Practices for Lesson 1: Mysql Overview and Architecture

Chapter 1

Practice 1-1: Quiz – Architecture Overview

Overview
In this practice, you answer questions about the MySQL architecture.

Duration
This practice should take approximately 10 minutes to complete.

Quiz Questions
Choose the best answer from those provided for each multiple choice or True/False question.

1. All of the MySQL client and non-client programs communicate with the MySQL server.
    a. True
    b. False
2. When running a MySQL server under Windows, client programs accessing that server also must run under Windows.
    a. True
    b. False
3. A command-line program commonly used to communicate with the MySQL server is called `mysqld`.
    a. True
    b. False
4. Consider the following list of ways to connect to the MySQL server. Which do not depend on the operating system? (Choose all that apply.)
    a. TCP/IP
    b. ODBC
    c. Shared memory
    d. Named pipe
    e. UNIX socket file
5. MySQL uses disk space to store the following:
    a. Server and client programs, and their libraries
    b. Log files and status files
    c. Databases
    d. Table format (`.frm`) files, data files and index files
    e. Internal temporary tables that have crossed the size threshold for being converted from in-memory tables to on-disk tables
    f. All of the above

6. The MySQL server allocates memory for the following:
   a. Connection handlers (every connection uses memory)
   b. Buffers and caches
   c. A copy of the grant tables
   d. The host cache and the table cache
   e. The query cache
   f. All of the above
7. The MySQL server uses __________________ as memory set aside to temporarily hold data for the purpose of avoiding costly disk access I/O.
   a. MEMORY table
   b. internal temporary tables
   c. shared memory
   d. buffers (or caches)

Solutions 1-1: Quiz – Architecture Overview

Quiz Solutions

1. **b.** False. There are MySQL programs that do not communicate with the server, but work directly on data or log files. Examples include `innochecksum` and `mysqlbinlog`.
2. **b.** False. MySQL can be used in heterogeneous environments. For example, a server running on a UNIX host can be accessed by clients running on Windows machines.
3. **b.** False. The most commonly used command-line program is called `mysql`, not `mysqld`. The latter is the MySQL server.
4. **a, b.** TCP/IP is not operating system–dependent. ODBC is also not operating system–dependent but is available for MySQL only on operating systems where MySQL Connector/ODBC is supported.
5. **f.** All of those listed are stored using disk space.
6. **f.** MySQL allocates memory for all of those listed.
7. **d.** The MySQL server uses buffers (or caches) as memory set aside to temporarily hold data for the purpose of avoiding costly disk access I/O.

Practice 1-1: Installing the MySQL Server

Overview

In this practice, you install the individual Linux RPM files.

Assumptions

The MySQL Server Linux RPM files have been downloaded as part of the Oracle class environment and are in the `/stage` directory.

Duration

This practice should take approximately 10 minutes to complete.

Tasks

1. Install the individual RPM files located in the `/stage/mysql` directory (as `root`).
   a. Execute the following commands for each file:
   - `rpm -hi --replacefiles MySQL-server*.rpm`
   - `rpm -hi MySQL-client*.rpm`
   - `rpm -hi MySQL-devel*.rpm`
   - `rpm -hi MySQL-shared*.rpm`
   - `rpm -hi MySQL-test*.rpm`

   Note: If your system comes with pre-installed MySQL libraries from a previous version, you can add the `--replacefiles` option to your `rpm` install command. In the Oracle classroom, the `server` package requires this option.
2. Start the MySQL server.
3. Secure the MySQL server.

   Note: You will need the initial `root` password saved in the location `/root/.mysql_secret`.

Solutions 3-1: Installing the MySQL Server

Tasks

1. Install the individual RPM files located in the `/stage/mysql` directory (as `root`).

a. Execute the following commands for each file in a terminal window and receive the results shown:

```
$ su –
Password: oracle
# cd /stage/mysql
# rpm -hi --replacefiles MySQL-server*.rpm
...
A RANDOM PASSWORD HAS BEEN SET FOR THE MySQL root USER !
You will find that password in '/root/.mysql_secret'.

You must change that password on your first connect,
no other statement but 'SET PASSWORD' will be accepted.
See the manual for the semantics of the 'password expired' flag.

Also, the account for the anonymous user has been removed.

In addition, you can run:

  /usr/bin/mysql_secure_installation

which will also give you the option of removing the test database.
This is strongly recommended for production servers.
...
New default config file was created as /usr/my.cnf and
will be used by default by the server when you start it.
You may edit this file to change server settings

WARNING: Default config file /etc/my.cnf exists on the system
This file will be read by default by the MySQL server
...
# rpm -hi MySQL-client*.rpm
# rpm -hi MySQL-devel*.rpm
# rpm -hi MySQL-shared*.rpm
# rpm -hi MySQL-test*.rpm
```

2. Start the MySQL server.

   a. Enter the following in a terminal window, and receive the result shown below:

```
# service mysql start
Starting MySQL..                                              [ OK ]
```

3. Secure the MySQL server.

   a. Enter the following in a terminal window, enter the answers appropriate to your installation, and receive the results shown below:

```
# cat /root/.mysql_secret
# The random password set for the root user ...: TeDRFkYP
# mysql -uroot -p
Enter password: TeDRFkYP
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.6.10-enterprise-commercial-advanced

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights
reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

mysql> SET PASSWORD=PASSWORD('oracle');
Query OK, 0 rows affected (0.00 sec)

mysql> EXIT
Bye
# /usr/bin/mysql_secure_installation
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MySQL
SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!
...
Enter current password for root (enter for none): oracle
OK, successfully used password, moving on...
...
You already have a root password set, so you can safely answer 'n'.

Set root password? [Y/n] n
 ... skipping.
...
Remove anonymous users? [Y/n] Y
 ... Success!
...

Disallow root login remotely? [Y/n] Y
 ... Success!
...
Remove test database and access to it? [Y/n] Y
```

```
 ... Success!
...
Reload privilege tables now? [Y/n] Y
 ... Success!
...
All done!  If you've completed all of the above steps, your MySQL
installation should now be secure.
Thanks for using MySQL!
```

Note: Although this solution shows the `mysql_secure_installation` script run as the Linux `root` user, you can run it as a normal user.

.

Practice 1-2: MySQL Data Directory

Overview

In this practice, you review the data directory associated with the MySQL server. This practice requires you to use the `mysql` client to load the `world_innodb` database. To view the pre-made tables in the `world_innodb` database, you create the database (empty) and then upload the file containing the table data.

Tasks

1. Create and populate the `world_innodb` database.
   a. Enter the `mysql` client. Enter the following in a terminal window:
   ```
   $ mysql –uroot –poracle
   ```
   – Use the username and password established during the MySQL server installation.

   b. Create the empty `world_innodb` database using the following `CREATE DATABASE` statement. In the `mysql` client, enter the following:
   ```
   mysql> CREATE DATABASE world_innodb;
   ```
   c. Select the `world_innodb` database with the `USE` statement:
   ```
   mysql> USE world_innodb
   ```
   d. Build and populate the `world_innodb` database tables using the **`SOURCE`** statement to run the `world_innodb.sql` script:
   ```
   mysql> SET autocommit=0;
   Query OK, 0 rows affected (0.00 sec)

   mysql> SOURCE /labs/world_innodb.sql
   ```
   – The `autocommit` option is described in the “Transactions and Locking” lesson. Resetting it here speeds up the script execution.

   – Several “`Query OK`…” messages scroll by before the command is completed.

   e. Set the `autocommit` option:
   ```
   mysql> SET autocommit=1;
   Query OK, 0 rows affected (0.00 sec)
   ```

2. List the current databases.
   a. In the `mysql` client, enter the following:

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| world_innodb       |
+--------------------+
4 rows in set (0.00 sec)
```

   – At this point, four databases are listed.

3. Locate the local MySQL server data directory.

   In the `mysql` client, enter the following:

```
mysql> SHOW VARIABLES LIKE 'datadir'\G
```

4. Review the contents of the data directory.

   Using the result from the preceding `SHOW VARIABLES` statement, enter the following in another terminal window:

```
$ cd /var/lib/mysql
$ ls
```

   – Each database on the MySQL server has a directory of its own.

5. Review the contents of the `mysql` database directory.

   As `root` user, change directory to the `mysql` directory and review the contents. Enter the following in a terminal window:

```
$ su –
Password: oracle
# cd /var/lib/mysql/mysql
# ls
```

   – Note the many `.frm` files contained in the directory, representing the table formats. Also note the `.MYD` and `.MYI` files for MyISAM tables, as well as some `.ibd` files for InnoDB table data, and `.CSV` and `.CSM` files for the `slow_log` and its Metafile respectively.

6. Review the contents of the `world_innodb` database directory.

   Change directory to the `world_innodb` directory and review the contents. Enter the following in the terminal window used in the previous step:

```
# cd ../world_innodb
# ls
```

   – Note the table `.frm` files, and InnoDB `.idb` files contained in the directory.

Solutions 1-2: MySQL Data Directory

Tasks

1. Create and populate the `world_innodb` database.

   a. Enter the `mysql` client. Type the following in a terminal window, and receive the result shown below:

```
$ mysql -uroot -poracle
...
Server version: 5.6.8-enterprise-commercial-advanced-log MySQL
Enterprise Server - Advanced Edition (Commercial)
...
mysql>
```

   b. Create the empty `world_innodb` database:

```
mysql> CREATE DATABASE world_innodb;
Query OK, 1 row affected (0.02 sec)
```

   c. Select the `world_innodb` database:

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

Database changed
```

   d. Build and populate the `world_innodb` database tables using the `SOURCE` statement to run the `world_innodb.sql` script:

```
mysql> SET autocommit=0;
Query OK, 0 rows affected (0.00 sec)

mysql> SOURCE /labs/world_innodb.sql
```

   – The `autocommit` option is described in the "Transactions and Locking" lesson. Resetting it here speeds up the script execution.

   – Several "`Query OK`…" messages scroll by before the command is completed.

   e. Set the `autocommit` option:

```
mysql> SET autocommit=1;
Query OK, 0 rows affected (0.00 sec)
```

2. List the current databases.

   In the `mysql` client, enter the following, and receive the result shown below:

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| world_innodb       |
+--------------------+
4 rows in set (0.00 sec)
```

   – At this point, the server contains four databases.

3. Locate the local MySQL server data directory.

   In the `mysql` client, enter the following, and receive the result shown below:

```
mysql> SHOW VARIABLES LIKE 'datadir'\G
*************************** 1. row ***************************
Variable_name: datadir
        Value: /var/lib/mysql/
1 row in set (0.01 sec)
```

4. Review the contents of the data directory.

   Enter the following in another terminal window, and receive the result shown below:

```
$ cd /var/lib/mysql
$ ls
auto.cnf     ib_logfile1     mysql               RPM_UPGRADE_HISTORY
ibdata1      hostname.err    mysql.sock          RPM_UPGRADE_MARKER-
LAST
ib_logfile0  hostname.pid    performance_schema  world_innodb
```

5. Review the contents of the `mysql` database directory:

   Enter the following in a terminal window, and receive the result shown below:

```
$ su –
Password: oracle
# cd /var/lib/mysql/mysql
# ls
columns_priv.frm   innodb_index_stats.frm    slave_worker_info.ibd
columns_priv.MYD   innodb_index_stats.ibd    slow_log.CSM
columns_priv.MYI   innodb_table_stats.frm    slow_log.CSV
...
help_topic.MYD     slave_relay_log_info.ibd
help_topic.MYI     slave_worker_info.frm
```

   – Note the many `.frm` files contained in the directory, representing the table formats. Also note the `.MYD` and `.MYI` files for MyISAM tables, as well as some `.ibd` files for InnoDB table data, and `.CSV` and `.CSM` files for the `slow_log` and its Metafile respectively.

6. Review the contents of the `world_innodb` database directory.

   Change to the `world_innodb` directory and review the contents. Enter the following in a terminal window, and receive the result shown below:

   ```
   # cd ../world_innodb
   # ls
   City.frm  Country.frm  CountryLanguage.frm  db.opt
   City.ibd  Country.ibd  CountryLanguage.ibd
   ```

   – Note the table `.frm` files, and InnoDB `.idb` files contained in the directory.

Practice 1-3: Starting and Stopping the MySQL Server

Overview
In this practice, you examine how to start and stop the MySQL server in the Linux environment.

Assumptions
The MySQL server installation has been completed prior to this exercise.

Duration
This practice should take approximately five minutes to complete.

Tasks
1. Prior to manipulating the server, log in with `admin` (or `root`) privileges.
2. Check the status of the MySQL server. (See the solution note if the server is not running.)
3. Stop the MySQL server.
4. Check the status of the MySQL server again.
5. Start the MySQL server again.
6. Check the status of the MySQL server again.

Solutions 1-3: Starting and Stopping the MySQL Server

Tasks

1. Prior to manipulating the server, you must have `admin` (or `root`) privileges.

   Enter the following in a terminal window:

   ```
   $ su –
   Password: oracle
   ```

2. Check the status of the MySQL server.

   Enter the following in a terminal window, and receive the result shown below:

   ```
   # service mysql status
   MySQL running (1117)                             [ OK ]
   ```

   – **Note:** The server should be running at this point—it was started as part of a preceding practice. If for some reason it is *not* running, run the `start` command before running the `stop` command for the next step. The process number may differ from that shown in the preceding output.

3. Stop the MySQL server.

   Enter the following in a terminal window, and receive the result shown below:

   ```
   # service mysql stop
   Shutting down MySQL......                           [ OK ]
   ```

4. Check the status of the MySQL server again.

   Enter the following in a terminal window, and receive the result shown below:

   ```
   # service mysql status
   MySQL is not running                                [FAILED]
   ```

5. Start the MySQL server again.

   Enter the following in a terminal window, and receive the result shown below:

   ```
   # service mysql start
   Starting MySQL                                   [ OK ]
   ```

   – Leave the server running because you will use it in later practices.

6. Check the status of the MySQL server.

   Enter the following in a terminal window, and receive the result shown below:

   ```
   # service mysql status
   MySQL running (1401)                                [ OK ]
   ```

Practices for Lesson 2: Server Configuration

Chapter 2

Practices for Lesson 2: Overview

Practices Overview

In these practices, you test your knowledge of MySQL server configuration. The hands-on practices are written for the Oracle Linux operating system environment, which is provided in Oracle classrooms. For non-Oracle classrooms, some adjustments may need to be made regarding file locations.

Assumptions

- The MySQL server is already installed.
- You are logged in as the `root` user in a terminal window.
- You can access the `mysql` client from a command line prompt.
- You are familiar with `gedit` or another text editor within Linux.

Practice 2-1: Quiz – MySQL Server Configuration

Overview

In this practice, you answer questions about MySQL server configuration.

Quiz Questions

Choose the best answer from those provided for each multiple choice or True/False question.

1. Which of the following statements regarding SQL modes are true? (Choose all that apply.)
   a. The setting for `sql_mode` can only be changed globally.
   b. If you want to set two SQL modes (for example, the `STRICT_ALL_TABLES` and `ERROR_FOR_DIVISION_BY_ZERO` modes), you must issue two `SET sql_mode` statements.
   c. Unless explicitly declared as global, setting SQL modes affects only the client session that sets the modes.
   d. SQL modes affect the behavior of the server. For example, they influence the way the server handles invalid input data.
   e. SQL modes affect the features that the server provides for a client. For example, you could turn InnoDB support on and off using SQL modes.
2. Is this the proper syntax for changing the SQL mode to `STRICT_TRANS_TABLES` and `PIPES_AS_CONCAT`?

   ```
   mysql> SET sql_mode = 'STRICT_TRANS_TABLES,PIPES_AS_CONCAT';
   ```

   a. Yes
   b. No
3. All MySQL programs read startup options from plain text option files named `my.ini`, `my.cnf`, or `~/.my.cnf`.
   a. True
   b. False
4. If you invoke `mysql` with the `-h 127.0.0.1` option, which of the following statements are true?
   a. `mysql` establishes a connection to a remote server.
   b. `mysql` uses a TCP/IP connection to the local instance.
   c. `mysql` works only for a specific operating system.

5. On Linux, the server writes errors to the standard error output (normally the terminal). You can write error output to a given file instead by starting the server with the `--log-error=<`*`file_name`*`>` option.
   a. True
   b. False
6. By default, the error log is written in text format to the data directory with the suffix ____________, and can be viewed using any program that displays text files.
   a. .log
   b. .error_log
   c. .err
   d. .bin_err
7. You can start the MySQL server without specifying options. To override default option values, specify the new options on the command line.
   a. True
   b. False
8. The following statements are true for binary logs:
   a. They are turned on randomly depending on the log file name.
   b. All actions are logged in one large file.
   c. They are created with a numeric, sequential, ascending extension.
   d. They are stored in binary format, not text format.

Solutions 2-1: Quiz – MySQL Server Configuration

Quiz Solutions

1. Which of the following statements regarding SQL modes are true?

   c. Unless explicitly declared as global, setting SQL modes affects only the client session that sets the modes.

   d. SQL modes affect the behavior of the server; for example, they influence the way in which the server handles invalid input data.

2. Is this the proper syntax for changing the SQL mode to `STRICT_TRANS_TABLES` and `PIPES_AS_CONCAT`?

   ```
   mysql> SET sql_mode = 'STRICT_TRANS_TABLES,PIPES_AS_CONCAT';
   ```

   a. Yes

3. All MySQL programs read startup options from plain text option files named `my.ini`, `my.cnf`, or ~/`.my.cnf`.

   a. True

4. Suppose that you invoke `mysql` with the `-h 127.0.0.1` option, which of the following statements are true?

   b. `mysql` uses a TCP/IP connection to the local instance.

5. On Linux, the server writes errors to the standard error output (normally the terminal). You can write error output to a given file instead by starting the server with the `--log-error=<`*`file_name`*`>` option.

   a. True

6. By default, the error log is written in text format to the data directory with the suffix ____________, and can be viewed using any program that displays text files.

   c. .err

7. You can start the MySQL server without specifying options. To override default option values, specify the new options on the command line.

   b. False. You can also place options in a startup option file.

8. The following statements are true about binary logs:

   c. They are created with a numeric, sequential, ascending extension.

   d. They are stored in binary format, not text format.

Practice 2-2: Editing and Creating a Configuration File

Overview

In this practice, you edit the `my.cnf` configuration file and create a new configuration file.

Tasks

1. To have a common baseline configuration, take a copy of the `my.cnf` configuration file template in the `/etc` directory and place it in `/root`.
2. Open the `/etc/my.cnf` file in a text editor. Locate the `[mysqld]` section and change the default port designation to 3309. Add a `[client]` section with a port designation of 3309.
3. In the `[mysqld]` section, indicate that the following logs are to be turned on: General, Binary, and Slow Query. Give the binary file a base name of `mybinlog`.
4. Save and close the configuration (config) file.
5. Stop and restart the server as `root`.
6. Confirm that the log files specified in your config file are now being created. View the `/var/lib/mysql` directory to see the log files.
7. In a terminal, logged in as the `oracle` user, create an additional config file with the following options: password = `oracle`, user name = `root`, turn on compress mode, show warnings, and customize the `mysql` prompt. Call this file `my_opts.txt` and save it in the home directory (`~/`).

   Note: Enter the following line in your config file to change the `mysql` prompt:

   ```
   prompt = \R:\m \d>\_
   ```

   – This option changes the prompt to include the current system time and database. You learn how to customize the `mysql` prompt in the lesson titled "Clients and Tools."
8. Invoke the `mysql` client with your new config file.
9. Confirm the settings from the new config file:
   a. Use the `STATUS` command to check the status of the `compression` option for this `mysql` client session.
   b. Note the new appearance of the prompt. Change the current database to `world_innodb` and verify that the database name appears as part of the prompt.
   c. Exit the `mysql` session.
10. Add the options from the `my_opts.txt` to the default config file (`/etc/my.cnf`), with the exception of the `password` setting, so that these options are set each time the client is invoked.
11. Logged in as the `oracle` user, invoke the `mysql` client without specifying the new config file. Confirm the settings.

12. Logged in as the `oracle` user, invoke `mysql_config_editor`, providing login settings for the `root` user in the default login path.
13. Use `mysql_config_editor` to display all stored login paths for the current user.
14. Use the **`cat`** command to display the contents of the `~/.mylogin.cnf` file.
15. Logged in as the `oracle` user, invoke the `mysql` client without providing any command-line options. When you have successfully logged in, exit the `mysql` client.
16. Remove the default login path by using `mysql_config_editor`.
17. Invoke the `mysql_config_editor` again, providing login settings for the `root` user, storing them in the login path **`admin`**.
18. Invoke the `mysql` client, specifying the login path `admin`.
19. Create a “clean” (original) version of the default config file by copying the existing `/root/my.cnf` file (created in step 1) into the `/etc` directory. After saving your new file, restart the server.

Solutions 2-2: Editing and Creating a Configuration File

Tasks

1. To have a common baseline configuration, take a copy of the `my.cnf` configuration file template in the `/etc` directory and place it in `/root`.
   a. Enter the following in a terminal window:
   ```
   $ su –
   Password: oracle
   ```
   b. Copy the `my.cnf` file to `/root`.
   ```
   # cp /etc/my.cnf /root
   ```
   c. Open the `my.cnf` configuration file in `gedit` (or your preferred text editor).
   ```
   # gedit /etc/my.cnf
   ```
   Use `vim` or `emacs` (or another editor) if you are more comfortable with those editors.
2. Open the `/etc/my.cnf` file in a text editor. Locate the `[mysqld]` section and add a port designation of 3309. Add a `[client]` section with a port designation of 3309. Modify the following lines in the **`my.cnf`** file:
   ```
   [mysqld] datadir=/var/lib/mysql
   socket=/var/lib/mysql/mysql.sock
   port=3309
   user=mysql
   ...
   [client]
   port=3309
   ```
   – In this case, the port number used when listening for TCP/IP connections is set to 3309. The port number must be 1024 or higher, unless the server is started by the `root` system user.
3. In the `[mysqld]` section, indicate that the following logs are to be turned on: General, Binary, and Slow Query. Add the following lines to the `[mysqld]` section:
   ```
   [mysqld]
   ...
   symbolic-links=0
   general_log
   log-bin=mybinlog
   slow_query_log
   ...
   ```
   – MySQL enables the general log, binary log, and slow log the next time that the server is restarted.
4. Save and close the configuration (config) file.
   – In `gedit`, press Ctrl + S, and then Ctrl + Q.

5. Stop and restart the server as `root`.

   Enter the following in a terminal window, and receive the results shown below:

   - Be sure to exit any `mysql` client that you currently have running prior to stopping the server.

