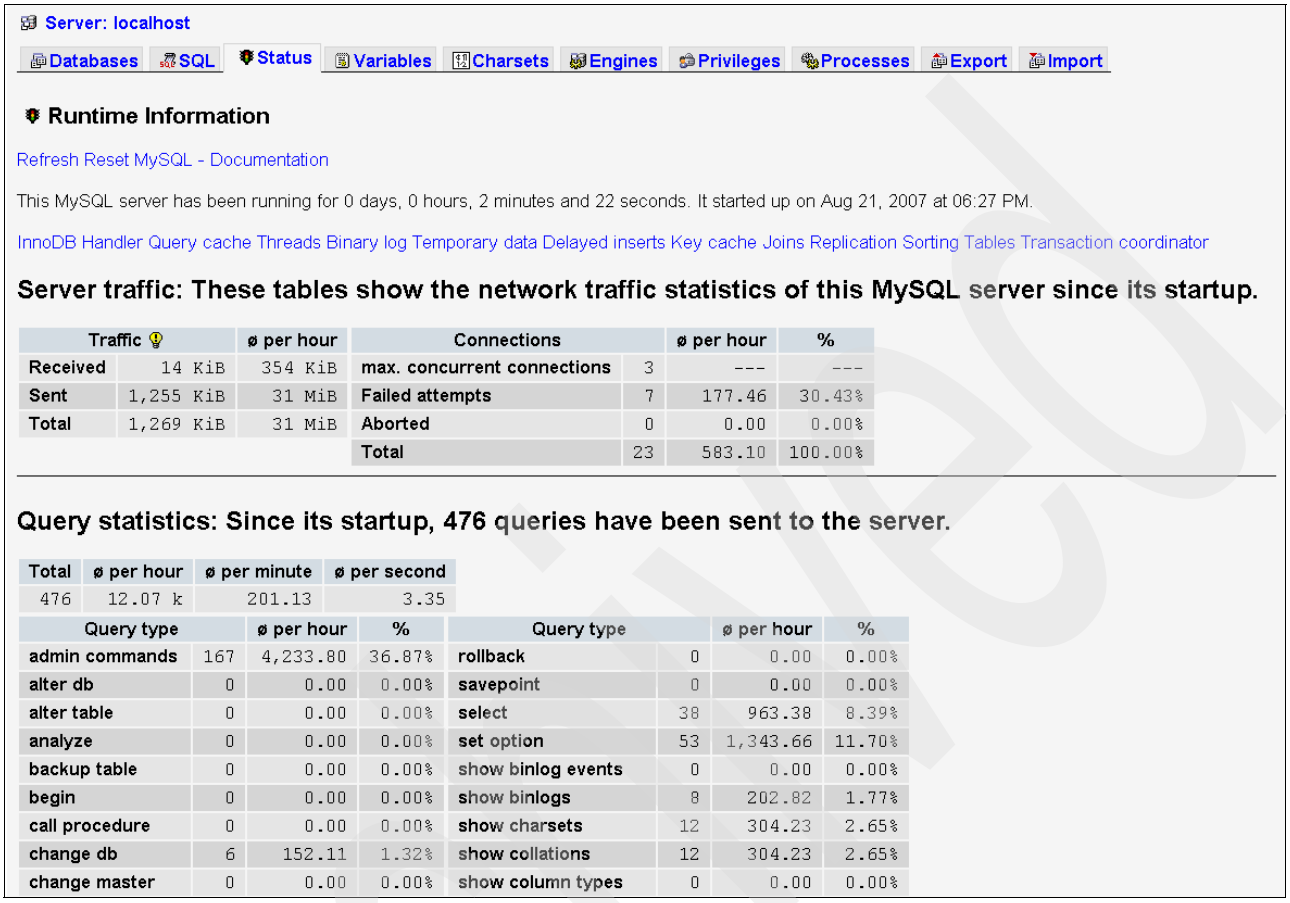
`http://yourserver:port/path_to_phpMyAdmin/`

Indicate your server, port, and path. Again, if you are unable to connect to the server by using the correct access data, your MySQL instance might be down. In such a case, you see an error message similar to the one in Figure B-10.

