Part 4. MySQL DBA II Exam

Table of Contents

Chapter 33. Using Stored Routines and Triggers for Administration

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:

One purpose of using stored procedures is to control access to sensitive tables. Why can't this always be done simply by granting the appropriate user privileges to those tables? How would you control access using stored procedures?

Question 2:

Suppose that you have a table where it should be possible to insert data only on business days between 8 a.m. and 11 a.m. Can this restriction be implemented with the MySQL user account system?

Question 3:

To control insertions into a table, which of the following measures would be appropriate?

a. A stored procedure that checks whether data values to be inserted are within a reasonable range and rejects the insertion attempt if they are not.

b. A `BEFORE` trigger that checks whether data values to be inserted are within a reasonable range and modifies them to be within range if they are not.

c. An `AFTER` trigger that checks whether data values to be inserted are within a reasonable range and rejects the insertion attempt if they are not.

Question 4:

Describe three ways that stored routines can improve overall performance.

Answers to Exercises

Answer 1:

In many cases, it's sufficient to grant the appropriate user privileges so that users can or cannot access particular tables. However, sometimes it's necessary for users to have access to information derived from the tables (for example, summary reports), but not be able to access the tables directly. This cannot be handled by the MySQL user privilege system.

In such cases, you would write a stored procedure that accesses the sensitive tables on behalf of the user who wants to generate derived information, without granting the user direct access to those tables. This can be done by using the `DEFINER` security characteristics of the routine, and by granting the `EX-ECUTE` privilege to those users who need to access the derived information.

Answer 2:

In MySQL, you cannot set up user accounts with privileges that grant access only at certain times. Instead, you would write a stored procedure that restricts write access to the table in question to the times you specify in that routine.

Answer 3:

To achieve more fine-grained control over data insertions, you could either use stored procedures or `BEFORE` triggers. `AFTER` triggers aren't suited well for this purpose because they are activated only after the rows have already been inserted.

Answer 4:

a. Network bandwidth may be a limited resource, in which case it might make sense to avoid sending anything but the final results of a complex operation over the network.

b. Network bandwidth and network latency may influence the time needed to send results from server to client (or even to send queries from client to server), so that you might want to reduce that traffic to its absolute minimum.

c. Clients such as mobile devices may not be equipped to do much computation so that it's desirable to move most of that work from the client application to the server side.

Chapter 34. User Management

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:

Having installed MySQL, you want to make sure that there are no MySQL accounts that could connect to the server without specifying a password. How can you do this?

Question 2:

Assume that you want to remove all users from your MySQL installation that either have no user name or that have a user name of `root`. How can you do that?

Question 3:

You want to set up a MySQL administrator account with all privileges. This administrator should be called `superuser`, and the password should be `s3creT`. `superuser` should only be able to connect from the local host, and should be allowed to create new MySQL users. How would you create this account and grant the privileges?

Question 4:

What `GRANT` statement would you issue to set up an account for user `steve`, who should only be able to manipulate data of tables in the `accounting` database? `steve` should be able to connect to the server from anywhere. The account password should be `some_password1`.

Question 5:

What `GRANT` statement would you issue to set up an account for user `pablo`, who should be able to do all kinds of operations on the tables in the `marketing` database and should also be able to grant permissions to do those operations to other MySQL users? `pablo` should be able to connect to the server from the local network where the IP numbers of all machines start with `192.168`. The account password should be `some_password2`.

Question 6:

What `GRANT` statement would you issue to set up an account for user `admin`, who should be able to administer the database server, including performing all operations on all its databases and tables? `admin` should not, however, be able to grant privileges to other accounts. `admin` should be able to connect to the server only from the local host. The account password should be `some_password3`.

Question 7:

Consider the following privilege settings for the accounts associated with a given MySQL username, where the `Select_priv` column indicates the setting for the global `SELECT` privilege:

```
mysql> SELECT
    ->  Host, User, Select_priv
    ->  FROM mysql.user
    ->  WHERE User = 'icke'
    -> ;
+--------------+------+-------------+
| Host         | User | Select_priv |
+--------------+------+-------------+
| 62.220.12.66 | icke | N           |
| 62.220.12.%  | icke | Y           |
| 62.220.%     | icke | N           |
+--------------+------+-------------+
```

The `Select_priv` column indicates that the `SELECT` privilege for the second entry has been granted on a global scope (`*.*`). Will user `icke` be able to select data from any table on the MySQL server when connecting from the following hosts:

a. `62.220.12.66`

b. `62.220.12.43`

c. `62.220.42.43`

d. `localhost`

Assume that the `icke` accounts are not granted privileges in any of the other grant tables.

Question 8:

Consider the following statements:

```
mysql> CREATE USER 'lennart';
mysql> GRANT CREATE ON not_yet_existent_db.* TO 'lennart';
```

Which of the following statements are true?

a. User `lennart` can connect to the server from anywhere.

b. User `lennart` does not have to specify a password when connecting to the server. This is insecure.

c. If the `not_yet_existent_db` database does not exist, the `GRANT CREATE` statement will result in an error.

d. If the `not_yet_existent_db` exists, user `lennart` may create tables in that database.

e. User `lennart` may create tables in that database, but he cannot populate them.

Question 9:

Which of the following statements are true?

a. To create a user, you can use either a `CREATE USER` statement or a `GRANT` statement.

b. To grant an existing user additional privileges, you can use either an `ALTER USER` or a `GRANT` statement.

c. To remove a user, you can use either a `DROP USER` or a `REVOKE` statement.

Question 10:

Consider the following privileges granted to a user:

```
mysql> SHOW GRANTS FOR 'someone';
+------------------------------------------------------------------------+
| Grants for someone@%                                                   |
+------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'someone'@'%'                                    |
| GRANT SELECT ON `test`.* TO 'someone'@'%'                              |
| GRANT INSERT ON `world`.`City` TO 'someone'@'%'                        |
| GRANT INSERT ON `world`.`Country` TO 'someone'@'%' WITH GRANT OPTION   |
+------------------------------------------------------------------------+
```

Show three ways of revoking all privileges from that user.

Question 11:

Assume that a new user has been created, but that user account hasn't been granted any privileges yet. What can that user do at this point?

a. Connect to the server.

b. Issue statements like `SHOW VARIABLES`.

c. Issue statements like `SHOW DATABASES`.

Question 12:

Which of the following statements about the effects of privilege changes for existing connections are true?

a. When a user is dropped, connections to the server by that user are terminated automatically.

b. Global privilege changes don't affect existing connections of a user; they take effect next time the user attempts to connect only.

c. Database privilege changes only take effect the next time a user attempts to connect.

d. Database privilege changes only take effect the next time a user issues `USE` *`database`*.

e. Table and column privileges take immediate effect.

Question 13:

How would you limit the maximum number of connections of user `someone@localhost` to 10 per hour, the maximum number of statements that user can issue to 1000 per hour, and the maximum number of statements that modify data to 100? Assume that user doesn't exist yet, and that you don't want to grant privileges at this point, except for the privilege to connect to the server.

Question 14:

`CREATE USER` has a simpler syntax than `GRANT`, but it has limitations. Which of the following statements about `CREATE USER` are true?

a. You cannot assign a password to a user.

b. You can only grant access from everywhere (`%`).

c. You cannot grant specific privileges to the user.

Question 15:

How would you change the password for user `someone@%`?

Question 16:

Assume that you set up an administrator for the MySQL server named `superuser`, and that this is the only account with the full set of privileges. In particular, this is the only account that can grant privileges to other accounts or shut down the server using `mysqladmin shutdown`. Unfortunately, you've forgotten the password for `superuser`. Assume that you can log on to the host where the MySQL server runs, and that you can do so as some administrative account (such as `Administrator` for Windows or `root` for Unix). What can you do to set up the MySQL `superuser` account with a new password, and what safety precautions would you take?

Answers to Exercises

Answer 1:

To determine which accounts, if any, can be used without specifying a password, use the following statement:

```
mysql> SELECT Host, User FROM mysql.user WHERE Password = '';
```

If any such accounts exist, you can delete them as follows:

```
mysql> DELETE FROM mysql.user WHERE Password = '';
mysql> FLUSH PRIVILEGES;
```

The `FLUSH PRIVILEGES` statement is necessary because `DELETE` does not cause the server to reread the grant tables into memory.

Answer 2:

First, you have to remove the unwanted users (if there are any) from the `mysql.user` and `mysql.db` tables:

```
mysql> DELETE FROM mysql.user WHERE User = '' OR User = 'root';

mysql> DELETE FROM mysql.db WHERE User = '' OR User = 'root';
```

Then, you have to reload the grant tables:

```
mysql> FLUSH PRIVILEGES;
```

Answer 3:

To create the `superuser` account, use `CREATE USER`:

```
mysql> CREATE USER 'superuser'@'localhost' IDENTIFIED BY 's3creT';
```

To grant privileges to the account, use `GRANT`:

```
mysql> GRANT
```

```
    ->       ALL PRIVILEGES ON *.*
    ->       TO 'superuser'@'localhost'
    ->       WITH GRANT OPTION
    -> ;
```

You can also create the account and grant privileges in one step with GRANT:

```
mysql> GRANT
    ->       ALL PRIVILEGES ON *.*
    ->       TO 'superuser'@'localhost'
    ->       IDENTIFIED BY 's3creT'
    ->       WITH GRANT OPTION
    -> ;
```

Answer 4:

This statement sets up an account for steve:

```
mysql> GRANT
    ->       SELECT, INSERT, UPDATE, DELETE
    ->       ON accounting.*
    ->       TO 'steve'@'%'
    ->       IDENTIFIED BY 'some_password1'
    -> ;
```

Answer 5:

This statement sets up an account for pablo:

```
mysql> GRANT
    ->       ALL PRIVILEGES
    ->       ON marketing.*
    ->       TO 'pablo'@'192.168.%'
    ->       IDENTIFIED BY 'some_password2'
    ->       WITH GRANT OPTION
    -> ;
```

Answer 6:

This statement sets up an account for admin:

```
mysql> GRANT
    ->       ALL PRIVILEGES
    ->       ON *.*
    ->       TO 'admin'@'localhost'
    ->       IDENTIFIED BY 'some_password3'
    -> ;
```

Answer 7:

a. 62.220.12.66 is the most specific entry that matches the host user icke is trying to connect from. Because the SELECT privilege for that entry is N, user icke cannot select from any table on the server.

b. The most specific entry that matches 62.220.12.43 is 62.220.12.%. Because the SELECT privilege for that entry is Y, user icke can select from any table on the server.

c. The most specific entry that matches `62.220.42.43` is `62.220.%`. Because the `SELECT` privilege for that entry is `N`, user `icke` cannot select from any table on the server.

d. There is no entry that would match `icke@localhost`. Therefore, user `icke` cannot even connect to the server from the local host.

Answer 8:

The privileges granted to user `lennart` can be viewed with a `GRANT PRIVILEGES` statement:

```
mysql> SHOW GRANTS FOR 'lennart';
+------------------------------------------------------------+
| Grants for lennart@%                                       |
+------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'lennart'@'%'                        |
| GRANT CREATE ON `not_yet_existent_db`.* TO 'lennart'@'%' |
+------------------------------------------------------------+
```

This shows that the user may connect to the server from anywhere (`%`) without specifying a password (otherwise, there would be an `IDENTIFIED BY` clause). This is insecure. The `CREATE` privilege allows `lennart` to create the `not_yet_existent_db` database if it doesn't exist yet. In other words: It's possible to grant a `CREATE` privilege on an object that doesn't exist. The `CREATE` privilege includes the privilege of creating tables, which means that `lennart` may create as many tables in that database as he wants. However, he cannot do anything else with those tables; for example, he cannot populate them because he lacks the `INSERT` privilege.