```
# service mysql restart
Shutting down MySQL....                                    [ OK ]
Starting MySQL....                                         [ OK ]
```

6. Confirm that the log files specified in your config file are now being created. View the `/var/lib/mysql` directory to see the log files. Enter the following in a terminal window, and receive the results shown below:

```
# ls /var/lib/mysql
auto.cnf            ib_logfile0       mysql.sock
host-name.log       ib_logfile1       performance_schema
host-name.pid       mybinlog.000001   RPM_UPGRADE_HISTORY
host-name-slow.log  mybinlog.index    RPM_UPGRADE_MARKER-LAST
ibdata1             mysql             world_innodb
```

7. In a terminal window, logged in as the `oracle` user, create an additional config file with the following options: password = `oracle`, user name = `root`, turn on compress mode, show warnings, and customize the `mysql` prompt. Call this file `my_opts.txt` and save it in the home directory (~/).

   a. In the ~ directory (`/home/oracle` when you are logged in as `oracle`), create the `my_opts.txt` configuration file using `gedit` (or your preferred text editor).

```
$ cd
$ pwd
/home/oracle
$ gedit my_opts.txt
```

   b. Add the following to the `my_opts.txt` file:

```
[client]
password = oracle
user = root

[mysql]
compress
show-warnings
prompt = \R:\m \d>\_
```

   c. Save and close the new config file.

8. Invoke the `mysql` client with the `my_opts.txt` file. Enter the following in a terminal window, and receive the result shown below:

```
$ mysql --defaults-extra-file=~/my_opts.txt
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.6.10-enterprise-commercial-advanced-log MySQL
Enterprise Server - Advanced Edition (Commercial)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights
reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

10:21 (none)>
```

   – The prompt now reflects the current time of the system, and shows that no database is currently being used.

9. Confirm the settings from the new config file:

   a. Use the STATUS command to check the status of the `compression` option for this `mysql` client session. Enter the following at the `mysql` prompt, and receive the result shown below:

```
10:21 (none)> STATUS
--------------
mysql  Ver 14.14 Distrib 5.6.10, for Linux (x86_64) using  EditLine
wrapper

Connection id:          1
Current database:
Current user:           root@localhost
SSL:                    Not in use
Current pager:          stdout
Using outfile:          ''
Using delimiter:        ;
Server version:         5.6.10-enterprise-commercial-advanced-log
MySQL Enterprise Server - Advanced Edition (Commercial)
Protocol version:       10
Connection:             Localhost via UNIX socket
Server characterset:    latin1
Db     characterset:    latin1
Client characterset:    utf8
Conn.  characterset:    utf8
```

```
UNIX socket:            /var/lib/mysql/mysql.sock
Protocol:               Compressed
Uptime:                 8 min 40 sec

Threads: 1  Questions: 8  Slow queries: 0  Opens: 70  Flush tables: 1
Open tables: 63  Queries per second avg: 0.015
--------------
```

b. Note the new appearance of the prompt. Change the current database to `world_innodb` and verify that the database name appears as part of the prompt. Enter the following command and receive the result shown below:

```
10:22 (none)> USE world_innodb
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
10:23 world_innodb>
```

c. Exit the `mysql` session, and receive the result shown below:

```
10:23 world_innodb> EXIT
Bye
$
```

10. Add the options from the `my_opts.txt` to the default config file (`/etc/my.cnf`), with the exception of the `password` setting, so that these options are set each time the client is invoked. Logged in as `root`, add the following to the existing `/etc/my.cnf` file:

```
...
[client]
port=3309
user = root

[mysql]
compress
show-warnings
prompt = \R:\m \d>\_
```

- Placing the password in the default option file is not secure and a bad practice.
- You do not have to restart the `mysql` server when you change client options.

11. Logged in as the `oracle` user, invoke the `mysql` client without specifying the new config file. Confirm the settings.

Enter the following in a terminal window, logged in as the `oracle` user, and receive the result shown below:

```
$ mysql -p
Enter password: oracle
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.6.8-enterprise-commercial-advanced-log MySQL
Enterprise Server - Advanced Edition (Commercial)
12:01 (none)>
```

   – You do not need to provide a value for **user**, because it is hard-coded into the configuration file.

12. Logged in as the `oracle` user, invoke the `mysql_config_editor`, providing login settings for the `root` user in the default login path.

    Enter the following in a new terminal window, logged in as the `oracle` user, and receive the result shown below:

```
$ mysql_config_editor set --user=root --password
Enter password: oracle
```

13. Use `mysql_config_editor` to display all stored login paths for the current user. Enter the following in a terminal window, logged in as the `oracle` user, and receive the result shown below:

```
$ mysql_config_editor print --all
[client] user =
root password =
*****
```

   – The command displays the plain-text contents of the newly created `~/.mylogin.cnf` file, obscuring the password.

14. Use the `cat` command to display the contents of the `~/.mylogin.cnf` file. Enter the following in a terminal window, logged in as the `oracle` user, and receive the result shown below:

```
$ cat ~/.mylogin.cnf
     ─────────────────────────-
────────────────────────────────────────────────────────────────────────
     2�U�1�浣�OB@1@��`{���$�]Nq � �E� Aq��-
��)^�|{�j��s�x��W�$
```

   – The file contents are unreadable, and you cannot see the username or password stored within.

15. Logged in as the `oracle` user, invoke the `mysql` client without providing any command-line options. When you have successfully logged in, exit the `mysql` client.

    a. Enter the following in a terminal window, logged in as the `oracle` user, and receive the result shown below:

```
$ mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.6.10-enterprise-commercial-advanced-log MySQL
Enterprise Server - Advanced Edition (Commercial)
...
```

   – You are logged in with the username and password provided in step 12. If you provide no command-line options, the `mysql` client uses the default login path. In step 12, you did not specify a login path name, so the settings were stored under the default path.

b. To exit, enter the following at the `mysql` prompt:

```
10:36 (none)> EXIT
Bye
```

16. Remove the default login path by using `mysql_config_editor`. Enter the following in a terminal window, logged in as the `oracle` user, and receive the result shown below:

```
$ mysql_config_editor remove
WARNING : No login path specified, so default login path will be
removed.
 Continue? (Press y|Y for Yes, any other key for No) : y
```

17. Invoke the `mysql_config_editor` again, providing login path settings for the `root` user, storing them in the login path `admin`. Enter the following in a terminal window, logged in as the `oracle` user, and receive the result shown below:

```
$ mysql_config_editor set --login-path=admin --user=root --password
Enter password: oracle
```

18. Invoke the `mysql` client, specifying the login path `admin`.

```
$ mysql --login-path=admin
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
...
```

- You are logged in with the username and password provided in step 17. When you use the `--login-path` option, the `mysql` client uses the credentials stored in that login path.

To exit, enter the following at the `mysql` prompt:

```
10:39 (none)> EXIT
Bye
```

19. Create a “clean” (original) version of the default config file by copying the existing `/root/my.cnf` file (created in step 1) into the `/etc` directory. After saving your new file, restart the server.

a. Enter the following in a terminal window, logged in as `root`:

```
# cp /root/my.cnf /etc/
cp: overwrite `/etc/my.cnf'? y
```

- Part of the reason for doing this is to go back to the default port setting. If you leave it set to 3309 (or other non-default), you must specify the port for all client connections, which can be time-consuming and a source of error. Revert to the default now to avoid later misunderstandings.

b. Restart the server. Enter the following in a terminal window, and receive the results shown below:

```
# service mysql restart
Shutting down MySQL......                                 [ OK ]
Starting MySQL......                                    [ OK ]
```

Practice 2-3: Additional Practice – Server Configuration

Overview

In this practice, you use the `world_innodb` database and the information from this lesson regarding MySQL server configuration.

Tasks

1. Check to see whether the general and slow query logs are turned on, by using the `SHOW GLOBAL VARIABLES LIKE '%log'` statement. If they are not on, use `SET GLOBAL` statements to turn on the log files.
2. Make sure that the logs are table-based by checking the `log_output` variable value. If needed, set the variable value to **`TABLE`**, and then empty the log tables by using a `TRUNCATE` statement.
3. Make a new copy of the `world_innodb` database, called `world2`. To accomplish this, create a new database called `world2` and source the `world_innodb.sql` database file again when connected to `world2`.
4. Count the number of `CREATE TABLE` statements in the general log, by using the following `SELECT` statement:

   ```
   mysql> SELECT COUNT(*) FROM mysql.general_log
        -> WHERE argument LIKE 'CREATE TABLE%';
   ```
5. Create a query that makes it into the slow query log:
   a. Example: A `SELECT` that takes some time

   ```
   mysql> SELECT SLEEP(11);
   ```
   b. View it from the `slow_log` table in the `mysql` database.
6. Determine whether the binary logging is enabled by checking the `log_bin` variable value. Then exit the current `mysql` session.
7. Edit the `my.cnf` file (located in the `/etc` directory) in the `[mysqld]` section to turn on binary logging. After saving your edit, stop and restart the server.
8. In a new `mysql` session, make sure that the binary log is enabled and erase all previous logs (if present), by using the `RESET MASTER` statement.
9. Perform a data changing operation:
   a. For instance, create and drop a database:

   ```
   mysql> CREATE DATABASE foo;
   mysql> DROP DATABASE foo;
   ```
   b. List it from the binary log by using the `SHOW BINLOG EVENTS` statement.
10. Rotate the binary logs explicitly using `FLUSH BINARY LOGS` and perform some more data changing queries (similar to the previous step).

11. List all your binary log files. Then display the latest log entries by using an appropriate `SHOW` statement with the `mysql-bin.000002` log file.
12. Purge the first binary log.
13. Perform some more data modifying queries (similar to the previous steps) but with the database name `foo2`.
14. Use `mysqlbinlog` to display the binary log. Identify which entries are new since you last inspected the log in step 11.
15. Look in the `mysql-bin.000002` binary log file. The log file records only the events that have taken place since the logs were flushed (the events resulting from step 13). What was the event number for the drop of the `foo2` database?
16. From the `mysql` prompt, load the Audit Log plugin.
17. Execute some modification statements (similar to the previous steps).
18. View the value of the `audit_log_file` system variable, and use that value to locate and view the contents of the audit log.
19. Unload the Audit Log plugin, noting any warnings.
20. Execute some modification statements (similar to the previous steps).
21. Exit the current `mysql` session.
22. View the last few lines of the audit log.
23. Configure the `/etc/my.cnf` file to enable the Audit Log plugin at server start-up, and to prevent it from being unloaded at runtime. Restart the server when you have done this.
24. Attempt to unload the Audit Log plugin, and note the result.
25. View the audit log file again, noting any changes.
26. Remove all settings related to Audit Log from the `/etc/my.cnf` file, and restart the MySQL server.

Solutions 2-3: Additional Practice – Server Configuration

Tasks

1. Check to see whether the general and slow query logs are turned on, by using the `SHOW GLOBAL VARIABLES LIKE '%log'` statement. If they are not on, use `SET GLOBAL` statements to turn on the log files.

   a. Invoke the `mysql` client from a terminal logged in as `oracle`, specifying the login path `admin`.

   ```
   $ mysql --login-path=admin
   Your MySQL connection id is 1
   ...
   ```

   b. View the status variables that end with the word "log." Enter the following at the `mysql` prompt, and receive the result shown below:

   ```
   mysql> SHOW GLOBAL VARIABLES LIKE '%log';
   +---------------------------------+-------+
   | Variable_name                   | Value |
   +---------------------------------+-------+
   | back_log                        | 80    |
   | general_log                     | OFF   |
   | innodb_api_enable_binlog        | OFF   |
   | innodb_locks_unsafe_for_binlog  | OFF   |
   | relay_log                       |       |
   | slow_query_log                  | OFF   |
   | sync_binlog                     | 0     |
   | sync_relay_log                  | 10000 |
   +---------------------------------+-------+
   8 rows in set (0.00 sec)
   ```

   c. Turn on the log files. Enter the following and receive the results shown below:

   ```
   mysql> SET GLOBAL general_log = ON;
   Query OK, 0 rows affected (0.06 sec)
   mysql> SET GLOBAL slow_query_log = ON;
   Query OK, 0 rows affected (0.07 sec)
   ```

d. View the new status variables that end with the word "log". Enter the following at the mysql prompt, and receive the result shown below:

```
mysql> SHOW GLOBAL VARIABLES LIKE '%log';
+--------------------------------+-------+
| Variable_name                  | Value |
+--------------------------------+-------+
| back_log                       | 80    |
| general_log                    | ON    |
| innodb_api_enable_binlog       | OFF   |
| innodb_locks_unsafe_for_binlog | OFF   |
| relay_log                      |       |
| slow_query_log                 | ON    |
| sync_binlog                    | 0     |
| sync_relay_log                 | 10000 |
+--------------------------------+-------+
8 rows in set (0.00 sec)
```

2. Make sure that the logs are table-based by checking the `log_output` variable value. If needed, set the variable value to `TABLE`, and then empty the log tables by using a `TRUNCATE` statement.

a. Check the `log_output` variable value. Enter the following, and receive the result shown below:

```
mysql> SHOW GLOBAL VARIABLES LIKE 'log_output';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| log_output    | FILE  |
+---------------+-------+
1 row in set (0.00 sec)
```

b. If it not set to `TABLE`, then type the following, and receive the result shown below:

```
mysql> SET GLOBAL log_output = 'TABLE';
Query OK, 0 rows affected (0.01 sec)
```

c. Verify that the `log_output` variable is now set to `TABLE`. Enter the following, and receive the result shown below:

```
mysql> SHOW GLOBAL VARIABLES LIKE 'log_output';
+---------------+--------+
| Variable_name | Value  |
+---------------+--------+
| log_output    | TABLE  |
+---------------+--------+
1 row in set (0.00 sec)
```

d. Empty the log tables. Enter the following, and receive the results shown below:

```
mysql> TRUNCATE mysql.general_log;
Query OK, 0 rows affected (0.01 sec)

mysql> TRUNCATE mysql.slow_log;
Query OK, 0 rows affected (0.01 sec)
```

3. Make a new copy of the `world_innodb` database, called `world2`. To accomplish this, create a new database called `world2` and source the **`world_innodb.sql`** database file again when connected to `world2`.

   Create and populate the `world2` database. Enter the following, and receive the results shown below:

```
mysql> CREATE DATABASE world2;
Query OK, 1 row affected (0.02 sec)

mysql> USE world2;
Database changed
mysql> SET autocommit=0;
Query OK, 0 rows affected (0.00 sec)

mysql> SOURCE /labs/world_innodb.sql
Query OK ...
...
Query OK, 0 rows affected (0.00 sec)

mysql> SET autocommit=1;
Query OK, 0 rows affected (0.04 sec)
```

   – Resetting the `autocommit` option speeds up the import.

4. Count the number of `CREATE TABLE` statements in the general log, by using the following `SELECT` statement. Enter the following, and receive the result shown below:

```
mysql> SELECT COUNT(*) FROM mysql.general_log
    -> WHERE argument LIKE 'CREATE TABLE%';
+----------+
| COUNT(*) |
+----------+
| 3        |
+----------+
```

5. Create a query that makes it into the slow query log:

   a. Example: A `SELECT` that takes some time. Enter the following, and receive the result shown below:

```
mysql> SELECT SLEEP(11);
+------------+
| SLEEP (11) |
+------------+
|          0 |
+------------+
```

b. View it from the `slow_log` table in the `mysql` database. Enter the following query, and receive the result shown below:

```
mysql> SELECT * FROM mysql.slow_log\G
*************************** 1. row ***************************
    start_time: 2013-02-01 11:06:36
     user_host: root[root] @ localhost []
    query_time: 00:00:11
     lock_time: 00:00:00
     rows_sent: 1
 rows_examined: 0
            db: world2
last_insert_id: 0
     insert_id: 0
     server_id: 0
      sql_text: SELECT SLEEP(11)
     thread_id: 1
1 row in set (0.00 sec)
```

6. Determine whether the binary logging is enabled by checking the `log_bin` variable value:

a. Check the status of the log. Enter the following, and receive the result shown below:

```
mysql> SHOW GLOBAL VARIABLES LIKE 'log_bin';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| log_bin       | OFF   |
+---------------+-------+
```

b. Then exit the current `mysql` session.

```
mysql> EXIT
Bye
```

7. Edit the `my.cnf` file (located in the `/etc` directory) in the `[mysqld]` section to turn on binary logging. After saving your edit, stop and restart the server.

a. In the terminal logged in as `root`, open the `my.cnf` configuration file in `gedit` (or your preferred text editor).

