Part 3. MySQL DBA I Exam

Table of Contents

Chapter 23. MySQL Architecture

Almost all examples and exercises in this study guide use the *world database* as the sample data set. The accompanying CD-ROM contains the data for this database and instructions that describe how to create and populate the database for use with your own MySQL installation.

Question 1:

Name a utility program that accesses database tables directly, without communicating with the server. Why do you have to use this type of program with extra care?

Question 2:

Name two character-based programs and two graphical programs that access tables by communicating with the server.

Question 3:

Consider the following list of ways to connect to the MySQL server. Which are operating system-dependent? Which will work only for connections to the local server, and which will also work for remote connections to a remote server?

a. TCP/IP

b. ODBC

c. Shared memory

d. Named pipe

e. Unix socket file

Question 4:

MySQL uses a two-tier processing model. What are those two tiers?

Question 5:

Which of the following are implications that arise out of MySQL's two-tier model?

a. Some index types are available only for particular storage engines.

b. Transactions that consist of more than one SQL statement are available only for particular storage engines.

Question 6:

For what kinds of files and directories does MySQL use disk space?

Question 7:

For what kinds of information does the MySQL server allocate memory?

Answers to Exercises

Answer 1:

`myisamchk` is one example of a utility program that accesses table data directly. You have to take care when using programs that access tables directly, which usually means you have to make sure that the server won't access those tables at the same time. If two or more programs access the same tables at the same time, this could lead to incorrect results or even table damage.

Answer 2:

`mysql` and `mysqlimport` are two examples of programs that access tables by communicating with the server. MySQL Query Browser and MySQL Administrator are graphical tools that communicate with the server and access tables that way.

Answer 3:

Shared memory and named pipes are available only under Windows, and Unix socket files are available only under Unix-like operating systems. TCP/IP is not operating system-dependent. ODBC is not operating system-dependent in itself, but is available for MySQL only on operating systems where MySQL Connector/ODBC is supported. Any of the connection methods can be used to connect to a local server. TCP/IP and ODBC can be used to connect to a remote server.

Answer 4:

MySQL has an upper tier that includes the SQL parser and the optimizer. The lower tier comprises a set of storage engines such as `MyISAM` or `InnoDB`.

Answer 5:

Some index types, for example `FULLTEXT` or `SPATIAL`, are available only for the `MyISAM` storage engine. This means that you cannot create those index types for other storage engines; in other words: You cannot use the `FULLTEXT` and `SPATIAL` keywords in an SQL statement like `CREATE TABLE` or `ALTER TABLE` for a storage engine other than `MyISAM`. (If you do, you get an error.)

It doesn't make sense to use some SQL statements for storage engines that don't support the concepts which those statements are for. For example, you might use the transactional commands `COMMIT` and `ROLLBACK` for `MyISAM` tables (without getting an error), but they make sense only for storage engines that support multi-statement transactions, like `InnoDB`.

Apart from SQL statements that refer to properties or behavior of particular storage engines, SQL statements generally are not bound to particular storage engines.

Answer 6:

MySQL uses disk space to store the following:

- The server and client programs, and their libraries
- Log files and status files
- Databases
- Table format (`.frm`) files for all storage engines, and data files and index files for some storage engines
- `InnoDB` tablespace files (if the `InnoDB` storage engine is enabled)
- Internal temporary tables that have crossed the size threshold for being converted from in-memory tables to on-disk tables

Answer 7:

MySQL allocates memory for the following:

- Connection handlers (every connection uses memory)
- Buffers and caches
- A copy of the grant tables
- Internal temporary tables that are below the size threshold for being converted from in-memory tables to on-disk tables
- The host cache and the table cache
- The query cache
- `MEMORY` table contents (note that each `MEMORY` table also requires disk space to store its `.frm` file)

Chapter 24. Starting, Stopping, and Configuring MySQL

Almost all examples and exercises in this study guide use the *world database* as the sample data set. The accompanying CD-ROM contains the data for this database and instructions that describe how to create and populate the database for use with your own MySQL installation.

Question 1:

How would you start the server so that it logs errors? How do you access the information in the error log?

Question 2:

What's the explanation for the following error?

```
ERROR 1251: Client does not support authentication protocol
requested by server; consider upgrading MySQL client
```

Question 3:

How would you start the server so that it logs two things: queries that take longer than 10 seconds to perform and queries that use no indexes? How can you access the information logged this way?

Question 4:

How would you start the server so that it logs all operations that change data in tables? How can you access the information logged this way?

Question 5:

As the `root` login user on a Linux host, how can you start the MySQL server so that it doesn't run as `root`, without having to log in as another user? How can you make sure that the server will always run as a user different from `root`?

Question 6:

Suppose that you used RPM files on a Linux host to install the MySQL server and clients. Can you start working with MySQL right away, or are there any further steps you must perform first?

Question 7:

After installing MySQL on Linux from a `tar` file, the server cannot be started. Looking in the error log, you find that the server wrote this error message before terminating: `Can't find file: './mysql/host.frm' (errno: 13)`. What is the reason for this error, and how could you solve the problem?

Question 8:

Name some reasons why you would compile MySQL from source, rather than using a binary distribution provided by MySQL AB.

Question 9:

You want to see the errors reported by the server at startup. Under Windows, how would you direct that

output to your terminal rather than to the error log?

Question 10:

You want to see the errors reported by the server at startup. Under Unix, how would you direct that output to your terminal rather than to the error log?

Question 11:

Suppose that you want to use an option file for the server. The file is named `my_config.ini` and is located in the `C:\Programme\MySQL\MySQL Server 5.0` directory.

a. How would you instruct the server to read that option file when you start it from the command line?

b. How would you instruct the server to read that option file when you install a Windows service that starts the server?

Question 12:

To enable support for named time zones, what do you have to do?

Question 13:

What must you do to make the server behave more like other “traditional” database servers? What is the effect of altering the server's behavior that way?

Question 14:

After installing MySQL, the server fails to start properly. How can you find out what prevented the server from starting?

Question 15:

What options are mandatory for the server to be able to start? How could you specify these options?

Question 16:

Suppose that the MySQL installation directory is `C:\Programme\MySQL\MySQL Server 5.0`. How do you let the server know this when it starts? Is it necessary to specify that directory at all?

Answers to Exercises

Answer 1:

On Windows, the server logs errors to a file by default. You need do nothing to enable the error log. If you want to indicate a specific filename for the log, start the server with the `--log-error=`*`file_name`* option.

On Unix, the server writes errors to the standard error output. Normally, this is the terminal. You can write error output to a given file instead by starting the server with the `--log-error=`*`file_name`* option.

To see the error log, look for a file with an `.err` suffix in the data directory (unless you've specified a filename of your own). The error log is written in text format and can be viewed using any program that displays text files. It can also be viewed using MySQL Administrator.

Answer 2:

Servers for MySQL 4.1 and up use an authentication mechanism that is more secure and provides better passwords than in older versions. However, client programs from older versions do not understand this mechanism and an error occurs when they attempt to connect to a newer server, unless it has been configured to support accounts that have old-format passwords.

Answer 3:

To log slow queries, enable the slow query log by starting the server with the `--log-slow-queries` option. This logs queries that take a long time to perform. By default, "a long time" is defined as 10 seconds. Using the `--log-queries-not-using-indexes` option in addition to the `--log-slow-queries` option instructs the server to also write queries that use no indexes to the slow query log.

The slow query log is in text format and can be viewed with any program that displays text files.

Answer 4:

Enable the binary log by starting the server with the `--log-bin` option. The server writes to this log all statements that modify data. Its contents are in binary format, but can be viewed using the `mysqlbinlog` program.

Answer 5:

To start the server so that it runs as user `mysql`, you can start it with a `--user` option like this:

```
shell> mysqld --user=mysql
```

The `mysqld_safe` script also accepts a `--user` option. To make sure that the server will always start as that user, put the option in an option file (for example, `/etc/my.cnf`):

```
[mysqld]
user=mysql
```

Answer 6:

The RPM installation gives you a MySQL system that is ready to work with, if no problems occurred during the installation procedure:

- The `mysql` login and group accounts are created
- The MySQL server and all client programs are installed
- The data directory is set up
- The grant tables are set up and initialized
- The startup script is registered
- The MySQL server is started

However, the initial MySQL accounts stored in the grant tables have no passwords and should be modified to make MySQL secure.

Answer 7:

The error message indicates that the grant tables were not installed. The server tries to read the grant

tables at startup, and if they aren't present, it reports an error for the first table file it doesn't find (`host.frm`). This can occur if the grant tables do not exist or if they exist but the server cannot read them. If the tables do not exist, run the `mysql_install_db` script to create them. If the grant tables exist, make sure that their ownership and access permissions allow the server to read them.

Answer 8:

Reasons why you might choose to compile MySQL yourself using a source distribution include the following:

- There is no binary distribution available for your platform.
- You need to enable a feature that is not available in any of the precompiled distributions.
- You want to disable an unneeded feature to produce a server that uses less memory.
- You want to run a server built from the current development source.

Answer 9:

Under Windows, start the server with the `--console` option:

```
shell> mysqld --console
```

Answer 10:

Under Unix, start the server directly (that is, without using a startup script such as `mysqld_safe` or `mysql.server`):

```
shell> mysqld
```

The server will send its diagnostics to the standard error output (normally your terminal).

Answer 11:

a. Start the server from the command line like this:

```
shell> mysqld
--defaults-file="C:\Programme\MySQL\MySQL Server 5.0\my_config.ini"
```

b. Install the server as a Windows service from the command line like this:

```
shell> mysqld --install MySQL
--defaults-file="C:\Programme\MySQL\MySQL Server 5.0\my_config.ini"
```

Each command shown is displayed on multiple lines, but should be entered using a single line.

Answer 12:

The way to enable named time zones depends on the operating system of the host on which the server is running. If the operating system provides its own time zone files, you can load them using the `mysql_tzinfo_to_sql` program; for example, like this:

```
shell> mysql_tzinfo_to_sql /usr/share/zoneinfo | mysql -u root mysql
```

If the operating system doesn't provide those files, you can download already populated time zone `MyISAM` tables from the MySQL web site, and copy these over the empty time zone tables that are installed by default in the `mysql` database. The server should be stopped when you do this.

Before you can populate the time zone tables on Unix, you should make sure that the time zone tables in the `mysql` database exist. Those tables are, among others, installed by invoking the `mysql_install_db` script. That script is either invoked explicitly, or implicitly for installation using RPM files.

Answer 13:

You can start the server using the `--sql-mode=TRADITIONAL` option. You might do this to run the server with a mode that is more careful about accepting invalid data or creating MySQL user accounts. Once enabled, the server enforces restrictions on input data values that are like other (more “traditional”) database servers, rather than MySQL's more forgiving behavior. It also will not allow new user accounts to be created with the `GRANT` statement unless a password is specified.

Answer 14:

Look in the error log, which is where the server writes error messages indicating why it couldn't start. The error log is located in the data directory and is named *`host_name`*`.err`, where *`host_name`* is the name of the machine on which the MySQL server is running. (If you've specified a filename of your own for the error log file, look for a file of that name.) Another way to see error messages is to send them to the console:

- On Windows, invoke the server at the command line with the `--console` option.
- On Unix, invoke `mysqld` directly (without using `mysqld_safe`) and without specifying the `--log-error=`*`file_name`* option.

Answer 15:

The server may be started without specifying any options. If you want to override the built-in default values, you would specify options either on the command line or in an option file.

Answer 16:

You can specify the installation directory either on the command line when starting the server, or in an option file. From the command line, issue this command:

```
shell> mysqld --basedir="C:\Programme\MySQL\MySQL Server 5.0"
```

In an option file, specify the location like this:

```
[mysqld]
basedir="C:/Programme/MySQL/MySQL Server 5.0"
```

You have to specify that installation directory in either of those ways, because otherwise the server assumes that it's installed in its default location, which for Windows is `C:\mysql`.