Figure B-10 Error message indicating that the phpMyAdmin server is not responding

Valid user name and password: You must use a valid user name and password combination for your mysql phpMyAdmin installation.

If there is no error message, you are taken to the phpMyAdmin index page. On this page, click **Show MySQL runtime information** to see a more detailed status of your server, similar to Figure B-11.



Server: localhost

Databases | SQL | Status | Variables | Charsets | Engines | Privileges | Processes | Export | Import

Runtime Information

Refresh Reset MySQL - Documentation

This MySQL server has been running for 0 days, 0 hours, 2 minutes and 22 seconds. It started up on Aug 21, 2007 at 06:27 PM.

InnoDB Handler Query cache Threads Binary log Temporary data Delayed inserts Key cache Joins Replication Sorting Tables Transaction coordinator

Server traffic: These tables show the network traffic statistics of this MySQL server since its startup.

Traffic		ø per hour	Connections		ø per hour	%
Received	14 KiB	354 KiB	max. concurrent connections	3	---	---
Sent	1,255 KiB	31 MiB	Failed attempts	7	177.46	30.43%
Total	1,269 KiB	31 MiB	Aborted	0	0.00	0.00%
			Total	23	583.10	100.00%

Query statistics: Since its startup, 476 queries have been sent to the server.

Total	ø per hour	ø per minute	ø per second
476	12.07 k	201.13	3.35

Query type		ø per hour	%	Query type		ø per hour	%
admin commands	167	4,233.80	36.87%	rollback	0	0.00	0.00%
alter db	0	0.00	0.00%	savepoint	0	0.00	0.00%
alter table	0	0.00	0.00%	select	38	963.38	8.39%
analyze	0	0.00	0.00%	set option	53	1,343.66	11.70%
backup table	0	0.00	0.00%	show binlog events	0	0.00	0.00%
begin	0	0.00	0.00%	show binlogs	8	202.82	1.77%
call procedure	0	0.00	0.00%	show charsets	12	304.23	2.65%
change db	6	152.11	1.32%	show collations	12	304.23	2.65%
change master	0	0.00	0.00%	show column types	0	0.00	0.00%

Figure B-11 phpMyAdmin status page

Tip: You can navigate through several options to see more information about your system. For a detailed review of the options, see the phpMyAdmin documentation in the Documentation.html file in your code, or check the phpMyAdmin wiki on the Web at:

`http://wiki.cihar.com/`

Automating the starting and stopping tasks

The MySQL Database Server runs as a server within IBM i PASE. Many users want the ability to start the server automatically. A method that loads only the server by using a call to QP2SHELL can start the server, but it normally starts in the batch subsystem that is set by your profile. If this batch subsystem is QBATCH and you have the subsystem set to a single batch stream, no other jobs can be loaded.

To overcome this problem, we create several objects through which the job can be submitted to its own subsystem and the programs can start and end the MySQL Database Server automatically when you start or end the IBM i subsystem. In this example, we use MYSQLLIB

as the name of the library. You can use any name that you prefer, but the parameters must be set as accordingly:

1. Grant permissions on mysql and the auxiliary directories.

   We created the tables by using `-user=mysql`, but had problems with the authority. To correct the problems, we changed the authority on the following directories:

   – /QOpenSys/usr/local/mysql/mysql/
   – /QOpenSys/usr/local/mysql/data
   – /tmp
   – /etc

   We changed the authority by calling the following command for each of the directories (that are listed in Figure B-12):