```
# gedit /etc/my.cnf
```

b. Add the following line controlling log updates to the `[mysqld]` section:

```
[mysqld]
log-bin=mysql-bin
datadir=/var/lib/mysql
...
```

c. Save and close the configuration file.

d. After saving your edit, stop and restart the server. Enter the following in a terminal window, and receive the results shown below:

```
# service mysql restart
Shutting down MySQL...                                           [  OK  ]
Starting MySQL.                                                  [  OK  ]
```

8. In a new `mysql` session, make sure that the binary log is enabled and erase all previous logs (if present), by using the `RESET MASTER` statement.

   a. In a Linux terminal logged in as `oracle`, launch the `mysql` client:

```
$ mysql --login-path=admin
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
...
```

   b. Check the status of the log. Enter the following, and receive the result shown below:

```
mysql> SHOW GLOBAL VARIABLES LIKE 'log_bin';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| log_bin       | ON    |
+---------------+-------+
```

   c. Erase all logs. Enter the following, and receive the result shown below:

```
mysql> RESET MASTER;
Query OK, 0 row affected (0.09 sec)
```

9. Perform a data changing operation and list it from the binary log.

   a. For instance, create and drop a database. Enter the following, and receive the results shown below:

```
mysql> CREATE DATABASE foo;
Query OK, 1 row affected (0.00 sec)

mysql> DROP DATABASE foo;
Query OK, 0 row affected (0.00 sec)
```

   b. List it from the binary log by using the `SHOW BINLOG EVENTS` statement. Enter the following, and receive the result shown below:

```
mysql> SHOW BINLOG EVENTS;
+------------------+-----+------------+-----------+----------+---------------------
| Log_name         | Pos | Event_type  | Server_id | End_log… | Info
+------------------+-----+------------+-----------+----------+---------------------
| mysql-bin.000001 |   4 | Format_desc |         1 |      120 | Server ver: 5.6.10..
| mysql-bin.000001 | 120 | Query       |         1 |      211 | CREATE DATABASE foo
| mysql-bin.000001 | 211 | Query       |         1 |      294 | DROP DATABASE foo
+------------------+-----+------------+-----------+----------+--------------------
3 rows in set (0.00 sec)
```

10. Rotate the binary logs explicitly by using `FLUSH BINARY LOGS` and perform some more data changing queries (similar to the previous step).

    a. Flush the binary log. Enter the following:

    ```
    mysql> FLUSH BINARY LOGS;
    Query OK, 0 row affected (0.04 sec)
    ```

    b. Change data. Enter the following, and receive the results shown below:

    ```
    mysql> CREATE DATABASE foo;
    Query OK, 1 row affected (0.00 sec)

    mysql> DROP DATABASE foo;
    Query OK, 0 row affected (0.02 sec)
    ```

11. List all your binary log files. Then display the latest log entries by using an appropriate `SHOW` statement with the `mysql-bin.000002` log file.

    a. List logs. Enter the following, and receive the result shown below:

    ```
    mysql> SHOW MASTER LOGS;
    +------------------+-----------+
    | Log_name         | File_size |
    +------------------+-----------+
    | mysql-bin.000001 |       341 |
    | mysql-bin.000002 |       294 |
    +------------------+-----------+
    2 rows in set (0.00 sec)
    ```

    – `SHOW MASTER LOGS` and `SHOW BINARY LOGS` are functionally equivalent.

    b. List changes. Enter the following, and receive the result shown below:

    ```
    mysql> SHOW BINLOG EVENTS IN 'mysql-bin.000002';
    +------------------+-----+-------------+-----------+----------+---------------------
    | Log_name         | Pos | Event_type  | Server_id | End_log… | Info
    +------------------+-----+-------------+-----------+----------+---------------------
    | mysql-bin.000002 |   4 | Format_desc |         1 |      120 | Server ver: 5.6.10..
    | mysql-bin.000002 | 120 | Query       |         1 |      211 | CREATE DATABASE foo
    | mysql-bin.000002 | 211 | Query       |         1 |      294 | DROP DATABASE foo
    +------------------+-----+-------------+-----------+----------+--------------------
    3 rows in set (0.00 sec)
    ```

12. Purge the first binary log. Enter the following, and receive the result shown below:

    ```
    mysql> PURGE MASTER LOGS TO 'mysql-bin.000002';
    Query OK, 0 row affected (0.05 sec)
    ```

    – `PURGE MASTER LOGS` and `PURGE BINARY LOGS` are functionally equivalent.

13. Perform some more data modifying queries (similar to the previous steps) but with the database name `foo2`. Enter the following, and receive the results shown below:

    ```
    mysql> CREATE DATABASE foo2;
    Query OK, 1 row affected (0.00 sec)

    mysql> DROP DATABASE foo2;
    Query OK, 0 row affected (0.00 sec)
    ```

14. Use mysqlbinlog to display the binary log. Identify which entries are new since you last inspected the log in step 11.

    Enter the following in a terminal window logged in as root, and receive the result shown below:

```
# mysqlbinlog /var/lib/mysql/mysql-bin.000002
/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=1*/;
/*!40019 SET @@session.max_insert_delayed_threads=0*/;
/*!50003 SET
@OLD_COMPLETION_TYPE=@@COMPLETION_TYPE,COMPLETION_TYPE=0*/;
DELIMITER /*!*/;
# at 4
#130202 12:02:05 server id 1  end_log_pos 120 CRC32 0x061acade  Start:
binlog v 4, server v 5.6.10-enterprise-commercial-advanced-log created
130202 12:02:05
# Warning: this binlog is either in use or was not closed properly.
BINLOG '
PQANUQ8BAAAAdAAAAHgAAAABAAQANS42LjEwLWVudGVycHJpc2UtY29tbWVyY2lhbC1hZH
ZhbmNl
ZC1sb2cAAAAAAAAAAAAAAAAAEzgNAAgAEgAEBAQEEgAAXAAEGggAAAAICAgCAAAACgoKGR
kAAd7K
GgY=
'/*!*/;
# at 120
#130202 12:02:15 server id 1  end_log_pos 211 CRC32 0x9d08e28b  Query
thread_id=1     exec_time=0     error_code=0
SET TIMESTAMP=1359806535/*!*/;
SET @@session.pseudo_thread_id=1/*!*/;
SET @@session.foreign_key_checks=1, @@session.sql_auto_is_null=0,
@@session.unique_checks=1, @@session.autocommit=1/*!*/;
SET @@session.sql_mode=1075838976/*!*/;
SET @@session.auto_increment_increment=1,
@@session.auto_increment_offset=1/*!*/;
/*!\C utf8 *//*!*/;
SET
@@session.character_set_client=33,@@session.collation_connection=33,@@
session.collation_server=8/*!*/;
SET @@session.lc_time_names=0/*!*/;
SET @@session.collation_database=DEFAULT/*!*/;
CREATE DATABASE foo
/*!*/;
# at 211
#130202 12:02:21 server id 1  end_log_pos 294 CRC32 0x24ec5ec1  Query
thread_id=1     exec_time=0     error_code=0
SET TIMESTAMP=1359806541/*!*/;
DROP DATABASE foo
/*!*/;
# at 294
```

```
#130202 12:06:19 server id 1  end_log_pos 388 CRC32 0xa8736010  Query
thread_id=1     exec_time=0     error_code=0
SET TIMESTAMP=1359806779/*!*/;
CREATE DATABASE foo2
/*!*/;
# at 388
#130202 12:06:19 server id 1  end_log_pos 473 CRC32 0xee26c05a  Query
thread_id=1     exec_time=0     error_code=0
SET TIMESTAMP=1359806779/*!*/;
DROP DATABASE foo2
/*!*/;
DELIMITER ;
# End of log file
ROLLBACK /* added by mysqlbinlog */;
/*!50003 SET COMPLETION_TYPE=@OLD_COMPLETION_TYPE*/;
/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=0*/;
```

15. Look in the `mysql-bin.000002` binary log file. The log file records only the events that have taken place since the logs were flushed (the events resulting from step 13). What was the event number for the drop of the `foo2` database? 388
    - The event position number combined with the binary log file name identifies the specific event.
16. From the `mysql` prompt, load the Audit Log plugin. Enter the following, and receive the results shown below:

```
mysql> INSTALL PLUGIN audit_log SONAME 'audit_log.so';
Query OK, 0 rows affected (0.19 sec)
```

17. Execute some modification statements (similar to the previous steps). Enter the following, and receive the results shown below:

```
mysql> CREATE DATABASE foo;
Query OK, 1 row affected (0.00 sec)

mysql> DROP DATABASE foo;
Query OK, 0 row affected (0.00 sec)
```

18. View the value of the `audit_log_file` system variable, and use that value to locate and view the contents of the audit log.
    a. Enter the following, and receive the results shown below:

```
mysql> SHOW VARIABLES LIKE 'audit_log_file';
+----------------+-----------+
| Variable_name  | Value     |
+----------------+-----------+
| audit_log_file | audit.log |
+----------------+-----------+
1 row in set (0.20 sec)
```

   - The `audit.log` file is in the server's data directory.

b. In a Linux terminal window, logged in as `root`, enter the following and receive the results shown:

```
# cat /var/lib/mysql/audit.log
<?xml version="1.0" encoding="UTF-8"?>
<AUDIT>
  <AUDIT_RECORD TIMESTAMP="2013-02-02T12:22:45" NAME="Audit"
SERVER_ID="1" VERSION="1" STARTUP_OPTIONS="/usr/sbin/mysqld --
basedir=/usr --datadir=/var/lib/mysql --plugin-
dir=/usr/lib64/mysql/plugin --user=mysql --log-
error=/var/log/mysqld.log --pid-file=/var/lib/mysql/host-name.pid --
socket=/var/lib/mysql/mysql.sock" OS_VERSION="x86_64-Linux"
MYSQL_VERSION="5.6.10-enterprise-commercial-advanced-log"/>
  <AUDIT_RECORD TIMESTAMP="2013-02-02T12:22:45" NAME="Query"
CONNECTION_ID="1" STATUS="0" SQLTEXT="INSTALL PLUGIN audit_log SONAME
'audit_log.so'"/>
  <AUDIT_RECORD TIMESTAMP="2013-02-02T12:23:22" NAME="Query"
CONNECTION_ID="1" STATUS="0" SQLTEXT="CREATE DATABASE foo"/>
  <AUDIT_RECORD TIMESTAMP="2013-02-02T12:23:25" NAME="Query"
CONNECTION_ID="1" STATUS="0" SQLTEXT="DROP DATABASE foo"/>
  <AUDIT_RECORD TIMESTAMP="2013-02-02T12:23:25" NAME="Query"
CONNECTION_ID="1" STATUS="0" SQLTEXT="SELECT DATABASE()"/>
  <AUDIT_RECORD TIMESTAMP="2013-02-02T12:23:51" NAME="Query"
CONNECTION_ID="1" STATUS="0" SQLTEXT="SHOW VARIABLES LIKE
'audit_log_file'"/>
```

19. Unload the Audit Log plugin, noting any warnings. Enter the following, and receive the results shown below:

```
mysql> UNINSTALL PLUGIN audit_log;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> SHOW WARNINGS;
+---------+------+---------------------------------------------------+
| Level   | Code | Message                                           |
+---------+------+---------------------------------------------------+
| Warning | 1620 | Plugin is busy and will be uninstalled on shutdown |
+---------+------+---------------------------------------------------+
1 row in set (0.00 sec)
```

20. Execute some modification statements (similar to the previous steps). Enter the following, and receive the results shown below:

```
mysql> CREATE DATABASE foo;
Query OK, 1 row affected (0.00 sec)

mysql> DROP DATABASE foo;
Query OK, 0 row affected (0.00 sec)
```

21. Exit the current `mysql` session. Enter the following, and receive the results shown below:

```
mysql> EXIT
Bye
```

22. View the last few lines of the audit log. Enter the following in a Linux terminal window logged in as `root`, and receive the results shown below:

```
# cat /var/lib/mysql/audit.log
<?xml version="1.0" encoding="UTF-8"?>
<AUDIT>
...
  <AUDIT_RECORD TIMESTAMP="2013-02-02T12:23:25" NAME="Query"
CONNECTION_ID="1" STATUS="0" SQLTEXT="DROP DATABASE foo"/>
  <AUDIT_RECORD TIMESTAMP="2013-02-02T12:23:25" NAME="Query"
CONNECTION_ID="1" STATUS="0" SQLTEXT="SELECT DATABASE()"/>
  <AUDIT_RECORD TIMESTAMP="2013-02-02T12:23:51" NAME="Query"
CONNECTION_ID="1" STATUS="0" SQLTEXT="SHOW VARIABLES LIKE
'audit_log_file'"/>
  <AUDIT_RECORD TIMESTAMP="2013-02-02T12:27:31" NAME="Query"
CONNECTION_ID="1" STATUS="0" SQLTEXT="UNINSTALL PLUGIN audit_log"/>
  <AUDIT_RECORD TIMESTAMP="2013-02-02T12:27:31" NAME="NoAudit"
SERVER_ID="1"/>
</AUDIT>
```

   – The XML file contains a closing tag, closing the `AUDIT` root element, and there is no record of the final modification statements.

23. Configure the `/etc/my.cnf` file to enable the Audit Log plugin at server start-up, and to prevent it from being unloaded at runtime. Restart the server when you have done this.

   a. Edit the file `/etc/my.cnf` as follows:

```
[mysqld]
plugin-load=audit_log.so
audit-log=FORCE_PLUS_PERMANENT
log-bin=mysql-bin
datadir=/var/lib/mysql
...
```

   b. Enter the following in a terminal window, and receive the results shown below:

```
# service mysql restart
Shutting down MySQL...                                      [  OK  ]
Starting MySQL.                                             [  OK  ]
```

24. Reconnect to `mysql` as `root`. Attempt to unload the Audit Log plugin, and note the result. Enter the following, and receive the results shown below:

```
$ mysql --login-path=admin
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.6.10-enterprise-commercial-advanced-log MySQL
Enterprise Server - Advanced Edition (Commercial)
...
mysql> UNINSTALL PLUGIN audit_log;
ERROR 1702 (HY000): Plugin 'audit_log' is force_plus_permanent and can
not be unloaded
```

25. View the audit log file again, noting any changes. Enter the following at a terminal prompt as `root`, and receive the results shown below:

```
# cat /var/lib/mysql/audit.log
<?xml version="1.0" encoding="UTF-8"?>
<AUDIT>
...
  <AUDIT_RECORD TIMESTAMP="2013-02-02T12:27:31" NAME="Query"
CONNECTION_ID="2" STATUS="0" SQLTEXT="UNINSTALL PLUGIN audit_log"/>
  <AUDIT_RECORD TIMESTAMP="2013-02-02T12:27:31" NAME="NoAudit"
SERVER_ID="1"/>
  <AUDIT_RECORD TIMESTAMP="2013-02-02T12:35:00" NAME="Audit"
SERVER_ID="1" VERSION="1" STARTUP_OPTIONS="/usr/sbin/mysqld --
basedir=/usr --datadir=/var/lib/mysql --plugin-
dir=/usr/lib64/mysql/plugin --user=mysql --log-
error=/var/log/mysqld.log --pid-file=/var/lib/mysql/EDTDR20P1.pid --
socket=/var/lib/mysql/mysql.sock" OS_VERSION="x86_64-Linux"
MYSQL_VERSION="5.6.10-enterprise-commercial-advanced-log"/>
  <AUDIT_RECORD TIMESTAMP="2013-02-02T12:39:16" NAME="Connect"
CONNECTION_ID="1" STATUS="0" USER="root" PRIV_USER="root" OS_LOGIN=""
PROXY_USER="" HOST="localhost" IP="" DB=""/>
  <AUDIT_RECORD TIMESTAMP="2013-02-02T12:39:16" NAME="Query"
CONNECTION_ID="1" STATUS="0" SQLTEXT="select @@version_comment limit
1"/>
  <AUDIT_RECORD TIMESTAMP="2013-02-02T12:40:05" NAME="Query"
CONNECTION_ID="1" STATUS="1702" SQLTEXT="UNINSTALL PLUGIN audit_log"/>
```

   – The XML file has been re-opened by the Audit Log plugin, and it has recorded the second attempt to uninstall the plugin, along with its error number, but it has not unloaded the plugin.

26. Remove all settings related to Audit Log from the `/etc/my.cnf` file, and restart the MySQL server.

   a. Remove the following lines from the `/etc/my.cnf` file:

   ```
   plugin-load=audit_log.so
   audit-log=FORCE_PLUS_PERMANENT
   ```

   The top of the file should look similar to the following:

   ```
   [mysqld]
   log-bin=mysql-bin
   datadir=/var/lib/mysql
   ...
   ```

   b. Restart the MySQL server. Enter the following in a terminal window, and receive the results shown below:

   ```
   # service mysql restart
   Shutting down MySQL...                                      [  OK  ]
   Starting MySQL.                                             [  OK  ]
   ```

Practice 2-4: Invoking the `mysql` Client

Overview

In this practice, you invoke the `mysql` client using some of the common options.

Tasks

1. Display the `mysql` client version on the local host.
2. Display the `mysql` client options.
3. Start the `mysql` client with username and password options. Exit the client.
4. Start the `mysql` client with a login path of `admin`, and include a `SELECT` statement (that displays the current date and time in HTML format) as part of the command line execution.
5. Start the `mysql` client with username, port (set to 3306), and a tee file for this session. Allow the client to request the password. Enter the client password when requested. Exit the client and confirm the tee file creation.
6. Start the `mysql` client in Safe Updates mode.
7. Within the `mysql` client, display the session status.
8. List server-side help:
   a. Find the overall categories.
   b. List the commands related to account management.
   c. Show information related to setting passwords.
9. Display the available databases. Exit the client.

Solutions 2-4: Invoking the `mysql` Client

Tasks

1. Display the `mysql` client version on the local host. Enter the following in a terminal window, and receive the result shown below:

```
$ mysql -V
mysql  Ver 14.14 Distrib 5.6.10, for Linux (x86_64) using  EditLine wrapper
```

2. Display the `mysql` client options. Enter the following in a terminal window, and receive the result shown below:

```
$ mysql --help
mysql  Ver 14.14 Distrib 5.6.10, for Linux (x86_64) using  EditLine
wrapper
Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights
reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Usage: mysql [OPTIONS] [database]
  -?, --help          Display this help and exit.
  -I, --help          Synonym for -?
  --auto-rehash       Enable automatic rehashing. One doesn't need to
use
...
max-join-size                     1000000
secure-auth                       TRUE
show-warnings                     FALSE
plugin-dir                        (No default value)
default-auth                      (No default value)
histignore                        (No default value)
binary-mode                       FALSE
server-public-key-path            (No default value)
```

3. Start the `mysql` client with username and password options. Exit the client. Enter the following in a terminal window, and receive the result shown below:

```
$ mysql -uroot -poracle
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.6.10-enterprise-commercial-advanced-log MySQL
Enterprise Server - Advanced Edition (Commercial)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights
reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

mysql> EXIT
Bye
```

   – Note that you could also use the following syntax for the user and password options: `--user=root --password=oracle`. Note also that it is considered best practice to avoid using the password on the command line, and that you should use alternative methods (for example `--login-path`) where possible.

4. Start the `mysql` client with a login path of `admin`, and include a `SELECT` statement (that displays the current date and time in HTML format) as part of the command line execution. Enter the following in a terminal window, and receive the result shown below:

```
$ mysql --login-path=admin --html -e \
    "SELECT CURRENT_DATE(), CURRENT_TIME()"
<TABLE BORDER=1><TR><TH>CURRENT_DATE()</TH><TH>CURRENT_TIME()</TH>
</TR><TR><TD>2013-02-03</TD><TD>09:56:34</TD></TR></TABLE>[prompt]$
```

   – This returns the output for the date and time stamp query in HTML format, instead of the default tabular format. You can also redirect this output to a file by adding the following to the command: `> <path>/date_time.html`

5. Start the **`mysql`** client with username, port (set to 3306), and a tee file for this session. Allow the client to request the password. Enter the client password when requested. Exit the client and confirm the tee file creation.
   a. Enter the following in a terminal window, and receive the result shown below:

```
$ mysql -uroot -p -P 3306 --tee=tee_1.txt
Logging to file 'tee_1.txt'
Enter password: oracle
Welcome to the MySQL monitor.  Commands end with ; or \g.
...
mysql> EXIT
Bye
```

   b. View the tee file to make sure that it was created correctly by entering the following in a terminal window, and receive the result shown below:

```
$ cat tee_1.txt
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.6.10-enterprise-commercial-advanced-log MySQL
Enterprise Server - Advanced Edition (Commercial)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights
reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

mysql> EXIT
```

6. Start the `mysql` client in Safe Updates mode. Enter the following in a terminal window, and receive the result shown below:

```
$ mysql --login-path=admin --safe-updates
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
...
```

7. Within the `mysql` client, display the session status. Enter the following in a terminal window, and receive the result shown below:

```
mysql> STATUS
--------------
mysql  Ver 14.14 Distrib 5.6.10, for Linux (x86_64) using  EditLine
wrapper

Connection id:          5
Current database:
Current user:           root@localhost
SSL:                    Not in use
Current pager:          stdout
Using outfile:          ''
Using delimiter:        ;
Server version:         5.6.10-enterprise-commercial-advanced-log
MySQL Enterprise Server - Advanced Edition (Commercial)
Protocol version:       10
Connection:             Localhost via UNIX socket
Server characterset:    latin1
Db     characterset:    latin1
Client characterset:    utf8
Conn.  characterset:    utf8
UNIX socket:            /var/lib/mysql/mysql.sock
Uptime:                 2 hours 28 min 59 sec

Threads: 1  Questions: 18  Slow queries: 0  Opens: 71  Flush tables: 1
Open tables: 64  Queries per second avg: 0.000

Note that you are running in safe_update_mode:
UPDATEs and DELETEs that don't use a key in the WHERE clause are not
allowed.
(One can force an UPDATE/DELETE by adding LIMIT # at the end of the
command.)
SELECT has an automatic 'LIMIT 1000' if LIMIT is not used.
Max number of examined row combination in a join is set to: 1000000

--------------
```

8. List server-side help:

   a. Find the overall categories. Enter the following in a terminal window, and receive the result shown below:

```
mysql> HELP CONTENTS;
You asked for help about help category: "Contents"
For more information, type 'help <item>', where <item> is one of the
following categories:
   Account Management
   Administration
   Compound Statements
   Data Definition
   Data Manipulation
   Data Types
   Functions
   Functions and Modifiers for Use with GROUP BY
   Geographic Features
   Help Metadata
   Language Structure
   Plugins
   Procedures
   Storage Engines
   Table Maintenance
   Transactions
   User-Defined Functions
   Utility
```

   b. List the commands related to account management. Enter the following in a terminal window, and receive the result shown below:

```
mysql> HELP Account Management;
You asked for help about help category: "Account Management"
For more information, type 'help <item>', where <item> is one of the
following topics:
   ALTER USER
   CREATE USER
   DROP USER
   GRANT
   RENAME USER
   REVOKE
   SET PASSWORD
```

c. Show information related to setting passwords. Enter the following in a terminal window, and receive the result shown below:

```
mysql> HELP SET PASSWORD;
Name: 'SET PASSWORD'
Description:
Syntax:
SET PASSWORD [FOR user] =
    {
        PASSWORD('cleartext password')
      | OLD_PASSWORD('cleartext password')
      | 'encrypted password'
    }

The SET PASSWORD statement assigns a password to an existing MySQL
user account. When the read_only system variable is enabled, the SUPER
...
```

9. Display the available databases. Exit the client. Enter the following in a terminal window, and receive the result shown below:

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| world2             |
| world_innodb       |
+--------------------+
5 rows in set (0.00 sec)
mysql> EXIT
Bye
```

Practice 2-5: Invoking the `mysqladmin` Client

Overview

In this practice, you invoke the `mysqladmin` client, using some of the common options.

Tasks

1. Display the `mysqladmin` version running on the local host.
2. Display the `mysqladmin` client options.
3. Start the `mysqladmin` client with username, password, and list current variable settings.