Chapter 25. Client Programs for DBA Work

Almost all examples and exercises in this study guide use the *world database* as the sample data set. The accompanying CD-ROM contains the data for this database and instructions that describe how to create and populate the database for use with your own MySQL installation.

Question 1:

Which MySQL client programs allow you to create a database? Which programs allow you to drop a database?

Question 2:

Which MySQL client programs allow you to create a table? Which programs allow you to drop a table?

Question 3:

Which MySQL client programs allow you to find out which clients are connected to the server? Which programs allow you to kill clients? If there are character-based programs for this, what would the commands be for displaying and killing connections?

Question 4:

Which MySQL client programs allow you to restart the server? Are there limitations, for example, do you have to run the client program on the same host as the server?

Question 5:

Which MySQL client programs allow you to find out whether the server is running? Can you get that information for both local and remote servers?

Question 6:

Which MySQL client programs allow you to stop the server? Are there limitations, for example, do you have to run the client program on the same host as the server? Are there any platform dependencies?

Question 7:

Which client or utility programs can be used to create a database named `landmarks`?

Question 8:

Is the following statement true?

You can load any files created by `mysqldump` with the `mysqlimport` client program.

Question 9:

Which MySQL client programs allow you to change the contents of the `my.cnf` or `my.ini` option file?

Question 10:

Which MySQL client programs allow you to change the password of a MySQL user?

Question 11:

Consider the `FLUSH PRIVILEGES` SQL statement, which you can issue using `mysql`. What command-line program could you use to accomplish the same task? What would the command look like?

Question 12:

Consider the `SHOW PROCESSLIST` SQL statement, which you can issue using `mysql`. What command-line program could you use to accomplish the same task? What would the command look like?

Answers to Exercises

Answer 1:

You can create and drop databases with `mysql`, `mysqladmin`, and MySQL Administrator.

Answer 2:

You can create and drop tables with `mysql` and MySQL Administrator.

Answer 3:

You can display connected clients and kill clients with `mysql`, `mysqladmin`, and MySQL Administrator.

With `mysql`, you would issue SQL statements like these:

```
mysql> SHOW PROCESSLIST;
+-----+--------+----------+------+---------+------+-------+------------------+
| Id  | User   | Host     | db   | Command | Time | State | Info             |
+-----+--------+----------+------+---------+------+-------+------------------+
| 789 | stefan | art:3828 | NULL | Sleep   |  146 |       | NULL             |
| 792 | stefan | art:2971 | NULL | Query   |    0 | NULL  | SHOW PROCESSLIST |
+-----+--------+----------+------+---------+------+-------+------------------+
2 rows in set (0.00 sec)

mysql> KILL 789;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW PROCESSLIST;
+-----+--------+----------+------+---------+------+-------+------------------+
| Id  | User   | Host     | db   | Command | Time | State | Info             |
+-----+--------+----------+------+---------+------+-------+------------------+
| 792 | stefan | art:2971 | NULL | Query   |    0 | NULL  | SHOW PROCESSLIST |
+-----+--------+----------+------+---------+------+-------+------------------+
1 row in set (0.00 sec)
```

With `mysqladmin`, you would issue command like these:

```
shell> mysqladmin processlist
+-----+--------+----------+----+---------+------+-------+------------------+
| Id  | User   | Host     | db | Command | Time | State | Info             |
+-----+--------+----------+----+---------+------+-------+------------------+
| 781 | stefan | art:2616 |    | Sleep   | 978  |       |                  |
| 786 | stefan | art:2637 |    | Query   | 0    |       | show processlist |
+-----+--------+----------+----+---------+------+-------+------------------+
shell> mysqladmin kill 781
shell> mysqladmin processlist
+-----+--------+----------+----+---------+------+-------+------------------+
| Id  | User   | Host     | db | Command | Time | State | Info             |
+-----+--------+----------+----+---------+------+-------+------------------+
```

```
| 788 | stefan | art:2637 |    | Query   | 0    |       | show processlist |
+-----+--------+----------+----+---------+------+-------+------------------+
```

Answer 4:

In general, you cannot restart the server with MySQL client programs. This is usually done either by starting the server directly or with a script. There is one exception: You can start the server with MySQL Administrator when it's run on the same host as the server, and when the operating system is Windows.

Answer 5:

You can use the `mysqladmin ping` command to find out if a server is running; that command works both for local and remote servers. A successful result means that the server is running. An unsuccessful result might mean that the server is not running, but it might also mean only that the server is unreachable due to network problems.

MySQL Administrator reports the server status (both local and remote servers) in the `Server Information` tab. (However, if the server isn't running at the time you try to connect to it using MySQL Administrator, you will get a `Could not connect to the specified instance` error. You can then use the `Ping` button to find out if the server host is reachable. This is different from what `mysqladmin` does, which pings the MySQL server, not just the host.)

With the `mysql` client program, if you can connect to the server, you can deduce from that fact that it is running. There is no other way of finding that out with this program.

Answer 6:

`mysqladmin` can shut down both the local server and remote servers. The program has no platform dependencies.

With MySQL Administrator, you can stop a local server on Windows if it's running as a Windows service. This is done by stopping the Windows service that manages the MySQL server.

With `mysql`, you cannot stop a server.

Answer 7:

To create the database from the command line, run `mysqladmin`:

```
shell> mysqladmin create landmarks
```

From within the `mysql` client, use the CREATE DATABASE statement:

```
mysql> CREATE DATABASE landmarks;
```

With MySQL Administrator, you can create a database in the `Catalogs` tab.

Answer 8:

You can load some but not all files created by `mysqldump` with the `mysqlimport` client program. The SQL-format files that `mysqldump` can produce should be loaded using the `mysql` program or MySQL Administrator. However, you can use `mysqlimport` to load data files that were created by running `mysqldump` with the `--tab` option.

Answer 9:

Only MySQL Administrator allows you to edit the `my.cnf` or `my.ini` option files. This requires that you run MySQL Administrator on the same host as the server.

Answer 10:

You can change user account passwords with `mysql`, `mysqladmin`, and MySQL Administrator.

Answer 11:

You could use `mysqladmin flush-privileges` to accomplish the same task.

Answer 12:

You could use `mysqladmin processlist` to accomplish the same task.

Chapter 26. MySQL Administrator

Almost all examples and exercises in this study guide use the *world database* as the sample data set. The accompanying CD-ROM contains the data for this database and instructions that describe how to create and populate the database for use with your own MySQL installation.

Question 1:

Which of these tasks can be performed by both MySQL Administrator and MySQL Query Browser?

a. Create databases (schemas)

b. Create tables

c. Modify table records

Question 2:

You want to track the number of temporary tables created by the MySQL server at a given time. How do you create a custom graph?

Question 3:

What type of backup does MySQL Administrator produce?

Question 4:

Can MySQL Administrator restore SQL dump files created with other tools?

Question 5:

Are changes to startup variables in MySQL Administrator applied to the server immediately?

Question 6:

Can MySQL Administrator schedule backups?

Question 7:

Can MySQL Administrator back up individual tables?

Question 8:

How can MySQL Administrator support replication?

Question 9:

What log files can MySQL Administrator browse?

Question 10:

Does MySQL Administrator show an entire log file at once?

Question 11:

On Windows, you can use MySQL Administrator to start a server that isn't running. How would you do

that? Could you start servers on any operating system using that method?

Question 12:

Can selective restores be performed? That is, can you restore particular tables from a file that contains a backup of entire databases?

Question 13:

What is the difference between the threads and user connections displays in the Server Connections section?

Question 14:

How many MySQL server instances can the MySQL System Tray Monitor manage?

Question 15:

What actions does the MySQL System Tray Monitor support?

Question 16:

Can you edit a table's structure (columns, indexes, and so forth) in MySQL Administrator, or do you have to switch to MySQL Query Browser to do this?

Question 17:

MySQL Administrator stores connection profiles in a file named `mysqlx_user_connections.xml`. Which statements about this file are true?

a. The file stores connection profiles for MySQL Administrator only.

b. The file is platform-specific, but you may copy it (for example, from one Windows machine to another Windows machine).

c. You can edit the file's contents only in the Connection dialog.

d. You can edit the file's contents in the connection editor.

Question 18:

Which statements about the Options dialog are true?

a. You can configure MySQL Administrator-specific options using the Options dialog in MySQL Query Browser, and the other way around.

b. You can manage connection profiles for MySQL Query Browser and for MySQL Administrator in the Options dialog of either tool.

c. You can configure the MySQL Table Editor that is used by both tools in the Options dialog of either tool.

Answers to Exercises

Answer 1:

Both tools can create databases (schemas) and tables, but only MySQL Query Browser can modify table records.

Answer 2:

1. Right-click the graph area, choose `Add a Group`, and name the group `Temporary tables`.
2. Right-click within the new group, and choose `Add a Graph`.
3. Choose a line graph, and use `^[created_tmp_tables]` as the formula.

Answer 3:

A text file containing a series of SQL statements, similar to the default output of `mysqldump`.

Answer 4:

Possibly, but you must take special care to ensure that the proper character set is selected before restoring.

Answer 5:

No. When you click `Apply Changes` the configuration file is modified, and you must restart the server for changes to take effect.

Answer 6:

Yes, on a daily, weekly, or monthly basis.

Answer 7:

Yes. Choose the schema you want to back up, then uncheck the boxes next to the tables you do not want to back up.

Answer 8:

The Replication tab of the Startup Variables section can be used to configure replication.

If the slaves have been configured to report themselves to the master (by setting at least the host names of the slave servers in the Replication tab of the Startup Variables section), you can use the Replication Status section to track the status of all slaves.

Answer 9:

- The general query log
- The slow query log
- The error log

Answer 10:

It depends. Large log files are paged and you can browse one page at a time.

Answer 11:

On Windows, you can hold down the `Ctrl` key and select the `Skip` button in the Connection dialog. This causes MySQL Administrator to present its main window in configure-service mode. In the Start/Stop Service tab of the Service Control section, you can select the Start Service button to start the server. This is possible only if MySQL Administrator and the server are running on the same machine. It's also necessary that the server has been configured to run as a Windows service.

Answer 12:

Yes. Load the restore file, click `Analyze Backup File Content` and uncheck the databases and tables you do not want to restore.

Answer 13:

A single user can have multiple threads. While a user may appear multiple times in the threads tab, a given user will only appear once on the User Connections tab.

Answer 14:

As many server instances as your operating system supports.

Answer 15:

The MySQL System tray Monitor can start a server instance, stop an instance, and configure an instance.

Answer 16:

Both MySQL Administrator and MySQL Query Browser provide access to the MySQL Table Editor, which allows you to edit a table's structure. In MySQL Administrator, select the Catalogs section, select a schema (database), and choose the table you want to edit.

Answer 17:

Only the last statement is true. Connection profiles stored in the `mysqlx_user_connections.xml` file are shared among various MySQL tools; for example, MySQL Query Browser uses the same information. The information is stored in plain text (in XML format) and can therefore be used by any MySQL Administrator (or MySQL Query Browser) installation, independent of the operating system. The file's contents can be edited using a plain text editor, in the Connection dialog, or in the connection editor that is accessible via the `Tools` menu.

Answer 18:

The Administrator section appears only when you are running MySQL Administrator, and the Browser section appears only when you are running MySQL Query Browser. Thus, you cannot configure MySQL Query Brower-specific options in MySQL Administrator, or the other way around. The other two statements are true.

Chapter 27. Character Set Support

Almost all examples and exercises in this study guide use the *world database* as the sample data set. The accompanying CD-ROM contains the data for this database and instructions that describe how to create and populate the database for use with your own MySQL installation.

Question 1:

To tell the server to use only specific character sets, what would you do?

1. Issue a SET statement like this:

   ```
   mysql> SET @@global.character_set = latin1, utf8, ucs2;
   ```

2. Start the server with the `--character-set` option, like this:

   ```
   shell> mysqld --character-set = "latin1, utf8, ucs2"
   ```

3. You can do this only by compiling the server from source.

Question 2:

Suppose that you plan to store values that all have the same length, using the `ucs2` character set. What data type would be better suited for this, CHAR or VARCHAR?