Answer 9:

The first statement is true; both `CREATE USER` and `GRANT` can create a new user. There's no `ALTER USER` statement; if you want to grant additional privileges, you should use `GRANT`. You can remove a user completely with a `DROP USER` statement. Using `REVOKE`, you might be able to revoke all of the user's privileges, but the user account will still remain, as this example illustrates:

```
mysql> SHOW GRANTS FOR 'lennart';
+-------------------------------------+
| Grants for lennart@%                |
+-------------------------------------+
| GRANT USAGE ON *.* TO 'lennart'@'%' |
+-------------------------------------+
1 row in set (0.00 sec)

mysql> REVOKE usage ON *.* FROM 'lennart';
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS FOR 'lennart';
+-------------------------------------+
| Grants for lennart@%                |
+-------------------------------------+
| GRANT USAGE ON *.* TO 'lennart'@'%' |
+-------------------------------------+
1 row in set (0.00 sec)

mysql> DROP USER 'lennart';
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS FOR 'lennart';
ERROR 1141 (42000): There is no such grant defined for
user 'lennart' on host '%'
```

Answer 10:

1. Remove every single privilege, one after one:

   ```
   mysql> REVOKE SELECT ON test.* FROM 'someone';
   mysql> REVOKE INSERT ON world.City FROM 'someone';
   mysql> REVOKE INSERT, GRANT OPTION ON world.Country FROM 'someone';
   ```

   However, this still leaves the capability to connect to the server:

   ```
   mysql> SHOW GRANTS FOR 'someone';
   +--------------------------------------+
   | Grants for someone@%                 |
   +--------------------------------------+
   | GRANT USAGE ON *.* TO 'someone'@'%' |
   +--------------------------------------+
   ```

2. Remove all privileges with one statement:

   ```
   mysql> REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'someone';
   ```

   However, this still leaves the capability to connect to the server:

   ```
   mysql> SHOW GRANTS FOR 'someone';
   +--------------------------------------+
   | Grants for someone@%                 |
   +--------------------------------------+
   | GRANT USAGE ON *.* TO 'someone'@'%' |
   +--------------------------------------+
   ```

3. Remove the user account, revoking all privileges while doing so:

   ```
   mysql> DROP USER 'someone';
   ```

   Because the user account is gone, there's no way for that user to even connect to the server any more:

   ```
   mysql> SHOW GRANTS FOR 'someone';
   ERROR 1141 (42000): There is no such grant defined for
   user 'someone' on host '%'
   ```

Answer 11:

The user account can be used to connect to and ‘look around’ on the server. Like any other user, that user gets the full output from statements that display information about the server, like `SHOW VARIABLES` or `SHOW STATUS`. The user cannot get information about database object because the account doesn't have any privileges on them. `SHOW DATABASES` will display the `INFORMATION_SCHEMA` database only (which is a database that doesn't have a physical representation in the file system.)

Answer 12:

a. False. When a user is dropped, connections to the server by that user are *not* terminated automatically.

b. True. Global privilege changes don't affect existing connections of a user; they take effect next time the user attempts to connect only.

c. False; see next item.

d. True. Database privilege changes only take effect the next time a user issues `USE database`.

e. True. Table and column privileges take immediate effect.

Answer 13:

```
mysql> GRANT USAGE ON *.* TO 'someone'@'localhost' WITH
    ->  MAX_CONNECTIONS_PER_HOUR 10
    ->  MAX_QUERIES_PER_HOUR 1000
    ->  MAX_UPDATES_PER_HOUR 100;

mysql> SHOW GRANTS FOR 'someone'@'localhost'\G
*************************** 1. row ***************************
Grants for someone@localhost: GRANT USAGE ON *.* TO
'someone'@'localhost' WITH MAX_QUERIES_PER_HOUR 1000
MAX_UPDATES_PER_HOUR 100 MAX_CONNECTIONS_PER_HOUR 10
```

Answer 14:

`CREATE USER` allows you to control access to the server just like this would be done with `GRANT`. You can assign a password and specify access for a particular host (not just for `%`). However, `CREATE USER` doesn't allow you to grant specific privileges to a user:

```
mysql> CREATE USER 'lennart'@'localhost' IDENTIFIED BY 'sEcR3t';

mysql> SHOW GRANTS FOR 'lennart'@'localhost'\G
*************************** 1. row ***************************
Grants for lennart@localhost: GRANT USAGE ON *.* TO
'lennart'@'localhost' IDENTIFIED BY PASSWORD
'*6C22407CDD2BA32BEF5B2D0E10CBBAAD6B35C2AC'
```

Answer 15:

There are two ways to change a password for an existing user:

1.
```
mysql> SET PASSWORD FOR 'someone'@'%' = PASSWORD('miteeSekure');
```

2.
```
mysql> GRANT USAGE ON *.* TO 'someone'@'%' IDENTIFIED BY 'miteeSekure';
```

Answer 16:

To set up a new password for `superuser`, you could use the following procedure:

1. Bring down the MySQL server by means of the operating system. If you run the server as a Win-

dows service, you can stop the service. On Unix, you might have to forcibly terminate the server by using the `kill` command.

2. Restart the server in a way that it will not read the grant tables. As a safety precaution, make sure that no clients can connect to it other than from the local host:

   ```
   shell> mysqld --skip-grant-tables --skip-networking
   ```

3. Connect to the server from the local host:

   ```
   shell> mysql
   ```

   No username is needed here because the server is not using the grant tables.

4. Update the `Password` column in the `mysql.user` table entry for the `superuser` account, and then end the `mysql` session:

   ```
   mysql> UPDATE mysql.user
       ->  SET Password = PASSWORD('NeverForget')
       ->  WHERE User = 'superuser'
       -> ;
   mysql> EXIT;
   ```

5. Shut down the server normally:

   ```
   shell> mysqladmin shutdown
   ```

   The `UPDATE` statement in the previous step does not cause the server to refresh the in-memory grant tables, so no password is needed here.

6. Start the server using your normal startup procedure.

7. If you had to forcibly terminate the server, it would be a good idea to check all tables:

   ```
   shell> mysqlcheck -u root -p --all-databases
   ```

Chapter 35. Securing the MySQL Installation

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 components of MySQL must you protect on the filesystem level?

Question 2:

What security-related arguments speak in favor of running MySQL on a dedicated machine?

Question 3:

Why wouldn't you want the MySQL server to run as the `root` system user?

Question 4:

In which respects may log files increase security?

Question 5:

What's the procedure for securing the initial MySQL user accounts that modifies the grant tables directly?

Question 6:

How can you make sure that no one uses `GRANT` to create a MySQL user without a password?

Question 7:

To disable TCP/IP connections to the MySQL server, what would you do? When TCP/IP is disabled, what interfaces are left to connect to the server?

Question 8:

For a MySQL server running on Windows, is it possible to disable TCP/IP connections without enabling named pipes or shared memory? Or does this mean that the server will become inaccessible to clients?

Question 9:

How do you enable encryption of the traffic between MySQL Cluster nodes?

Question 10:

Assume that you have a table `fedTable` in the `world` database that stores these credentials:

```
mysql://wuser:wpass@world.example.com/world/City
```

How could a user access that information?

Question 11:

Assume that you have three users who have login accounts on a host where a MySQL server is running. Users `pablo` and `charlton` need to communicate with the MySQL server, but user `steve` doesn't. How would you set the file permissions for the `/usr/local/mysql/data` directory so that `pablo` and `charlton` can access their databases located there but `steve` cannot?

Answers to Exercises

Answer 1:

On the filesystem level, you must protect the following:

- Databases and their tables, so that unauthorized users cannot access them directly.
- Log files and status files, so that unauthorized users cannot access them directly.
- Configuration files, so that unauthorized users cannot replace or modify them.
- Programs and scripts that manage and access databases, so that users cannot replace or modify them.

Answer 2:

- Minimizing the number of user accounts minimizes the risk that these accounts are used against the MySQL installation. On a dedicated machine, you only need user accounts for administering the MySQL installation.
- Minimizing the number of tasks that the server host is used for reduces the complexity of the system, making it easier to secure the host.
- You can reduce the number of open ports on the host to a minimum.

Answer 3:

A server that has the privileges of the `root` login account has more filesystem access than necessary and constitutes a security risk. There are operations performed by the server that involve reading or writing files in the server host filesystem. Running the server as `root` is a bad idea because doing so gives it `root` privileges and vastly increases the extent of the filesystem that the server can access or modify.

Answer 4:

- The *binary log* is needed for recovery operations to restore data in case the server was compromised by attackers modifying or deleting data.
- The *general query log* contains information about connecting clients that might help uncover malicious activity.

Answer 5:

You could issue the following statements:

```
shell> mysql -u root mysql
mysql> UPDATE user SET Password = PASSWORD('rootpass')
    -> WHERE User = 'root';
mysql> DELETE FROM user
    -> WHERE User = 'root' AND Host = '%';
```

```
mysql> DELETE FROM user WHERE User = '';
mysql> DELETE FROM db WHERE User = '';
mysql> FLUSH PRIVILEGES;
```

To make sure that there are no anonymous users or users without passwords left, issue this statement:

```
mysql> SELECT * FROM user
    -> WHERE User = '' OR Password = '';
Empty set (0.00 sec)
```

Answer 6:

By running the server with the `NO_AUTO_CREATE_USER` SQL mode enabled:

```
mysql> SET GLOBAL sql_mode = 'NO_AUTO_CREATE_USER';

mysql> SELECT @@global.sql_mode;
+---------------------+
| @@global.sql_mode   |
+---------------------+
| NO_AUTO_CREATE_USER |
+---------------------+
```

Answer 7:

To disable TCP/IP connection, start the server with the `--skip-networking` option. When TCP/IP connections are disabled, you can connect to the server using Unix domain sockets (on Unix-like systems only), or named pipes or shared memory (on Windows systems only).

Answer 8:

If you disable TCP/IP and don't enable any of the local interfaces available under Windows (either named pipes or shared memory), the server will be inaccessible to clients. However, you can restrict TCP/IP connections to the loopback interface, which effectively means that the server cannot be accessed from remote machines. This is done by starting the server with this entry in its option file:

```
[mysqld]
bind-address=127.0.0.1
```

Answer 9:

Traffic between MySQL Cluster nodes is unencrypted because of performance reasons. There's no built-in mechanism to enable it. This means that a MySQL Cluster should always run in a trusted environment, and behind a firewall.

Answer 10:

- By issuing `SHOW CREATE TABLE world.fedTable`.
- By selecting `world` as the default database, and then issuing `SHOW TABLE STATUS LIKE 'fedTable'`.
- By issuing this statement:

  ```
  mysql> SELECT TABLE_COMMENT FROM INFORMATION_SCHEMA.TABLES
      -> WHERE TABLE_SCHEMA = 'world' AND TABLE_NAME = 'fedTable';
  +----------------------------------------------------+
  ```

```
| TABLE_COMMENT                                    |
+--------------------------------------------------+
| mysql://wuser:wpass@world.example.com/world/City |
+--------------------------------------------------+
```

- By viewing the contents of the `fedTable.frm` file:

```
shell> cat fedTable.frm
...
? ?   0mysql://wuser:wpass@world.example.com/world/City
...
```

Answer 11:

Neither `pablo` nor `charlton` needs file system-level access to their database directories. If they want to access their databases, they should do this through the MySQL server (for example, by using the `mysql` client program). Therefore, the `/usr/local/mysql/data` directory should be accessible only to user `mysql`, assuming that this is the system user the server runs as.

Chapter 36. Upgrade-Related Security Issues

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 statements are true?

The `mysql_fix_privilege_tables` script does this:

a. It changes the current set of privileges of existing users to make the MySQL installation more secure.

b. It adds additional privileges to the grant tables in the `mysql` database.

c. It adds additional privileges to the grant tables in the `mysql` database. It also removes outdated privileges so you have to make sure your applications or users don't rely on those outdated privileges.

d. It determines whether it's run under Windows, and if so, it reduces the privileges of all users to a minimum, to enforce security.

e. It overwrites the existing grant tables in the `mysql` database with newer versions that contain more privileges than the old ones. You will have to recreate the users.