Solutions 2-5: Invoking the `mysqladmin` Client

Tasks

1. Display the `mysqladmin` version running on the local host. Enter the following in a terminal window, and receive the result shown below:

```
$ mysqladmin -V
mysqladmin  Ver 8.42 Distrib 5.6.10, for Linux on x86_64
```

2. Display the `mysqladmin` client options. Enter the following in a terminal window, and receive the result shown below:

```
$ mysqladmin --help
mysqladmin  Ver 8.42 Distrib 5.6.10, for Linux on x86_64
Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights
reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Administration program for the mysqld daemon.
Usage: mysqladmin [OPTIONS] command command....
  --bind-address=name IP address to bind to.
  -c, --count=#       Number of iterations to make. This works with -i
                      (--sleep) only.
  --debug-check       Check memory and open file usage at exit.
  --debug-info        Print some debug info at exit.
...
  status                Gives a short status message from the server
  start-slave           Start slave
  stop-slave            Stop slave
  variables             Prints variables available
  version               Get version info from server
```

3. Start the `mysqladmin` client with username, password, and list current variable settings. Enter the following in a terminal window, and receive the result shown below:

```
$ mysqladmin -uroot -poracle variables
+-------------------------------+------------------+
| Variable_name                 | Value            |
+-------------------------------+------------------+
| audit_log_buffer_size         | 1048576          |
| audit_log_file                | audit.log        |
| audit_log_flush               | OFF              |
| audit_log_policy              | ALL              |
| audit_log_rotate_on_size      | 0                |
| audit_log_strategy            | ASYNCHRONOUS     |
| auto_increment_increment      | 1                |
| auto_increment_offset         | 1                |
| autocommit                    | ON               |
| automatic_sp_privileges       | ON               |
| back_log                      | 80               |
| basedir                       | /usr             |
...
| version_compile_machine       | x86_64           |
| version_compile_os            | Linux            |
| wait_timeout                  | 28800            |
+-------------------------------+------------------+
```

   – You can also redirect this output to a file by adding the following to the command:
     `> <path>/mysqladmin_variables.txt`

Practice 2-6: Performing System Administration Tasks with MySQL Workbench

Overview

In this practice, you install MySQL Workbench, and use it to create and populate a database, and work with its system administration features.

Duration

This practice should take approximately 25 minutes to complete.

Tasks

1. Install MySQL Workbench from its RPM file in `/stage/mysql`.
2. Launch MySQL Workbench by selecting Applications > Programming > MySQL Workbench.
3. At the Home screen, create a new connection to the `world_innodb` schema and name it `World`.
4. Use MySQL Workbench to launch a Query Database window.
5. Query the `world_innodb.Country` table with MySQL Workbench.
6. Query the `world_innodb.City` table with MySQL Workbench.
7. Query both of the preceding tables at the same time.
8. Use the Plug-ins menu to convert keywords in your query to uppercase.
9. View the list of columns for the `world_innodb.CountryLanguage` table.
10. Load the `sakila` database from scripts in your `/labs` directory, and refresh the Schemas list to confirm that it has loaded correctly.
11. Create a new server instance for the Server Administration module.
12. Connect to the new server instance.
13. On the Server Status page, view the various status graphs showing system load, memory in use, connections in use, traffic, query cache hit-rate, and key efficiency.
14. Select the Startup / Shutdown page and view the Database Server Status.
15. Select the "Status and System Variables" page.
16. On the Status Variables tab, select InnoDB/Stats. Note the InnoDB status variables that appear.
17. Select the Options File page.
18. View the configuration file location, and note that it is `/etc/my.cnf`.
19. Browse the other pages and tabs within the "Admin (Local server)" tab.
20. Close the "Admin (Local server)" tab.

Solutions 2-6: Performing System Administration Tasks with MySQL Workbench

1. Install MySQL Workbench from its RPM file in `/stage/mysql`.

   Execute the following commands in a terminal window and receive the results shown:

   ```
   $ su –
   Password: oracle
   # cd /stage/mysql
   # rpm -hi --nodeps mysql-workbench*.rpm
   ```

2. Launch MySQL Workbench by selecting Applications > Programming > MySQL Workbench.
3. At the Home screen, create a new connection to the `world_innodb` schema and name it `World`.
   a. Under SQL Development, click New Connection.
   b. In the Setup New Connection dialog box, note the default Connection Method: Standard (TCP/IP), the default Hostname: 127.0.0.1, the default Port: 3306, and the default Username: root.
   c. For Connection Name, enter `World`.
   d. For Default Schema, enter `world_innodb`.
   e. Select the Advanced tab.
   f. Note the default settings, but do not change any.
   g. Click Test Connection.
   h. Enter the password `oracle` and click OK.
   i. Click OK in the dialog box.
   j. Click OK to save the new connection.
4. Use MySQL Workbench to launch a Query Database window.
   a. Right-click the newly created World connection and select Query Database.
   b. Enter the password `oracle` and select the “Save password in keychain” option.
   c. Click OK.
   d. If this is the first time a keychain has been used on this system, you see another dialog box. Enter the password `oracle` once each in the Password field and the “Confirm password” field, and click Create.
   e. Note the following:
      – A new SQL Editor (World) tab appears in MySQL Workbench.
      – The SCHEMAS pane contains a list of user databases within the MySQL server, which currently consists of the databases `world2` and `world_innodb`.
      – An empty “Query 1” pane appears, within which you can enter some SQL statements.

5. Query the `world_innodb.Country` table with MySQL Workbench.

   a. Enter the following statement in the Query 1 pane:

   ```
   select * from Country;
   ```

   b. Press Ctrl + Return to execute the statement.

6. Query the `world_innodb.City` table with MySQL Workbench.

   a. Enter the following statement on a new line beneath the preceding statement in the Query 1 pane:

   ```
   select * from City;
   ```

   b. Press Ctrl + Return to execute the statement. Ensure that the cursor is flashing at the end of the query in the preceding step.

7. Query both of the preceding tables at the same time.

   Press Ctrl + Shift + Return to execute all statements in the editor window.

   Note: Two tabs appear in the results pane, one for each result set returned from statements in the query editor window.

8. Use the Plug-ins menu to convert keywords in your query to uppercase.

   In the menu, select Plugins > Utilities > “Make keywords in query uppercase.”

9. View the list of columns for the `world_innodb.CountryLanguage` table.

   In the SCHEMAS pane, browse to the “world_innodb” > Tables > CountryLanguage > Columns container, and browse the list of columns within the `CountryLanguage` table. When you click each column, note its definition in the Object Info pane.

10. Load the `sakila` database from scripts in your `/labs` directory, and refresh the Schemas list to confirm that it has loaded correctly.

    a. Select File > Open SQL Script.

    b. Browse to `/labs/sakila-db/sakila-schema.sql`.

    c. Press Ctrl + Shift + Return to execute the entire script.

    d. Select File > Open SQL Script.

    e. Browse to `/labs/sakila-db/sakila-data.sql`.

    f. In the Unknown File Encoding dialog box, click OK.

    g. Press Ctrl + Shift + Return to execute the entire script.

    h. In the Schemas pane, click the Refresh icon. Note that the `sakila` database appears.

11. Create a new server instance for the Server Administration module.

    a. Click the Home tab to return to the main screen. Under Server Administration, click New Server Instance.

    b. A Create New Server Instance Profile dialog box appears, showing the first of a series of pages similar to those covered in the “MySQL Clients” lesson.

    c. The first such page is labeled “Specify the Host Machine the Database Server is running on.” Click Next to accept the default, localhost.

    d. On the “Set the Database Connection Values” page, accept the default values and click Next. To test the connection details, you must enter the appropriate password.
    e. In the “Please enter password for the following service:” dialog box, enter the password `oracle`.
    f. On the “Testing the Database Connection” page, note the “Database connection tested successfully” message and click Next.
    g. On the “Specify the installation type for your target operating system” page, select the following:
        - Operating System: Linux (already selected)
        - MySQL Installation Type: Generic Linux (MySQL tar package)
    h. Click Next**.**
    i. The process checks the configuration settings, and brings you to a screen labeled “Testing Host Machine Settings.” Note the “Testing host machine settings is done” message, and click Next.
    j. In the “Review settings” dialog box, click Continue.
    k. On the “Create the Instance Profile” page, enter `Local server` for the service instance name, and click Finish.

12. Connect to the new server instance.
    a. On the main screen, note the new “Local server” entry under Server Administration. Double-click the “Local server” entry.
    b. If prompted, enter the password **`oracle`**.
    c. A new tab appears in Workbench, labeled “Admin (Local server),” with a number of pages on the left, divided into sections titled “Management,” “Configuration,” “Security,” and “Data Export/Restore.”
13. On the Server Status page, view the various status graphs showing system load, memory in use, connections in use, traffic, query cache hit-rate, and key efficiency.
14. Select the Startup / Shutdown page and view the Database Server Status.
15. Select the “Status and System Variables” page.
16. In the Status Variables tab, select InnoDB/Stats. Note the InnoDB status variables that appear.
17. Select the Options File page.
18. View the Configuration File location, and note that it is `/etc/my.cnf`.
19. Browse the other pages and tabs within the “Admin (Local server)” tab.
20. Close the “Admin (Local server)” tab.

Practices for Lesson 3: Obtaining Metadata

Chapter 3

Practice 3-1: Obtaining Metadata by Using `INFORMATION_SCHEMA`

Overview

In this practice, you query the `INFORMATION_SCHEMA` database for metadata.

Tasks

1. Obtain metadata about the `world_innodb` database (schema), by using a `SELECT` statement against the `SCHEMATA` table in the `INFORMATION_SCHEMA` database.
2. Select the name and engine of the tables in the `world_innodb` database (schema), by using a `SELECT` statement against the `TABLES` table in the `INFORMATION_SCHEMA` database.
3. List the number of tables, per storage engine, for each database (schema), by using a `SELECT` statement with a `GROUP BY` clause against the `TABLES` table in the `INFORMATION_SCHEMA` database.
4. List the data length of the `City` table in the `world_innodb` database, by using a `SELECT` statement against the `TABLES` table in the `INFORMATION_SCHEMA` database.
5. List the number of table columns in the `world_innodb` database that are using the `CHAR` or the `VARCHAR` data types, by using a `SELECT` statement with the `COUNT(*)` option against the `COLUMNS` table in the `INFORMATION_SCHEMA` database.

Solutions 3-1: Obtaining Metadata by Using `INFORMATION_SCHEMA`

Tasks

1. Obtain metadata about the `world_innodb` database (schema), by using a `SELECT` statement against the `SCHEMATA` table in the `INFORMATION_SCHEMA` database. Enter the following in a terminal window, and receive the result shown below:

```
$ mysql -uroot -poracle
...

mysql> SELECT *
    -> FROM INFORMATION_SCHEMA.SCHEMATA
    -> WHERE SCHEMA_NAME = 'world_innodb'\G
*************************** 1. row ***************************
              CATALOG_NAME: def
               SCHEMA_NAME: world_innodb
DEFAULT_CHARACTER_SET_NAME: latin1
    DEFAULT_COLLATION_NAME: latin1_swedish_ci
                  SQL_PATH: NULL
1 row in set (0.00 sec)
```

2. Select the name and engine of the tables in the `world_innodb` database (schema), by using a `SELECT` statement against the `TABLES` table in the `INFORMATION_SCHEMA` database. Enter the following in a terminal window, and receive the result shown below:

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

Database changed
mysql> SELECT TABLE_NAME, ENGINE
    -> FROM TABLES
    -> WHERE TABLE_SCHEMA = 'world_innodb';
+-----------------+--------+
| TABLE_NAME      | ENGINE |
+-----------------+--------+
| City            | InnoDB |
| Country         | InnoDB |
| CountryLanguage | InnoDB |
+-----------------+--------+
```

3. List the number of tables, per storage engine, for each database (schema), by using a `SELECT` statement with a `GROUP BY` clause against the `TABLES` table in the `INFORMATION_SCHEMA` database. Enter the following in a terminal window, and receive the result shown below:

```
mysql> SELECT TABLE_SCHEMA, ENGINE, COUNT(*)
    -> FROM TABLES
    -> GROUP BY TABLE_SCHEMA, ENGINE;
```

```
+--------------------+--------------------+----------+
| TABLE_SCHEMA       | ENGINE             | COUNT(*) |
+--------------------+--------------------+----------+
| information_schema | MEMORY             |       50 |
| information_schema | MyISAM             |       10 |
| mysql              | CSV                |        2 |
| mysql              | InnoDB             |        5 |
| mysql              | MyISAM             |       21 |
| performance_schema | PERFORMANCE_SCHEMA |       52 |
| sakila             | NULL               |        7 |
| sakila             | InnoDB             |       15 |
| sakila             | MyISAM             |        1 |
| test               | InnoDB             |        1 |
| world2             | InnoDB             |        3 |
| world_innodb       | InnoDB             |        3 |
+--------------------+--------------------+----------+
12 rows in set (0.01 sec)
```

4. List the data length of the `City` table in the `world_innodb` database, by using a `SELECT` statement against the `TABLES` table in the `INFORMATION_SCHEMA` database. Enter the following in a terminal window, and receive the result shown below:

```
mysql> SELECT TABLE_NAME, DATA_LENGTH
    -> FROM TABLES
    -> WHERE TABLE_SCHEMA = 'world_innodb'
    -> AND TABLE_NAME = 'City';
+------------+-------------+
| TABLE_NAME | DATA_LENGTH |
+------------+-------------+
| City       |      409600 |
+------------+-------------+
```

5. List the number of table columns in the `world_innodb` database that are using the `CHAR` or the `VARCHAR` data types, by using a `SELECT` statement with the `COUNT(*)` option against the `COLUMNS` table in the `INFORMATION_SCHEMA` database. Enter the following in a terminal window, and receive the result shown below:

```
mysql> SELECT DATA_TYPE, COUNT(*)
    -> FROM COLUMNS
    -> WHERE TABLE_SCHEMA = 'world_innodb'
    -> AND DATA_TYPE IN ('CHAR', 'VARCHAR')
    -> GROUP BY DATA_TYPE;
+-----------+----------+
| DATA_TYPE | COUNT(*) |
+-----------+----------+
| char      |       12 |
+-----------+----------+
```

   – There are no `VARCHAR` data types in the `world_innodb` database; therefore, it does not show up in the output.

Practice 3-2: Obtaining Metadata by Using `SHOW` and `DESCRIBE`

Overview

In this practice, you use the `SHOW` and `DESCRIBE` statements to obtain database metadata.

Duration

This practice should take approximately 10 minutes to complete.

Tasks

1. List all the available databases.
2. List all the databases that have the letter ‘o’ in their name.
3. List all the available `INFORMATION_SCHEMA` database tables.
4. Show detailed information about the columns in the `City` table from the `world_innodb` database.
5. Show the index information for the `City` table from the `world_innodb` database.
6. Show the structure of the `CountryLanguage` table from the `world_innodb` database.
7. List all character sets available.
8. List all collations available.
9. Exit the `mysql` client.

Solutions 3-2: Obtaining Metadata by Using SHOW and DESCRIBE

Tasks

1. List all the available databases. Using the `mysql` client that you opened in the previous practice, enter the following in a terminal window, and receive the result shown below:

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| test               |
| world2             |
| world_innodb       |
+--------------------+
7 rows in set (0.00 sec)
```

2. List all the databases that have the letter ‘o’ in their name. Enter the following in a terminal window, and receive the result shown below:

```
mysql> SHOW DATABASES LIKE '%o%';
+--------------------+
| Database (%o%)     |
+--------------------+
| information_schema |
| performance_schema |
| world2             |
| world_innodb       |
+--------------------+
4 rows in set (0.00 sec)
```

3. List all the available `INFORMATION_SCHEMA` database tables. Enter the following in a terminal window, and receive the results shown below:

   a. Change the database to `information_schema`:

```
mysql> USE information_schema;
Database changed
```

b. List the tables in the `information_schema` database:

```
mysql> SHOW TABLES;
+---------------------------------------+
| Tables_in_information_schema          |
+---------------------------------------+
| CHARACTER_SETS                        |
| COLLATIONS                            |
| COLLATION_CHARACTER_SET_APPLICABILITY |
| COLUMNS                               |
| COLUMN_PRIVILEGES                     |
...
| INNODB_BUFFER_POOL_STATS              |
| INNODB_SYS_COLUMNS                    |
| INNODB_SYS_FOREIGN                    |
+---------------------------------------+
60 rows in set (0.00 sec)
```

4. Show detailed information about the columns in the `City` table from the `world_innodb` database. Enter the following in a terminal window, and receive the result shown below:

a. Change the database to `world_innodb`:

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

Database changed
```

b. List all the column information for the `City` table:

```
mysql> SHOW FULL COLUMNS FROM City\G
*************************** 1. row ***************************
     Field: ID
      Type: int(11)
 Collation: NULL
      Null: NO
       Key: PRI
   Default: NULL
     Extra: auto_increment
Privileges: select,insert,update,references
   Comment:
*************************** 2. row ***************************
     Field: Name
      Type: char(35)
 Collation: latin1_swedish_ci
      Null: NO
       Key:
   Default:
     Extra:
Privileges: select,insert,update,references
   Comment:
*************************** 3. row ***************************
     Field: CountryCode
      Type: char(3)
 Collation: latin1_swedish_ci
```

```
      Null: NO
       Key: MUL
   Default:
     Extra:
Privileges: select,insert,update,references
   Comment:
*************************** 4. row ***************************
     Field: District
      Type: char(20)
 Collation: latin1_swedish_ci
      Null: NO
       Key:
   Default:
     Extra:
Privileges: select,insert,update,references
   Comment:
*************************** 5. row ***************************
     Field: Population
      Type: int(11)
 Collation: NULL
      Null: NO
       Key:
   Default: 0
     Extra:
Privileges: select,insert,update,references
   Comment:
5 rows in set (0.00 sec)
```

5. Show the index information for the `City` table from the `world_innodb` database. Enter the following in a terminal window, and receive the result shown below:

```
mysql> SHOW INDEX FROM City\G
*************************** 1. row ***************************
        Table: City
   Non_unique: 0
     Key_name: PRIMARY
 Seq_in_index: 1
  Column_name: ID
    Collation: A
  Cardinality: 4188
     Sub_part: NULL
       Packed: NULL
         Null:
   Index_type: BTREE
      Comment:
Index_comment:
*************************** 2. row ***************************
        Table: City
   Non_unique: 1
     Key_name: CountryCode
 Seq_in_index: 1
  Column_name: CountryCode
    Collation: A
  Cardinality: 465
     Sub_part: NULL
       Packed: NULL
         Null:
   Index_type: BTREE
```

```
      Comment:
Index_comment:
2 rows in set (0.01 sec)
```

Note: The `INFORMATION_SCHEMA` tables `TABLE_CONSTRAINTS` and `STATISTICS` also contain index metadata.

6. Show the structure of the `CountryLanguage` table from the `world_innodb` database. Enter the following in a terminal window, and receive the result shown below:

```
mysql> DESCRIBE CountryLanguage;
+-------------+---------------+------+-----+---------+-------+
| Field       | Type          | Null | Key | Default | Extra |
+-------------+---------------+------+-----+---------+-------+
| CountryCode | char(3)       | NO   | PRI |         |       |
| Language    | char(30)      | NO   | PRI |         |       |
| IsOfficial  | enum('T','F') | NO   |     | F       |       |
| Percentage  | float(4,1)    | NO   |     | 0.0     |       |
+-------------+---------------+------+-----+---------+-------+
4 rows in set (0.03 sec)
```

7. List all character sets available. Enter the following in a terminal window, and receive the result shown below:

```
mysql> SHOW CHARACTER SET;
+---------+-------------------------+---------------------+--------+
| Charset | Description             | Default collation   | Maxlen |
+---------+-------------------------+---------------------+--------+
| big5    | Big5 Traditional Chinese | big5_chinese_ci     |      2 |
| dec8    | DEC West European       | dec8_swedish_ci     |      1 |
| cp850   | DOS West European       | cp850_general_ci    |      1 |
| hp8     | HP West European        | hp8_english_ci      |      1 |
| koi8r   | KOI8-R Relcom Russian   | koi8r_general_ci    |      1 |
...
| eucjpms | UJIS for Windows Japanese | eucjpms_japanese_ci |      3 |
+---------+-------------------------+---------------------+--------+
40 rows in set (0.02 sec)
```

   – A full list of character sets on the current system is displayed.

8. List all collations available. Enter the following in a terminal window, and receive the result shown below:

```
mysql> SHOW COLLATION;
+--------------------+---------+-----+---------+----------+---------+
| Collation          | Charset | Id  | Default | Compiled | Sortlen |
+--------------------+---------+-----+---------+----------+---------+
| big5_chinese_ci    | big5    |   1 | Yes     | Yes      |       1 |
| big5_bin           | big5    |  84 |         | Yes      |       1 |
| dec8_swedish_ci    | dec8    |   3 | Yes     | Yes      |       1 |
| dec8_bin           | dec8    |  69 |         | Yes      |       1 |
| cp850_general_ci   | cp850   |   4 | Yes     | Yes      |       1 |
| cp850_bin          | cp850   |  80 |         | Yes      |       1 |
...
| eucjpms_bin        | eucjpms |  98 |         | Yes      |       1 |
+--------------------+---------+-----+---------+----------+---------+
219 rows in set (0.00 sec)
```

   – A full list of collations on the current system is displayed.

9. Exit the `mysql` client. Enter the following in a terminal window, and receive the result shown below:

```
mysql> EXIT
Bye
```

Practice 3-3: Obtaining Metadata by Using `mysqlshow`

Overview

In this practice, you use the `mysqlshow` client to obtain database metadata.

Duration

This practice should take approximately 5 minutes to complete.

Tasks

1. Using the `mysqlshow` application from the command line, list all the available databases.
2. List all the tables in the `world_innodb` database, from the command line.
3. Show the structure of the `CountryLanguage` table in the `world_innodb` database, from the command line.

Solutions 3-3: Obtaining Metadata by Using `mysqlshow`

Tasks

1. Using the `mysqlshow` application from the command line, list all the available databases. Enter the following in a terminal window, and receive the result shown below:

```
$ mysqlshow -uroot -poracle
Warning: Using a password on the command line interface can be
insecure.
+--------------------+
|     Databases      |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| test               |
| world2             |
| world_innodb       |
+--------------------+
```

2. List all the tables in the `world_innodb` database, from the command line. Enter the following in a terminal window, and receive the result shown below:

```
$ mysqlshow world_innodb -uroot -poracle
+-----------------+
|     Tables      |
+-----------------+
| City            |
| Country         |
| CountryLanguage |
+-----------------+
```

3. Show the structure of the `CountryLanguage` table in the `world_innodb` database, from the command line. Enter the following in a terminal window, and receive the result shown below:

```
# mysqlshow world_innodb CountryLanguage -uroot -oracle
Database: world_innodb  Table: CountryLanguage
+-------------+--------------+-------------------+------+-----+------
---+-------+---------------------------------+---------+
| Field       | Type         | Collation         | Null | Key |
Default | Extra | Privileges                      | Comment |
+-------------+--------------+-------------------+------+-----+------
---+-------+---------------------------------+---------+
| CountryCode | char(3)      | latin1_swedish_ci | NO   | PRI |
|       | select,insert,update,references |         |
| Language    | char(30)     | latin1_swedish_ci | NO   | PRI |
|       | select,insert,update,references |         |
| IsOfficial  | enum('T','F') | latin1_swedish_ci | NO   |     | F
|       | select,insert,update,references |         |
| Percentage  | float(4,1)   |                   | NO   |     | 0.0
|       | select,insert,update,references |         |
+-------------+--------------+-------------------+------+-----+------
---+-------+---------------------------------+---------+
```

Practices for Lesson 4: Transaction, Locking and Innodb Storage Engine

Practice 4-1: Quiz – Transactions and Locking

Overview

In this practice, you answer questions pertaining to transactions and locking using the MySQL server.

Quiz Questions

Choose the best answer from those provided for each multiple choice or True/False question.

1. A transaction is a collection of data manipulation execution steps that are treated as a single unit of work.
   a. True
   b. False
2. When the term *ACID compliant* is used, it refers to the fact that the transactions in MySQL are ___________, consistent, __________, and durable.
   a. atomic, isolated
   b. automatic, isolated
   c. atomic, independent
   d. always on, intended for transactions
3. If `AUTOCOMMIT` is not enabled, ___________________. The transaction is then explicitly committed or rolled back using the `COMMIT` and `ROLLBACK` statements respectively. After the transaction terminates, a new transaction is implicitly started.
   a. transactions include only a single statement at a time
   b. the server asks you for the value of the `AUTOCOMMIT` setting before proceeding
   c. transactions span multiple statements by default
4. The four transaction isolation levels include `READ UNCOMMITTED`, `READ COMMITTED`, `REPEATABLE`, and `ISOLATED`.
   a. True
   b. False
5. Three types of inconsistencies can occur when two simultaneous transactions access the same table: ____________________, non-repeatable read, and/or phantom row/read.
   a. repeatable read
   b. “dirty” read
   c. “polluted” read
6. A `SERIALIZABLE` level allows non-repeatable reads.
   a. True
   b. False

7. Locking is a mechanism to prevent concurrency problems, which is managed by the MySQL server and locks for one client to _____________.
    a. isolate the master client
    b. remove any non-MySQL clients
    c. restrict other clients
8. The two locking modifiers that InnoDB supports are `LOCK IN SHARE MODE` and `FOR UPDATE`.
    a. True
    b. False
9. The InnoDB locking modifiers `LOCK IN SHARE MODE` and `FOR UPDATE` lock ______________ to prevent problems that might occur when there is simultaneous access of table data.
    a. lock(s)
    b. table(s)
    c. read(s)
    d. row(s)

Solutions 4-1: Quiz – Transactions and Locking

Quiz Solutions

1. **a.** True.
2. **a.** ACID = atomic, consistent, isolated, durable
3. **c.** span multiple statements by default
4. **b.** False. The last two are `REPEATABLE READ` and `SERIALIZABLE`.
5. **b.** “dirty” read.
6. **b.** False. `SERIALIZABLE` completely isolates the effects of one transaction from others. It is similar to `REPEATABLE READ` with the additional restriction that rows selected by one transaction cannot be changed by another until the first transaction finishes.
7. **c.** restrict other clients. MySQL does this using either a shared or an exclusive lock.
8. **a.** True.
9. **d.** row(s).

Practice 4-2: Using Transaction Control Statements

Overview

In this practice, you use the transaction control statements to start, manipulate, and commit SQL transactions.

Tasks

1. Determine whether any transactional storage engines are available on your system, and determine which one is the default engine, by using the `SHOW ENGINES` command.
2. Enable `AUTOCOMMIT` by using the `SET AUTOCOMMIT` statement.
3. Prepare to use the `world_innodb` database, and confirm that the `City` table uses the transactional storage engine, InnoDB.
4. Explicitly start a new transaction by using the `START TRANSACTION` statement.
5. Delete a row.
6. Roll back the open transaction by using the `ROLLBACK` statement.
7. Start another new transaction.
8. Delete the same row again.
9. Stop and commit the transaction by using the `COMMIT` statement.
10. Confirm that the row is now gone.
11. Try to roll back the committed transaction.

Solutions 4-2: Using Transaction Control Statements

Tasks

1. Determine whether any transactional storage engines are available on your system, and determine which one is the default engine, by using the `SHOW ENGINES` command. Enter the following in a terminal window, and receive the result shown below:

```
$ mysql -uroot -poracle
...

mysql> SHOW ENGINES\G
...
*************************** 5. row ***************************
      Engine: MyISAM
     Support: YES
     Comment: MyISAM storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
...
*************************** 8. row ***************************
      Engine: InnoDB
     Support: DEFAULT
     Comment: Supports transactions, row-level locking, and foreign
keys
Transactions: YES
          XA: YES
  Savepoints: YES
...
```

   – The order of the list might be different on your system. Regardless of the order, the InnoDB storage engine is indicated as the default, and includes transactions.

2. Enable `AUTOCOMMIT` by using the `SET AUTOCOMMIT` statement. Enter the following in a terminal window, and receive the results shown below:

   a. Set the `AUTOCOMMIT` mode to on:

```
mysql> SET AUTOCOMMIT = 1;
Query OK, 0 rows affected (0.06 sec)
```

   b. Check the value to confirm:

```
mysql> SELECT @@AUTOCOMMIT;
+--------------+
| @@AUTOCOMMIT |
+--------------+
|            1 |
+--------------+
1 row in set (0.02 sec)
```

3. Prepare to use the `world_innodb` database, and confirm that the `City` table uses the transactional storage engine, InnoDB. Enter the following in a terminal window, and receive the results shown below:

    a. Change to the `world_innodb` database:

```
mysql> USE world_innodb
...
Database changed
```

    b. Display the `City` table structure, and confirm that the value of `ENGINE` is InnoDB:

```
mysql> SHOW CREATE TABLE City\G
*************************** 1. row ***************************
       Table: City
Create Table: CREATE TABLE `City` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `Name` char(35) NOT NULL DEFAULT '',
  `CountryCode` char(3) NOT NULL DEFAULT '',
  `District` char(20) NOT NULL DEFAULT '',
  `Population` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`ID`),
  KEY `CountryCode` (`CountryCode`),
  CONSTRAINT `city_ibfk_1` FOREIGN KEY (`CountryCode`) REFERENCES
`Country` (`Code`)
) ENGINE=InnoDB AUTO_INCREMENT=4080 DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```

4. Explicitly start a new transaction by using the `START TRANSACTION` statement. Enter the following in a terminal window, and receive the result shown below:

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

5. Delete a row. Enter the following in a terminal window, and receive the results shown below:

    a. Select the row that contains the city named “Manta”:

```
mysql> SELECT * FROM City WHERE Name = 'Manta';
+-----+-------+-------------+----------+------------+
| ID  | Name  | CountryCode | District | Population |
+-----+-------+-------------+----------+------------+
| 600 | Manta | ECU         | Manabí   |     164739 |
+-----+-------+-------------+----------+------------+
1 row in set (0.06 sec)
```

    b. Delete the row that contains the city named “Manta”:

```
mysql> DELETE FROM City WHERE Name = 'Manta';
Query OK, 1 row affected (0.02 sec)
```

c. Select the row that contains the city named “Manta”:

```
mysql> SELECT * FROM City WHERE Name = 'Manta';
Empty set (0.00 sec)
```

– The first SELECT confirms the existence of the city of Manta. The delete returns OK, indicating that one row was affected. The second SELECT shows that the row is now empty.

6. Roll back the open transaction by using the ROLLBACK statement. Enter the following in a terminal window, and receive the results shown below:

   a. Roll back the open transaction:

```
mysql> ROLLBACK;
Query OK, 0 rows affected (0.05 sec)
```

   b. Confirm that the row is now back to its original state:

```
mysql> SELECT * FROM City WHERE Name = 'Manta';
+-----+-------+-------------+----------+------------+
| ID  | Name  | CountryCode | District | Population |
+-----+-------+-------------+----------+------------+
| 600 | Manta | ECU         | Manabí   |     164739 |
+-----+-------+-------------+----------+------------+
1 row in set (0.06 sec)
```

   – The result shows that the row is now back.

7. Start another new transaction. Enter the following in a terminal window, and receive the result shown below:

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

8. Delete the same row again. Enter the following in a terminal window, and receive the results shown below:

```
mysql> DELETE FROM City WHERE Name = 'Manta';
Query OK, 1 row affected (0.02 sec)
```

9. Stop and commit the transaction by using the COMMIT statement. Enter the following in a terminal window, and receive the result shown below:

```
mysql> COMMIT;
Query OK, 0 rows affected (0.04 sec)
```

10. Confirm that the row is now gone. Enter the following in a terminal window, and receive the result shown below:

```
mysql> SELECT * FROM City WHERE Name = 'Manta';
Empty set (0.00 sec)
```

   – The final SELECT shows that the row does not exist anymore, and that this DELETE cannot be undone.

11. Try to roll back the committed transaction. Enter the following in a terminal window, and receive the results shown below:

    a. Roll back the open transaction:

    ```
    mysql> ROLLBACK;
    Query OK, 0 rows affected (0.05 sec)
    ```

    b. Confirm the current status of the row:

    ```
    mysql> SELECT * FROM City WHERE Name = 'Manta';
    Empty set (0.00 sec)
    ```

    – The result shows that the row is still empty because the transaction was committed before this rollback took place.

Practice 4-3: Additional Practice with Transactions and Locking

Overview

In this practice, you use transaction and locking statements.

Tasks

1. From a `mysql` client session, check the current isolation level.
2. Using the `PROMPT` statement, change the prompt to `t1` in the `mysql` session opened in the preceding step, to differentiate between this and future client sessions.
3. Start a new transaction in the `t1` `mysql` session.
4. In the `t1` session, select all rows from the `City` table where the ID > 4070.
5. In a separate terminal window, open a second `mysql` session. Change the prompt to `t2` in the `mysql` session.
6. Start a transaction in the `t2` `mysql` session.
7. In the `t2` session, from the `world_innodb` database, select all rows from the `City` table where the ID > 4070.
8. In the `t2` session, insert a new row to the `City` table. Confirm that the new row has been added. Enter the following in the `t2` terminal window:

```
t2> INSERT INTO City (Name, CountryCode) VALUES ('New City', 'ATA');
```

9. In the `t1` session, select the city with ID > 4070 again. Do you now see the row?
10. In the `t2` session, commit the transaction.
11. In the `t1` session, commit the transaction. Select the city with ID > 4070 again.
12. Start a new transaction in the `t1` `mysql` session.
13. In the `t1` session, remove the row that was inserted to the `City` table. Confirm the removal of the row.

Isolation level change:

14. In the `t2` session, change the isolation level to `READ UNCOMMITTED` by using the `SET SESSION tx_isolation` command, and confirm the change.
15. Start a new transaction in the `t2` session. Select the city with ID > 4070. What is the problem with this read?
16. In the `t1` session, cancel the transaction by using the `ROLLBACK` statement. Select the city with ID > 4070 again.
17. In the `t2` session, select the city with ID > 4070. Are the results the same as in the same query to the `t1` session in the previous step?
18. In the `t2` session, cancel any transaction that was registered by this session.

19. In preparation for the next practice, exit the both `mysql` sessions, as well as both terminal windows.

Solutions 4-3: Additional Practice with Transactions and Locking

Tasks

1. From a `mysql` client session, check the current Isolation level. Enter the following in a terminal window, and receive the results shown below:

```
$ mysql -uroot -poracle
...

mysql> SELECT @@global.tx_isolation;
+-----------------------+
| @@global.tx_isolation |
+-----------------------+
| REPEATABLE-READ       |
+-----------------------+
1 row in set (0.00 sec)
```

2. Change the prompt to `t1` in the `mysql` session opened in the preceding step, to differentiate between this and future client sessions. Enter the following in a terminal window, and receive the result shown below:

```
mysql> PROMPT t1> ;
PROMPT set to 't1> '
t1>
```

3. Start a new transaction in the `t1` `mysql` session. Enter the following in the `t1` terminal window, and receive the result shown below:

   Start the new transaction:

```
t1> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)
```

4. In the `t1` session, from the `world_innodb` database, select all rows from the `City` table where the ID > 4070. Enter the following in the `t1` terminal window, and receive the results shown below:

```
t1> USE world_innodb
...
Database changed
t1> SELECT * FROM City WHERE ID > 4070;
+------+--------------+-------------+------------+------------+
| ID   | Name         | CountryCode | District   | Population |
+------+--------------+-------------+------------+------------+
| 4071 | Mount Darwin | ZWE         | Harare     |     164362 |
| 4072 | Mutare       | ZWE         | Manicaland |     131367 |
| 4073 | Gweru        | ZWE         | Midlands   |     128037 |
| 4074 | Gaza         | PSE         | Gaza       |     353632 |
| 4075 | Khan Yunis   | PSE         | Khan Yunis |     123175 |
| 4076 | Hebron       | PSE         | Hebron     |     119401 |
| 4077 | Jabaliya     | PSE         | North Gaza |     113901 |
| 4078 | Nablus       | PSE         | Nablus     |     100231 |
| 4079 | Rafah        | PSE         | Rafah      |      92020 |
+------+--------------+-------------+------------+------------+
9 rows in set (0.00 sec)
```

5. In a separate terminal window, open a second `mysql` session. Change the prompt to `t2` in the `mysql` session. Enter the following in a terminal window, and receive the results shown below:

```
$ mysql -uroot -poracle
...
mysql> PROMPT t2> ;
PROMPT set to 't2> '
t2>
```

6. Start a transaction in the `t2` `mysql` session. Enter the following in the `t2` terminal window, and receive the result shown below:

```
t2> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)
```

7. In the `t2` session, from the `world_innodb` database, select all rows from the `City` table where the ID > 4070. Enter the following in the `t2` terminal window, and receive the result shown below:

```
t2> USE world_innodb
Database changed

t2> SELECT * FROM City WHERE ID > 4070;
+------+-------------+-------------+------------+------------+
| ID   | Name        | CountryCode | District   | Population |
+------+-------------+-------------+------------+------------+
| 4071 | Mount Darwin | ZWE         | Harare     |     164362 |
| 4072 | Mutare      | ZWE         | Manicaland |     131367 |
| 4073 | Gweru       | ZWE         | MIDlands   |     128037 |
| 4074 | Gaza        | PSE         | Gaza       |     353632 |
| 4075 | Khan Yunis  | PSE         | Khan Yunis |     123175 |
| 4076 | Hebron      | PSE         | Hebron     |     119401 |
| 4077 | Jabaliya    | PSE         | North Gaza |     113901 |
| 4078 | Nablus      | PSE         | Nablus     |     100231 |
| 4079 | Rafah       | PSE         | Rafah      |      92020 |
+------+-------------+-------------+------------+------------+
9 rows in set (0.00 sec)
```

8. In the `t2` session, insert a new row to the `City` table. Confirm that the new row has been added.

   a. Enter the following in the `t2` terminal window, and receive the results shown below:

```
t2> INSERT INTO City (Name, CountryCode) VALUES ('New City', 'ATA');
Query OK, 1 row affected (0.00 sec)
```

   − The insert completed and shows that it affected one row.

b. Select the cities with ID > 4070:

```
t2> SELECT * FROM City WHERE ID > 4070;
+------+-------------+------------+-----------+------------+
| ID   | Name        | CountryCode | District  | Population |
+------+-------------+------------+-----------+------------+
| 4071 | Mount Darwin | ZWE        | Harare    |     164362 |
| 4072 | Mutare      | ZWE        | Manicaland |     131367 |
| 4073 | Gweru       | ZWE        | MIDlands  |     128037 |
| 4074 | Gaza        | PSE        | Gaza      |     353632 |
| 4075 | Khan Yunis  | PSE        | Khan Yunis |     123175 |
| 4076 | Hebron      | PSE        | Hebron    |     119401 |
| 4077 | Jabaliya    | PSE        | North Gaza |     113901 |
| 4078 | Nablus      | PSE        | Nablus    |     100231 |
| 4079 | Rafah       | PSE        | Rafah     |      92020 |
| 4080 | New City    | ATA        |           |          0 |
+------+-------------+------------+-----------+------------+
10 rows in set (0.00 sec)
```

9. In the t1 session, select the city with ID > 4070 again.

a. Enter the following in the t1 terminal window, and receive the result shown below:

```
t1> SELECT * FROM City WHERE ID > 4070;
+------+-------------+------------+-----------+------------+
| ID   | Name        | CountryCode | District  | Population |
+------+-------------+------------+-----------+------------+
...
| 4079 | Rafah       | PSE        | Rafah     |      92020 |
+------+-------------+------------+-----------+------------+
9 rows in set (0.00 sec)
```

b. Do you now see the row? The row is still not there. This occurs due to the REPEATABLE READ isolation level.

10. In the t2 session, commit the transaction. Enter the following in the t2 terminal window, and receive the result shown below:

```
t2> COMMIT;
Query OK, 0 rows affected (0.01 sec)
```

11. In the t1 session, commit the transaction. Select the city with ID > 4070 again. Enter the following in the t1 terminal window, and receive the results shown below:

```
t1> COMMIT;
Query OK, 0 rows affected (0.01 sec)

t1> SELECT * FROM City WHERE ID > 4070;
+------+-------------+------------+-----------+------------+
| ID   | Name        | CountryCode | District  | Population |
+------+-------------+------------+-----------+------------+
...
| 4080 | New City    | ATA        |           |          0 |
+------+-------------+------------+-----------+------------+
10 rows in set (0.00 sec)
```

– Note that the row that was inserted in the t2 session is now available to the t1 session due to the transactions being committed in each session.

12. Start a new transaction in the `t1` `mysql` session. Enter the following in the `t1` terminal window, and receive the result shown below:

```
t1> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)
```

13. In the `t1` session, remove the row that was inserted to the `City` table. Confirm the removal of the row. Enter the following in the `t1` terminal window, and receive the results shown below:

```
t1> DELETE FROM City WHERE ID = 4080;
Query OK, 1 row affected (0.00 sec)

t1> SELECT * FROM City WHERE ID > 4070;
+------+-------------+------------+-----------+------------+
| ID   | Name        | CountryCode | District  | Population |
+------+-------------+------------+-----------+------------+
...
| 4079 | Rafah       | PSE        | Rafah     |      92020 |
+------+-------------+------------+-----------+------------+
9 rows in set (0.00 sec)
```

   – The row has been removed.