Question 3:

Suppose that you plan to store values that all have the same length, using the `utf8` character set. What data type would be better suited for this, CHAR or VARCHAR?

Answers to Exercises

Answer 1:

You can do this only by compiling the server from source.

Answer 2:

CHAR would be suited better because `ucs2` stores every character in two bytes. VARCHAR would use an extra length byte per value.

Answer 3:

VARCHAR would probably be suited better because `utf8` stores characters in one to three bytes, depending on the kind of character. For the theoretical case that *all* values you store use three bytes per character, CHAR would be the better choice because it saves one byte per value (the length byte of VARCHAR).

Chapter 28. Locking

Almost all examples and exercises in this study guide use the *world database* as the sample data set. The accompanying CD-ROM contains the data for this database and instructions that describe how to create and populate the database for use with your own MySQL installation.

Question 1:

In principle, what operations must a reader block, and what operations must a writer block?

Question 2:

What are reasons to acquire explicit locks, rather than relying on the server to acquire implicit locks?

Question 3:

Which privileges do you need to lock a table?

Question 4:

What prerequisite must be satisfied for acquiring a `READ LOCAL` lock?

Question 5:

Normally, a `WRITE` lock is acquired when no other clients are using the table. In which way does this behavior change if you request a `LOW_PRIORITY WRITE` lock?

Question 6:

What locking levels does MySQL have, and what storage engines do those levels apply to?

Question 7:

With locking, deadlocks may occur, but there's one exception. What is it?

Question 8:

In an application, you want to use the advisory locking mechanism provided by MySQL. How would you implement the following procedure?

1. Check whether the `app lock` lock is free.
2. If it's free, get the lock.
3. Release the lock.

Question 9:

Assume that you've locked the `City` table with a `READ` lock. If you try to select data from the table, what will happen?

Question 10:

Assume that you've locked the `City` table with a `READ` lock. If you try to insert data into the table, what will happen?

Question 11:

Assume that one client has locked the `City` table with a `READ` lock. If a second client tries to insert data into the table, what will happen?

Question 12:

The connection with ID `2098` requests a `WRITE` lock on the table `City`. However, the connection with ID `2099` has already obtained a `READ` lock on the same table. Which of the following statements correctly describes how the server handles the `WRITE` lock request?

a. The `WRITE` lock request results in an error.

b. The server automatically releases the `READ` lock of connection `2099` and gives connection `2098` a `WRITE` lock.

c. The server makes the `WRITE` lock request of connection `2098` wait until connection `2099` releases its `READ` lock.

d. Connection `2098` cannot obtain the `WRITE` lock as long as connection `2099` has the `READ` lock, so the server silently converts the `WRITE` lock request by connection `2098` to a `READ` lock request.

Question 13:

Under what circumstances is an explicit lock on a table released?

Question 14:

What table or tables will the following statements lock, if any?

```
mysql> LOCK TABLES City READ; LOCK TABLES Country WRITE;
```

Question 15:

Assume that you've locked the `City` table with a `READ` lock. If another client now tries to select data from the table, what will happen?

Question 16:

Assume that you've locked the `City` table with a `READ` lock. Who can release the lock?

Answers to Exercises

Answer 1:

A reader must block writers, but not other readers. A writer must block both readers and writers.

Answer 2:

a. An implicit lock lasts for the duration of a single query only, which is unsuitable should you want to perform a multiple-statement update that requires no interference by other clients.

b. Acquiring explicit locks can improve performance over letting the server acquire implicit locks. The server would acquire locks for each single statement, whereas an explicit lock can be acquired for a group of statements.

c. Acquiring explicit locks can improve performance over letting the server acquire implicit locks because, for a group statements that change data, index flushing is reduced.

Answer 3:

You need the `LOCK TABLES` privilege and the `SELECT` privilege for each table to be locked.

Answer 4:

The prerequisite is the same for acquiring a `READ LOCAL` lock as for a `READ` lock. That is, the table must not be in use. However, if the locked table is a non-fragmented `MyISAM` table, the `READ LOCAL` lock differs from a `READ` lock in that it will allow concurrent inserts by other clients. A `READ` lock does not allow concurrent inserts.

Answer 5:

While the `LOW_PRIORITY WRITE` request is waiting to get its lock, it will allow other clients that request a `READ` lock to get their lock. A pending request for a regular `WRITE` lock causes `READ` lock requests that arrive later to wait.

Answer 6:

MySQL implements three locking levels:

1. Table-level locking (`MyISAM`, `MEMORY`, `MERGE` storage engines)
2. Page-level locking (`BDB` storage engine)
3. Row-level locking (`InnoDB` storage engine)

Answer 7:

With *table*-level locking, deadlocks cannot occur.

Answer 8:

1.

```
mysql> SELECT IS_FREE_LOCK('app lock');
+--------------------------+
| IS_FREE_LOCK('app lock') |
+--------------------------+
|                        1 |
+--------------------------+
```

`1` indicates that the lock is available.

2.

```
mysql> SELECT GET_LOCK('app lock', 10);
+--------------------------+
| GET_LOCK('app lock', 10) |
+--------------------------+
|                        1 |
+--------------------------+
```

`1` indicates that the lock could be acquired within 10 seconds. (In the `GET_LOCK()` call, `10` is

timeout value.)

You should check the result of the `GET_LOCK()` call even if `IS_FREE_LOCK()` indicated that the lock is available because another client might have acquired the lock in the meantime.

3.

```
mysql> SELECT RELEASE_LOCK('app lock');
+---------------------------+
| RELEASE_LOCK('app lock') |
+---------------------------+
|                         1 |
+---------------------------+
```

`1` indicates that the lock was successfully released.

Answer 9:

You acquired the `READ` lock, so you can read data from the table:

```
mysql> LOCK TABLES City READ;
mysql> SELECT * FROM City LIMIT 1;
+----+-------+---------+----------+------------+
| ID | name  | Country | District | Population |
+----+-------+---------+----------+------------+
|  1 | Kabul | AFG     | Kabol    |    1780000 |
+----+-------+---------+----------+------------+
```

Answer 10:

Trying to insert data after acquiring a `READ` lock results in an error:

```
mysql> INSERT INTO City (ID, name) VALUES (10000, 'Test City');
ERROR 1099 (HY000): Table 'City' was locked with a READ lock and
can't be updated
```

Answer 11:

When one client holds a `READ` lock on a table, attempts by a second client to insert data into the table results in that client waiting for the lock to be released:

```
mysql> INSERT INTO City (ID, name) VALUES (10000, 'Test City');
(nothing happens here until the first client releases the lock)
```

Answer 12:

The `WRITE` lock request of connection `2098` waits until connection `2099` releases the `READ` lock.

Answer 13:

A lock can be released several ways:

- When the client that acquired the lock issues an `UNLOCK TABLES` statement.
- When the client that acquired the lock issues another `LOCK TABLES` statement (no matter whether for the same or different tables).

- When the client that acquired the lock is terminated (either by ending it normally, or when the client is killed by another client).
- An administrator can kill a client connection. This releases locks held by the client.

Answer 14:

The first statement locks table `City`, but the lock is released immediately with the next lock request. As a result, table `Country` will be locked for read and write requests by other clients. If you want to lock both tables at the same time, you have to issue this statement:

```
mysql> LOCK TABLES City READ, Country WRITE;
```

Answer 15:

The `READ` lock that you acquired does not prevent other reads. All other clients can still read data from the table:

```
mysql> SELECT * FROM City limit 1;
+----+-------+---------+----------+------------+
| ID | name  | Country | District | Population |
+----+-------+---------+----------+------------+
|  1 | Kabul | AFG     | Kabol    |    1780000 |
+----+-------+---------+----------+------------+
```

Answer 16:

Only the client that acquired the lock can release it. One way to do so is with the `UNLOCK TABLES` statement:

```
mysql> UNLOCK TABLES;
Query OK, 0 rows affected
```

The lock also is released if the client acquires another lock with `LOCK TABLES`, closes the connection, or is aborted. (An administrator who has appropriate privileges can abort the connection by using a `KILL` statement.)

Chapter 29. Storage Engines

Almost all examples and exercises in this study guide use the *world database* as the sample data set. The accompanying CD-ROM contains the data for this database and instructions that describe how to create and populate the database for use with your own MySQL installation.

Question 1:

How can the `MyISAM` storage engine be disabled?

a. Only when compiling MySQL from source, by using the `--without-myisam` option.

b. By starting the server with the `--skip-myisam` option.

c. By issuing the statement `SET GLOBAL have_myisam = 0`. You need the `SUPER` privilege to do this.

d. The `MyISAM` storage engine cannot be disabled.

Question 2:

What do all storage engines have in common?

Question 3:

How can you turn the `MyISAM` table `City` into an `InnoDB` table?

Question 4:

How can you find out which storage engine is used for the table `City` in the `world` database?

Question 5:

How can you find out which storage engines are available, which ones are compiled in but disabled, and which ones aren't compiled in?

Question 6:

With table-level locking, what's the default behavior of the server?

a. Read requests take priority over write requests.

b. Write requests take priority over read requests.

Question 7:

For `MyISAM` tables, how can you modify the server's default scheduling priorities for read and write requests?

Question 8:

Assume that you have a `MERGE` table that is a collection of three underlying `MyISAM` tables. To lock all of these tables, what would you have to do?

Question 9:

Which statements are true for `InnoDB` tables?

a. The tablespace consist of one or more files that are either regular files or raw partitions.

b. If `InnoDB` is configured to use one tablespace per table, the number of tablespace files is equal the number of `InnoDB` tables.

Question 10:

What is a transaction?

Question 11:

Can you use transactions that consist of more than one statement if the autocommit mode is enabled?

Question 12:

Give an example of using a savepoint.

Question 13:

What kind of locking methods does `MyISAM` use?

a. Table-level locking.

b. Page-level locking.

c. Row-level locking with lock escalation.

d. Row-level locking without lock escalation.

Question 14:

In `InnoDB`, what happens if a deadlock occurs?

Question 15:

How can you set the transaction level to `SERIALIZE` at server startup?

Question 16:

Consider the `MyISAM` table `CountryLanguage` in the `world` database. How would you add a foreign key constraint to the `CountryCode` column of that table that references the `Code` column of the `Country` table and that should have the following effects?

- If a country (code) is deleted in the `Country` table, corresponding rows in the `CountryLanguage` table should also be deleted.
- Changing the country code in the `Code` column of the `Country` table should be disallowed if there are corresponding rows in the `CountryLanguage` table.

Question 17:

Which of the following alternatives correctly describes what the server does when it finds that the disk is full during an update operation on a `MyISAM` table?

a. It cancels the update operation silently.

b. It cancels the update operation with an error message.

c. It waits until free space becomes available.

d. It deletes rows in the table until there's enough free space to complete the operation.

e. It replaces existing rows with the data the update operation would insert or change.

Question 18:

Is the following statement about the `InnoDB` storage engine true or false?

The `InnoDB` storage engine has its own error log.

Question 19:

Is the following statement about the `InnoDB` storage engine true or false?

The `InnoDB` storage engine uses both its own log files and the MySQL binary log (if enabled).

Question 20:

Is the following statement about the `InnoDB` storage engine true or false?

The `InnoDB` storage engine uses row-level locking, which provides better query concurrency than page-level or table-level locking and gives better performance in environments where there are many reads and writes at the same time.

Question 21:

Is the following statement about the `InnoDB` storage engine true or false?

The `SELECT ... FOR UPDATE` SQL statement makes sense for a transactional storage engine like `InnoDB`, but not for a non-transactional storage engine like `MyISAM`.

Question 22:

Where would you set the `innodb_flush_log_at_trx_commit` option? What will you have to do after setting it to cause it to take effect??