Question 2:

Which of the following statements are true?

a. When you upgrade to a MySQL version that contains the `TRADITIONAL` SQL mode, that mode is activated by default.

b. If you're running applications that rely on the old, “forgiving” behavior of MySQL (for example, silent data conversions), you need to activate the `TRADITIONAL` SQL mode to ensure backward compatibility.

c. If you're running applications that rely on the old, “forgiving” behavior of MySQL (for example, silent data conversions), you shouldn't activate the `TRADITIONAL` SQL mode (which is disabled by default).

Question 3:

What's the effect of turning on the `NO_AUTO_CREATE_USER` SQL mode?

a. The `GRANT` statement is disabled.

b. The `GRANT` statement only works if you specify an `IDENTIFIED BY` clause.

c. You cannot create new users with the `GRANT` command.

d. You cannot create new users with the `GRANT` command unless you specify an `IDENTIFIED BY` clause.

Answers to Exercises

Answer 1:

It adds additional privileges to the grant tables in the `mysql` database.

Answer 2:

If you're running applications that rely on the old, “forgiving” behavior of MySQL (for example, silent data conversions), you shouldn't activate the `TRADITIONAL` SQL mode (which is disabled by default).

Answer 3:

You cannot create new users with the `GRANT` command unless you specify an `IDENTIFIED BY` clause.

Chapter 37. Optimizing Queries

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 three log files that can be helpful for getting an overall picture of the type of queries processed by your MySQL server.

Question 2:

Why would you want to have the `PROCESS` privilege when running `SHOW PROCESSLIST` statements?

Question 3:

What kind of statements can `EXPLAIN` be applied to?

a. Only to `SELECT` statements that use at least one table.

b. Only to `SELECT` statements that join at least two tables.

c. To all `SELECT` statements.

d. To all SQL statements.

Question 4:

Assume that you frequently retrieve data from the `City` table, using queries similar to those shown here:

```
mysql> SELECT * FROM City WHERE Name BETWEEN 'E' AND 'G'
    -> ORDER BY Name;
+------+------------------+-------------+--------------+------------+
| ID   | Name             | CountryCode | District     | Population |
+------+------------------+-------------+--------------+------------+
|  735 | East London      | ZAF         | Eastern Cape |     221047 |
| 3963 | East Los Angeles | USA         | California   |     126379 |
| 1845 | East York        | CAN         | Ontario      |     114034 |
|  533 | Eastbourne       | GBR         | England      |      90000 |
| 1720 | Ebetsu           | JPN         | Hokkaido     |     118805 |
|  ... | ...              | ...         | ...          |        ... |

mysql> SELECT * FROM City WHERE CountryCode >= 'Y'
    -> ORDER BY name;
+------+------------+-------------+----------------+------------+
| ID   | Name       | CountryCode | District       | Population |
+------+------------+-------------+----------------+------------+
| 1781 | Aden       | YEM         | Aden           |     398300 |
| 1784 | al-Mukalla | YEM         | Hadramawt      |     122400 |
|  721 | Alberton   | ZAF         | Gauteng        |     410102 |
|  724 | Benoni     | ZAF         | Gauteng        |     365467 |
| 1792 | Beograd    | YUG         | Central Serbia |    1204000 |
|  ... | ...        | ...         | ...            |        ... |
```

How would you determine the number of rows MySQL estimates that it must inspect to calculate the result sets?

Question 5:

Consider the `City` and the `Country` table. The tables are related: `CountryCode` in `City` references `Code` in `Country`. What information does the following `EXPLAIN` statement give you regarding possible optimization of the query?

```
mysql> EXPLAIN
    -> SELECT
    ->  City.Name, City.Population, Country.Name
    ->  FROM City INNER JOIN Country
    ->  ON City.CountryCode = Country.Code
    ->  WHERE City.Population > 10000000
    ->  ORDER BY City.Population DESC
    -> \G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: City
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 4079
        Extra: Using where; Using filesort
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: Country
         type: eq_ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 3
          ref: City.CountryCode
         rows: 1
        Extra:
```

Question 6:

Based on the information provided by the `EXPLAIN` in the previous question, what would you do to optimize the query performance?

Question 7:

Consider once again the `EXPLAIN` output for the `Country` and `City` tables from the previous two questions. How would you roughly “measure” the performance for the unoptimized query? For the optimized query?

Question 8:

Which of the following statements is true?

a. `ANALYZE TABLE` should be run frequently for tables that change often.

b. `ANALYZE TABLE` can be run for any storage engine.

c. `ANALYZE TABLE` is used to reorder table indexes.

d. `ANALYZE TABLE` is used to update table statistics.

Question 9:

Most of the time, the MySQL optimizer makes the right choice of indexes to use for a query. However, you suspect that, for a certain `SELECT` statement, the optimizer is not making the right choice. How can you determine whether the optimizer is choosing the index you want it to use?

Question 10:

Most of the time, the MySQL optimizer makes the right choice of indexes to use for a query. However, you suspect that, for a certain `SELECT` statement, the optimizer is not making the right choice. How could you rewrite the statement to determine whether it runs faster without using an index?

Question 11:

Most of the time, the MySQL optimizer makes the right choice of indexes to use for a query. However, you suspect that, for a certain query, the optimizer is not making the right choice. How could you force MySQL to use an index that is different from the index which the optimizer would choose?

Question 12:

For which purpose can `SHOW WARNINGS` be used?

a. For identifying statements that the optimizer didn't optimize.

b. For identifying statements that might be unnecessary.

Question 13:

Suppose that you want to cache indexes for the three `world` database tables (`City`, `Country`, `CountryLanguage`) in a special key cache called `world_cache` that should have a size of 4MB. Furthermore, the table indexes should be preloaded into this cache, and this should happen every time the server starts.

What must you do to accomplish this?

Question 14:

Assume that you frequently retrieve data from the `City` table, using queries similar to those shown here:

```
mysql> SELECT * FROM City WHERE Name BETWEEN 'E' AND 'G'
    -> ORDER BY Name;
+------+-------------------+-------------+---------------+------------+
| ID   | Name              | CountryCode | District      | Population |
+------+-------------------+-------------+---------------+------------+
|  735 | East London       | ZAF         | Eastern Cape  |     221047 |
| 3963 | East Los Angeles  | USA         | California    |     126379 |
| 1845 | East York         | CAN         | Ontario       |     114034 |
|  533 | Eastbourne        | GBR         | England       |      90000 |
| 1720 | Ebetsu            | JPN         | Hokkaido      |     118805 |
|  ... | ...               | ...         | ...           |        ... |


mysql> SELECT * FROM City WHERE CountryCode >= 'Y'
    -> ORDER BY name;
+------+------------+------------+---------------+-----------+
| ID   | Name       | CountryCode | District       | Population |
+------+------------+-------------+----------------+------------+
| 1781 | Aden       | YEM         | Aden           |     398300 |
| 1784 | al-Mukalla | YEM         | Hadramawt      |     122400 |
|  721 | Alberton   | ZAF         | Gauteng        |     410102 |
|  724 | Benoni     | ZAF         | Gauteng        |     365467 |
| 1792 | Beograd    | YUG         | Central Serbia |    1204000 |
|  ... | ...        | ...         | ...            |        ... |
```

What index or indexes would you add to the table to speed up the queries?

Question 15:

Assume that you frequently retrieve data from the `City` table, using queries similar to those shown here:

```
mysql> SELECT * FROM City WHERE Name BETWEEN 'E' AND 'G'
    -> ORDER BY Name;
+------+------------------+-------------+--------------+------------+
| ID   | Name             | CountryCode | District     | Population |
+------+------------------+-------------+--------------+------------+
|  735 | East London      | ZAF         | Eastern Cape |     221047 |
| 3963 | East Los Angeles | USA         | California   |     126379 |
| 1845 | East York        | CAN         | Ontario      |     114034 |
|  533 | Eastbourne       | GBR         | England      |      90000 |
| 1720 | Ebetsu           | JPN         | Hokkaido     |     118805 |
|  ... | ...              | ...         | ...          |        ... |
```

```
mysql> SELECT * FROM City WHERE CountryCode >= 'Y'
    -> ORDER BY name;
+------+------------+-------------+----------------+------------+
| ID   | Name       | CountryCode | District       | Population |
+------+------------+-------------+----------------+------------+
| 1781 | Aden       | YEM         | Aden           |     398300 |
| 1784 | al-Mukalla | YEM         | Hadramawt      |     122400 |
|  721 | Alberton   | ZAF         | Gauteng        |     410102 |
|  724 | Benoni     | ZAF         | Gauteng        |     365467 |
| 1792 | Beograd    | YUG         | Central Serbia |    1204000 |
|  ... | ...        | ...         | ...            |        ... |
```

Further assume you've added indexes to the `City` table so that it now looks like this:

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

In addition to adding those indexes to the `City` table, what else can be done, with regard to the table's columns, to improve performance?

Question 16:

Consider once again the new table structure and the sample queries shown for the `City` table in the previous question. How would you find out whether the new indexes on the table are actually used to resolve the queries?

Answers to Exercises

Answer 1:

General query log, binary log, and slow query log.

Answer 2:

If you have the PROCESS privilege, SHOW PROCESSLIST will display queries being run by all clients. Otherwise, you will see only your own queries.

Answer 3:

EXPLAIN can be applied to all SELECT statements.

Answer 4:

You can use EXPLAIN to determine the number of rows MySQL estimates that it must inspect to calculate the result sets:

```
mysql> EXPLAIN SELECT * FROM City WHERE Name BETWEEN 'E' AND 'G'
    -> ORDER BY Name\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: City
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 4079
        Extra: Using where; Using filesort
mysql> EXPLAIN SELECT * FROM City WHERE CountryCode >= 'Y'
    -> ORDER BY Name\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: City
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 4079
        Extra: Using where; Using filesort
```

The EXPLAIN output shows that MySQL would not use indexes to process the queries. All rows (4,079) must be scanned to calculate the results. This is indicated by the ALL value in the type column as well.

Answer 5:

EXPLAIN provides the following information:

- For table City, EXPLAIN indicates that all table rows must be scanned to find the desired information (ALL). There are no keys on the columns that should be retrieved, nor on the column mentioned in the ORDER BY clause, so no keys are used as indicated by the NULL entries for possible_keys, key, key_len, and ref. Therefore, all 4,079 table rows are scanned. Using filesort indicates that MySQL needs to do an extra pass to find out how to retrieve the rows in sorted order.

- For table Country, EXPLAIN shows a join type of eq_ref. This is the one of the best join types; it means that only one row is read from this table for each row from the previous table. This join type is possible because the index used for table Country is a primary key, as indicated by the PRIMARY entries for possible_keys and key. The primary key has the same length as the column itself (three characters, as indicated by key_len). ref shows which column is used with the key to select rows from the table: the CountryCode column of the City table. The rows value of 1 thus indicates that MySQL must examine one row of the Country table to find the match for each CountryCode value selected from the City table.

Answer 6:

To optimize the query shown by the EXPLAIN in the last question, you could create an index for the Population column of the City table because it is used both in the WHERE clause to determine which rows to retrieve and in the ORDER BY clause, to sort the result:

```
mysql> ALTER TABLE City
    ->  ADD INDEX (Population)
    -> ;
Query OK, 4079 rows affected (0.68 sec)
Records: 4079  Duplicates: 0  Warnings: 0
```

With the new index, EXPLAIN displays the following result for the same query:

```
mysql> EXPLAIN
    -> SELECT
    ->  City.Name, City.Population, Country.Name
    ->  FROM City INNER JOIN Country
    ->  ON City.CountryCode = Country.Code
    ->  WHERE City.Population > 10000000
    ->  ORDER BY City.Population DESC
    -> \G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: City
         type: range
possible_keys: Population
          key: Population
      key_len: 4
          ref: NULL
         rows: 9
        Extra: Using where
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: Country
         type: eq_ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 3
          ref: City.CountryCode
         rows: 1
        Extra:
```

The EXPLAIN output for the Country table is unchanged, but the output for the City table indicates a much improved search. It shows that only rows within a given range of `Population` values will be retrieved (`type`: `range`), using an index to select the rows. The possible key `Population` is actually used with its full key length (4). Due to the use of the new index, MySQL now has to inspect only nine rows to resolve the `WHERE` clause.