Isolation level change:

14. In the `t2` session, change the isolation level to `READ UNCOMMITTED` by using the `SET SESSION tx_isolation` command, and confirm the change. Enter the following in the `t1` terminal window, and receive the results shown below:

```
t2> SET SESSION tx_isolation = 'READ-UNCOMMITTED';
Query OK, 0 rows affected (0.00 sec)

t2> SELECT @@tx_isolation;
+----------------------+
| @@ tx_isolation      |
+----------------------+
| READ-UNCOMMITTED     |
+----------------------+
1 row in set (0.00 sec)
```

15. Start a new transaction in the `t2` session. Select the city with ID > 4070. Enter the following in the `t2` terminal window, and receive the results shown below:

```
t2> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

t2> SELECT * FROM City WHERE ID > 4070;
+------+-------------+------------+-----------+------------+
| ID   | Name        | CountryCode | District  | Population |
+------+-------------+------------+-----------+------------+
...
| 4079 | Rafah       | PSE        | Rafah     |      92020 |
+------+-------------+------------+-----------+------------+
9 rows in set (0.00 sec)
```

   a. What is the problem with this read? It is a dirty read. Due to the `READ UNCOMMITTED` isolation level, the `SELECT` result reflects the uncommitted transaction in the `t1` session.

16. In the `t1` session, cancel the transaction by using the `ROLLBACK` statement. Select the city with ID > 4070 again. Enter the following in the `t1` terminal window, and receive the results shown below:

```
t1> ROLLBACK;
Query OK, 0 rows affected (0.01 sec)

t1> SELECT * FROM City WHERE ID > 4070;
+------+-------------+------------+-----------+------------+
| ID   | Name        | CountryCode | District  | Population |
+------+-------------+------------+-----------+------------+
...
| 4080 | New City    | ATA        |           |          0 |
+------+-------------+------------+-----------+------------+
10 rows in set (0.00 sec)
```

   – The `DELETE` transaction was rolled back. Therefore, the row that was inserted earlier still exists.

17. In the `t2` session, select the city with ID > 4070. Enter the following in the `t2` terminal window, and receive the results shown below:

```
t2> SELECT * FROM City WHERE ID > 4070;
+------+-------------+------------+-----------+------------+
| ID   | Name        | CountryCode | District  | Population |
+------+-------------+------------+-----------+------------+
...
| 4080 | New City    | ATA        |           |          0 |
+------+-------------+------------+-----------+------------+
10 rows in set (0.00 sec)
```

Are the results the same as in the same query to the `t1` session in the previous step? Yes, the table contents remain as they were prior to the `DELETE`.

18. In the `t2` session, cancel any transaction that was registered by this session. Enter the following in the `t2` terminal window, and receive the results shown below:

    ```
    t2> ROLLBACK;
    Query OK, 0 rows affected (0.01 sec)
    ```

19. In preparation for the next practice, exit the both `mysql` sessions, as well as both terminal windows:

    ```
    t1> EXIT
    Bye
    $ exit

    t2> EXIT
    Bye
    $ exit
    ```

Practice 4-4: Quiz – InnoDB Storage Engine

Overview

In this practice, you answer questions about the InnoDB storage engine.

Quiz Questions

Choose the best answer from those provided for each multiple choice or True/False question.

1. How can the InnoDB storage engine be disabled?
   a. Only when compiling MySQL from source, by using the `--without-innodb` option.
   b. By starting the server with the `--skip-innodb` option.
   c. By issuing the statement `SET GLOBAL have_innodb = 0`. You need the **`SUPER`** privilege to do this.
   d. Storage engines cannot be disabled.
2. In the default InnoDB configuration, the number of `.ibd` files is equal to the number of InnoDB tables.
   a. True
   b. False
3. ________________ is possible because InnoDB does not acquire locks during a transaction until they are needed.
   a. Row-level locking
   b. Deadlock
   c. Phantom read
   d. All of the above.
4. You can remove a data file from the system tablespace to decrease its size.
   a. True
   b. False
5. InnoDB provides referential integrity by using the __________ and __________ keys to enforce table relationships.
   a. candidate and unique
   b. primary and unique
   c. primary and foreign
   d. unique and foreign

Solutions 4-4: Quiz – InnoDB Storage Engine

Quiz Solutions

1. **b.** By starting the server with the `--skip-innodb` option.

   Note: When you use the `--skip-innodb` option, set the `default-storage-engine` option to an engine other than InnoDB. InnoDB is the default engine starting from MySQL Server version 5.5. If no `default-storage-engine` is configured, the server does not start when the
   `--skip-innodb` option is used

2. **a.** True
3. **b.** Deadlock
4. **b.** False
5. **c.** primary and foreign

Practice 4-5: Setting and Confirming InnoDB Settings

Overview

In this practice, use a variety of methods to set and display InnoDB settings for the current MySQL session.

Tasks

1. Edit the **[**mysqld**]** section of the current configuration file (my.cnf, from the preceding practices) to set the InnoDB buffer pool to approximately 50% of your physical RAM and create one buffer pool instance per GB of RAM in the buffer pool.
2. Stop and restart the MySQL server to implement the config file changes.
3. Start the mysql client in a new terminal window, and use the SELECT statement to confirm the default storage engine for the current session.
4. Confirm that the INFORMATION_SCHEMA database contains tables related to InnoDB.
5. List the settings for all InnoDB-specific server variables.
   a. Is the multiple tablespace (file-per-table) setting ON or OFF?
   b. What is the auto-extend increment setting?
   c. What is the InnoDB data file path?
   d. Is there any purge lag?
   e. What is the buffer pool size?
   f. How many buffer pool instances are there?
6. Edit the existing config file (/etc/my.cnf) to change the auto-extend increment to 128 MB.
7. Confirm the changes made to the config file by listing the settings for all InnoDB-specific server variables again. Compare the results to the previous list.

   What is the auto-extend increment setting?
8. Create a new table called CityLanguage in the world_innodb database using the MEMORY storage engine. Include definitions for City, Country, CountryCode, and Language columns.
   a. Enter the following CREATE TABLE statement at a mysql prompt, and receive the results shown below:

```
mysql> CREATE TABLE CityLanguage (
    -> City CHAR(35),
    -> Country CHAR(35),
    -> CountryCode CHAR(3),
    -> Language CHAR(30)
    -> ) ENGINE=MEMORY;
```

   b. Confirm the storage engine setting.

9. Modify the new `CityLanguage` table to use the InnoDB storage engine, and confirm the change.

Solutions 4-5: Setting and Confirming InnoDB Settings

Tasks

1. Edit the **[mysqld]** section of the current configuration file (my.cnf, from the preceding practices) to set the InnoDB buffer pool to approximately 50% of your physical RAM and create one buffer pool instance per GB of RAM in the buffer pool.

   a. To find the total physical memory, enter the following in a terminal window:

   ```
   $ su -
   Password: oracle
   # head -1 /proc/meminfo
   MemTotal:        4059764 kB
   ```

   The total memory shown in the preceding output is approximately 4 GB. If your server has 4 GB RAM, set the buffer pool size to 2 GB and the number of buffer pool instances to two in the following steps. Choose different values as appropriate. For example, if your server has 8 GB RAM, choose a buffer pool size of 4 GB, and four buffer pool instances.

   b. Open the my.cnf configuration file in gedit (or your preferred text editor).

   ```
   # gedit /etc/my.cnf
   ```

   c. Add the following lines to the [mysqld] section. Choose values appropriate to your system as found in step 1a:

   ```
   [mysqld]
   innodb_buffer_pool_size=2GB
   innodb_buffer_pool_instances=2
   ...
   ```

   d. Save and close the config file.

2. Stop and restart the MySQL server to implement the config file changes. Enter the following in a terminal window, and receive the results shown below:

   ```
   # service mysql restart
   Shutting down MySQL......                  [ OK ]
   Starting MySQL......                       [ OK ]
   ```

3. Start the mysql client in a new terminal window, and use a SELECT statement to confirm the default storage engine for the current session. Enter the following in a terminal window, and receive the results shown below:

   ```
   $ mysql -uroot -poracle
   ...
   mysql> SELECT @@default_storage_engine;
   +--------------------------+
   | @@default_storage_engine |
   +--------------------------+
   | InnoDB                   |
   +--------------------------+
   1 row in set (0.00 sec)
   ```

4. Confirm that the `INFORMATION_SCHEMA` database contains tables related to InnoDB. Enter the following in a terminal window, and receive the result shown below:

```
mysql> USE INFORMATION_SCHEMA
...
Database changed

mysql> SHOW TABLES LIKE 'INNODB%';
+----------------------------------------+
| Tables_in_information_schema (INNODB%) |
+----------------------------------------+
| INNODB_LOCKS                           |
| INNODB_TRX                             |
| INNODB_SYS_DATAFILES                   |
| INNODB_LOCK_WAITS                      |
| INNODB_SYS_TABLESTATS                  |
| INNODB_CMP                             |
| INNODB_FT_BEING_DELETED                |
| INNODB_CMP_RESET                       |
| INNODB_CMP_PER_INDEX                   |
| INNODB_CMPMEM_RESET                    |
| INNODB_FT_DELETED                      |
| INNODB_BUFFER_PAGE_LRU                 |
| INNODB_FT_INSERTED                     |
| INNODB_CMPMEM                          |
| INNODB_SYS_INDEXES                     |
| INNODB_SYS_TABLES                      |
| INNODB_SYS_FIELDS                      |
| INNODB_CMP_PER_INDEX_RESET             |
| INNODB_BUFFER_PAGE                     |
| INNODB_FT_DEFAULT_STOPWORD             |
| INNODB_FT_INDEX_TABLE                  |
| INNODB_FT_INDEX_CACHE                  |
| INNODB_SYS_TABLESPACES                 |
| INNODB_METRICS                         |
| INNODB_SYS_FOREIGN_COLS                |
| INNODB_FT_CONFIG                       |
| INNODB_BUFFER_POOL_STATS               |
| INNODB_SYS_COLUMNS                     |
| INNODB_SYS_FOREIGN                     |
+----------------------------------------+
29 rows in set (0.01 sec)
```

5. List the settings for all InnoDB-specific server variables. Enter the following in a terminal window, and receive the result shown below:

```
mysql> SHOW VARIABLES LIKE '%innodb%';
+-------------------------------------------+------------------------+
| Variable_name                             | Value                  |
+-------------------------------------------+------------------------+
| ignore_builtin_innodb                     | OFF                    |
...
| innodb_api_trx_level                      | 0                      |
| innodb_autoextend_increment               | 64                     |
| innodb_autoinc_lock_mode                  | 1                      |
| innodb_buffer_pool_dump_at_shutdown       | OFF                    |
| innodb_buffer_pool_dump_now               | OFF                    |
| innodb_buffer_pool_filename               | ib_buffer_pool         |
| innodb_buffer_pool_instances              | 2                      |
| innodb_buffer_pool_load_abort             | OFF                    |
| innodb_buffer_pool_load_at_startup        | OFF                    |
| innodb_buffer_pool_load_now               | OFF                    |
| innodb_buffer_pool_size                   | 2147483648             |
| innodb_change_buffer_max_size             | 25                     |
...
| innodb_concurrency_tickets                | 5000                   |
| innodb_data_file_path                     | ibdata1:12M:autoextend |
| innodb_data_home_dir                      |                        |
...
| innodb_file_format_max                    | Antelope               |
| innodb_file_per_table                     | ON                     |
| innodb_flush_log_at_timeout               | 1                      |
...
| innodb_max_dirty_pages_pct_lwm            | 0                      |
| innodb_max_purge_lag                      | 0                      |
| innodb_max_purge_lag_delay                | 0                      |
...
| innodb_version                            | 1.2.10                 |
| innodb_write_io_threads                   | 4                      |
+-------------------------------------------+------------------------+
115 rows in set (0.01 sec)
```

a. Is the multiple tablespace (file-per-table) setting ON or OFF? **ON**

b. What is the auto-extend increment setting? **64**

c. What is the InnoDB data file path? **ibdata1:12M:autoextend**

d. Is there any purge lag? No; the values of **innodb_max_purge_lag** is **0.**

e. What is the buffer pool size? **2147483648** bytes in the preceding output. The answers to this and the following question depend on the values used in step 1c.

f. How many buffer pool instances are there? **Two** in the preceding output.

6. Edit the existing config file (`/etc/my.cnf`) to change the auto-extend increment to 128 MB.

   Note: You must launch the editor from your terminal logged in as `root`.

   a. Edit the `/etc/my.cnf` file as follows:

   ```
   [mysqld]
   innodb_autoextend_increment=128
   innodb_buffer_pool_size=2GB
   ...
   ```

   b. Save and close the config file.

   c. Stop and restart the server to implement the change. Enter the following in a terminal window, and receive the results shown below:

      – Be sure to exit any running `mysql` clients prior to stopping the server.

   ```
   # service mysql restart
   Shutting down MySQL......                          [ OK ]
   Starting MySQL......                               [ OK ]
   ```

7. Confirm the changes made to the config file by listing the settings for all InnoDB-specific server variables again. Compare the results to the previous list. Enter the following in a terminal window, and receive the result shown below:

   ```
   $ mysql -uroot -poracle
   ...
   mysql> SHOW VARIABLES LIKE '%innodb%';
   +----------------------------------+-------------------------+
   | Variable_name                    | Value                   |
   +----------------------------------+-------------------------+
   | ignore_builtin_innodb            | OFF                     |
   | innodb_adaptive_flushing         | ON                      |
   ...
   | innodb_autoextend_increment      | 128                     |
   ...
   +----------------------------------+-------------------------+
   ```

   a. What is the auto-extend increment setting? **128**

8. Create a new table called `CityLanguage` in the `world_innodb` database using the `MEMORY` storage engine. Include definitions for `City`, `Country`, `CountryCode`, and `Language` columns.

   a. Enter the following `CREATE TABLE` statement at a `mysql` prompt, and receive the results shown below:

```
mysql> USE world_innodb;
...
Database changed
mysql> CREATE TABLE CityLanguage (
    -> City CHAR(35),
    -> Country CHAR(35),
    -> CountryCode CHAR(3),
    -> Language CHAR(30)
    -> ) ENGINE=MEMORY;
Query OK, 0 rows affected (0.11 sec)
```

   b. Confirm the storage engine setting:

```
mysql> SHOW CREATE TABLE CityLanguage\G
*************************** 1. row *********************
       Table: CityLanguage
Create Table: CREATE TABLE `citylanguage` (
  `City` char(35) DEFAULT NULL,
  `Country` char(35) DEFAULT NULL,
  `CountryCode` char(3) DEFAULT NULL,
  `Language` char(30) DEFAULT NULL
) ENGINE=MEMORY DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```

9. Modify the new `CityLanguage` table to use the InnoDB storage engine, and confirm the change. Enter the following at a `mysql` prompt, and receive the results shown below:

```
mysql> ALTER TABLE CityLanguage ENGINE=InnoDB;
Query OK, 0 rows affected (0.80 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> SHOW CREATE TABLE CityLanguage\G
*************************** 1. row ***************************
       Table: CityLanguage
Create Table: CREATE TABLE `CityLanguage` (
  `City` char(35) DEFAULT NULL,
  `Country` char(35) DEFAULT NULL,
  `CountryCode` char(3) DEFAULT NULL,
  `Language` char(30) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```

Practices for Lesson 5: Partitioning

Chapter 5

Practice 5-1: Quiz – MySQL Partitioning

Overview

In this quiz, you answer questions about MySQL partitioning.

Quiz Questions

Choose the best answer from those provided for each multiple choice or True/False question.

1. The primary reason for using partitioning is to improve database performance.
   a. True
   b. False
2. Assigning different rows of a table to different physical partitions is called ________________ __________________.
   a. partitioning databases
   b. physical partitioning
   c. horizontal partitioning
   d. none of the above
3. The statement used to determine whether or not your current MySQL server supports partitioning is:

   ```
   SHOW ENGINES;
   ```

   a. True
   b. False
4. You use the _______________ partition type when selecting partitions based on columns matching one of a set of discrete values.
   a. `LIST`
   b. subpartition
   c. `RANGE`
   d. `LINEAR`
5. You use the `COLUMNS` partitioning variant (with the `RANGE` and `LIST` types) to enable the use of multiple columns. These columns are then taken into account for placing rows in partitions, and for determining which partitions are to be checked for matching rows in partition pruning.
   a. True
   b. False

6. You can subpartition tables that are partitioned by the following partition type(s), to further divide each partition.
   a. `LIST` and `LINEAR`
   b. `HASH` and `KEY`
   c. `RANGE` only
   d. `LIST` and `RANGE`
7. Query the `PARTITIONS` table of the `INFORMATION_SCHEMA` database to list the partition names (with respective partitioning descriptions) for a specific table (using the `WHERE` clause).
   a. True
   b. False

Solutions 5-1: Quiz – MySQL Partitioning

Quiz Solutions

1. **a**. True
2. **c**. horizontal partitioning
3. **b**. False. The statement is as follows:

```
SHOW PLUGINS;
```

4. **a**. `LIST`
5. **a**. True
6. **d**. `LIST` and `RANGE`. The subpartitions themselves can use `HASH`, `LINEAR HASH`, `KEY`, or `LINEAR KEY` partitioning.
7. **a**. True

Practice 5-2: Creating and Modifying a Partitioned Table

Overview

In this practice, you create a partitioned table and modify that table.

Tasks

1. Start a new `mysql` client session. Verify that the system variable `innodb_file_per_table` is turned on (this is the default since MySQL 5.6).
2. Create a new table called `City_part` that has the same column definitions as `City`, within the `world_innodb` database.
3. Use the `SHOW TABLE STATUS` statement to determine whether the new table is partitioned.
4. Modify the new table to add four `RANGE` type partitions (using the `ID` column) by using an `ALTER TABLE` statement:
   - `p0` (values less than 1000)
   - `p1` (values less than 2000)
   - `p2` (values less than 3000)
   - `p3` (values less than maximum value)
5. Now that you have modified the new table, display the table partitioning status again. Does it indicate partitioning?
6. Complete the copy of the `City` table into `City_part`, by inserting all the rows from the original table.
7. In a separate terminal window, check the MySQL data directory for the new table files (`.par` and `.ibd`). Note the exact number of `.ibd` files, and their file names and sizes.

   Note: The individual `.ibd` partition files are created due to the `info_file_per_table` being set to `ON`.
8. Confirm the `City_part` table partitions by using `EXPLAIN PARTITIONS` to show which partitions are used for a query of all table data. List the partitions in the result.
9. Determine which partitions would be used for a query of `City_part` table data, where the `ID` value is less than `2000`. Does it use all the partitions? If not, which partitions are used for this level of query?
10. Redefine the `City_part` table to use `KEY` partitioning with three separate partitions.
11. Confirm the table partition modifications by using `EXPLAIN PARTITIONS` to show partitions used for a query of all table data.
12. Check the MySQL data directory for the modified table files. Note the exact number of `.ibd` files, and their file names and sizes. What is the difference between these files and the previously `RANGE` partitioned table files?

13. Query the `PARTITIONS` table from the `INFORMATION_SCHEMA` database, for the partition names in the `City_part` table. How many partitions are listed, and what are their names?

    Note: Keep the current `City_part` table intact for use in the next practice.

Solutions 5-2: Creating and Modifying a Partitioned Table

Tasks

1. Start a new `mysql` client session. Verify that the system variable `innodb_file_per_table` is turned on (this is the default since MySQL 5.6).

   Enter the following in a terminal window, and receive the results shown below:

```
$ mysql --login-path=admin
...

mysql> SHOW VARIABLES LIKE 'innodb_file_per_table';
+-----------------------+-------+
| Variable_name         | Value |
+-----------------------+-------+
| innodb_file_per_table | ON    |
+-----------------------+-------+
1 row in set (0.00 sec)
```

2. Create a new table called `City_part` that has the same column definitions as `City`, within the `world_innodb` database. Enter the following in a terminal window, and receive the results shown below:

```
mysql> USE world_innodb
...
Database changed
mysql> CREATE TABLE City_part LIKE City;
Query OK, 0 rows affected (0.00 sec)
```

3. Use the `SHOW TABLE STATUS` statement to determine whether the new table is partitioned. Enter the following in a terminal window, and receive the result shown below:

```
mysql> SHOW TABLE STATUS LIKE 'City_part'\G
*************************** 1. row ***************************
           Name: City_part
         Engine: InnoDB
        Version: 10
     Row_format: Compact
           Rows: 0
 Avg_row_length: 0
    Data_length: 16384
Max_data_length: 0
   Index_length: 16384
      Data_free: 0
 Auto_increment: 1
    Create_time: 2012-10-15 18:00:02
    Update_time: NULL
     Check_time: NULL
      Collation: latin1_swedish_ci
       Checksum: NULL
 Create_options:
        Comment:
1 row in set (0.00 sec)
```

   There is no indication of partitioning in this table.