Question 23:

When specifying a shared tablespace file in a MySQL option file, you normally indicate a fixed size for it (for example, `ibdata2:100M`). What happens when all tablespace files are full? What could you do to make tablespace files dynamic? Give an example how you would do that.

Question 24:

Can you have a shared tablespace that uses regular files and raw partitions (device files) at the same time?

Question 25:

How can you check the amount of free space available in the InnoDB tablespace?

Question 26:

Consider the following session listing for one client:

```
mysql> START TRANSACTION;
mysql> SELECT * FROM trans;
Empty set
mysql> INSERT INTO trans VALUES (1),(2),(3);
Query OK, 3 rows affected
Records: 3  Duplicates: 0  Warnings: 0
```

Now, a second client issues the following statements:

```
mysql> SELECT * FROM trans;
```

How many rows will the second client see if both clients are running with an InnoDB isolation level of REPEATABLE READ?

Question 27:

Suppose that CountryList is an InnoDB table. Consider the following session listing:

```
mysql> SELECT COUNT(*) FROM CountryList;
+----------+
| COUNT(*) |
+----------+
|      192 |
+----------+
mysql> START TRANSACTION;
mysql> DELETE FROM CountryList;
mysql> SELECT COUNT(*) FROM CountryList;
+----------+
| COUNT(*) |
+----------+
|        0 |
+----------+
mysql> ROLLBACK;
```

How many rows will the table have after the ROLLBACK statement has been issued?

Question 28:

Can you undo a ROLLBACK or a COMMIT? If so, how would you do that?

Question 29:

Suppose that a client is in the middle of performing a transaction. How does the server handle the transaction under the following conditions:

a. The client loses the connection before ending the transaction.

b. The client itself closes the connection before ending the transaction.

Question 30:

Are there circumstances under which an `INSERT LOW_PRIORITY` statement might never be performed?

Question 31:

According to the following session listing, it appears that the `SELECT` query took longer than two minutes to execute. What is the most probable reason for that?

```
mysql> INSERT DELAYED INTO City (ID, name) VALUES (20000, 'Delayne');
Query OK, 1 row affected (0.00 sec)
mysql> SELECT ID, name FROM City WHERE ID = 20000;
+-------+---------+
| ID    | name    |
+-------+---------+
| 20000 | Delayne |
+-------+---------+
1 row in set (2 min 5.61 sec)
```

Question 32:

Which of the following statements are true for `MEMORY` tables?

a. Table structure, data, and indexes are held in memory only.

b. They are read-only.

c. They support row-level locking.

d. They have extremely high performance.

Question 33:

What kind of indexes can you use for `MEMORY` tables?

Question 34:

What statements are true about `FEDERATED` tables?

a. They don't have a `.frm` file because they're located on a remote server.

b. `FEDERATED` tables are always located on a remote server.

c. `FEDERATED` tables do not support transactions.

d. MySQL doesn't use any locking for `FEDERATED` tables.

e. One of their main benefitis is that they allow for accessing tables from different servers with a single query.

f. `FEDERATED` tables are created like any other tables. The storage engine, however, must be given as `ENGINE = FEDERATED`, and they need a special string in the `COMMENT` table option that contains connection information.

Question 35:

What's the difference between “NDB Cluster” and “MySQL Cluster.”?

Question 36:

What are the two main architectural properties of MySQL Cluster?

Question 37:

Which storage engines are always available, that is, they cannot be disabled at compile time or at runtime?

Question 38:

How do you disable the `InnoDB` storage engine at compile time?

Question 39:

How do you disable the `InnoDB` storage engine at server startup?

Question 40:

Name some benefits of the `MyISAM` storage engine.

Question 41:

What row-storage formats does the `MyISAM` storage engine support?

Question 42:

Can you create a `MERGE` table that is a collection of `MyISAM` tables that includes both compressed and uncompressed tables?

Question 43:

Which data-modifying operation can be disabled for `MERGE` tables?

a. `INSERT` operations.

b. `UPDATE` operations.

c. `DELETE` operations.

Question 44:

What does the acronym `ACID` stand for?

Question 45:

What does the term *implicit commit* mean?

Question 46:

What kind of locking methods does `InnoDB` use?

a. Table-level locking.

b. Page-level locking.

c. Row-level locking with lock escalation.

d. Row-level locking without lock escalation.

Question 47:

In `InnoDB`, how can you lock rows of the `City` table with a `SELECT` statement?

Question 48:

Suppose that you want a table to include a column that has a `FULLTEXT` index. What storage engines could you use?

Question 49:

Which SQL statements tell you what storage engine a given table `mytable` has?

Question 50:

In an application, you expect to have a high number of reads as well as a high number of writes at the same time. Which storage engine is best suited to handle this?

Question 51:

How would you disable the `InnoDB` storage engine? Why might you want to do that?

Question 52:

Is the following statement about the `InnoDB` storage engine true or false?

The `InnoDB` storage engine keeps table data and indexes in a shared tablespace.

Question 53:

Is the following statement about the `InnoDB` storage engine true or false?

The `InnoDB` storage engine keeps table definitions in `.frm` files.

Question 54:

Is the following statement about the `InnoDB` storage engine true or false?

The `InnoDB` storage engine can use compressed tables.

Question 55:

Is the following statement about the `InnoDB` storage engine true or false?

The `InnoDB` storage engine is multi-versioned, which means that different transactions use different versions of the data.

Question 56:

Is the following statement about the `InnoDB` storage engine true or false?

The `InnoDB` storage engine is multi-versioned, which means that all versions of `InnoDB` can use the same data files.

Question 57:

Is the following statement about the `InnoDB` storage engine true or false?

Deadlock can occur with the `InnoDB` storage engine, unlike with the `MyISAM` storage engine.

Question 58:

Is the following statement about the `InnoDB` storage engine true or false?

`SELECT ... LIMIT` is a MySQL extension to standard SQL that cannot be used with the `InnoDB` storage engine.

Question 59:

What is the effect of setting the `innodb_flush_log_at_trx_commit` option to 1 or 0?

Question 60:

By default, an `InnoDB` shared tablespace is stored in a single regular file named `ibdata1` in the MySQL data directory, but you can configure the tablespace to have more than one component by setting the `innodb_data_file_path` option in the `[mysqld]` section of a MySQL option file. What other kind of file can you use in a tablespace, and what would be the advantage of not using a regular file?

Question 61:

When creating an `InnoDB` table, how do you control which tablespace file `InnoDB` will use for storing that table's contents?

Question 62:

For an auto-extending shared tablespace file, how can you prevent it from expanding to use all available space on the filesystem where it is located?

Question 63:

With `InnoDB`, you can create tables that are larger than filesystem limits on maximum file size. Why is that so? Give an example.

Question 64:

Can an `InnoDB` shared tablespace be distributed across different filesystems?

Question 65:

How can you view status information about `InnoDB`?

Question 66:

Is the following statement about the `InnoDB` rollback mechanism true or false?

`InnoDB` uses information in its log files to perform rollbacks.

Question 67:

Is the following comment about the `InnoDB` rollback mechanism true or false?

`InnoDB` uses a data structure called the rollback segment in the `InnoDB` shared tablespace to store transaction undo information. If your operations will require large transactions, you must ensure that the tablespace is large enough to store that information.

Question 68:

How would you define “multi-versioning” as used by the `InnoDB` storage engine?

Question 69:

After the following transaction has been ended with `ROLLBACK`, what will the contents of the table `t` be? Why?

```
mysql> CREATE TABLE t (i INT) ENGINE = InnoDB;
mysql> CREATE TABLE t2 (i INT) ENGINE = InnoDB;
mysql> START TRANSACTION;
mysql> INSERT INTO t SET i = 1;
mysql> DROP TABLE t2;
mysql> ROLLBACK;
```

Question 70:

Consider the following `InnoDB` table and the session listing:

```
mysql> DESCRIBE CountryList;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| Code      | char(3)     | NO   |     |         |       |
| Name      | char(52)    | NO   |     |         |       |
| IndepYear | smallint(6) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
mysql> SELECT * FROM CountryList;
Empty set (0.00 sec)

mysql> SET AUTOCOMMIT=0;
mysql> INSERT INTO CountryList VALUES('XXX','XLand',2003);
Query OK, 1 row affected (0.00 sec)

mysql> ROLLBACK;
```

What are the contents of `CountryList` after the `ROLLBACK` statement has been issued?

Question 71:

Consider the following `InnoDB` table and the session listing:

```
mysql> DESCRIBE CountryList;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| Code      | char(3)     | NO   |     |         |       |
| Name      | char(52)    | NO   |     |         |       |
| IndepYear | smallint(6) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
mysql> SELECT * FROM CountryList;
Empty set (0.01 sec)

mysql> SET AUTOCOMMIT=0;
mysql> INSERT INTO CountryList VALUES('XXX','XLand',2003);
Query OK, 1 row affected (0.00 sec)

mysql> START TRANSACTION;
mysql> INSERT INTO CountryList VALUES('YYY','YLand',2004);
Query OK, 1 row affected (0.00 sec)
```

```
mysql> ROLLBACK;
```

What are the contents of `CountryList` after the `ROLLBACK` statement has been issued?

Question 72:

Consider the following `InnoDB` table and the session listing:

```
mysql> DESCRIBE CountryList;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| Code      | char(3)     | NO   |     |         |       |
| Name      | char(52)    | NO   |     |         |       |
| IndepYear | smallint(6) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
mysql> SELECT * FROM CountryList;
Empty set (0.01 sec)

mysql> SET AUTOCOMMIT=0;
mysql> START TRANSACTION;
mysql> START TRANSACTION;
mysql> INSERT INTO CountryList VALUES('XXX','XLand',2003);
Query OK, 1 row affected (0.00 sec)

mysql> COMMIT;
mysql> INSERT INTO CountryList VALUES('YYY','YLand',2004);
Query OK, 1 row affected (0.00 sec)

mysql> ROLLBACK;
```

What are the contents of `CountryList` after the `ROLLBACK` statement has been issued?

Question 73:

Consider the following `InnoDB` table and the session listing:

```
mysql> DESCRIBE CountryList;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| Code      | char(3)     | NO   |     |         |       |
| Name      | char(52)    | NO   |     |         |       |
| IndepYear | smallint(6) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
mysql> SELECT * FROM CountryList;
Empty set (0.01 sec)

mysql> SET AUTOCOMMIT=0;
mysql> INSERT INTO CountryList VALUES('XXX','XLand',2003);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO CountryList VALUES('YYY','YLand',2004);
Query OK, 1 row affected (0.00 sec)

mysql> ROLLBACK;
```

What are the contents of `CountryList` after the `ROLLBACK` statement has been issued?

Question 74:

You want to use `INSERT` to add data to a `MyISAM` table, but you suspect that the table is locked by a lock request of another client. You're using an interactive client (for example `mysql`), so you would have to wait for that lock to be released before you can continue with your work. How can you solve this problem, and still insert the data?

Question 75:

Assume that you have a `MyISAM` table that is subject to many read (`SELECT`) requests. Compared to the number of reads, you have only a few write (`INSERT`) requests taking place. Furthermore, you consider the reads more important than the write requests. What could you do to give read requests priority over write requests?

Question 76:

Which storage engines are always available in MySQL? Which storage engines support transactions, and what other storage engines do you know?

Answers to Exercises

Answer 1:

The `MyISAM` storage engine cannot be disabled.

Answer 2:

Every table in a database has a format (`.frm`) file in the database directory.

Answer 3:

By issuing `ALTER TABLE City ENGINE = InnoDB`.