   ```
   CHGAUT OBJ('path_to_change') USER(MYSQL) DTAAUT(*RWX) OBJAUT(*OBJMGT *OBJEXIST
   *OBJALTER *OBJREF) SUBTREE(*ALL)
   ```

```
                        Change Authority (CHGAUT)

Type choices, press Enter.

Object . . . . . . . . . . . . > '/QOpenSys/usr/local/mysql/mysql'

User . . . . . . . . . . . . . > MYSQL        Name, *PUBLIC, *NTWIRF
              + for more values
New data authorities . . . . . > *RWXE        *SAME, *NONE, *RWX, *RX...
New object authorities . . . . .  *SAME       *SAME, *NONE, *ALL...
              + for more values
Authorization list . . . . . .                Name, *NONE
Directory subtree  . . . . . . > *YES         *NONE, *ALL
Symbolic link  . . . . . . . .   *NO          *NO, *YES

                                                                      Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure B-12 Sample CHGAUT panel for the /QOpenSys/usr/local/mysql/mysql directory

Alternative: If you do not perform this step, you can start the server by using QP2TERM and the `/QOpenSys/usr/local/mysql/mysql/bin/mysqld_safe`, but you would not be able to start the following programs because of authority issues.

2. Create a library to hold all of the objects:

   ```
   CRTLIB LIB(MYSQLLIB) TEXT('MySQL Lib')
   ```

3. Create a source file to hold the programs that we will use for starting and ending the server:

   ```
   CRTSRCPF FILE(MYSQLLIB/QCLSRC) TEXT('Source File to hold mySQL programs')
   ```

4. Create the members with the code of your choice.

   We provide the minimum code. Consider adding error checking and cleanup routines. You must choose the adequate program depending on whether you are implementing the startup with mysqld_safe or mysqlmanager.

If you want to use the mysqld_safe script to start the database, use the code shown in Example B-2.

Example: B-2 Code sample for startup if you are using mysqld _safe

```
PGM
SBMJOB     CMD(SBMJOB CMD(CALL PGM(QP2SHELL) +
             PARM('/QOpenSys/usr/local/mysql/mysql/bin/mysqld_safe' +
             '--no-defaults' '--user=mysql')) +
             JOB(MYSQLD) JOBD(MYSQLLIB/MYSQLJOBD) +
             JOBQ(MYSQLLIB/MYSQLJOBQ))
ENDPGM
```

Otherwise, if you want to use mysqlmanager to start the database, use the program shown in Example B-3.

Example: B-3 Code sample for startup if you are using mysqlmanager

```
PGM
DCL        VAR(&CMD) TYPE(*CHAR) LEN(33)
SBMJOB     CMD(CALL PGM(QP2SHELL) +
             PARM('/QOpenSys/usr/local/mysql/mysql/bin/mysqlmanager' +
             '--run-as-service')) JOB(MYSQLD) +
             JOBD(MYSQLLIB/MYSQLJOBD) JOBQ(MYSQLLIB/MYSQLJOBQ)
ENDPGM
```

5. Create the job queue object:

   ```
   CRTJOBQ JOBQ(MYSQLLIB/MYSQLJOBQ) TEXT('MySQL JOBQ')
   ```

6. Create the job description with the routing data and request data to call the startup program:

   ```
   CRTJOBD JOBD(MYSQLLIB/MYSQLJOBD) JOBQ(MYSQLLIB/MYSQLJOBQ) TEXT('Mysql Job
   Description') USER(MYSQL) RTGDTA('MYSQL') RQSDTA('call mysqllib/strmysql')
   ```

7. Create a class:

   ```
   CRTCLS CLS(MYSQLLIB/MYSQLCLS) RUNPTY(50) TEXT('MySQL Class')
   ```

8. Create the subsystem description:

   ```
   CRTSBSD SBSD(MYSQLLIB/MYSQLSBS) POOLS((1 *BASE)) TEXT('MySQL Subsystem')
   ```

9. Add a job queue entry to link the job queue that we created previously to the subsystem:

   ```
   ADDJOBQE SBSD(MYSQLLIB/MYSQLSBS) JOBQ(MYSQLLIB/MYSQLJOBQ) MAXACT(*NOMAX)
   ```

10. Add two routing entries to ensure that the job routing is carried out:

    ```
    ADDRTGE SBSD(MYSQLLIB/MYSQLSBS) SEQNBR(100) CMPVAL(MYSQL) PGM(QCMD)
    CLS(MYSQLLIB/MYSQLCLS)
    ADDRTGE SBSD(MYSQLLIB/MYSQLSBS) SEQNBR(999) CMPVAL(*ANY) PGM(QCMD)
    ```

11. Add an autostart job to the subsystem that will be called when the subsystem is started:

    ```
    ADDAJE SBSD(MYSQLLIB/MYSQLSBS) JOB(AUTOSTART) JOBD(MYSQLLIB/MYSQLJOBD)
    ```

Now when you start the MYSQLSBS subsystem, you automatically start the MySQL Database Server.

Tip: You must ensure that the mysql user has the proper authorizations to the library and programs that were created.

Starting and ending MySQL Database Server subsystem

We created a subsystem similar to others that you might find in IBM i, so that the subsystems operate in the same way.

To start the new subsystem, enter the following command:

```
STRSBS MYSQLLIB/MYSQLSBS
```

To stop the MySQL Database Server, end the subsystem by entering the following command:

```
ENDSBS SBS(MYSQLSBS)
```

Alternatively, you can use the *IMMED option.

Error messages: If the server fails to start, various error messages are returned depending on your configuration. To identify the problems, you can look in the out queue QPRINT, where you should find a printout from the jobs that failed. No job logs are created in the IBM i PASE environment for the failing processes. You can also look in the error log that is created as part of the installation. In our example, the error log is RCHAS55.RCHLAND.local.err, where *RCHAS55* is the system name and *RCHLAND* is the domain.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 196. Note that some of the documents referenced here might be available in softcopy only.

- *Bringing PHP to Your IBM eServer iSeries Server*, REDP-3639
- *PHP: Zend for i5/OS*, SG24-7327
- *Discovering MySQL in IBM i5/OS*, SG24-7398
- *Porting UNIX Applications Using AS/400 PASE*, SG24-5970
- *The System Administrator's Companion to AS/400 Availability and Recovery*, SG24-2161

Online resources

These Web sites are also relevant as further information sources:

- The most recent information about MySQL on IBM i and the IBMDB2I storage engine
  http://www.ibm.com/systems/i/software/mysql/index.html
- IBM i Domain Redbooks publications
  http://www.redbooks.ibm.com/portals/systemi
- IBM DB2 for i portal
  http://www-03.ibm.com/systems/i/software/db2/index.html
- MySQL 5.1 Reference Manual
  http://dev.mysql.com/doc/refman/5.1/en/
- MySQL Storage Engines
  http://dev.mysql.com/doc/refman/5.0/en/storage-engines.html
- MySQL Community Server downloads page
  http://dev.mysql.com/downloads/mysql/5.0.html
- Recommended IBM i fixes (including database)
  http://www-912.ibm.com/s_dir/slkbase.nsf/recommendedfixes
- Current IBM i PASE PTFs by IBM i release
  http://www.ibm.com/servers/enable/site/porting/iseries/pase/misc.html
- Pluggable Storage Engine Architecture
  http://solutions.mysql.com/engines.html

- phpMyAdmin official home Web site and downloads

  http://phpmyadmin.net
- The Perl directory

  http://www.perl.org/

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

M

N

T

U

V

W

X

Using IBM DB2 for i as a Storage Engine of MySQL

(0.2”spine)
0.17”<->0.473”
90<->249 pages

Using IBM DB2 for i as a Storage Engine of MySQL

Discover how to configure and manage the IBMDB2I Storage Engine

Integrate and consolidate MySQL and DB2 in one place

Access PHP Applications data through native interfaces

With the Apache, MySQL, and PHP (AMP) stack, IBM i has the open source middleware to run thousands of PHP applications and scripts that have been written to the MySQL database. MySQL is a database that is used on millions of Web sites. To support the wide variety of usage, the developers of MySQL has developed an open storage engine architecture for data functionality and storage. Over a dozen storage engines are available for MySQL. IBM and Sun Microsystems have worked together to deliver a DB2 for i Storage Engine for MySQL. With this support, PHP applications written to MySQL database can have the data stored in the DB2 for i database. This approach provides management benefits for the IBM i customer because DB2 is integrated into IBM i and customers already know how to manage, back up, and protect DB2 data. In addition, the DB2 for i Storage Engine provides access to the MySQL data from IBM i environments such as RPG, CL, and DB2 Web Query. The DB2 for i Storage Engine offers the management and data access integration that can make IBM i the preferred platform for running open source applications for IBM i customers.

This IBM Redbooks publication provides broad information to help you understand this storage engine. The book also helps you install, tailor, and configure DB2 for i Storage Engine for MySQL support.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7705-00 ISBN 0738432407