Answer 7:

As a rough measure of performance, take the product of the `rows` output of the `EXPLAIN` statements before and after the addition of the index: In the original unoptimized situation, the product of the `rows` values is 4,079 × 1 = 4,079. With the index added to optimize the query, the product is only 9 × 1 = 9. This lower value indicates that performance is better with the new index.

Answer 8:

`ANALYZE TABLE` should be run frequently for tables that change often, because the statistics for such tables will get out of date soon. Table statistics that are up to date help the optimizer make the right decisions. The `ANALYZE TABLE` statement can be run for `MyISAM` and `InnoDB` tables only.

Answer 9:

To find out which indexes the optimizer will use, prefix your `SELECT` statement with `EXPLAIN`. For example:

```
EXPLAIN SELECT Name FROM City;
```

Answer 10:

To rewrite a query that forces MySQL not to use a specific index that the optimizer would otherwise choose, you would use the `IGNORE INDEX` (or `IGNORE KEY`) option. For example:

```
SELECT Name FROM City IGNORE INDEX (idx_name);
```

Answer 11:

To force the optimizer to use a specific index, you would use the `FORCE INDEX` (or `FORCE KEY`) option. For example:

```
SELECT Name FROM City FORCE INDEX (idx_name);
```

Another option is `USE INDEX` (or `USE KEY`), but this provides only a hint whereas `FORCE INDEX` requires the index to be used.

Answer 12:

`SHOW WARNINGS` can be used to identify statements that might be unnecessary.

Answer 13:

You need to perform the following steps:

1. Create the `world_cache` key cache:

   ```
   mysql> SET GLOBAL world_cache.key_buffer_size = 4 * 1024 * 1024;
   ```

2. Assign the three tables to the cache:

```
mysql> CACHE INDEX City, Country, CountryLanguage IN world_cache;
+-----------------------+--------------------+----------+----------+
| Table                 | Op                 | Msg_type | Msg_text |
+-----------------------+--------------------+----------+----------+
| world.City            | assign_to_keycache | status   | OK       |
| world.Country         | assign_to_keycache | status   | OK       |
| world.CountryLanguage | assign_to_keycache | status   | OK       |
+-----------------------+--------------------+----------+----------+
```

3. Preload the table indexes of all three tables:

```
mysql> LOAD INDEX INTO CACHE City, Country, CountryLanguage;
+-----------------------+--------------+----------+----------+
| Table                 | Op           | Msg_type | Msg_text |
+-----------------------+--------------+----------+----------+
| world.City            | preload_keys | status   | OK       |
| world.Country         | preload_keys | status   | OK       |
| world.CountryLanguage | preload_keys | status   | OK       |
+-----------------------+--------------+----------+----------+
```

4. Put the preceding statements in an initialization file (we'll call it `init_cache.sql`):

```
SET GLOBAL world_cache.key_buffer_size = 4 * 1024 * 1024;
CACHE INDEX City, Country, CountryLanguage IN world_cache;
LOAD INDEX INTO CACHE City, Country, CountryLanguage;
```

5. Instruct the server to execute the statements in that file when it starts (you'll need to add a path to the filename if you don't place it in the data directory):

```
shell> mysqld --init-file=init_cache.sql
```

   Alternatively, the `--init-file` option could be placed in an option file that the server reads at startup.

Answer 14:

To improve performance, indexes should be added to the `Name` and `CountryCode` columns because those are the columns used in the comparisons that determine which rows to return. Also, because `Name` is used in the `ORDER BY` clause, an index on `Name` can speed up sorting operations.

For the `Name` column, the results of the queries in question indicate that an index with a prefix length that is shorter than the full column length is likely to improve performance even more. However, the prefix length should be long enough to differentiate cities that begin with words like “East Lo...”, so we choose a prefix length of `10`:

```
mysql> ALTER TABLE City
    ->  ADD INDEX (Name(10)),
    ->  ADD INDEX (CountryCode)
    -> ;
```

Answer 15:

Another means of making table lookups faster is to declare the table's columns to be `NOT NULL`. Assume that `City` must contain a city name in each row, as well as a country code for each city. To disallow `NULL` values in the `Name` and `CountryCode` columns, you could alter the table with this SQL statement:

```
mysql> ALTER TABLE City
    ->  MODIFY Name CHAR(35) NOT NULL,
    ->  MODIFY CountryCode CHAR(3) NOT NULL
    -> ;
Query OK, 4079 rows affected (0.21 sec)
Records: 4079  Duplicates: 0  Warnings: 0

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

Answer 16:

To check whether MySQL actually uses the new indexes to resolve the queries, use `EXPLAIN` once again:

```
mysql> EXPLAIN SELECT * FROM City WHERE Name BETWEEN 'E' AND 'G'
    -> ORDER BY Name\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: City
         type: range
possible_keys: Name
          key: Name
      key_len: 5
          ref: NULL
         rows: 146
        Extra: Using where; Using filesort

mysql> EXPLAIN SELECT * FROM City WHERE CountryCode >= 'Y'
    -> ORDER BY Name\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: City
         type: range
possible_keys: CountryCode
          key: CountryCode
      key_len: 3
          ref: NULL
         rows: 76
        Extra: Using where; Using filesort
```

The `EXPLAIN` output shows that the indexes you would expect to be used actually are used by MySQL to resolve the queries. Compared to the previous results from `EXPLAIN` (three questions earlier), the number of rows inspected drops dramatically from 4,079 to 146 and 76 due to the use of indexes.

Chapter 38. Optimizing Databases

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 most common table optimizations that are independent of the storage engine used.

Question 2:

Assume that you've created a table `District` and populated it with data from the `City` table, like this:

```
mysql> CREATE TABLE District
    -> (DName CHAR(50), DPop INT)
    -> ENGINE = MyISAM;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO District
    -> SELECT District, SUM(Population) FROM City
    -> GROUP BY District;
Query OK, 1367 rows affected (0.05 sec)
Records: 1367  Duplicates: 0  Warnings: 0
```

How could you find out whether your column layout could be improved? What could you actually improve?

Question 3:

What are the basic goals of normalization?

Question 4:

Why would you specify the `MAX_ROWS` option for a table?

Question 5:

Can you specify that a `MyISAM` table should have a fixed row format, even if it contains variable-width columns like `VARCHAR`? If so, how would you do that?

Question 6:

On a Windows server host, you want to compress the `MyISAM` table `City`, found in the `world` database. Assuming that MySQL is installed in `C:\mysql` and that MySQL's data directory is in the default location, what commands would you issue to compress the table if your current location is the `C:\mysql\bin` directory?

Question 7:

You want to compress and then unpack the `MyISAM` table `City`, found in the `world` database. After you've unpacked the compressed table, how would you verify that the table actually has been unpacked?

Question 8:

Can you use both compressed and uncompressed `MyISAM` tables in one `MERGE` table?

Question 9:

What are the advantages of fixed-row and of dynamic-row tables? Can you gain the advantages of both, and if so, how would you do that?

Question 10:

Consider the following table, which has two single-column `FULLTEXT` indexes:

```
mysql> DESCRIBE faq;
+----------+-----------+------+-----+---------+-------+
| Field    | Type      | Null | Key | Default | Extra |
+----------+-----------+------+-----+---------+-------+
| cdate    | timestamp | YES  |     | NULL    |       |
| question | char(150) | NO   | MUL |         |       |
| answer   | char(250) | NO   | MUL |         |       |
+----------+-----------+------+-----+---------+-------+
mysql> SHOW INDEX FROM faq;
+-------+------------+----------+-    -+-------------+-    -+------------+-
| Table | Non_unique | Key_name | ...  | Column_name | ...  | Index_type | ...
+-------+------------+----------+-    -+-------------+-    -+------------+-
| faq   |          1 | question | ...  | question    | ...  | FULLTEXT   | ...
| faq   |          1 | answer   | ...  | answer      | ...  | FULLTEXT   | ...
+-------+------------+----------+-    -+-------------+-    -+------------+-
```

With `MATCH … AGAINST()`, you can search the answers and the questions stored in the table. How would you search for the search term `'Access'` in either the `question` or the `answer` column?

Question 11:

Suppose that `mix1` is a `MyISAM` table that has the following structure and indexes:

```
mysql> DESCRIBE mix1;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int(11)     | NO   | PRI | 0       |       |
| name  | varchar(20) | YES  |     | NULL    |       |
| story | text        | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
mysql> SHOW KEYS FROM mix1;
+-------+------------+----------+-
| Table | Non_unique | Key_name | ...
+-------+------------+----------+-
| mix1  |          0 | PRIMARY  | ...
+-------+------------+----------+-
```

Assume that you have many seeks on the `mix1` table, most of which use `id` or `name` as a search term. Searches are becoming considerably slow. What can you do to improve the situation?

Question 12:

Assume that you hit a filesystem limit on file size with a `MyISAM` table. That table contains a `FULL-TEXT` index, so you cannot switch to another storage engine. Also, assume that it isn't possible to change the filesystem you're using. What else could you do to overcome the filesystem size limit?

Question 13:

You can influence the maximum size of a `MyISAM` table by specifying `MAX_ROWS`. If your tables always need to be big, what else could you do, rather than specifying `MAX_ROWS` for each new (or altered) table? How could you find out about the maximum size of a `MyISAM` table when there's no `MAX_ROWS` clause specified?

Question 14:

How can you rebuild `InnoDB` tables? Why would you want to do that regularly? What's the positive side effect of rebuilding tables that were created in an older MySQL version?

Question 15:

How can you make sure that `MEMORY` tables don't use up huge amounts of memory?

Question 16:

Consider the `City` table which has rows like these:

```
mysql> SELECT * FROM City LIMIT 5;
+----+----------------+-------------+---------------+------------+
| ID | Name           | CountryCode | District      | Population |
+----+----------------+-------------+---------------+------------+
|  1 | Kabul          | AFG         | Kabol         |    1780000 |
|  2 | Qandahar       | AFG         | Qandahar      |     237500 |
|  3 | Herat          | AFG         | Herat         |     186800 |
|  4 | Mazar-e-Sharif | AFG         | Balkh         |     127800 |
|  5 | Amsterdam      | NLD         | Noord-Holland |     731200 |
+----+----------------+-------------+---------------+------------+
```

Its structure looks like this:

```
mysql> DESCRIBE City;
+-------------+----------+------+-----+---------+----------------+
| 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       |                |
+-------------+----------+------+-----+---------+----------------+
```

Explain why this table is in the third normal form (3NF).

Question 17:

If you don't specify `MAX_ROWS` for a `MyISAM` table, how large can a newly created table grow by default? What happens when you try to insert more data than the maximum size allows for?

Question 18:

Name two ways of finding out about the row format of the `City` table in the `world` database.

Question 19:

You want to compress the `MyISAM` table `City`, found in the `world` database. After you've done so, how would you verify that the table actually has been compressed?

Question 20:

You want to compress the `MyISAM` table `City`, found in the `world` database. What operations will you no longer be able to perform on `City` after the table has been compressed?

Question 21:

On a Windows server host, you want to compress the `MyISAM` table `City`, found in the `world` database. Assume that MySQL is installed in `C:\mysql`, that MySQL's data directory is in the default location, and that your current location is the `C:\mysql\bin` directory.

a. After you've compressed the table, how can you unpack it?

b. What alternatives to using `myisamchk --unpack` could you use to unpack the table after it has been compressed?

Question 22:

With `MyISAM` tables, what would you use `ANALYZE TABLE` and `OPTIMIZE TABLE` for?

Question 23:

Consider the following table, which has two single-column `FULLTEXT` indexes:

```
mysql> DESCRIBE faq;
+----------+-----------+------+-----+---------+-------+
| Field    | Type      | Null | Key | Default | Extra |
+----------+-----------+------+-----+---------+-------+
| cdate    | timestamp | YES  |     | NULL    |       |
| question | char(150) | NO   | MUL |         |       |
| answer   | char(250) | NO   | MUL |         |       |
+----------+-----------+------+-----+---------+-------+
mysql> SHOW INDEX FROM faq;
+-------+------------+----------+-    -+-------------+-    -+------------+-
| Table | Non_unique | Key_name | ...  | Column_name | ...  | Index_type | ...
+-------+------------+----------+-    -+-------------+-    -+------------+-
| faq   |          1 | question | ...  | question    | ...  | FULLTEXT   | ...
| faq   |          1 | answer   | ...  | answer      | ...  | FULLTEXT   | ...
+-------+------------+----------+-    -+-------------+-    -+------------+-
```

With `MATCH … AGAINST()`, you can search the answers and the questions stored in the table. How would you search for the search term `'MySQL'` in the `question` column?

Question 24:

Suppose that `mix1` is a `MyISAM` table that has the following structure and indexes:

```
mysql> DESCRIBE mix1;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int(11)     | NO   | PRI | 0       |       |
| name  | varchar(20) | YES  |     | NULL    |       |
| story | text        | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
mysql> SHOW KEYS FROM mix1;
+-------+------------+----------+-
| Table | Non_unique | Key_name | ...
+-------+------------+----------+-
```

```
| mix1  |          0 | PRIMARY  | ...
+-------+-----------+----------+-
```

Assume that you have many seeks on the `mix1` table, most of which look for a search term in the `story` column. What can you do to speed up those searches?

Question 25:

Name the most common table optimizations for `InnoDB` tables.

Answers to Exercises

Answer 1:

- Proper indexing makes lookups faster.
- Short columns require less storage, compare faster, and make for smaller indexes when the whole column is indexed.
- Columns used to join tables should be as short as possible and use the same data type.
- Columns declared `NOT NULL` are processed faster.

Answer 2:

By using `PROCEDURE ANALYSE()` and comparing its output with the output of `DESCRIBE TABLE`:

```
mysql> SELECT * FROM District PROCEDURE ANALYSE()\G
*************************** 1. row ***************************
             Field_name: world.District.DName
              Min_value:
              Max_value: #
             Min_length: 0
             Max_length: 20
       Empties_or_zeros: 1
                  Nulls: 0
Avg_value_or_avg_length: 8.7915
                    Std: NULL
      Optimal_fieldtype: VARCHAR(20) NOT NULL
*************************** 2. row ***************************
             Field_name: world.District.DPop
              Min_value: 167
              Max_value: 26316966
             Min_length: 3
             Max_length: 8
       Empties_or_zeros: 0
                  Nulls: 0
Avg_value_or_avg_length: 1045764.3628
                    Std: 2204524.0754
      Optimal_fieldtype: INT(8) UNSIGNED NOT NULL

mysql> DESCRIBE District;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| DName | char(50) | YES  |     | NULL    |       |
| DPop  | int(11)  | YES  |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
```

Comparison of the output from the two statements yields the following conclusions:

- Both columns should be declared `NOT NULL`.
- The `DName` column is much too large (50 characters wide, whereas 20 would be sufficient).
- The `DName` column should be `VARCHAR`, rather than `CHAR`, because its values differ in length (between 0 and 20).
- The `DPop` column should be declared `UNSIGNED`. It contains positive values only.

Answer 3:

- To remove redundant data.
- To enable more flexible access to data.
- To eliminate possibilities of anomalies when modifying data that would make the data become inconsistent.

Answer 4:

For a `MyISAM` table, specifying `MAX_ROWS` allows the server to make a better estimate of how large to make the table's internal row pointers. A large `MAX_ROWS` value is useful when you expect a table to become very large and to contain more rows than the default pointer size will allow. For a table that you know will remain small, a small `MAX_ROWS` value saves space because the row pointers can be smaller than the default size.

Answer 5:

In MySQL 5.0, you may specify the row format explicitly. (In prior versions, the row format was implicitly set by the column types used.) To specify a fixed row format, you create (or alter) a table with the `ROW_FORMAT = FIXED` clause. This works as long as the table does not contain `TEXT` or `BLOB` columns.

Answer 6:

The `City` table is located in the `world` directory under the MySQL data directory, `C:\mysql\data`. To refer to that directory from within the `C:\mysql\bin` directory, you can use either the absolute pathname `C:\mysql\data\world` or the relative pathname `..\data\world`. The commands shown here use the latter.

To compress the `City` table, stop the server, and then issue this command:

```
shell> myisampack ..\data\world\City.MYI
Compressing ..\data\world\City.MYD: (4079 records)
- Calculating statistics
- Compressing file
70.94%
Remember to run myisamchk -rq on compressed tables
```

As indicated by the last output line, you should now run `myisamchk` with the `--recover` (or `-r`) and `--quick` (or `-q`) options:

```
shell> myisamchk --recover --quick ..\data\world\City.MYI
```

```
- check key delete-chain
- check record delete-chain
- recovering (with sort) MyISAM-table '..\data\world\City.MYI'
Data records: 4079
- Fixing index 1
```

Now you can restart the server.

Answer 7:

To verify that the table actually has been uncompressed, use `SHOW TABLE STATUS`, as shown here:

```
shell> mysql -e "SHOW TABLE STATUS LIKE 'City'" world
+------+--------+---------+------------+------+----------------+-
| Name | Engine | Version | Row_format | Rows | Avg_row_length | ...
+------+--------+---------+------------+------+----------------+-
| City | MyISAM | 10      | Fixed      | 4079 |             67 | ...
+------+--------+---------+------------+------+----------------+-
```

Answer 8:

Yes.

Answer 9:

Fixed-row tables can be retrieved faster, whereas dynamic-row tables have lower storage requirements (on average). Sometimes it's possible to gain advantages of both table types by splitting an existing `MyISAM` table into a fixed-row table and a dynamic-row table. You could do this by using `CREATE TABLE` and `ALTER TABLE` statements, and declaring one table with `ROW_FORMAT = FIXED` and the other with `ROW_FORMAT = DYNAMIC`. Both tables need a primary key so their rows can be properly retrieved.

Answer 10:

A search for `'Access'` in either the `question` or the `answer` column could be performed as follows. (To limit the width of the output, the query displays only the first 20 characters of retrieved values.)

```
mysql> SELECT
    ->  LEFT(question,20), LEFT(answer,20)
    ->  FROM faq
    ->  WHERE MATCH(question) AGAINST('Access')
    ->  OR MATCH(answer) AGAINST('Access')
    -> ;
```

The result of the query could look like this:

```
+----------------------+----------------------+
| LEFT(question,20)    | LEFT(answer,20)      |
+----------------------+----------------------+
| Is there a database  | Access will most pro |
| Is Microsoft Access  | It's sold as a datab |
+----------------------+----------------------+
```

Note that `OR` in the preceding query means that you're looking for the word “Access” whether it appears only in the question, only in the answer, or in both the question and the answer. To find records that contain “Access” in both the question and the answer, you would use `AND` instead of `OR` in the query.

Answer 11:

To improve searches on the `id` and `name` columns, you essentially have two choices:

- You could add an index to the `name` column, thus improving searches for names.
- You could split the table into two separate tables, thus avoiding disk I/O caused by the `TEXT` column when MySQL has to scan the table. The `mix1` table could be split as shown here:

  ```
  mysql> DESCRIBE mix1; DESCRIBE mix2;
  +-------+-------------+------+-----+---------+-------+
  | Field | Type        | Null | Key | Default | Extra |
  +-------+-------------+------+-----+---------+-------+
  | id    | int(11)     | NO   | PRI | 0       |       |
  | name  | varchar(20) | YES  |     | NULL    |       |
  +-------+-------------+------+-----+---------+-------+
  +---------+---------+------+-----+---------+-------+
  | Field   | Type    | Null | Key | Default | Extra |
  +---------+---------+------+-----+---------+-------+
  | mix1_id | int(11) | NO   | PRI | 0       |       |
  | story   | text    | YES  |     | NULL    |       |
  +---------+---------+------+-----+---------+-------+
  ```

You could also combine both of the strategies just described.

Answer 12:

In that scenario, the only solution would be to use `MERGE` tables, and to split up the `MyISAM` tables into a number of smaller `MyISAM` tables, each of which will not hit the filesystem size limit.

Answer 13:

To generally make `MyISAM` tables have a larger maximum size, you can set the row pointer size from its default value of 6 to a higher value. You can find out about a table's maximum size by using a `SHOW TABLE STATUS` statement:

```
mysql> SELECT TABLE_NAME, MAX_DATA_LENGTH
    -> FROM INFORMATION_SCHEMA.TABLES
    -> WHERE TABLE_SCHEMA = 'world'
    -> AND TABLE_NAME = 'City';
+------------+-----------------+
| TABLE_NAME | MAX_DATA_LENGTH |
+------------+-----------------+
| City       |    287762808831 |
+------------+-----------------+

mysql> SELECT @@global.myisam_data_pointer_size;
+-----------------------------------+
| @@global.myisam_data_pointer_size |
+-----------------------------------+
|                                 6 |
+-----------------------------------+

mysql> SET @@global.myisam_data_pointer_size = 7;

mysql> ALTER TABLE City ENGINE = MyISAM;

mysql> SELECT TABLE_NAME, MAX_DATA_LENGTH
    -> FROM INFORMATION_SCHEMA.TABLES
```

```
    -> WHERE TABLE_SCHEMA = 'world'
    -> AND TABLE_NAME = 'City';
+------------+---------------------+
| TABLE_NAME | MAX_DATA_LENGTH     |
+------------+---------------------+
| City       | 4827858800541171711 |
+------------+---------------------+
```

Answer 14:

You can rebuild `InnoDB` tables just like any other tables by dumping them with `mysqldump` and then reloading the dump with `mysql`. An alternative method is to issue an `ALTER TABLE` statement, like this:

```
ALTER TABLE world.City ENGINE = InnoDB;
```

Rebuilding `InnoDB` tables regularly helps clean up index fragmentation that occurs from data-modifying operations. A positive side effect of rebuilding tables that were created in an older version of MySQL is that, as of MySQL 5, `InnoDB` implements a table format that results in savings of about 20% for disk and memory.

Answer 15:

You can control this by setting the `max_heap_table_size` system variable:

```
mysql> SET GLOBAL max_heap_table_size = 1024 * 1024;

mysql> SELECT @@global.max_heap_table_size;
+------------------------------+
| @@global.max_heap_table_size |
+------------------------------+
|                      1048576 |
+------------------------------+
```

Answer 16:

- The `City` table is in the first normal form (1NF) because it doesn't have repeating groups within its rows.
- It's in the second normal form (2NF) because it's in 1NF and all its non-index values depend fully on the primary key value (the `ID` column).
- It's in the third normal form (3NF) because it's in 2NF and all its non-index values depend directly on the primary key (and not on some other non-index value).

Answer 17:

Without `MAX_ROWS` specified, a `MyISAM` table can hold up to 256TB of data. If you try to insert more data, you will get a `data file full` error.

Answer 18:

1. The `Row_format` line in a `SHOW TABLE STATUS` statement indicates the row format:

   ```
   mysql> SHOW TABLE STATUS LIKE 'City'\G
   *************************** 1. row ***************************
   ```

```
           Name: City
         Engine: MyISAM
        Version: 10
     Row_format: Fixed
           Rows: 4079
 Avg_row_length: 67
    Data_length: 273293
Max_data_length: 18858823439613951
   Index_length: 43008
      Data_free: 0
 Auto_increment: 4080
    Create_time: 2005-05-29 01:20:22
    Update_time: 2005-05-30 01:54:51
     Check_time: NULL
      Collation: latin1_swedish_ci
       Checksum: NULL
 Create_options:
        Comment:
```

2. The INFORMATION_SCHEMA database contains row-format information:

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

Answer 19:

To verify that the table actually has been compressed, you could issue this command:

```
shell> mysql -e "SHOW TABLE STATUS LIKE 'City'" world
+------+--------+---------+------------+------+----------------+-
| Name | Engine | Version | Row_format | Rows | Avg_row_length | ...
+------+--------+---------+------------+------+----------------+-
| City | MyISAM | 10      | Compressed | 4079 |             19 | ...
+------+--------+---------+------------+------+----------------+-
```

Answer 20:

You can read from a compressed table, but you can no longer modify it, as these examples show:

```
mysql> USE world;
mysql> INSERT INTO City (ID, name) VALUES (10000, 'Test City');
ERROR 1036 (HY000): Table 'City' is read only
mysql> ALTER TABLE City ADD testcolumn INT;
ERROR 1036 (HY000): Table 'City' is read only
```

Answer 21:

The City table is located in the world directory under the MySQL data directory, C:\mysql\data. To refer to that directory from within the C:\mysql\bin directory, you can use either the absolute pathname C:\mysql\data\world or the relative pathname ..\data\world. The commands shown here use the latter.

To unpack the compressed table, use this command:

```
shell> myisamchk --unpack ..\data\world\City.MYI
- recovering (with keycache) MyISAM-table '..\data\world\City.MYI'
Data records: 4079
```

There are also other ways to unpack the table. One alternative is to use `mysqldump` to dump the table, followed by `mysql` to reimport it, as shown here:

```
shell> mysqldump world City > CityDump.sql
shell> mysql world < CityDump.sql
```

Another alternative is to execute these statements with `mysql`:

```
mysql> CREATE TABLE CityCopy LIKE City;
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO CityCopy SELECT * FROM City;
Query OK, 4079 rows affected (0.07 sec)
Records: 4079  Duplicates: 0  Warnings: 0

mysql> DROP TABLE City;
Query OK, 0 rows affected (0.00 sec)

mysql> ALTER TABLE CityCopy RENAME TO City;
Query OK, 0 rows affected (0.00 sec)
```

Answer 22:

`ANALYZE TABLE` updates the table's internal statistics and thus helps the optimizer make faster decisions. You don't need to run `ANALYZE TABLE` if you run `OPTIMIZE TABLE` regularly, because the latter updates the internal statistics of a `MyISAM` table, too. `OPTIMIZE TABLE` is particularly important for tables that contain variable-length data types because data-modifying operations tend to fragment such tables.

Answer 23:

A search for `'MySQL'` in the `question` column could be performed only as follows. (To limit the width of the output, the query displays only the first 20 characters of retrieved values.)

```
mysql> SELECT
    ->  LEFT(question,20), LEFT(answer,20)
    ->  FROM faq
    ->  WHERE MATCH(question) AGAINST('MySQL')
    -> ;
```

The result of the query could look like this:

```
+------------------------+----------------------+
| LEFT(question,20)      | LEFT(answer,20)      |
+------------------------+----------------------+
| Does MySQL support t   | Yes, as of version 3 |
| When will MySQL supp   | This is on the TODO  |
| Does MySQL support f   | Yes, as of version 3 |
| Does MySQL support s   | Not yet, but stored  |
| Is MySQL available u   | Yes, you can buy a l |
| When was MySQL relea   | MySQL was first rele |
+------------------------+-----------------------+
```

Answer 24:

To improve searches on the `story` column, you could add a `FULLTEXT` index to that column, like this:

```
mysql> ALTER TABLE mix1 ADD FULLTEXT (story);
mysql> SHOW KEYS FROM mix1;
+-------+------------+----------+-     -+------------+
| Table | Non_unique | Key_name |  ...  | Index_type |
+-------+------------+----------+-     -+------------+
| mix1  |          0 | PRIMARY  |  ...  | BTREE      |
| mix1  |          1 | story    |  ...  | FULLTEXT   |
+-------+------------+----------+-     -+------------+
```

Answer 25:

- Use a primary key in each table.
- Use `VARCHAR`, rather than `CHAR`.
- Avoid explicit locking (`FOR UPDATE` or `LOCK IN SHARE MODE`) if there is no index that `InnoDB` can use to look up rows.
- Avoid `SELECT COUNT(*)` statements.
- Make sure the `AUTOCOMMIT` session mode is set to `OFF` before performing multiple data modifications that could easily be grouped in a transaction.
- Rebuild `InnoDB` tables regularly to avoid index fragmentation.

Chapter 39. Optimizing the Server

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:

Using programs available from MySQL AB, there are several ways to access server status information. What are they?

Question 2:

What can you learn from server system variables and server status variables? Give an example for each type of variable.

Question 3:

In your MySQL option file, you want to have the following settings for the server:

a. The installation directory should be set to `D:\mysql`.

b. The data directory should be set to `D:\mysql_datafiles`.

c. The `MyISAM` key cache size should be 24MB.

What are the appropriate entries in your option file?

Question 4:

Which of the following statements will succeed?

a. `SET GLOBAL key_buffer_size = 25165824;`

b. `SET GLOBAL key_buffer_size = 24*1024*1024;`

c. `SET GLOBAL key_buffer_size = 24M;`

Question 5:

What privileges do you need to set a global server system variable?

Question 6:

You can set system variables (server parameters) in a number of ways:

a. With options in MySQL option files

b. With options on the command line at server startup

c. With a `SET GLOBAL` statement

d. With a `SET SESSION` (or `SET LOCAL`) statement

Assume that you choose to set system variables using options in MySQL option files. What's the lifetime of the settings? What's the scope within which each setting applies? How would you set the `sort_buffer_size` variable to a value of `512000` bytes?

Question 7:

Assume that you choose to set system variables using a `SET GLOBAL` statement. What's the lifetime of the settings? What's the scope within which each setting applies? How would you set the `sort_buffer_size` variable to a value of `512000`?

Question 8:

What SQL statement will display the status variables that tell you how many times the server has executed each type of SQL statement?

Question 9:

Besides the status variables, what other sources of information are available for checking server load and performance?

Question 10:

Name three system variables that affect the performance of `SELECT` statements.

Question 11:

What does “key cache efficiency” mean?

Question 12:

How could you determine the number of clients that successfully connected to the server?

Question 13:

How could you determine the number of clients that either aborted their connections or were killed?

Question 14:

Name MySQL's per-client buffers.

Question 15:

How would you determine the sizes of MySQL's per-client record buffers?

Question 16:

How would you determine the size of MySQL's per-client join buffer?

Question 17:

The server's query cache can speed up `SELECT` queries. How would you check the system variables for the query cache?

Question 18:

Using the system variables for the query cache, how could you determine whether the query cache is enabled?

Question 19:

The server's query cache can speed up `SELECT` queries. How would you determine to what extent the query cache is effective?

Question 20:

For what reason would you issue the following set of statements?

```
mysql> SHOW VARIABLES LIKE 'table_cache';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| table_cache   | 64    |
+---------------+-------+

mysql> SHOW STATUS LIKE 'Open_tables';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Open_tables   | 5     |
+---------------+-------+

mysql> SHOW STATUS LIKE 'Opened_tables';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Opened_tables | 5     |
+---------------+-------+
```

Question 21:

Why would you want to disable a storage engine? How would you do that for the `InnoDB` engine and the `MyISAM` engine?

Question 22:

Using programs available from MySQL AB, there are several ways to access server system variables. What are they?

Question 23:

Suppose that you have just installed MySQL, but you're not sure where the installation or data directories are located. However, you can connect to the server using `mysql`. What SQL statement or statements would you issue to find out that information?

Question 24:

Can you set all server system variables while the server is running?

Question 25:

Assume that you choose to set system variables using options on the command line at server startup. What's the lifetime of the settings? What's the scope within which each setting applies? How would you set the `sort_buffer_size` variable to a value of `512000`?

Question 26:

Assume that you choose to set system variables using a `SET SESSION` (or `SET LOCAL`) statement. What's the lifetime of the settings? What's the scope within which each setting applies? How would you set the `sort_buffer_size` variable to a value of `512000`?

Question 27:

What might the following session listing indicate about server configuration? If further analysis would point out there is in fact a problem, how could you identify and solve it?

```
mysql> SHOW STATUS LIKE 'Opened_tables';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Opened_tables | 110   |
+---------------+-------+
mysql> SELECT * FROM faq LIMIT 1;
+----------------+---------------------
| cdate          | question             ...
+----------------+---------------------
| 20030318160028 | What is the name of t ...
+----------------+---------------------
mysql> SHOW STATUS LIKE 'Opened_tables';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Opened_tables | 111   |
+---------------+-------+
mysql> SELECT * FROM t LIMIT 1;
+------+
| i    |
+------+
|    6 |
+------+
mysql> SHOW STATUS LIKE 'Opened_tables';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Opened_tables | 112   |
+---------------+-------+
```

Question 28:

What values are necessary to calculate key cache efficiency?

Question 29:

How would you retrieve the values needed to calculate the key cache efficiency?

Question 30:

How can you improve key cache efficiency for better server performance?

Question 31:

How could you determine the number of clients that tried, but failed, to connect?

Question 32:

How could you determine the number of clients that closed their connections properly?

Question 33:

How would you determine the size of MySQL's per-client sort buffer?

Question 34:

How would you determine the size of MySQL's per-client communications buffer?

Answers to Exercises

Answer 1:

You can access server status variables using any of these methods:

- With the `mysql` client program, or by typing this query into the query area of the MySQL Query Browser graphical client:

  ```
  mysql> SHOW STATUS;
  ```

- With the `mysqladmin` program:

  ```
  shell> mysqladmin extended-status
  ```

- With the MySQL Administrator graphical client: Open a connection to the server, select the Health section, and then choose the Status Variables tab.

Answer 2:

System variables provide configuration details for the server. For example, you can determine the location of the data directory like this:

```
mysql> SHOW VARIABLES LIKE 'datadir';
+---------------+------------------------+
| Variable_name | Value                  |
+---------------+------------------------+
| datadir       | /usr/local/mysql/data/ |
+---------------+------------------------+
```

Status variables provide information about the activities of the server. For example, you can check how many clients are connected to the server:

```
mysql> SHOW STATUS LIKE 'Threads_connected';
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| Threads_connected | 17    |
+-------------------+-------+
```

Answer 3:

The entries should be made in the `[mysqld]` option group like this:

```
[mysqld]
basedir = D:/mysql
datadir = D:/mysql_datafiles
key_buffer_size = 24M
```

Answer 4:

The first two statements will succeed, but the third one won't. You cannot use the `K`, `M`, or `G` suffixes with a `SET` command.

Answer 5:

You need the `SUPER` privilege.

Answer 6:

Each time the server is started, it reads settings in option files. Settings specified in these files thus pertain to each invocation of the server. For any given invocation, the settings have global scope and persist until the server shuts down. Example:

```
[mysqld]
sort_buffer_size=512000
```

Answer 7:

Using `SET GLOBAL` requires the `SUPER` privilege. You would set a global system variable like this:

```
mysql> SET GLOBAL sort_buffer_size=512000;
```

As the statement indicates, the scope is global; it applies to any clients that connect after the variable is set. An alternative syntax for setting the variable value is as follows:

```
mysql> SET @@global.sort_buffer_size=512000;
```

Answer 8:

To find out how many times the server has executed each type of statement, examine the status variables that begin with `Com`:

```
mysql> SHOW STATUS LIKE 'Com%';
+-----------------------+-------+
| Variable_name         | Value |
+-----------------------+-------+
| Com_admin_commands    | 0     |
| Com_alter_table       | 0     |
| Com_analyze           | 0     |
| Com_backup_table      | 0     |
| Com_begin             | 0     |
| Com_change_db         | 3     |
  ...                     ...
| Com_show_status       | 26    |
| Com_show_innodb_status | 1    |
| Com_show_tables       | 3     |
| Com_show_variables    | 18    |
| Com_slave_start       | 0     |
| Com_slave_stop        | 0     |
| Com_truncate          | 0     |
| Com_unlock_tables     | 0     |
| Com_update            | 0     |
...
```

This will not list the number of `SELECT` statements that were processed using the query cache. You can obtain that number from the `Qcache_hits` status variable:

```
mysql> SHOW STATUS LIKE 'Qcache_hits';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Qcache_hits   | 11019 |
+---------------+-------+
```

Answer 9:

In addition to the server's status variables, sources of server load and performance information include the following:

- The error log provides information also about errors that are not fatal but might affect server performance (such as aborted connections).
- The slow query log provides information about queries that take a long time to perform, or that are not using indexes (that latter information is available when the server was started with the additional `--log-queries-not-using-indexes` option.
- The `STATUS` command of the `mysql` client program displays some statistical information about the server load, for example the total number of queries (called `Questions`), the average number of queries per second, and the number of tables that were opened (called `Opens`). This information also is available from `mysqladmin status`.

Answer 10:

`sort_buffer_size` affects sorting operations (`ORDER BY`, `GROUP BY`), and `join_buffer_size` affects join performance. For `MyISAM` tables, `key_buffer_size` affects index-related operations.

Answer 11:

Key cache efficiency is a measure of the number of index reads from the cache, relative to index reads that need to be done from disk. The value should be as close to 1 as possible, so a value of .9 is good, but a value of .99 is much better.

Answer 12:

The information can be determined using status information available from `SHOW STATUS`. The number of successful connections can be calculated as the number of connection attempts minus the number of unsuccessful connection attempts:

```
mysql> SHOW STATUS LIKE 'Connections';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Connections   | 18220 |
+---------------+-------+
mysql> SHOW STATUS LIKE 'Aborted_connects';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| Aborted_connects | 6     |
+------------------+-------+
```

In this case, it is 18,220 – 6, or 18,214 successful connections made.

Answer 13:

The information can be determined using status information available from `SHOW STATUS`. To get the number of connections that were improperly closed (aborted), use this statement:

```
mysql> SHOW STATUS LIKE 'Aborted_clients';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| Aborted_clients | 99    |
+-----------------+-------+
```

Answer 14:

The MySQL server uses these per-client buffers:

- Record buffers are used for sequential table scans (`read_buffer_size`) and when reading rows in sorted order after a sort, usually with the `ORDER BY` clause (`read_rnd_buffer_size`).
- The join buffer is used to perform table joins.
- The sort buffer is used for sorting operations.
- The communications buffer is used for exchanging information with the client. It begins with a size of `net_buffer_length`, but the server expands it up to a size of `max_allowed_packet` as necessary.

Answer 15:

Record buffers are used for sequential table scans (`read_buffer_size`), and when reading rows in sorted order after a sort, usually with the `ORDER BY` clause (`read_rnd_buffer_size`). To get their sizes, issue this statement:

```
mysql> SHOW VARIABLES LIKE 'read%buffer_size';
+----------------------+--------+
| Variable_name        | Value  |
+----------------------+--------+
| read_buffer_size     | 131072 |
| read_rnd_buffer_size | 262144 |
+----------------------+--------+
```

Answer 16:

The join buffer is used to perform table joins. To get its size, issue this statement:

```
mysql> SHOW VARIABLES LIKE 'join_buffer_size';
+------------------+--------+
| Variable_name    | Value  |
+------------------+--------+
| join_buffer_size | 131072 |
+------------------+--------+
```

Answer 17:

To display the server variables that show how the query cache is configured, issue this statement:

```
mysql> SHOW VARIABLES LIKE 'query_cache%';
+------------------------------+---------+
| Variable_name                | Value   |
+------------------------------+---------+
| query_cache_limit            | 1048576 |
| query_cache_min_res_unit     | 4096    |
| query_cache_size             | 1048576 |
| query_cache_type             | ON      |
| query_cache_wlock_invalidate | OFF     |
+------------------------------+---------+
```

Answer 18:

If `query_cache_type` is not `OFF` and the value of `query_cache_size` is greater than zero, the query cache is enabled.

Answer 19:

To check the server's usage of the query cache, examine the appropriate status variables:

```
mysql> SHOW STATUS LIKE 'Qcache%';
+-------------------------+----------+
| Variable_name           | Value    |
+-------------------------+----------+
| Qcache_free_blocks      | 77       |
| Qcache_free_memory      | 67099960 |
| Qcache_hits             | 11546    |
| Qcache_inserts          | 7598     |
| Qcache_lowmem_prunes    | 21       |
| Qcache_not_cached       | 5511     |
| Qcache_queries_in_cache | 179      |
| Qcache_total_blocks     | 505      |
+-------------------------+----------+
```

Answer 20:

This is done to determine if the table cache is big enough. (To be able to actually assess if the current value of `table_cache` is big enough, you'd have to issue that set of statements more than just once.) If `Open_tables` usually is at or near the value of `table_cache`, and the value of `Opened_tables` increases steadily, it indicates that the table cache is being used to capacity and that the server often has to close tables in the cache so it can open other tables. This is a sign that the table cache is too small and that you should increase the value of `table_cache`.

Answer 21:

Disabling unneeded storage engines is one way to save memory. This can be done by either compiling the server without the storage engine (or engines) that you don't need, or by disabling that storage engine (or those engines) at server startup.

To disable `InnoDB`, you could either compile the server with the `--without-innodb` configuration option, or you could start the server with the `--skip-innodb` option.

The `MyISAM` storage engine is always compiled in and cannot be disabled at runtime.

Answer 22:

You can access server system variables using any of these methods:

- With the `mysql` client program, or by typing this query into the query area of the MySQL Query Browser graphical client:

  ```
  mysql> SHOW VARIABLES;
  ```

- With the `mysqladmin` program:

```
shell> mysqladmin variables
```

- With the MySQL Administrator graphical client: Open a connection to the server, select the Health section, and then choose the System Variables tab.

Answer 23:

The server stores the base installation and data directory locations in its `basedir` and `datadir` system variables, which you can display using `SHOW VARIABLES` statements:

```
mysql> SHOW VARIABLES LIKE 'basedir';
+---------------+-----------+
| Variable_name | Value     |
+---------------+-----------+
| basedir       | c:\mysql\ |
+---------------+-----------+
mysql> SHOW VARIABLES LIKE 'datadir';
+---------------+----------------+
| Variable_name | Value          |
+---------------+----------------+
| datadir       | c:\mysql\data\ |
+---------------+----------------+
```

Answer 24:

No. Some system variables are static and can *only* be set at server startup. (You need not know which for the exam.)

Answer 25:

When the server is invoked with parameter settings specified on the command line, those settings pertain only to that particular invocation. The settings have global scope and persist until the server is shut down. Example:

```
shell> mysqld --sort_buffer_size=512000
```

Answer 26:

Using `SET SESSION` (or `SET LOCAL`) sets the variable to a value that applies only to the current connection (and thus not to other connected users). This is called a “session scope.” Example:

```
mysql> SET SESSION sort_buffer_size=512000;
```

An alternative syntax for setting the variable value is as follows:

```
mysql> SET @@session.sort_buffer_size=512000;
```

Answer 27:

The session listing seems to indicate that `Opened_tables` is incremented with each table that is accessed. This does not necessarily indicate there is a problem, because these queries could have been the first ones that accessed those two tables since the server started. If, however, you find `Opened_tables` being incremented steadily even when using tables that have been opened before, it might indicate a table cache that is too small. To see the size of the cache, check the value of the `ta-`

ble_cache variable:

```
mysql> SHOW VARIABLES LIKE 'table_cache';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| table_cache   | 1     |
+---------------+-------+
```

That output shows an extremely small value that is definitely too low. You can set it higher while the server is running by issuing a SET statement:

```
mysql> SET GLOBAL table_cache=64;
mysql> SHOW VARIABLES LIKE 'table_cache';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| table_cache   | 64    |
+---------------+-------+
```

But more likely you would set the value in an appropriate option file. Then the setting would take effect each time you restart the server.

Answer 28:

Key cache efficiency is calculated from two status variables using this formula:

```
1 - (Key_reads / Key_read_requests)
```

Answer 29:

To retrieve the values necessary for calculating key cache efficiency, issue this statement:

```
mysql> SHOW STATUS LIKE 'key_read%';
+-------------------+--------+
| Variable_name     | Value  |
+-------------------+--------+
| Key_read_requests | 280944 |
| Key_reads         | 5827   |
+-------------------+--------+
```

Using the formula given in the previous exercise, the values shown in this example yield the following efficiency:

```
1 - (5827 / 280944) = .98
```

That value is close to 1 and is reasonably efficient.

Answer 30:

To improve server performance, check the value of key_buffer_size. If the value is small and memory is available, increase the key cache size. The following example sets the size to 16MB:

```
mysql> SHOW VARIABLES LIKE 'key_buffer_size';
+-----------------+--------+
| Variable_name   | Value  |
+-----------------+--------+
| key_buffer_size | 512000 |
+-----------------+--------+
mysql> SET GLOBAL key_buffer_size=16777216;
mysql> SHOW VARIABLES LIKE 'key_buffer_size';
+-----------------+----------+
| Variable_name   | Value    |
+-----------------+----------+
| key_buffer_size | 16777216 |
+-----------------+----------+
```

Answer 31:

The information can be determined using status information available from `SHOW STATUS`. To get the number of unsuccessful connection attempts, use this statement:

```
mysql> SHOW STATUS LIKE 'Aborted_connects';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| Aborted_connects | 6     |
+------------------+-------+
```

Answer 32:

The information can be determined using status information available from `SHOW STATUS`. The number of connections that were closed properly can be calculated as the number of connection attempts, minus the number of unsuccessful attempts, minus the number of aborted clients:

```
mysql> SHOW STATUS LIKE 'Connections';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| Connections     | 18222 |
+-----------------+-------+
mysql> SHOW STATUS LIKE 'Aborted_connects';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| Aborted_connects | 99    |
+------------------+-------+
mysql> SHOW STATUS LIKE 'Aborted_clients';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| Aborted_clients | 4     |
+-----------------+-------+
```

In this case, it is 18,222 – 99 – 4, or 18,119 connections properly closed.

Answer 33:

The sort buffer is used for sorting operations. To get its size, issue this statement:

```
mysql> SHOW VARIABLES LIKE 'sort_buffer_size';
+------------------+--------+
| Variable_name    | Value  |
+------------------+--------+
| sort_buffer_size | 524280 |
+------------------+--------+
```

Answer 34:

The communications buffer is used for exchanging information with the client. It begins with a size of `net_buffer_length`, but the server expands it up to a size of `max_allowed_packet` as necessary. To get the sizes of these buffers, issue the following statements:

```
mysql> SHOW VARIABLES LIKE 'net_buffer_length';
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| net_buffer_length | 16384 |
+-------------------+-------+
mysql> SHOW VARIABLES LIKE 'max_allowed_packet';
+--------------------+---------+
| Variable_name      | Value   |
+--------------------+---------+
| max_allowed_packet | 1048576 |
+--------------------+---------+
```

Chapter 40. Interpreting Diagnostic Messages

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 the following statements are true?

a. The error log logs errors only.

b. The error log contains information about auto-recovery after a crash.

c. Storage engines like `InnoDB` and `MyISAM` write to the error log when they perform automatic repairs.

d. The error log contains information about server starts and stops, but this can be ignored when doing error analyses.

Question 2:

The slow query log contains, as its name implies, information about slow queries. How can you determine what “slow” means on your particular server, and what is “slow” by default? Can you change that variable's value at runtime?

Question 3:

What kinds of errors might the server report in its diagnostic output?

Answers to Exercises

Answer 1:

a. False. The error log does not only log errors, but also incidents like server starts and stops.

b. True. The error log contains information about auto-recovery after a crash.

c. True. Storage engines like `InnoDB` and `MyISAM` write to the error log when they perform automatic repairs, for example on a restart after a server crash.

d. False. It's right that the error log contains information about server starts and stops, but this should not be ignored when doing error analyses. Many restarts normally indicate that there are serious problems that should be taken care of. An extreme example of this is when the server restarts every few seconds; typically this is the case when there is a misconfiguration in an option file that prevents the server from starting successfully, and the `mysqld_safe` script tries to restart the server permanently.

Answer 2:

You can find out what “slow” means by issuing this SQL statement:

```
mysql> SHOW VARIABLES LIKE 'long_query_time';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| long_query_time | 10    |
+-----------------+-------+
```

The default value is `10` seconds.

The value can be changed at runtime like this:

```
mysql> SET long_query_time = 5;
mysql> SHOW VARIABLES LIKE 'long_query_time';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| long_query_time | 5     |
+-----------------+-------+
```

Answer 3:

The errors reported by the server include the following:

- Unrecognized startup options
- Failure to open its network interfaces (TCP/IP port, Windows named pipe, Windows shared memory, Unix socket file)
- Storage engine initialization failure
- Failure to find SSL certificate or key files
- Inability of the server to change its user ID
- Problems related to replication

Chapter 41. Optimizing the Environment

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 three limits that the operating system imposes on a MySQL server, and describe how these limits affect server performance.

Question 2:

Name the hardware components that influence the server performance.

Question 3:

Which parts of your MySQL installation should be distributed onto different disks to improve overall performance?

Question 4:

Does the filesystem's directory lookup time influence performance when you have a large number of tables per database?

a. This is true only for `MyISAM` tables because they are represented by three files per table.

b. This is true for all storage engines.

Question 5:

What is the maximum size for a `MyISAM` table?

a. 2MB

b. 4GB

c. 65,536TB

Question 6:

Why would you want to relocate a database using the symlinking technique?

Question 7:

On Windows, assume that the MySQL data directory is in `C:\Program Files\MySQL\MySQL Server 5.0\data`. How would you relocate the `mydb` database so that it's physically located in `I:\mysqldata\db1`?

Question 8:

On Unix, assume that the MySQL data directory is in `/var/lib/mysql`. How would you relocate the `mydb` database so that it's physically located in `/home/stefan/mysqldata/db1`?

Question 9:

Where does MySQL store the format, data, and index files for a `MyISAM` table by default? What options do you use to tell the server to place the files in locations different than the default? Are there restrictions on the use of these options?

Answers to Exercises

Answer 1:

The limits to a MySQL server imposed by the operating system include the following:

- The per-process limit on number of open files. This limits the maximum size of the table cache that holds file descriptors for table files.
- The number of threads allowed to each process by the operating system. This limits the number of clients that can connect to a MySQL server at the same time.
- The backlog allowed by the operating system. This affects the maximum number of queued network connections for clients that are waiting to connect.

Answer 2:

- Memory: More memory reduces disk activity.
- Processors: 64-bit systems are superior to 32-bit systems. MySQL is multi-threaded and will benefit from multiple processors.
- Network: A faster network enables the server to transfer information more quickly which consequently reduces resource contention.
- Disks: The server makes heavy use of disk resources, so it's important to use fast disks.

Answer 3:

- Log files
- Storage location of temporary files
- Databases (may also be distributed on several disks)

Answer 4:

This is true for all storage engines because each table is represented by at least one file (the `.frm` table definition file) in the database directory. The statement is particularly true for `MyISAM` tables because these are represented by three files per table.

Answer 5:

The `MyISAM` storage engine has an internal file size limit of about 65,536TB, but `MyISAM` tables cannot actually use files that large unless the filesystem allows it.

Answer 6:

There are two main reasons for relocating databases:

- To free disk space on the data directory filesystem, if databases are moved to different filesystems.
- To distribute disk I/O if databases are moved to filesystems on different physical devices.

Answer 7:

1. Stop the server.
2. Move the `C:\Program Files\MySQL\MySQL Server 5.0\data\mydb` directory to `I:\mysqldata\db1` (renaming `mydb` to `db1` at the same time).
3. In `C:\Program Files\MySQL\MySQL Server 5.0\data`, create a text file named `mydb.sym`.
4. Edit the `mydb.sym` text file so that it contains this line: `I:\mysqldata\db1`
5. Restart the server.

Answer 8:

1. Stop the server.
2. Move the `/var/lib/mysql/mydb` directory to `/home/stefan/mysqldata/db1` (renaming `mydb` to `db1` at the same time).
3. In `/var/lib/mysql`, create a symlink named `mydb`:

   ```
   shell> ln -s /home/stefan/mysqldata/db1 mydb
   ```

4. Restart the server.

Answer 9:

By default, MySQL places the format, data, and index files for a `MyISAM` table under the data directory, in the database directory for the database that contains the table. There is no option for creating the format file anywhere else. It is always stored in the database directory. The data file and index file can be placed elsewhere by using the table options `DATA DIRECTORY='`*`path_name`*`'` and `INDEX DIRECTORY='`*`path_name`*`'`. One restriction on these options is that *`path_name`* must be a full pathname to the directory, not a relative path. Other restrictions are that these options work only on operating systems that support symlinks, and only when you have not started the server with the `--skip-symbolic-links` option.

Chapter 42. Scaling 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:

When running two servers on the same host, you'll have to make sure that they don't share resources they need exclusively. What are those resources?

Question 2:

On Windows, what are the commands to set up Windows services for two MySQL servers so that the first service is named `MySQL50` and the second one is named `MySQL51`? Additionally, the second service should read an option file that is located in the `C:\MySQLOptions` directory and has a name of `mysql51.conf`. Both servers should be capable of using named pipes.

Question 3:

How many masters can a replication slave have?

Question 4:

How many slaves can a replication master have?

Question 5:

Describe the communication process between master and client, the role of the three replication threads in that communication, and the two kinds of log files used by master and slave.

Question 6:

For replication purposes, you should set up a special user that has nothing but the `REPLICATION SLAVE` privilege. Where do you have to set up that user?

- On the master
- On the slave
- On both master and slave

Question 7:

When setting up a replication slave, you issue a `CHANGE MASTER` statement. For subsequent slave server restarts, do you have to issue that statement each time, or should you store information about the master host in an option file, or is this done otherwise?

Question 8:

Assume that a replication master manages five databases: `africa`, `america`, `asia`, `australia`, and `europe`. Its option file contains these entries:

```
binlog-do-db = america
binlog-do-db = asia
binlog-do-db = europe
```

You are planning to set up a replication network so that slave A replicates `africa` and `america`, slave B replicates `asia` and `europe`, and slave C replicates `asia` and `australia`. What would the appropriate option file entries for the slaves look like?

Question 9:

How could you find out if a slave has caught up with its master's statements?

Question 10:

Assume that you want to run two MySQL servers on one Windows machine. You have set up two system services called `MySQL_Development` and `MySQL_Production`. What would the absolute minimum of entries in the `C:\Windows\my.ini` option file look like for those two servers? (Assume that both servers are reading that option file only.)

Question 11:

Regarding a replicated database, what are the roles of master and slave?

Question 12:

Why is it a good idea to backup the original set of databases that are to be replicated using `mysqldump` with the `--master-data` option?

Question 13:

After issuing `STOP SLAVE SQL_THREAD`, which of the following statements is true?

a. The master continues sending events from its binary log to the slave.

b. The master stops sending events from its binary log until you issue `START SLAVE` on the slave.

c. The master stops sending events from its binary log until you issue `START SLAVE` *`server-id`* on the master.

d. The master stops sending events from its binary log until you issue a `CHANGE MASTER` statement on the slave.

e. The slave continues writing events to its relay log file.

f. The slave stops writing events to its relay log file.

Question 14:

It's best to have master and slaves use the same MySQL version, but often replication will still work even if these versions differ. What is the general rule for this?

a. Older slaves may replicate from a newer master.

b. Newer slaves may replicate from an older master.

Answers to Exercises

Answer 1:

The following resources cannot be shared among MySQL servers:

- The network interfaces (TCP/IP port, Windows named pipe or shared memory, or Unix socket file)
- The Windows service name
- The log files (including the `InnoDB` logs)
- The process ID file
- The `InnoDB` tablespace files

In addition, although it is sometimes possible for servers to share a data directory, doing so isn't recommended.

Answer 2:

Both servers should be capable of using named pipes, so you have to use `mysqld-nt`, rather than `mysqld`. For setting up the first service, you would change directory to where the `mysqld-nt.exe` is located, and then issue this command:

```
shell> mysqld-nt --install MySQL50
```

For the second service, change directory into its appropriate installation directory, and issue this command:

```
shell> mysqld-nt --install
           MySQL51 --defaults-file=C:\MySQLOptions\mysql51.conf
```

Answer 3:

In MySQL, a replication slave can have only one master.

Answer 4:

In MySQL, a master can have an almost unlimited number of replication slaves, although in practice the number is limited by the `max_connections` variable. If that variable's value is low compared to the number of slaves and if there are many other concurrent connections to the server, a slave might have to wait a very long time for a replication connection. (In theory, the number of slaves is limited by the fact that you must assign a unique ID value to each server that will participate in your replication setup. ID values are positive integers in the range from 1 to $2^{32} - 1$, which means you couldn't have much more than 4 billion slaves.)

Answer 5:

Replication threads interact as follows:

1. To begin receiving replication events, an I/O thread starts on the slave server and connects to the master server.
2. The master starts a thread as a connection handler for the slave I/O thread.
3. The master server sends events from its binary log files to the slave I/O thread, which records them in the slave's relay log files. The relay log stores events to be executed later.

4. The slave SQL thread processes the contents of the relay log files. When it starts, it reads events from the relay logs and executes them. As it finishes processing each relay log file, it deletes it if the I/O thread is writing to a newer relay log. If the SQL thread is reading the same relay log that the I/O thread is writing, the SQL thread pauses until more events are available in the file.

Answer 6:

The special replication user has to be created on the master only.

Answer 7:

The values given in the `CHANGE MASTER` statement are stored by the slave in its `master.info` file, so there's no need of issuing the statement on each server start, or storing it in an option file.

Answer 8:

Slave A's option file entries look like this:

```
replicate-do-db = africa
replicate-do-db = america
```

However, slave A will replicate only `america` because the `africa` database isn't being replicated by the master.

Slave B's option file entries look like this:

```
replicate-do-db = asia
replicate-do-db = europe
```

Slave B will replicate both databases because both are being replicated by the master.

Slave C's option file entries look like this:

```
replicate-do-db = asia
replicate-do-db = australia
```

However, slave C will replicate only `asia` because the `australia` database isn't being replicated by the master.

Answer 9:

You could do this by comparing the status of the master and of the slave. On the master, issue this statement:

```
mysql> SHOW MASTER STATUS;
+--------------------+----------+--------------+------------------+
| File               | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+--------------------+----------+--------------+------------------+
| apollon-bin.000007 |     1953 |              |                  |
+--------------------+----------+--------------+------------------+
```

On the slave, issue this statement:

```
mysql> SHOW SLAVE STATUS\G
*************************** 1. row ***************************
...
```

```
            Master_Log_File: apollon-bin.000007
        Read_Master_Log_Pos: 1953
...
```

If the reported values for the master's binary log file and log position are the same on master and slave, this means that the slave has caught up with its master.

Answer 10:

Running two MySQL servers on one host under Windows requires settings for at least the TCP/IP port and the data directory. An example might look like this:

```
[MySQL_Development]
port=3306
datadir=C:/mysql_1/data

[MySQL_Production]
port=3307
datadir=C:/mysql_2/data
```

By assigning a different port for each server, you make sure that they listen on different ports (otherwise, the second server wouldn't even start). By assigning a different `datadir` value, you make sure that the servers use different databases. If nothing else is specified, this will also make sure that the servers use different log files and `InnoDB` tablespace files because those will be created in the respective data directories by default.

Answer 11:

The master manages the *original* database, while the slave manages *a copy* of the original database.

Answer 12:

Backups created by `mysqldump` with the `--master-data=2` option write to the dump file a comment line containing the replication coordinates, which can be used when setting up replication slaves. That line has this format:

```
CHANGE MASTER TO MASTER_LOG_FILE='file_name', MASTER_LOG_POS=file_position;
```

Answer 13:

The master continues sending events from its binary log to the slave. The `STOP SLAVE SQL_THREAD` statement causes the slave's SQL thread to terminate, but its I/O thread is still active, so the slave continues writing events to its relay log file. The two slave threads are decoupled, so “half” of the slave's replication continues.

Answer 14:

In general, newer slaves may replicate from an older master.