Answer 4:

There are several ways to determine a table's storage engine:

- Use `SHOW CREATE TABLE`:

```
mysql> SHOW CREATE TABLE world.City\G
*************************** 1. row ***************************
       Table: City
Create Table: CREATE TABLE `City` (
  `ID` int(11) NOT NULL auto_increment,
  `Name` char(35) NOT NULL default '',
  `CountryCode` char(3) NOT NULL default '',
  `District` char(20) NOT NULL default '',
  `Population` int(11) NOT NULL default '0',
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1
```

- Use `SHOW TABLE STATUS`:

```
mysql> USE world; SHOW TABLE STATUS LIKE 'City'\G
Database changed
*************************** 1. row ***************************
           Name: City
         Engine: MyISAM
        Version: 10
```

```
     Row_format: Fixed
           Rows: 4079
 Avg_row_length: 67
    Data_length: 273293
Max_data_length: 4827858800541171711
   Index_length: 48128
      Data_free: 0
 Auto_increment: 4080
    Create_time: 2005-06-08 10:51:03
    Update_time: 2005-06-08 10:51:03
     Check_time: NULL
      Collation: latin1_swedish_ci
       Checksum: NULL
 Create_options:
        Comment:
```

- Use the TABLES table in the INFORMATION_SCHEMA database:

```
mysql> SELECT TABLE_NAME, ENGINE FROM INFORMATION_SCHEMA.TABLES
    -> WHERE TABLE_SCHEMA = 'world' AND TABLE_NAME = 'City';
+------------+--------+
| TABLE_NAME | ENGINE |
+------------+--------+
| City       | MyISAM |
+------------+--------+
```

Answer 5:

By issuing the SHOW ENGINES statement. The result of the statement consists of three column (of which two are displayed here):

```
+------------+----------+
| Engine     | Support  |
+------------+----------+
| MyISAM     | DEFAULT  |
| HEAP       | YES      |
| MEMORY     | YES      |
| MERGE      | YES      |
| MRG_MYISAM | YES      |
| ISAM       | NO       |
| MRG_ISAM   | NO       |
| InnoDB     | YES      |
| INNOBASE   | YES      |
| BDB        | YES      |
| BERKELEYDB | YES      |
| NDBCLUSTER | DISABLED |
| NDB        | DISABLED |
| EXAMPLE    | YES      |
| ARCHIVE    | NO       |
| CSV        | YES      |
| FEDERATED  | YES      |
| BLACKHOLE  | NO       |
+------------+----------+
```

A value of DEFAULT or YES means that the storage engine is compiled in and enabled. A value of DISABLED means it's compiled in but disabled, and a value of NO means it was not compiled in when this MySQL server was built. (For each NO engine, the engine could be made available by reconfiguring and recompiling.)

Answer 6:

Write requests take priority over read requests.

Answer 7:

- Write requests can be given a lower priority by using the scheduling modifiers `LOW_PRIORITY` or `DELAYED`. `LOW_PRIORITY` is available for all statements that change data (such as `INSERT`, `UPDATE`, `DELETE`, and `REPLACE`). `DELAYED` is available for `INSERT` and `REPLACE` operations only.
- Read requests can be given a higher priority by using the scheduling modifier `HIGH_PRIORITY`.

Answer 8:

It's sufficient to lock the `MERGE` table. This will lock all three underlying `MyISAM` tables.

Answer 9:

The `InnoDB` tablespace consists of at least one file that can be a regular file or a raw partition. The shared tablespace may contain of more than one file. There is always a shared tablespace, even if `InnoDB` is configured to use on tablespace per table. In that case, the number of tablespace files is equal to the number of `InnoDB` tables that were created *after* starting the server with that option, plus at least one file for the shared tablespace. Tables that were created in the shared tablespace *before* the option to use one tablespace per table was enabled will remain in the shared tablespace. There will be exactly the same number of tablespace files as there are `InnoDB` tables if the shared tablespace contains as many tables as there are shared tablespace files.

Answer 10:

A transaction is a logical grouping of statements that is handled by the database server as a single unit.

Answer 11:

Yes, by starting a transaction with the `START TRANSACTION` statement. Alternatively, you may use `BEGIN` or `BEGIN WORK`.

Answer 12:

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE City SET Name = 'Kabbul' WHERE Name LIKE 'Kab%';
Query OK, 3 rows affected (0.06 sec)
Rows matched: 3  Changed: 3  Warnings: 0

mysql> UPDATE City SET Name = 'Werbelina' WHERE Name LIKE 'Berl%';
Query OK, 1 row affected (0.04 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SAVEPOINT end_of_city_updates;
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE Country SET Code = 'zzz' WHERE Code = 'deu';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> ROLLBACK TO SAVEPOINT end_of_city_updates;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> COMMIT;
Query OK, 0 rows affected (0.29 sec)
```

Answer 13:

MyISAM uses table-level locking only.

Answer 14:

In most cases, InnoDB will detect a deadlock. In that case, it terminates one of the deadlocking transactions. When doing this, it tries to roll back the transaction that has modified the smallest number of rows. If the deadlock is not detected, the deadlocked transactions eventually begin to time out and InnoDB rolls them back as they do.

Answer 15:

You can put these lines in an option file:

```
[mysqld]
transaction-isolation = SERIALIZE
```

Alternatively, you can give that option on the command line:

```
shell> mysqld --transaction-isolation=serialize
```

Answer 16:

First of all, you need to change the storage engine from MyISAM to InnoDB, for the two tables that are affected:

```
mysql> ALTER TABLE Country ENGINE = InnoDB;
Query OK, 239 rows affected (0.08 sec)
Records: 239  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE CountryLanguage ENGINE = InnoDB;
Query OK, 984 rows affected (0.14 sec)
Records: 984  Duplicates: 0  Warnings: 0
```

Next, you need to specify the foreign key constraint. Note that there is no action specified for UPDATE operations. This means that updates (changes of existing data) will not be allowed for the Code column of the referenced table (Country).

```
mysql> ALTER TABLE CountryLanguage
    -> ADD FOREIGN KEY (CountryCode)
    -> REFERENCES Country (Code) ON DELETE CASCADE;
```

You can see that UPDATE operations will fail, while DELETE operations will cascade:

```
mysql> UPDATE Country SET Code = 'XYZ' WHERE Code = 'ABW';
ERROR 1217 (23000): Cannot delete or update a parent row:
a foreign key constraint fails
mysql> SELECT * FROM CountryLanguage WHERE CountryCode = 'ABW';
+-------------+------------+------------+------------+
| CountryCode | Language   | IsOfficial | Percentage |
+-------------+------------+------------+------------+
| ABW         | Dutch      | T          |        5.3 |
| ABW         | English    | F          |        9.5 |
```

```
| ABW         | Papiamento | F           |       76.7 |
| ABW         | Spanish    | F           |        7.4 |
+-------------+------------+-------------+------------+
4 rows in set (0.00 sec)

mysql> DELETE FROM Country WHERE Code = 'ABW';
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM CountryLanguage WHERE CountryCode = 'ABW';
Empty set (0.00 sec)
```

Answer 17:

The server waits until free space becomes available.

Answer 18:

False. The MySQL error log is used to store errors, but it is not `InnoDB`-specific.

Answer 19:

True. `InnoDB` records transaction activity in the `InnoDB` log, and MySQL records statements for all storage engines in the binary log if they modify data.

Answer 20:

True. When there are many reads and writes at the same time, row-level locking provides superior performance than does table-level locking.

Answer 21:

True. `SELECT ... FOR UPDATE` is used to select a set of rows that you also intend to update. It makes sense for `InnoDB`, which allows individual rows to be locked. It does not make sense for the `MyISAM` storage engine, which locks the entire table even if only some of its rows will be updated.

Answer 22:

The `innodb_flush_log_at_trx_commit` setting should be given in the `[mysqld]` section of the MySQL option file, or on the command line. Putting it in the option file is advisable because you don't have to remember it each time you start the server.

To cause the change to take effect, restart the server.

Answer 23:

When there is no more free space in a fixed size tablespace, `InnoDB` rolls back the next statement that tries to add data. (The application is expected to detect this error and perform a `ROLLBACK` operation to roll back the entire transaction.) To avoid running out of space, you can add the `autoextend` attribute to the specification of the last file in the tablespace. For example, to create a tablespace from two 100MB files in the data directory and make the second one auto-extending, you could put something like this in your `my.cnf` file:

```
[mysqld]
innodb_data_file_path = ibdata1:100M;ibdata2:100M:autoextend
```

Answer 24:

Yes. You can use a mix of regular files and raw partitions in the same tablespace.

Answer 25:

Issue a `SHOW TABLE STATUS` statement that includes output for at least one `InnoDB` table. The approximate amount of available space in the table's tablespace is displayed in the `Comment` field for every `InnoDB` table shown in the output. For example:

```
mysql> SHOW TABLE STATUS LIKE 't%';
+----------+--------+------------+-   -+------------------------+
| Name     | Type   | Row_format | ... | Comment                |
+----------+--------+------------+-   -+------------------------+
| t        | InnoDB | Compact    | ... | InnoDB free: 179200 kB |
| test     | InnoDB | Compact    | ... | InnoDB free: 179200 kB |
+----------+--------+------------+-   -+------------------------+
```

`InnoDB` free space information also is available in the `INFORMATION_SCHEMA.TABLES` table.

Answer 26:

The second client will see no rows because the first client has not issued a `COMMIT`. If the first client does issue a `COMMIT`, the second client then will see the three newly inserted rows.

Answer 27:

The table will have its original 192 rows. `ROLLBACK` rolls back the `DELETE` statement.

Answer 28:

A `ROLLBACK` cannot be undone. You must repeat your transaction. A `COMMIT` also cannot be undone.

Answer 29:

In either case (whether the connection closes abnormally or normally), the server treats connection termination as if the client had issued a `ROLLBACK` statement, so the transaction is rolled back.

Answer 30:

Yes. An `INSERT LOW_PRIORITY` statement waits until there are no read requests or normal-priority update requests in progress or pending on that table. This includes new requests that arrive while the `INSERT LOW_PRIORITY` is waiting. If a server is so busy that there is never a time when no read requests are in progress or pending, the `INSERT LOW_PRIORITY` might wait forever.

Answer 31:

Most probably there was a lock on that table obtained by another client. Only when that other client released its lock was the `INSERT DELAYED` statement performed. After that, the `SELECT` statement was performed. With no further information given, you can assume that the lock was in effect for at least 2 minutes and 5 seconds.

Answer 32:

a. False. Although the data and index information is stored in memory, the format (`.frm`) file is stored on disk.

b. False. You can insert, update, and delete data, just as you can with other tables.

c. False. `MEMORY` tables use table locking only.

d. True. The table's contents are always stored in memory and never need to be read from disk or writ-

ten to disk.

Answer 33:

- Hash indexes are the default for `MEMORY` tables. They are very fast but can be used only for comparisons that use the `=` or `<=>` operator.
- `BTREE` indexes can also use other comparison operators, such as `<` or `BETWEEN`.

Answer 34:

From the item list in the question, the following two items are false, and the rest are true:

- `FEDERATED` tables have a `.frm` file, just like any other table type.
- `FEDERATED` tables need not reside on a remote server. They may be located on another server that runs on the same machine, or they may even be tables from the same server.

Answer 35:

NDB Cluster refers to the cluster technology and is thus specific to the storage engine itself, whereas MySQL Cluster refers to a group of one or more MySQL servers that works as a "front end" to the NDB Cluster engine. That is, a MySQL Cluster consists of a group of one or more server hosts, each of which is usually running multiple processes that include MySQL servers, NDB management processes, and NDB database storage nodes.

Answer 36:

- MySQL Cluster is an *in-memory* database. Table contents are kept in RAM.
- MySQL Cluster is a *shared-nothing* system. No hardware components are shared between two nodes.

Answer 37:

`MyISAM`, `MERGE`, and `MEMORY`.

Answer 38:

By compiling from source with the `--without-innodb` option.

Answer 39:

By starting the server with the `--skip-innodb` option.

Answer 40:

- Has the most flexible `AUTO_INCREMENT` column handling of all storage engines.
- Can be compressed into fast, read-only tables.
- Supports `FULLTEXT` indexing.

- Support spatial data types.
- Is very fast for retrievals.
- Deadlock cannot occur.
- The table storage format is portable across platforms.

Answer 41:

Fixed-row, dynamic-row, and compressed format.

Answer 42:

Yes. `MERGE` tables can be created using a mix of compressed and uncompressed `MyISAM` tables.

Answer 43:

You can *explicitly* disable `INSERT` operations in the `CREATE TABLE ... ENGINE = MERGE` statement. *Implicitly*, all data-modifying operations are disabled if *all* underlying `MyISAM` tables are compressed read-only tables.

Answer 44:

Atomicity, consistency, isolation, durability.

Answer 45:

Normally, transactions are commit explicitly with the `COMMIT` statement. Some other statements, however, commit transactions, too. Examples include DDL statements like `ALTER TABLE` and `CREATE TABLE`, or locking statements like `LOCK TABLES` and `UNLOCK TABLES`.

Answer 46:

`InnoDB` uses row-level locking without lock escalation. However, you may explicitly lock tables at the table level with `LOCK TABLES`.

Answer 47:

You can acquire either a shared lock or an exclusive lock:

- `SELECT * FROM City WHERE Name LIKE 'A%' LOCK IN SHARE MODE;`
- `SELECT * FROM City WHERE Name LIKE 'A%' FOR UPDATE;`

Answer 48:

`MyISAM` is the only storage engine that supports `FULLTEXT` indexing.

Answer 49:

You can use `SHOW CREATE TABLE mytable` or `SHOW TABLE STATUS LIKE 'mytable'`, or select from the `INFORMATION_SCHEMA.TABLES` table.

Answer 50:

`InnoDB` provides the best concurrency in an environment consisting of mixed reads and writes. Of the

MySQL storage engines available, it provides the most fine-grained locking (row-level locking), and it also uses multi-versioning to give each transaction its own view of the database.

Answer 51:

To disable the `InnoDB` storage engine, you could either put the setting in a MySQL option file, or start the server with the appropriate option. In your option file, you would have these lines:

```
[mysqld]
skip-innodb
```

The preceding is the preferred way to disable the storage engine. To disable it only for the next time the server starts, you would use the option on the command line rather than putting it in an option file:

```
shell> mysqld --skip-innodb
```

Disabling an unneeded storage engine reduces the server's memory requirements because it need not allocate buffers and other data structures associated with the engine.

It's also possible to disable the `InnoDB` storage engine entirely by compiling the server without it.

Answer 52:

The statement is true for the default `InnoDB` configuration. `InnoDB` also can be configured to use a tablespace per table.

Answer 53:

The statement is true.

Answer 54:

False. Compressed tables are a feature of the `MyISAM` storage engine, not of `InnoDB`.

Answer 55:

The statement is true.

Answer 56:

False. Multi-versioning means that different transactions use different versions of the data.

Answer 57:

True. Deadlock can occur with `InnoDB` because it uses row-level locking and it might determine that additional locks are necessary during the course of query processing. The `MyISAM` storage engine uses table-level locking; this cannot lead to deadlock because the server determines all necessary locks before executing a query.

Answer 58:

False. `SELECT ... LIMIT` is a MySQL extension to standard SQL, but it works with all MySQL storage engines.

Answer 59:

With an `innodb_flush_log_at_trx_commit` setting of 1, `InnoDB` writes and flushes the log buffer to the log files after each commit, making the transaction changes permanent in the database. With a setting of 0, `InnoDB` writes and flushes the log buffer about once a second, but not after each

commit. It is possible with a setting of 0 for about a second's worth of committed transactions to be lost if a crash occurs.

Answer 60:

You can specify a raw partition (a device file) for an `InnoDB` tablespace. This will avoid a level of overhead normally incurred when using regular files in a filesystem. Raw partitions also are not bound by file size limits; `InnoDB` can use the entire partition.

Answer 61:

You cannot control that in the default `InnoDB` configuration that stores all tables in the shared tablespace. In this case, `InnoDB` is free to store table contents anywhere in the tablespace that it finds available space. It will use any or all files in the tablespace, if necessary. If you configure `InnoDB` to use per-table tablespaces, contents for a given table are stored in its own tablespace.

Answer 62:

To limit the size to which `InnoDB` allows an auto-extending file to grow, add a `max` specifier after `autoextend`. To allow `ibdata2` to grow to a maximum of 500MB, configure the tablespace like this:

```
[mysqld]
innodb_data_file_path = ibdata1:100M;ibdata2:100M:autoextend:max:500M
```

You can also limit the maximum size of a tablespace file indirectly if your operating system provides a quota system. This involves procedures that are not covered in this study guide.

Answer 63:

You can create an `InnoDB` shared tablespace from multiple files if you want. Any regular file that is part of the tablespace is subject to the size limitations of the filesystem, but `InnoDB` will store tables using more than one file if necessary. For example, if your filesystem imposes a limit of 2GB as the maximum size of a file, you can create the `InnoDB` tablespace from multiple files that are 2GB in size. To store a table that has 5GB of data, `InnoDB` could then use three such files, thus exceeding the filesystem's limitation. Another way to overcome file size limits is to use raw partitions that `InnoDB` can access directly (that is, not through the filesystem). The size of a raw partition that `InnoDB` can handle in a tablespace is constrained only by `InnoDB`'s internal size limit (4 billion database pages, where each page is 16KB by default).

Answer 64:

Yes. For example, you could have one tablespace file on a ReiserFS filesystem partition, and another tablespace file on an ext3 filesystem partition.

Answer 65:

The `SHOW ENGINE INNODB STATUS` statement displays information about the status of `InnoDB`. For example, you can issue the statement using the `mysql` client program (the output shown here has been shortened):

```
mysql> SHOW ENGINE INNODB STATUS\G
*************************** 1. row ***************************
Status:
=====================================
030322 20:54:14 INNODB MONITOR OUTPUT
=====================================
Per second averages calculated from the last 7 seconds
----------
SEMAPHORES
----------
```

```
OS WAIT ARRAY INFO: reservation count 11, signal count 11
Mutex spin waits 9, rounds 160, OS waits 1
RW-shared spins 18, OS waits 9; RW-excl spins 1, OS waits 1
------------
TRANSACTIONS
------------
Trx id counter 0 244754
Purge done for trx's n:o < 0 244747 undo n:o < 0 0
Total number of lock structs in row lock hash table 0
LIST OF TRANSACTIONS FOR EACH SESSION:
---TRANSACTION 0 244753, not started, OS thread id 1500
MySQL thread id 2, query id 881 localhost 127.0.0.1 superuser
SHOW ENGINE INNODB STATUS
--------
FILE I/O
--------
I/O thread 0 state: wait Windows aio
I/O thread 1 state: wait Windows aio
I/O thread 2 state: wait Windows aio
I/O thread 3 state: wait Windows aio
Pending normal aio reads: 0, aio writes: 0,
...
```

Answer 66:

False. `InnoDB` uses the rollback segment of the tablespace to perform rollbacks.

Answer 67:

True. The rollback segment contains undo information that is used to roll back transactions.

Answer 68:

In `InnoDB`, multi-versioning means that as transactions modify rows, `InnoDB` maintains isolation between them by maintaining multiple versions of the rows, and makes available to each transaction the appropriate version of the rows that it should see.

Answer 69:

The table `t` will contain the row added by the `INSERT` statement:

```
mysql> SELECT i FROM t;
+------+
| i    |
+------+
|    1 |
+------+
```

The reason for this is that `DROP TABLE` is one of the statements that causes an implicit commit of preceding uncommitted statements in the transaction.

Answer 70:

`CountryList` will contain no rows. Setting `AUTOCOMMIT` to `0` causes MySQL to treat the following SQL statements as a transaction. That transaction can be (and was) rolled back. To commit it instead of rolling it back, you would either have to issue `COMMIT` explicitly or issue another SQL statement that would commit the transaction implicitly.

Answer 71:

The table `CountryList` will contain the following row:

```
mysql> SELECT * FROM CountryList;
+------+-------+-----------+
| Code | Name  | IndepYear |
+------+-------+-----------+
| XXX  | XLand |      2003 |
+------+-------+-----------+
```

The `START TRANSACTION` statement causes an implicit commit of the preceding uncommitted `INSERT` statement, thus the first row is inserted. The second `INSERT`, however, is within a transaction that is rolled back.

Answer 72:

The table `CountryList` will contain the following row:

```
mysql> SELECT * FROM CountryList;
+------+-------+-----------+
| Code | Name  | IndepYear |
+------+-------+-----------+
| XXX  | XLand |      2003 |
+------+-------+-----------+
```

The explanation is *not* that the session has "nested" transactions. The second `START TRANSACTION` statement actually commits the first transaction implicitly and starts another transaction. `COMMIT` ends that transaction. Because the session is not running in autocommit mode, however, the next statement also begins a new transaction and it is unnecessary to issue an explicit `START TRANSACTION`. As a result, `ROLLBACK` rolls back the second `INSERT` statement.

Answer 73:

The table `CountryList` will contain no rows. In non-autocommit mode, everything is regarded as a transaction that is committed only when a `COMMIT` statement is issued. Thus, all statements are rolled back with `ROLLBACK`.

Answer 74:

You can insert the data using an `INSERT DELAYED` statement. This allows you to proceed immediately. The rows to be inserted will be buffered by the server and added to the table when it becomes free.

Answer 75:

To give read requests higher priority than write requests, you can use either of the following strategies:

- `INSERT DELAYED` will cause `INSERT` statements to wait until there are no more pending read requests on that table.
- `SELECT HIGH_PRIORITY` will give a `SELECT` statement priority over write requests.

Answer 76:

MySQL has storage engines that are always available (`MyISAM`, `MERGE`, and `MEMORY`). Transactional storage engines include `InnoDB` and `BDB`. There are other storage engines like `ARCHIVE`, `FEDERATED`, `BLACKHOLE`, `CSV`, `EXAMPLE`, `NDB`. The deprecated `ISAM` storage engine is no longer available as of MySQL 5.

Chapter 30. Table Maintenance

Almost all examples and exercises in this study guide use the *world database* as the sample data set. The accompanying CD-ROM contains the data for this database and instructions that describe how to create and populate the database for use with your own MySQL installation.

Question 1:

What is the main difference between `mysqlcheck` and `myisamchk` regarding how they access tables?

Question 2:

For which storage engines can you use `mysqlcheck`, and what can you do for each storage engine?

Question 3:

For which storage engines can you use `myisamchk`?

Question 4:

What types of table maintenance operations can you perform, and what SQL statements or tools does MySQL provide for those operations?

Question 5:

What SQL statement would you use to update a table's index statistics?

Question 6:

What command-line program would you use update a table's index statistics?

Question 7:

What is the effect of a table analysis?

Question 8:

What precautions should you observe when analyzing a table?

Question 9:

What does it mean to optimize a `MyISAM` table, and how would you do it?

Question 10:

Which programs or SQL statements can you use to check `InnoDB` tables?

Question 11:

Which programs or SQL statements can you use to repair `InnoDB` tables?

Question 12:

Using `mysqlcheck`, how would you perform the following operations?

a. Check the `Country` table in the `world` database.

b. Repair all tables in the `world` database (if necessary).

c. Optimize all tables in all databases.

Does all this work for all storage engines?

Question 13:

How would you repair the `City` table using `myisamchk`?

Question 14:

How would you start the server so that it automatically performs a quick recovery of `MyISAM` tables?

Answers to Exercises

Answer 1:

- `mysqlcheck` is a client program for the server; it determines which options were given on the command line, and then sends appropriate SQL statements to the MySQL server to perform the requested operation.
- `myisamchk` is not a client program. It performs operations on tables by accessing the table files directly.

Answer 2:

`mysqlcheck` can perform operations on `MyISAM` and `InnoDB` tables. It can check, repair, analyze, and optimize `MyISAM` tables, and it can check and optimize (but not repair or analyze) `InnoDB` tables.

Answer 3:

`myisamchk` works only for `MyISAM` tables.

Answer 4:

The following maintenance operations can be performed on tables. For each kind of operation MySQL provides an SQL statement (although it should be noted that not all statements work for all storage engines). These statements can also be called using the `mysqlcheck` client program or the `myisamchk` program (for `MyISAM` tables only). MySQL Administrator provides a graphical wrapper for these SQL statements, too, with the exception of `ANALYZE TABLE`.

a. Integrity check of tables. This can be done with the `CHECK TABLES` SQL statement.

b. Repair tables. This can be done with the `REPAIR TABLES` SQL statement.

c. Analyze tables. This can be done with the `ANALYZE TABLES` SQL statement.

d. Optimize tables. This can be done with the `OPTIMIZE TABLES` SQL statement.

Answer 5:

Use `ANALYZE TABLE` to update a table's index statistics:

```
mysql> ANALYZE TABLE City;
```

```
+------------+---------+----------+----------+
| Table      | Op      | Msg_type | Msg_text |
+------------+---------+----------+----------+
| world.City | analyze | status   | OK       |
+------------+---------+----------+----------+
```

Answer 6:

To update a table's statistics from the command line, use either `mysqlcheck` or `myisamchk`:

```
shell> mysqlcheck --analyze world City
world.City                                         OK

shell> myisamchk --analyze c:\mysql\data\world\City.MYI
Checking MyISAM file: c:\mysql\data\world\City.MYI
Data records:    4083   Deleted blocks:       0
- check file-size
- check key delete-chain
- check record delete-chain
- check index reference
```

Answer 7:

The effect of analyzing a table is twofold: A check is performed on the table and the table's index statistics are updated. Analyzing can thus help the query optimizer better process joins involving the table.

Answer 8:

- When using the `ANALYZE TABLE` statement or the `mysqlcheck` program, there are no special precautions to consider. The server automatically locks the table before performing the analysis.
- Before using `myisamchk --analyze`, you must ensure that no other program, such as the server, tries to access the table to be analyzed. One way to do this is to stop the server before you use `myisamchk`.

Answer 9:

Optimizing a `MyISAM` table means reclaiming unused space that results when rows are deleted from the table. Deleted rows can cause wasted space within a table, thus slowing down table reads. To optimize a `MyISAM` table, use the `OPTIMIZE TABLE` statement. For example:

```
mysql> OPTIMIZE TABLE City;
+------------+----------+----------+----------+
| Table      | Op       | Msg_type | Msg_text |
+------------+----------+----------+----------+
| world.city | optimize | status   | OK       |
+------------+----------+----------+----------+
```

You can also use `mysqlcheck --optimize`.

Answer 10:

To check an `InnoDB` table, you can use either of the following:

- The `CHECK TABLE` SQL statement

- The `mysqlcheck` program

Answer 11:

To repair an `InnoDB` table, you cannot use `REPAIR TABLE` or the `myisamchk` program (which, as its name indicates, is for `MyISAM` tables only). The method to use is to dump the `InnoDB` table with `mysqldump` and then reload it. (However, in most cases, `InnoDB`'s auto-recovery feature is sufficient to repair tables.)

Answer 12:

a.

```
shell> mysqlcheck --check world Country
```

The `--check` option doesn't have to be given because it's the default behavior. The command will work for `MyISAM` and `InnoDB` tables, but not for `MEMORY` tables.

b.

```
shell> mysqlcheck --repair world
```

One could also use the additional options `--quick` or `--force`. This command will work for `MyISAM` tables only.

c.

```
shell> mysqlcheck --optimize --all-databases
```

This command will work for `MyISAM` and `InnoDB` tables, but not for `MEMORY` tables.

Answer 13:

Before you start, you should make sure that no one accesses the table. This includes the server, so you should either stop the server or lock the table you're going to repair. Also note that the following command will only work for `MyISAM` tables:

```
shell> myisamchk --recover City
```

As an alternative to the `--recover` option, you could use the `--safe-recover` option, which is much slower.

Answer 14:

You would start the server like this:

```
shell> mysqld --myisam-recover=QUICK
```

Alternatively, you could put the `--myisam-recover` option in an option file.

Chapter 31. The INFORMATION_SCHEMA Database

Almost all examples and exercises in this study guide use the *world database* as the sample data set. The accompanying CD-ROM contains the data for this database and instructions that describe how to create and populate the database for use with your own MySQL installation.

Question 1:

The SHOW VARIABLES and SHOW STATUS SQL statements provide information about server system variables and status variables. How can you retrieve that information using the INFORMATION_SCHEMA database?

Question 2:

Using SHOW, how can you find out which tables are contained in the INFORMATION_SCHEMA database?

Question 3:

Assume that you want to retrieve the column names (and nothing else) of the City table in the world database. How can you do that using the INFORMATION_SCHEMA database, and what is the analogous SHOW statement?

Answers to Exercises

Answer 1:

The INFORMATION_SCHEMA database contains metadata only, so the information that SHOW VARIABLES and SHOW STATUS provide cannot be accessed through the INFORMATION_SCHEMA database.

Answer 2:

This SHOW statement tells you which tables are contained in the INFORMATION_SCHEMA database:

```
mysql> SHOW TABLES FROM information_schema;
+---------------------------------------+
| Tables_in_information_schema          |
+---------------------------------------+
| SCHEMATA                              |
| TABLES                                |
| COLUMNS                               |
| CHARACTER_SETS                        |
| COLLATIONS                            |
| COLLATION_CHARACTER_SET_APPLICABILITY |
| ROUTINES                              |
| STATISTICS                            |
| VIEWS                                 |
| USER_PRIVILEGES                       |
| SCHEMA_PRIVILEGES                     |
| TABLE_PRIVILEGES                      |
| COLUMN_PRIVILEGES                     |
| TABLE_CONSTRAINTS                     |
| KEY_COLUMN_USAGE                      |
+---------------------------------------+
```

Answer 3:

Using the INFORMATION_SCHEMA database, you would issue this statement:

```
mysql> SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS
    -> WHERE TABLE_SCHEMA = 'world' AND TABLE_NAME = 'City';
+-------------+
| COLUMN_NAME |
+-------------+
| ID          |
| Name        |
| CountryCode |
| District    |
| Population  |
+-------------+
```

You cannot retrieve just the column names using SHOW:

```
mysql> SHOW COLUMNS FROM City FROM world;
+-------------+----------+------+-----+---------+----------------+
| Field       | Type     | Null | Key | Default | Extra          |
+-------------+----------+------+-----+---------+----------------+
| ID          | int(11)  | NO   | PRI | NULL    | auto_increment |
| Name        | char(35) | NO   |     |         |                |
| CountryCode | char(3)  | NO   |     |         |                |
| District    | char(20) | NO   |     |         |                |
| Population  | int(11)  | NO   |     | 0       |                |
+-------------+----------+------+-----+---------+----------------+
```

Chapter 32. Data Backup and Recovery Methods

Almost all examples and exercises in this study guide use the *world database* as the sample data set. The accompanying CD-ROM contains the data for this database and instructions that describe how to create and populate the database for use with your own MySQL installation.

Question 1:

Name the principles of database backups in MySQL.

Question 2:

Why would you prefer text backups over binary backups when the latter are faster?

Question 3:

Assume that you want to make a binary copy of three `MyISAM` tables in the `world` database. Physically, that database is located under `/var/lib/mysql/`, and you want to back up the tables `City`, `Country`, and `CountryLanguage` to `/tmp`. What do you have to do?

Question 4:

How do you back up the tables `City` and `Country` from the `world` database into the file `/tmp/world_backup.sql` using a single `mysqldump` command?

Question 5:

How do you back up the databases `db1` and `db2` into the file `/tmp/db1_db2_backup.sql` with a single `mysqldump` command that ensures existing tables will be overwritten when you restore them?

Question 6:

What are the advantages and disadvantages of the multiple-row `INSERT` statements that `mysqldump` can produce?

Question 7:

Assume that you have the following tables in the `project` database:

```
mysql> SHOW TABLES;
+------------------+
| Tables_in_project |
+------------------+
| auth             |
| lang             |
| project          |
+------------------+
3 rows in set (0.00 sec)
```

You have backed up all tables in that database as follows:

```
shell> mysqldump project --tab=/tmp
```

Someone accidentally dropped the `project` database. Using `mysql` (and no other program), what

statements do you have to issue to restore the database and its tables?

Question 8:

Which backup programs and backup strategies require the server to be running, and which do not?

Question 9:

What commands do you have to issue to perform the following backup operations using `mysqldump`?

a. Back up the `City`, `Country`, and `CountryLanguage` tables in the `world` database into a file named `3tables.sql`.

b. Back up the `test` and `world` databases into a file named `2dbs.sql`.

c. Back up all databases into a file `alldbs.sql`.

Question 10:

When backing up `InnoDB` tables using `mysqldump`, what option is recommended to use?

Question 11:

When backing up data via a master's replication slave, what should you do on the slave side?

Question 12:

With MySQL Administrator, can you reload SQL dump files created by `mysqldump`? Can you use `mysqldump` to reload SQL dump files created by MySQL Administrator?

Question 13:

Assume that you've flushed the logs and made a full backup of your databases so that `bin.000003` and `bin.000004` contain all data-modifying statements performed after the full backup. However, you know that `bin.000004` contains a series of `DROP DATABASE` statements issued by a malicious user, starting on May 20, 2005, at a time of 20:00:00. After having reloaded your databases from the full backup, how can you replay the statements in the binary log files, but only until the point in time where the malicious user intervened?

Question 14:

Can you make a binary copy of the `InnoDB` tables stored on a machine while the server is running?

Question 15:

What methods can you use to copy `InnoDB` tables besides making a binary copy?

Question 16:

When the MySQL server or the server host crashes, what can you do to recover your `InnoDB` tables, other than restoring them from backups?

Question 17:

What are the prerequisites for using `mysqlhotcopy`?

Question 18:

You want to produce a backup of the `test` database into the file `/backups/structure.sql` using

`mysqldump`. However, the purpose of the backup file is to create an empty copy of each table on another server. What command do you issue?

Question 19:

How do you restore a backup of tables in the `test` database from a backup file called `/backups/back.sql`? When you use `mysql` to reload an SQL-format dump file produced by `mysqldump`, what option can you use to ensure that the restore operation doesn't stop if errors occur?

Question 20:

To copy a `MyISAM` table to another database server on another host, you can either copy the table files directly, or you can use `mysqldump` to dump the table contents into a file and then import that file into the other server. Suppose that a table named `mytable` exists in the `test` database of server 1, but not server 2. You want to copy the table from server 1 to server 2 without stopping either server for the copy operation. How might you do that?

Question 21:

Which backup programs and backup strategies work when used on the local host? Which can be used from a remote host?

Question 22:

Assume that you want to back up all tables in the `project` database using `mysqldump` on the local host. You do not want to dump the structure for each table, just its data. The data for the tables should be stored in text files in the `/tmp` directory, with columns separated by tabs and records separated by `\r\n`. What command would you issue?

Question 23:

Assume that you've issued the following command to back up all table data in the `project` database:

```
shell> mysqldump --no-create-info --tab=/tmp
           --lines-terminated-by="\r\n" project
```

How would you restore the tables?

Question 24:

The `test` database contains tables named `tbl1` and `tbl2`. Assume that you want to back up the data contained in the tables. The data should be written into text files in the `/backup` directory. In the text files, you want columns to be separated by commas and lines to be separated by DOS line endings (`\r\n`). What command do you issue? What will be the names of the resulting output files?

Question 25:

When you perform backups, it's not sufficient to back up only your tables. What else should you back up regularly?

Question 26:

`InnoDB` tablespace files are stored in a format that is machine independent and can be copied from one machine to another as long as certain conditions are met for both machines. Which of the following conditions must be true for the `InnoDB` tablespace to be machine independent?

a. Both machines must run under the same operating system.

b. You should create databases and tables using lowercase names.

c. The operating systems on both machines must use the same line-ending sequence. This is why you can't copy an `InnoDB` tablespace from a Windows machine (where lines end with `\r\n`) to a Mac OS X machine (where lines end with `\r`).

d. Both machines must use two's-complement integer arithmetic.

e. Both machines must have processors from the same family, such as Intel 586.

f. Both machines must use IEEE floating-point format, or else none of the `InnoDB` tables in the tablespace must contain any floating-point columns (`FLOAT` or `DOUBLE`).

Question 27:

Provided that the conditions for `InnoDB` binary portability are met, you can make a binary copy of the `InnoDB` tables stored on one machine and copy them to another machine. To do so, what files do you need to copy?

Answers to Exercises

Answer 1:

The principles of database backups in MySQL are:

- Backups should be made regularly.
- The binary log should be enabled to get incremental backups.
- Before a full backup, the logs should be flushed, to synchronize full backups with the incremental backup mechanism.
- Store backup files on a physical device separate from your data directory.
- Include storing your database backups in your regular filesystem backups.

Answer 2:

Text backups have a number of advantages over binary backups:

- They are more portable across different database management systems.
- They are portable across machine architectures.
- They are independent of the storage engines used.
- They don't require that you manually lock tables, or even shut down the server.
- They can be performed on remote servers.

Answer 3:

First, lock and flush the tables:

```
mysql> USE world;
mysql> LOCK TABLES City READ, Country READ, CountryLanguage READ;
mysql> FLUSH TABLES City, Country, CountryLanguage;
```

Then copy the tables:

```
shell> cd /var/lib/mysql/world
shell> cp City.* Country.* CountryLanguage.* /tmp
```

Unlock the tables:

```
mysql> UNLOCK TABLES;
```

Alternatively, stop the server, copy the files, and restart the server.

Answer 4:

```
shell> mysqldump world City Country > /tmp/world_backup.sql
```

Answer 5:

```
shell> mysqldump --add-drop-table --databases
           db1 db2 > /tmp/db1_db2_backup.sql
```

The `--add-drop-table` option causes the dump file to contain `DROP TABLE` statements, so that each table is dropped (if it exists) before it is restored. It is actually unnecessary to specify this option explicitly unless you also specify `--skip-opt`. By default, `--opt` is enabled, which also causes `--add-drop-table` to be enabled.

Answer 6:

Advantages: Multiple-row statements can be reloaded more efficiently.

Disadvantages: Multiple-row statements are less readable and less easily ported to other database management systems. For large tables, a multiple-row insert statement may fail because it exceeds the maximum size of the communication buffer.

Answer 7:

First, re-create the `project` database and make it the default database:

```
mysql> CREATE DATABASE project;
mysql> USE project;
```

Next, re-create the tables using the `.sql` files stored in `/tmp`:

```
mysql> SOURCE /tmp/auth.sql;
Query OK, 0 rows affected (0.00 sec)
mysql> SOURCE /tmp/lang.sql;
Query OK, 0 rows affected (0.00 sec)
mysql> SOURCE /tmp/project.sql;
Query OK, 0 rows affected (0.00 sec)
mysql> SHOW TABLES;
+-------------------+
| Tables_in_project |
+-------------------+
| auth              |
| lang              |
| project           |
+-------------------+
```

```
3 rows in set (0.00 sec)
```

Finally, load the data stored in the `.txt` files in `/tmp`:

```
mysql> LOAD DATA INFILE '/tmp/auth.txt' INTO TABLE auth;
Query OK, 1 row affected (0.02 sec)
Records: 1  Deleted: 0  Skipped: 0  Warnings: 0
mysql> LOAD DATA INFILE '/tmp/lang.txt' INTO TABLE lang;
Query OK, 9 rows affected (0.00 sec)
Records: 9  Deleted: 0  Skipped: 0  Warnings: 0
mysql> LOAD DATA INFILE '/tmp/project.txt' INTO TABLE project;
Query OK, 6 rows affected (0.00 sec)
Records: 6  Deleted: 0  Skipped: 0  Warnings: 0
```

By default, `mysqldump` uses the same column and line formatting options as the `LOAD DATA INFILE` statement, so you don't have to specify those explicitly when reloading the tables.

Answer 8:

- To back up data, you can use `mysql` to issue `SELECT ... INTO OUTFILE` statements. `mysql` requires the server to be running.
- You can also use `mysqldump` to back up data. The server must be running.
- Another way to perform data backup is to use the Perl script `mysqlhotcopy`. The server must be running for `mysqlhotcopy` to work. In addition, the script requires the Perl DBI module to be installed.
- InnoDB Hot Backup (`ibbackup`) performs backups by connecting to a running server, too.
- Performing backups using MySQL Administrator also requires the server to be running.
- If the server is *not* running, the only way to back up a `MyISAM` table is to copy its `.frm`, `.MYD`, and `.MYI` files directly.

Answer 9:

a. `mysqldump world City Country CountryLanguage > 3tables.sql`

b. `mysqldump --databases test world > 2dbs.sql`

c. `mysqldump --all-databases > alldbs.sql`

Answer 10:

For `InnoDB` tables, you should use the `--single-transaction` option to `mysqldump` so that the dump performs a consistent read.

Answer 11:

You should either shut down the slave server, or make the slave stop processing updates received from the master. The latter is done by issuing a `STOP SLAVE SQL_THREAD` statement. You should also make sure tables are flushed to disk by issuing `FLUSH TABLES`. After performing the backup, you should remember to reactivate the slave by issuing a `START SLAVE SQL_THREAD` statement.

Answer 12:

With MySQL Administrator, you can reload SQL dumps created by `mysqldump`, but you must take special care to ensure that the proper character set is selected before restoring. (In most cases, MySQL Administrator can detect the character set used when you click its `Auto-Detect Character Set` button.)

It's not possible to reload backup SQL files created by MySQL Administrator with `mysqldump`, because that program is for creating dumps, not reloading them. However, you should be able to reload backup files created by MySQL Administrator by using the `mysql` program.

Answer 13:

To do this, use the `mysqlbinlog` program with the appropriate options and arguments and pipe its output to the `mysql` program, like this:

```
shell> mysqlbinlog --stop-datetime="2005-05-20 19:59:59"
           bin.000003 bin.000004 | mysql
```

Answer 14:

You cannot make a binary copy of `InnoDB` tables while the server is running. Unlike the options available when using `MyISAM` tables (lock the tables, flush them to disk, and then copy them), you must tell the server to stop before making a copy of `InnoDB` files. This is necessary to ensure that `InnoDB` has completed any pending transactions before the copy is made.

Answer 15:

You can also copy `InnoDB` tables with the following programs and SQL statements:

- `mysqldump`,
- InnoDB Hot Backup (`ibbackup`),
- MySQL Administrator
- `SELECT … INTO OUTFILE`.

Answer 16:

Normally, you don't have to do anything after a crash except restart the server. `InnoDB` will recognize that it was not shut down correctly and perform an automatic recovery.

Answer 17:

- `mysqlhotcopy` runs on Unix and NetWare only.
- It requires Perl, and the DBI module must be installed.
- It must be run on the server host because it does a binary copy.
- The server must be running because `mysqlhotcopy` uses the server to lock and flush the tables to be backed up.

Answer 18:

```
shell> mysqldump --no-data test > /backups/structure.sql
```

Answer 19:

To restore tables, you can use this command:

```
shell> mysql test < /backups/back.sql
```

To cause `mysql` to continue processing queries even if errors occur, run it with the `--force` option.

Answer 20:

Two methods for copying the `mytable` table to another server are shown here. Other methods are possible.

- To copy `mytable` directly, you have to make sure that the server (and any other program) doesn't access the table files, and that the table is entirely flushed to disk. This would be done by issuing these statements:

  ```
  mysql> LOCK TABLES mytable READ;
  mysql> FLUSH TABLES;
  ```

  Now, you can copy the table files (`mytable.frm`, `mytable.MYD`, and `mytable.MYI`) to the other server host. The location to copy them to would be the `test` directory in that server's data directory. Afterward, you have to unlock the table on the first server:

  ```
  mysql> UNLOCK TABLES;
  ```

- Using `mysqldump`, you would issue this command:

  ```
  shell> mysqldump test mytable > mytabledump.sql
  ```

  After that, you can copy `mytabledump.sql` to the other server host, and import it by issuing this statement:

  ```
  shell> mysql test < mytabledump.sql
  ```

  This assumes that `mytabledump.sql` is located in the directory from which you invoke `mysql`.

Answer 21:

- `mysql`: To back up data, you can run the `SELECT ... INTO OUTFILE` statement under `mysql`. The statement can be run from any client host that can connect to the server, so you don't necessarily have to connect from the local host. The backup files, however, will always be located on the server host. This means that you must have the `FILE` privilege to perform backup operations using `mysql`.
- `mysqldump`: To back up data, you can also run `mysqldump`, from either a local client or from a remote client. The first of the following commands performs a dump that connects to the local server. The second command connects to a remote server.

  ```
  shell> mysqldump test > dump_db_test.sql
  shell> mysqldump test -h remote_host -u user_name -p > dump_db_test.sql
  ```

In both cases, the backup file is created on the client host where `mysqldump` is invoked.

- `mysqlhotcopy`: To back up data, you can also run the Perl script `mysqlhotcopy`. To do so, you must ensure that the script is run on the same host as the server.
- MySQL Administrator works with either a local server or a remote server.

Answer 22:

To dump the table data to text files, you can use the `--tab` option to `mysqldump`. By default, this produces output files in which columns are separated by tabs and records are separated by `\n`. The tab separators match the requirements stated but the record separators do not, so a `--lines-terminated-by` option is needed. By default, `mysqldump` will also produce SQL statements that can be used to re-create table structures. With the `--tab` option, this information will be stored in `.sql` files. To prevent creation of these files, use the `--no-create-info` option. Thus, to dump only the table data in the format required, you would issue this command:

```
shell> mysqldump --no-create-info --tab=/tmp
           --lines-terminated-by="\r\n" project
```

Answer 23:

To restore tables that have been backed up with the `--tab` option to `mysqldump`, you can use `LOAD DATA INFILE`. Because the records are terminated by `\r\n`, it will be necessary to use a `LINES TERMINATED BY` option. `LOAD DATA INFILE` must be issued for every table that was backed up. For example, to restore a table `table_1`, you would issue this statement:

```
mysql> LOAD DATA INFILE '/tmp/table_1.txt' INTO TABLE table_1
    -> LINES TERMINATED BY '\r\n';
```

Answer 24:

To back up the data as stated, you would issue this command:

```
shell> mysqldump --tab=/backup --fields-terminated-by=,
           --lines-terminated-by="\r\n" test tbl1 tbl2
```

The resulting files will be named `tbl1.sql`, `tbl1.txt`, `tbl2.sql`, and `tbl2.txt`.

Answer 25:

You should back up the following files:

- Binary log files. These are often needed for recovery operations.
- Option files used by the server (`my.cnf`, `my.ini`, `.my.cnf`, and whatever other option files you are using).
- Replication-related files (`master.info`, `relay-log.info`).

Answer 26:

The conditions for a machine-independent `InnoDB` tablespace are similar to those the `MyISAM` storage

engine:

- Both machines must use two's-complement integer arithmetic.
- Both machines must use IEEE floating-point format, or else none of the `InnoDB` tables in the tablespace must contain any floating-point columns (`FLOAT` or `DOUBLE`).

In addition, you should create databases and tables using lowercase names.

The other conditions stated are not required for machine independence of the tablespace.

Answer 27:

To make a binary copy of the `InnoDB` tablespace, you need to copy:

- All tablespace files, including the files for the shared tablespace and `.ibd` files if you have configured `InnoDB` to use per-table tablespaces
- All `InnoDB` log files
- The `.frm` file for each of your `InnoDB` tables
- The `InnoDB` tablespace and log configuration information stored in your MySQL option file (that is, the settings for `innodb_data_home_dir`, `innodb_data_file_path`, and perhaps other `InnoDB` options)