4. Modify the new table to add four `RANGE` type partitions (using the `ID` column) by using an `ALTER TABLE` statement. Enter the following in a terminal window, and receive the result shown below:

```
mysql> ALTER TABLE City_part PARTITION BY RANGE (ID) (
    -> PARTITION p0 VALUES LESS THAN (1000),
    -> PARTITION p1 VALUES LESS THAN (2000),
    -> PARTITION p2 VALUES LESS THAN (3000),
    -> PARTITION p3 VALUES LESS THAN MAXVALUE
    -> );
Query OK, 0 rows affected (4.34 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

5. Now that you have modified the new table, display the table partitioning status again.
   a. Enter the following in a terminal window, and receive the result shown below:

```
mysql> SHOW TABLE STATUS LIKE 'City_part'\G
*************************** 1. row ***************************
           Name: City_part
         Engine: InnoDB
        Version: 10
     Row_format: Compact
           Rows: 4
 Avg_row_length: 16384
    Data_length: 65536
Max_data_length: 0
   Index_length: 65536
      Data_free: 25165824
 Auto_increment: 1
    Create_time: NULL
    Update_time: NULL
     Check_time: NULL
      Collation: latin1_swedish_ci
       Checksum: NULL
 Create_options: partitioned
        Comment:
1 row in set (0.00 sec)
```

   b. Does it indicate partitioning? **Yes**

6. Complete the copy of the `City` table into `City_part`, by inserting all the rows from the original table. Enter the following in a terminal window, and receive the result shown below:

```
mysql> INSERT INTO City_part SELECT * FROM City;
Query OK, 4079 rows affected, 1 warning (1.63 sec)
Records: 4079  Duplicates: 0  Warnings: 1
```

The statement issues a warning because you are using statement-based logging while inserting records from another table into a table that uses an auto-increment column. This combination is not guaranteed to be safe for replication. Because you are not using the binary log for replication, you can safely ignore this warning in this lesson.

7. In a separate terminal window, check the MySQL data directory for the new table files (`.par` and `.ibd`).

   a. Enter the following in a separate terminal window, and receive the result shown below:

```
$ su –
Password: oracle
# cd /var/lib/mysql/world_innodb; ls -l
...
-rw-rw---- 1 mysql mysql   8710 Feb  6 04:55 City_part.frm
-rw-rw---- 1 mysql mysql     32 Feb  6 04:55 City_part.par
-rw-rw---- 1 mysql mysql 212992 Feb  6 04:57 City_part#P#p0.ibd
-rw-rw---- 1 mysql mysql 212992 Feb  6 04:57 City_part#P#p1.ibd
-rw-rw---- 1 mysql mysql 212992 Feb  6 04:57 City_part#P#p2.ibd
-rw-rw---- 1 mysql mysql 212992 Feb  6 04:57 City_part#P#p3.ibd
...
```

   b. Note the exact number of `.ibd` files, and their file names and sizes:

      – The `.frm` and `.par` files are very small.

      – Your file sizes may vary from those shown above. The most important thing is to note which files exist and the difference in their sizes.

8. Confirm the `City_part` table partitions by using `EXPLAIN PARTITIONS` to show which partitions are used for a query of all table data. Enter the following in the `mysql` window, and receive the result shown below:

```
mysql> EXPLAIN PARTITIONS SELECT * FROM City_part\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: City_part
   partitions: p0,p1,p2,p3
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 4079
        Extra:
1 row in set (0.00 sec)
```

   – `EXPLAIN` returns an estimate of the number of examine rows used for the query, which can be very different than the actual number of rows.

9. Determine which partitions would be used for a query of `City_part` table data, where the **`ID`** value is less than **`2000`**.

   a. Enter the following in a terminal window, and receive the result shown below:

```
mysql> EXPLAIN PARTITIONS SELECT * FROM City_part WHERE ID < 2000\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: City_part
   partitions: p0,p1
         type: range
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: NULL
         rows: 1996
        Extra: Using where
1 row in set (0.00 sec)
```

   b. Does it use all the partitions? **No**. Only `p0` and `p1` appear in the `partitions` column.

10. Redefine the `City_part` table to use **`KEY`** partitioning with three separate partitions. Enter the following in a terminal window, and receive the result shown below:

```
mysql> ALTER TABLE City_part
    -> PARTITION BY KEY (ID) PARTITIONS 3;
Query OK, 4079 rows affected (5.24 sec)
Records: 4079  Duplicates: 0  Warnings: 0
```

11. Confirm the table partition modifications by using `EXPLAIN PARTITIONS` to show partitions used for a query of all table data. Enter the following in a terminal window, and receive the result shown below:

```
mysql> EXPLAIN PARTITIONS SELECT * FROM City_part\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: City_part
   partitions: p0,p1,p2
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 2734
        Extra:
1 row in set (0.00 sec)
```

12. Check the MySQL data directory for the modified table files. Note the exact number of `.ibd` files, and their file names and sizes.

    a. Enter the following in a terminal window logged in as `root`, and receive the result shown below:

    ```
    # cd /var/lib/mysql/world_innodb; ls -l
    ...
    -rw-rw---- 1 mysql mysql   8710 Feb  6 05:06 City_part.frm
    -rw-rw---- 1 mysql mysql     32 Feb  6 05:06 City_part.par
    -rw-rw---- 1 mysql mysql 245760 Feb  6 05:06 City_part#P#p0.ibd
    -rw-rw---- 1 mysql mysql 278528 Feb  6 05:06 City_part#P#p1.ibd
    -rw-rw---- 1 mysql mysql 245760 Feb  6 05:06 City_part#P#p2.ibd
    ...
    ```

    b. What is the difference between these files and the previously `RANGE` partitioned table files? There are now three `.ibd` files, each containing approximately one-third of the table data.

       – Your file sizes may vary from those shown above. The most important thing is to note which files exist and the difference in their sizes.

13. Query the `PARTITIONS` table from the `INFORMATION_SCHEMA` database, for the partition names in the `City_part` table. Enter the following in a `mysql` window, and receive the result shown below:

    ```
    mysql> SELECT TABLE_NAME,
        -> GROUP_CONCAT(PARTITION_NAME)
        -> FROM INFORMATION_SCHEMA.PARTITIONS
        -> WHERE TABLE_SCHEMA='world_innodb'
        -> AND TABLE_NAME='City_part';
    +------------+------------------------------+
    | TABLE_NAME | GROUP_CONCAT(PARTITION_NAME) |
    +------------+------------------------------+
    | City_part  | p0,p1,p2                     |
    +------------+------------------------------+
    1 row in set (0.00 sec)
    ```

    Note: Keep the current `City_part` table intact for use in the next practice.

Practice 5-3: Removing Partitions from a Table

Overview

In this practice, you drop partitions from a partitioned table and remove all partitioning from a table.

Tasks

1. Using the `mysql` client from the previous practice, drop the first partition (`p0`) from the `City_part` table. Did this operation work? If not, why did it not work?
2. Return the current `City_part` table back to the first `RANGE`-partitioned configuration:
   - `p0` (values less than 1000)
   - `p1` (values less than 2000)
   - `p2` (values less than 3000)
   - `p3` (values less than maximum value)
3. Verify the file sizes for each of the new partitions.
4. Attempt to drop the first partition (`p0`) from the `City_part` table again. Did this operation work? Why?
5. Confirm the modifications made to `City_part` table partitions by using `EXPLAIN PARTITIONS` to show which partitions are now being used for a query of all table data. Which partitions are left?
6. Check the MySQL data directory. Does the dropped partition file still exist? Are the sizes of the remaining `.ibd` files the same as they were before the partition was dropped?
   - Compare to the output results from Practice 10-2, step 7.
7. Remove partitioning of the `City_part` table and return it to its original, non-partitioned status.
8. Now that you have modified `City_part` again, display the table partitioning status again. Does the output indicate that the table is partitioned?
9. Verify that the `City_part` table partitions are now gone, by using `EXPLAIN PARTITIONS`.
10. One last time, check the MySQL data directory. What `City_part` files are left? What are their names and sizes?

Solutions 5-3: Removing Partitions from a Table

Tasks

1. Using the `mysql` client from the previous practice, drop the first partition (`p0`) from the `City_part` table.
   a. Enter the following in a terminal window, and receive the result shown below:

```
mysql> ALTER TABLE City_part DROP PARTITION p0;
ERROR 1512 (HY000): DROP PARTITION can only be used on RANGE/LIST
partitions
```

   b. Did this operation work? No, because this is a KEY partitioned table.

2. Return the current `City_part` table back to the first RANGE-partitioned configuration. Enter the following in a terminal window, and receive the result shown below:

```
mysql> ALTER TABLE City_part PARTITION BY RANGE (id) (
    -> PARTITION p0 VALUES LESS THAN (1000),
    -> PARTITION p1 VALUES LESS THAN (2000),
    -> PARTITION p2 VALUES LESS THAN (3000),
    -> PARTITION p3 VALUES LESS THAN MAXVALUE
    -> );
Query OK, 4079 rows affected (7.73 sec)
Records: 4079  Duplicates: 0  Warnings: 0
```

3. Verify the file sizes for each of the new partitions. Enter the following in a terminal window logged in as `root`, and receive the result shown below:

```
# cd /var/lib/mysql/world_innodb; ls -l
...
-rw-rw---- 1 mysql mysql   8710 Feb  6 05:31 City_part.frm
-rw-rw---- 1 mysql mysql     32 Feb  6 05:31 City_part.par
-rw-rw---- 1 mysql mysql 327680 Feb  6 05:31 City_part#P#p0.ibd
-rw-rw---- 1 mysql mysql 311296 Feb  6 05:31 City_part#P#p1.ibd
-rw-rw---- 1 mysql mysql 360448 Feb  6 05:31 City_part#P#p2.ibd
-rw-rw---- 1 mysql mysql 311296 Feb  6 05:31 City_part#P#p3.ibd
...
```

4. Attempt to drop the first partition (`p0`) from the `City_part` table again.
   a. Enter the following in a terminal window, and receive the result shown below:

```
mysql> ALTER TABLE City_part DROP PARTITION p0;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

   b. Did this operation work? Yes. RANGE partitioned tables allow `DROP PARTITION`.

5. Confirm the modifications made to `City_part` table partitions by using `EXPLAIN PARTITIONS` to show which partitions are now being used for a query of all table data. Enter the following in a terminal window, and receive the result shown below:

```
mysql> EXPLAIN PARTITIONS SELECT * FROM City_part\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: City_part
   partitions: p1,p2,p3
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 3081
        Extra: NULL
1 row in set (0.00 sec)
```

Which partitions are left? Having deleted partition p0 in the preceding step, partitions **p1**, **p2**, and **p3** are left.

6. Check the MySQL data directory.

   a. Enter the following in a terminal window logged in as `root`, and receive the result shown below:

```
# cd /var/lib/mysql/world_innodb; ls -l
...
-rw-rw---- 1 mysql mysql   8710 Feb  6 05:21 City_part.frm
-rw-rw---- 1 mysql mysql     32 Feb  6 05:21 City_part.par
-rw-rw---- 1 mysql mysql 311296 Feb  6 05:12 City_part#P#p1.ibd
-rw-rw---- 1 mysql mysql 360448 Feb  6 05:12 City_part#P#p2.ibd
-rw-rw---- 1 mysql mysql 311296 Feb  6 05:12 City_part#P#p3.ibd
...
```

   b. Does the dropped partition file still exist? **No**

   c. Are the sizes of the remaining `.ibd` files the same as they were before the partition was dropped? **Yes**, these three partitions are the same as they were when the table was first altered to be a `RANGE` table. However, the `p0` partition and all its contents are now gone.

      – Your file sizes may vary from those shown above. The most important thing is to note which files exist and the difference in their sizes.

7. Remove partitioning of the `City_part` table and return it to its original, non-partitioned status. Enter the following in a terminal window, and receive the result shown below:

```
mysql> ALTER TABLE City_part REMOVE PARTITIONING;
Query OK, 3080 rows affected (0.12 sec)
Records: 3080  Duplicates: 0  Warnings: 0
```

8. Now that you have modified `City_part` again, display the table partitioning status again. Enter the following in a terminal window, and receive the result shown below:

```
mysql> SHOW TABLE STATUS LIKE 'City_part'\G
*************************** 1. row ***************************
           Name: City_part
         Engine: InnoDB
        Version: 10
     Row_format: Compact
           Rows: 3081
 Avg_row_length: 101
    Data_length: 311296
Max_data_length: 0
   Index_length: 98304
      Data_free: 0
 Auto_increment: 4081
    Create_time: 2013-02-06 05:35:01
    Update_time: NULL
     Check_time: NULL
      Collation: latin1_swedish_ci
       Checksum: NULL
 Create_options:
        Comment:
1 row in set (0.00 sec)
```

   a. Does the output indicate that the table is partitioned? No. The `Create_options` column is empty and does not contain the string "`partitioned`" to indicate that partitioning is enabled.

9. Verify that the `City_part` table partitions are now gone, by using `EXPLAIN PARTITIONS`. Enter the following in a terminal window, and receive the result shown below:

```
mysql> EXPLAIN PARTITIONS SELECT * FROM City_part\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: City_part
   partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 3081
        Extra:
1 row in set (0.00 sec)
```

10. One last time, check the MySQL data directory. What `City_part` files are left? What are their names and sizes? Enter the following in a terminal window, and receive the result shown below:

```
# cd /var/lib/mysql/world_innodb; ls -l
...
-rw-rw---- 1 mysql mysql   8710 Mar 23 19:12 City_part.frm
-rw-rw---- 1 mysql mysql 475136 Mar 23 19:12 City_part.ibd
...
```

Your file sizes may vary from those shown above. The most important thing is to note which files exist and the difference in their sizes.

Practices for Lesson 6: User Management

Chapter 6

Practice 6-1: Quiz – MySQL User Management

Overview

In this quiz, you answer questions about MySQL user management.

Quiz Questions

Choose the best answer from those provided for each multiple choice or True/False question.

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

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

   a. True

   b. False

2. 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 = 'Sasha';
+--------------+-------+-------------+
| Host         | User  | Select_priv |
+--------------+-------+-------------+
| 62.220.12.66 | Sasha | N           |
| 62.220.12.%  | Sasha | Y           |
| 62.220.%     | Sasha | N           |
+--------------+-------+-------------+
```

Note: The **`Select_priv`** column indicates that the **`SELECT`** privilege for the second entry has been granted with global scope (**`*.*`**). Assume that the **`Sasha`** accounts are not granted privileges in any of the other grant tables.

Can user **`Sasha`** 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`

3. Which of the following user manipulation statements is true?

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

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

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

4. Assume that a new user has been created, but that user account has not been granted any privileges yet. What can that user do at this point?
   a. Connect to the server
   b. Issue statements like **`SELECT`**
   c. Issue statements like **`SHOW DATABASES`**
5. 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 do not affect existing connections of a user. They take effect only the next time the user attempts to connect.
   c. Database privilege changes take effect only at the time that a user next attempts to connect.
   d. Database privilege changes take effect only at the time that a user next issues a **`USE`** ***`<database>`*** statement.
   e. Table and column privileges take immediate effect.
6. A server that has the privileges of the **`root`** operating system login account has more file system access than necessary and constitutes a security risk.
   a. True
   b. False
7. To disable a TCP/IP connection, start the server with the ______________ option.
   a. `--skip-tcp_ip`
   b. `--skip-networking`
   c. `--tcp-ip_skip`

Solutions 6-1: Quiz – MySQL User Management

Quiz Solutions

1. **a**. True
2. Can user **Sasha** select data from any table on the MySQL server when connecting from the following hosts?
   - **d.** **No**. **62.220.12.66** is the most specific entry that matches the host that user **Sasha** is trying to connect from. Because the **SELECT** privilege for that entry is **N**, **Sasha** cannot select from any table on the server.
   - e. **Yes**. The most specific entry that matches **62.220.12.43** is **62.220.12.%**. Because the **SELECT** privilege for that entry is **Y**, user **Sasha** can select from any table on the server.
   - f. **No**. The most specific entry that matches **62.220.42.43** is **62.220.%**. Because the **SELECT** privilege for that entry is **N**, user **Sasha** cannot select from any table on the server.
   - g. **No**. There is no entry that would match **Sasha@localhost**. Therefore, user **Sasha** cannot even connect to the server from the **localhost**.
3. **c**. To create a user, you can use either a **CREATE USER** statement or a **GRANT** statement. **ALTER USER** can only be used to expire a user's password; 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 remains, as this example illustrates:

```
mysql> SHOW GRANTS FOR 'lennart';
+--------------------------------------+
| Grants for lennart@%                 |
+--------------------------------------+
| GRANT USAGE ON *.* TO 'lennart'@'%' |
+--------------------------------------+
mysql> REVOKE usage ON *.* FROM 'lennart';
mysql> SHOW GRANTS FOR 'lennart';
+--------------------------------------+
| Grants for lennart@%                 |
+--------------------------------------+
| GRANT USAGE ON *.* TO 'lennart'@'%' |
+--------------------------------------+
mysql> DROP USER 'lennart';
mysql> SHOW GRANTS FOR 'lennart';
ERROR 1141 (42000): There is no such grant defined for user 'lennart'
on host '%'
```

4. **a**. Connect to the server. 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 objects, because the account does not have any privileges on them. **SHOW DATABASES** displays only the **INFORMATION_SCHEMA** database (which is a database that does not have a physical representation in the file system).

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

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

   b. **True**. Global privilege changes do not affect existing connections of a user. They take effect only the next time the user attempts to connect.

   c. **False**. See the next item.

   d. **True**. Database privilege changes take effect only at the time that a user next issues a **`USE`** ***`<database>`*** statement.

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

6. **a**. True

7. **b**. `--skip-networking`

Practice 6-2: Creating, Verifying, and Dropping a User

Overview

In this practice, you create, verify, and drop an anonymous user.

Tasks

1. Verify that there are no anonymous accounts on the server by using a `SELECT` statement.
2. Create an anonymous account by using the `CREATE USER` statement on the local host.
3. Confirm that the anonymous account now exists by re-running the previous `SELECT` statement.
4. Delete the anonymous account by using the `DROP USER` statement. Confirm that the anonymous account no longer exists.

Solutions 6-2: Creating, Verifying, and Dropping a User

Tasks

1. Verify that there are no anonymous accounts on the server by using a `SELECT` statement.

   Enter the following in a terminal window, and receive the result shown below:

   ```
   $ mysql --login-path=admin
   ...

   mysql> SELECT user, host, password FROM mysql.user
       -> WHERE user = '';
   Empty set (0.00 sec)
   ```

2. Create an anonymous account by using the `CREATE USER` statement on the local host.

   Enter the following at a `mysql` prompt, and receive the result shown below:

   ```
   mysql> CREATE USER ''@'localhost';
   Query OK, 0 rows affected (0.00 sec)
   ```

3. Confirm that the anonymous account now exists by re-running the previous **`SELECT`** statement.

   Enter the following at a `mysql` prompt, and receive the result shown below:

   ```
   mysql> SELECT user, host, password FROM mysql.user
       -> WHERE user = '';
   +------+-----------+----------+
   | user | host      | password |
   +------+-----------+----------+
   |      | localhost |          |
   +------+-----------+----------+
   ```

4. Delete the anonymous account by using the **`DROP USER`** statement.

   a. Enter the following at a `mysql` prompt, and receive the result shown below:

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

   b. Confirm that the anonymous account no longer exists:

   ```
   mysql> SELECT user, host, password FROM mysql.user
       -> WHERE user = '';
   Empty set (0.00 sec)
   ```

Practice 6-3: Setting Up a User for the `world_innodb` Database

Overview

In this practice, you create, drop, and verify a new user specifically for the `world_innodb` database.

Tasks

1. Create a new user called `student` at the host `pc.example.com` by using the `mysql` client session from the previous practice.
2. Confirm that the new user is now included in the `mysql` database by using a `SELECT` statement.
3. Assign the user `student` a password and privileges (`SELECT`, `INSERT`, `DELETE`, `UPDATE`) for the `world_innodb` database by using a `GRANT` statement.
4. Show the grants set for the user `student` on the `pc.example.com` host by using the `SHOW GRANTS` statement.
5. Revoke the `DELETE` and `UPDATE` privileges currently given to `student` on the `pc.example.com` host. Confirm the updated grants now set for the user `student` on the `pc.example.com` host.
6. Change the password for the `student` on the `pc.example.com` host to `NewPass` by using a `GRANT` statement.
7. Grant `ALL` privileges to the `student` on the `pc.example.com` host with password `NewPass`, and allow only 10 maximum connections per hour. Confirm the updated grants now set for the user `student` on the `pc.example.com` host.
8. Drop the user `student` at the host `pc.example.com`. Confirm that the account no longer exists.

Solutions 6-3: Setting Up a User for the `world_innodb` Database

Tasks

1. Create a new user called `student` at the host `pc.example.com` by using the `mysql` client session from the previous practice.

   Enter the following at a `mysql` prompt, and receive the result shown below:

   ```
   mysql> CREATE USER 'student'@'pc.example.com';
   Query OK, 0 rows affected (0.00 sec)
   ```

2. Confirm that the new user is now included in the `mysql` database by using a `SELECT` statement.

   Enter the following at a `mysql` prompt, and receive the result shown below:

   ```
   mysql> SELECT user, host, password
       -> FROM mysql.user WHERE user = 'student';
   +---------+----------------+----------+
   | user    | host           | password |
   +---------+----------------+----------+
   | student | pc.example.com |          |
   +---------+----------------+----------+
   1 row in set (0.00 sec)
   ```

3. Assign the user `student` a password and privileges (`SELECT`, `INSERT`, `DELETE`, `UPDATE`) for the `world_innodb` database by using a `GRANT` statement.

   Enter the following at a `mysql` prompt, and receive the result shown below:

   ```
   mysql> GRANT SELECT, INSERT, DELETE, UPDATE ON world_innodb.*
       -> TO 'student'@'pc.example.com'
       -> IDENTIFIED BY 'student_pass';
   Query OK, 0 rows affected (0.00 sec)
   ```

4. Show the grants set for the user `student` on the `pc.example.com` host by using the `SHOW GRANTS` statement.

   Enter the following at a `mysql` prompt, and receive the result shown below:

   ```
   mysql> SHOW GRANTS FOR 'student'@'pc.example.com'\G
   *************************** 1. row ***************************
   Grants for student@pc.example.com: GRANT USAGE ON *.* TO
   'student'@'pc.example.com' IDENTIFIED BY PASSWORD
   '*832CA3798551FEA378AD34E02713320E28194DA2'
   *************************** 2. row ***************************
   Grants for student@pc.example.com: GRANT SELECT, INSERT, UPDATE,
   DELETE ON `world_innodb`.* TO 'student'@'pc.example.com'
   2 rows in set (0.00 sec)
   ```

5. Revoke the DELETE and UPDATE privileges currently given to the student user on the pc.example.com host. Confirm the updated grants now set for the user student on the pc.example.com host.

   Enter the following at a mysql prompt, and receive the results shown below:

```
mysql> REVOKE DELETE, UPDATE ON world_innodb.*
    -> FROM 'student'@'pc.example.com';
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS for 'student'@'pc.example.com'\G
*************************** 1. row ***************************
Grants for student@pc.example.com: GRANT USAGE ON *.* TO
'student'@'pc.example.com' IDENTIFIED BY PASSWORD
'*832CA3798551FEA378AD34E02713320E28194DA2'
*************************** 2. row ***************************
Grants for student@pc.example.com: GRANT SELECT, INSERT ON
`world_innodb`.* TO 'student'@'pc.example.com'
2 rows in set (0.00 sec)
```

6. Change the password for the student user on the pc.example.com host to NewPass by using a GRANT statement.

   Enter the following at a mysql prompt, and receive the result shown below:

```
mysql> GRANT USAGE ON *.* TO 'student'@'pc.example.com'
    -> IDENTIFIED BY 'NewPass';
Query OK, 0 rows affected (0.00 sec)
```

7. Grant ALL privileges to the student user on the pc.example.com host with password NewPass, and allow only 10 maximum connections per hour. Confirm the updated grants now set for the user student on the pc.example.com host.

   Enter the following at a mysql prompt, and receive the results shown below:

```
mysql> GRANT ALL ON world_innodb.* TO 'student'@'pc.example.com'
    -> IDENTIFIED BY 'NewPass'
    -> WITH MAX_CONNECTIONS_PER_HOUR 10;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS for 'student'@'pc.example.com'\G
*************************** 1. row ***************************
Grants for student@pc.example.com: GRANT USAGE ON *.* TO
'student'@'pc.example.com' IDENTIFIED BY PASSWORD
'*9EE65E227E34770F6F707EC991CFF926CB294BC7' WITH
MAX_CONNECTIONS_PER_HOUR 10
*************************** 2. row ***************************
Grants for student@pc.example.com: GRANT ALL PRIVILEGES ON
`world_innodb`.* TO 'student'@'pc.example.com'
```

8. Drop the user `student` at the host `pc.example.com`. Confirm that the account no longer exists.

   a. Enter the following at a `mysql` prompt, and receive the result shown below:

   ```
   mysql> DROP USER 'student'@'pc.example.com';
   Query OK, 0 rows affected (0.00 sec)
   ```

   b. Confirm that the account no longer exists:

   ```
   mysql> SELECT user, host, password FROM mysql.user
       -> WHERE user='student';
   Empty set (0.00 sec)
   ```

Practice 6-4: Using the PAM Authentication Plugin

Overview

In this practice, you enable the PAM authentication plugin and log in to MySQL using operating system credentials.

Tasks

Note: This practice requires the use of multiple terminal windows for multiple `mysql` client sessions.

1. Execute the following commands at a terminal window logged in as `root` to create the Linux user `pamuser1` with the password `oracle1`.

```
useradd pamuser1
passwd pamuser1
```

2. Using a text editor, create a file in the `/etc/pam.d/` directory called `mysql-dba-course` containing the following text:

```
#%PAM-1.0
auth            include         password-auth
account         include         password-auth
```

3. Make the file `/etc/shadow` group-readable, and set its group to `mysql` so the MySQL server process can authenticate using PAM. Use the following Linux commands:

```
chmod 440 /etc/shadow
chgrp mysql /etc/shadow
```

4. Install the `authentication_pam` plugin.
5. Show all plugins to verify that the PAM authentication plugin is enabled.
6. Create a MySQL user called `pamuser1` that can log in from the local host. Ensure that it is identified with PAM authentication, using the PAM service `mysql-dba-course`.
7. Grant the `SELECT` privilege on the `world_innodb.City` table to the newly created `pamuser1` user.
8. At a new terminal window, attempt to log in to MySQL as `pamuser1`, using the `cleartext` plugin and the password `oracle1`.
9. Execute a statement to display the value of the `CURRENT_USER()` function.
10. Exit the mysql prompt logged in as `pamuser1`.
11. Create a Linux user `pamuser2` with the password `oracle2`, using the same technique as at the beginning of this practice.

12. Create a Linux group `dba`, and add the new `pamuser2` user to that group by using the following commands at a `root` terminal prompt:

```
groupadd dba
usermod -G dba pamuser2
```

13. Create a `mysql` user called `world_admin` that can log in from the local host. Give it a cryptic password. Grant all privileges on the `world_innodb` database to the new user.
14. Create a default proxy account with an empty user name and host name that is identified with PAM authentication, using the PAM service `mysql-dba-course` and the mapping `dba=world_admin`.
15. Grant the default proxy account the `PROXY` privilege on the `world_admin@localhost` account created previously.
16. At a new terminal window, attempt to log in to MySQL as `pamuser2`, using the `cleartext` plugin and the password `oracle2`.
17. Execute a statement to display the value of the `CURRENT_USER()` function.
18. Exit all `mysql` prompts.

Solutions 6-4: Using the PAM Authentication Plugin

Tasks

Note: This practice requires the use of multiple terminal windows for multiple `mysql` client sessions.

1. Execute the following commands at a terminal window logged in as `root` to create the Linux user `pamuser1` with the password `oracle1`.

   Enter the following at a terminal window, and receive the results shown below:

```
$ su –
Password: oracle
# useradd pamuser1
# passwd pamuser1
Changing password for user pamuser1.
New password: oracle1
BAD PASSWORD: it is based on a dictionary word
Retype new password: oracle1
passwd: all authentication tokens updated successfully.
```

2. Using a text editor, create a file in the `/etc/pam.d/` directory called `mysql-dba-course` containing the following text:

```
#%PAM-1.0
auth            include         password-auth
account         include         password-auth
```

   Enter the following at the terminal window used in the preceding step, and add the preceding contents to the file, before saving it as `/etc/pam.d/mysql-dba-course`:

```
# gedit /etc/pam.d/mysql-dba-course
```

   Note: Use any text editor that you feel comfortable with, but ensure that you are running the editor as `root` so you can save the file to the correct location.

3. Make the file `/etc/shadow` group-readable, and set its group to `mysql` so the MySQL server process can authenticate using PAM.

   Enter the following at the terminal window used in the preceding step, and receive the results shown below:

```
# chmod 440 /etc/shadow
# chgrp mysql /etc/shadow
```

4. Install the `authentication_pam` plugin.

   Enter the following at a `mysql` prompt and receive the results shown below:

```
mysql> INSTALL PLUGIN authentication_pam
    -> SONAME 'authentication_pam.so';
Query OK, 0 rows affected (0.00 sec)
```

   Note: To keep the plugin loaded across server restarts, use the `INSTALL PLUGIN` statement. See the “Installing and Uninstalling Plugins” page in the *MySQL Reference Manual* for details*:* http://dev.mysql.com/doc/refman/5.6/en/server-plugin-loading.html

5. Show all plugins to verify that the PAM authentication plugin is enabled.

   Enter the following at a `mysql` prompt, and receive the results shown below:

```
mysql> SHOW PLUGINS;
+--------------------------+----------+--------------------+----------------------+-------------+
| Name                     | Status   | Type               | Library              | License     |
+--------------------------+----------+--------------------+----------------------+-------------+
| binlog                   | ACTIVE   | STORAGE ENGINE     | NULL                 | PROPRIETARY |
| mysql_native_password    | ACTIVE   | AUTHENTICATION     | NULL                 | PROPRIETARY |
...
| daemon_memcached         | ACTIVE   | DAEMON             | libmemcached.so      | PROPRIETARY |
| authentication_pam       | ACTIVE   | AUTHENTICATION     | authentication_pam.so | PROPRIETARY |
+--------------------------+----------+--------------------+----------------------+-------------+
45 rows in set (0.00 sec)
```

6. Create a MySQL user called `pamuser1` that can log in from the local host. Ensure that it is identified with PAM authentication, using the PAM service `mysql-dba-course`.

   Enter the following at a `mysql` prompt, and receive the results shown below:

```
mysql> CREATE USER pamuser1@localhost
    -> IDENTIFIED WITH authentication_pam AS 'mysql-dba-course';
Query OK, 0 rows affected (0.00 sec)
```

   Note that you have not provided a password for this user to MySQL. Also note that you must add quotes around names that have special characters, but it is not required if the user or host does not have special characters, as with `pamuser` and `localhost`.

7. Grant the `SELECT` privilege on the `world_innodb.City` table to the newly created `pamuser1` user.

   Enter the following at a `mysql` prompt, and receive the results shown below:

```
mysql> GRANT SELECT ON world_innodb.City TO pamuser1@localhost;
Query OK, 0 rows affected (0.00 sec)
```

8. At a new terminal window, attempt to log in to MySQL as `pamuser1`, using the `cleartext` plugin and the password `oracle1`.

   Enter the following at a new terminal window, and receive the results shown below:

```
$ mysql --enable-cleartext-plugin -upamuser1 -p
Enter password: oracle1
Welcome to the MySQL monitor.  Commands end with ; or \g.
...
```

9. Execute a statement to display the value of the `CURRENT_USER()` function.

   Enter the following at the `mysql` prompt launched in the preceding step, and receive the results shown below:

```
mysql> SELECT CURRENT_USER();
+--------------------+
| CURRENT_USER()     |
+--------------------+
| pamuser1@localhost |
+--------------------+
1 row in set (0.00 sec)
```

10. Exit the mysql prompt logged in as `pamuser1`.

    Enter the following at the `mysql` prompt used in the preceding step, and receive the results shown below:

    ```
    mysql> EXIT
    Bye
    ```

11. Create a Linux user `pamuser2` with the password `oracle2`, using the same technique as at the beginning of this practice.

    Enter the following at the `root` terminal window used earlier, and receive the results shown below:

    ```
    # useradd pamuser2
    # passwd pamuser2
    Changing password for user pamuser2.
    New password: oracle2
    BAD PASSWORD: it is based on a dictionary word
    Retype new password: oracle2
    passwd: all authentication tokens updated successfully.
    ```

12. Create a Linux group `dba`, and add the new `pamuser2` user to that group by using the following commands at a `root` terminal prompt:

    Enter the following at the `root` terminal window used in the preceding step, and receive the results shown below:

    ```
    # groupadd dba
    # usermod -G dba pamuser2
    ```

13. Create a `mysql` user called `world_admin` that can log in from the local host. Give it a cryptic password. Grant all privileges on the `world_innodb` database to the new user.

    Enter the following at a `mysql` prompt logged in as `root`, and receive the results shown below:

    ```
    mysql> CREATE USER world_admin@localhost IDENTIFIED BY 'u=aX;yö#Qq';
    Query OK, 0 rows affected (0.00 sec)
    mysql> GRANT ALL PRIVILEGES ON world_innodb.*
        -> TO world_admin@localhost;
    Query OK, 0 rows affected (0.00 sec)
    ```

    The account is not used directly, so use a password that is as cryptic as you can make it.

14. Create a default proxy account with an empty user name and host name that is identified with PAM authentication, using the PAM service `mysql-dba-course` and the mapping `dba=world_admin`.

    Enter the following at the `mysql` prompt, and receive the results shown below:

    ```
    mysql> CREATE USER ''@''
        ->   IDENTIFIED WITH authentication_pam
        -> AS 'mysql-dba-course, dba=world_admin';
    Query OK, 0 rows affected (0.00 sec)
    ```

15. Grant the default proxy account the PROXY privilege on the world_admin@localhost account created previously.

    ```
    mysql> GRANT PROXY ON world_admin@localhost TO ''@'';
    Query OK, 0 rows affected (0.00 sec)
    ```

16. At a new terminal window, attempt to log in to MySQL as pamuser2, using the cleartext plugin and the password oracle2.

    Enter the following at a new terminal window, and receive the results shown below:

    ```
    $ mysql --enable-cleartext-plugin -upamuser2 -p
    Enter password: oracle2
    Welcome to the MySQL monitor.  Commands end with ; or \g.
    ...
    ```

17. Execute a statement to display the value of the CURRENT_USER() function.

    Enter the following at the mysql prompt launched in the preceding step, and receive the results shown below:

    ```
    mysql> SELECT CURRENT_USER();
    +-----------------------+
    | CURRENT_USER()        |
    +-----------------------+
    | world_admin@localhost |
    +-----------------------+
    1 row in set (0.00 sec)
    ```

    Having logged in as pamuser2, you have been authenticated using PAM with the Linux password, and mapped to the world_admin MySQL user using the default proxy account.

18. Exit all mysql prompts.

Practice 6-5: Additional Practice

Overview

In this practice, you use the information covered in this lesson to implement user accounts and privileges by using the `mysql` client with the `world_innodb` database.

Tasks

Note: This practice requires the use of multiple terminal windows for multiple `mysql` client sessions.

1. Log in to the `mysql` client with the `root` user account. Verify whether any anonymous accounts exist on the server. If there are any, remove them now.
2. Create a new user account called `Stefan` which can only connect locally, with a password of `weak`. Confirm that the new user is now included in the `mysql` database.
3. Give `Stefan` all privileges to the `world_innodb` database. Confirm the updated privileges.
4. Open a second terminal window, and log in to `mysql` as `Stefan`. Verify that the new account works.
5. The password for the `Stefan` account is weak. Set a new password of '`new_pass`'. Log out of this `Stefan` session
6. Using the `mysql` client session logged in as `root`, revoke all privileges on the `City` table previously granted to the `Stefan` account.
7. Allow `Stefan` to `SELECT` all columns in the `City` table. Log out of this `root` session. Confirm the updated privileges.
8. Using the second terminal window (opened earlier for the user `Stefan`), log in as `Stefan` again and verify that the account still works, by executing a `SELECT` on each `world_innodb` table. Log out of the `Stefan` session.
9. From the previous `root` session window, create a new user called `UserGroup4_01` who has the same privileges as `Stefan`, with a password of `0004nq2`. Confirm that the new user exists.
10. Remove the new user `UserGroup4_01`. Confirm that the user no longer exists by attempting to show the grants for this user.
11. Start two separate queries in different terminal windows, and use a third to show the processes as they are running.

    Note: For this exercise to work properly, observe the order and timing of the following instructions exactly. Read over the entire set of instructions and execute the instructions in quick succession.

    a. From the previous `root` session window, change the prompt to `t1` to distinguish it from the other two windows. Execute an initial `SHOW PROCESSLIST` statement that shows only the statement itself.

b. Open a second terminal window. Launch the `mysql` client by using the newly created user, `Stefan` (username: `Stefan`, password: `new_pass`). Change this `mysql` session prompt to `t2`. Change the database to use the `world_innodb` database.

c. Open a third terminal window. Launch the `mysql` client by using the `admin` login path. Change this `mysql` session prompt to `t3`. Change the database to use the `world_innodb` database.

12. Start the queries from the separate windows.

a. In the `t1` session, execute the following `SELECT` statement:

```
t1> SELECT SLEEP(60);
```

b. In the `t2` session, send the following `SELECT` statement:

```
t2> SELECT * FROM City, Country, CountryLanguage
 -> LIMIT 10000000;
```

c. In the `t3` session, within a few seconds resend the `SHOW PROCESSLIST` statement and note the existence of the two new processes.

d. In the `t3` session, within several more seconds resend `SHOW PROCESSLIST` again (with the `\G` terminator for easy-to-read results) and note the difference in the `Time` column, indicating the number of seconds the processes have been running.

Solutions 6-5: Additional Practice

Tasks

Note: This practice requires the use of multiple terminal windows for multiple `mysql` client sessions.

1. Log in to the `mysql` client with the `root` user account. Verify whether any anonymous accounts exist on the server. If there are any, remove them now.

   Enter the following at a terminal window, and receive the result shown below:

```
$ mysql --login-path=admin
...

mysql> SELECT user, host, password FROM mysql.user
    -> WHERE user='';
+------+------+----------+
| user | host | password |
+------+------+----------+
|      |      |          |
+------+------+----------+
1 row in set (0.00 sec)
mysql> DROP USER ''@'';
Query OK, 0 rows affected (0.00 sec)
```

2. Create a new user account called `Stefan` which can only connect locally, with a password of `weak`.

   a. Enter the following at a `mysql` prompt, and receive the results shown below:

```
mysql> CREATE USER 'Stefan'@'localhost'
    -> IDENTIFIED BY 'weak';
Query OK, 0 rows affected (0.00 sec)
```

   b. Confirm that the new user is now included in the `mysql` database:

```
mysql> SELECT user, host, password FROM mysql.user
    -> WHERE user = 'Stefan';
+--------+-----------+-------------------------------------------+
| user   | host      | password                                  |
+--------+-----------+-------------------------------------------+
| Stefan | localhost | *2924FCAB604BD710DD8B0B223B67ABEBF826D10D |
+--------+-----------+-------------------------------------------+
```

3. Give `Stefan` all privileges to the `world_innodb` database.

   a. Enter the following at a `mysql` prompt, and receive the results shown below:

```
mysql> GRANT ALL ON world_innodb.* TO 'Stefan'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

b. Confirm the updated privileges:

```
mysql> SHOW GRANTS FOR 'Stefan'@'localhost'\G
*************************** 1. row ***************************
Grants for Stefan@localhost: GRANT USAGE ON *.* TO
'Stefan'@'localhost' IDENTIFIED BY PASSWORD
'*2924FCAB604BD710DD8B0B223B67ABEBF826D10D'
*************************** 2. row ***************************
Grants for Stefan@localhost: GRANT ALL PRIVILEGES ON `world_innodb`.*
TO 'Stefan'@'localhost'
2 rows in set (0.00 sec)
```

4. Open a second terminal window, and log in to `mysql` as `Stefan`.

   Enter the following at a `mysql` prompt, and receive the results shown below:

```
$ mysql -uStefan –pweak
Welcome to the MySQL monitor.  Commands end with ; or \g.
...
mysql>
```

5. The password for the `Stefan` account is weak. Set a new password of `new_pass`. Log out of this `Stefan` session.

   a. Enter the following at a `mysql` prompt, and receive the result shown below:

```
mysql> SET PASSWORD FOR 'Stefan'@'localhost' = PASSWORD('new_pass');
Query OK, 0 rows affected (0.00 sec)

mysql> EXIT
Bye
```

   b. Verify that the new password works:

```
$ mysql -uStefan -pnew_pass
Welcome to the MySQL monitor.  Commands end with ; or \g.
...
mysql> EXIT
Bye
```

6. Using the `mysql` client session logged in as `root`, revoke all privileges on the `City` table previously granted to the `Stefan` account.

   Note: You must have the privilege being granted, along with the `GRANT OPTION` privilege to grant or revoke privileges. That is why you use the `root` account.

   a. Enter the following at a terminal window and receive the results shown below:

   ```
   $ mysql -uroot -p
   Enter password: oracle
   ...

   mysql> REVOKE ALL ON world_innodb.* FROM 'Stefan'@'localhost';
   Query OK, 0 rows affected (0.00 sec)

   mysql> GRANT ALL ON world_innodb.Country TO 'Stefan'@'localhost';
   Query OK, 0 rows affected (0.00 sec)

   mysql> GRANT ALL ON world_innodb.CountryLanguage
       -> TO 'Stefan'@'localhost';
   Query OK, 0 rows affected (0.00 sec)
   ```

   b. Confirm the updated privileges:

   ```
   mysql> SHOW GRANTS FOR 'Stefan'@'localhost'\G
   *************************** 1. row ***************************
   Grants for Stefan@localhost: GRANT USAGE ON *.* TO
   'Stefan'@'localhost' IDENTIFIED BY PASSWORD
   '*B6408F4D32E8BEC631EF224B6F743F3340E6E744'
   *************************** 2. row ***************************
   Grants for Stefan@localhost: GRANT ALL PRIVILEGES ON
   `world_innodb`.`CountryLanguage` TO 'Stefan'@'localhost'
   *************************** 3. row ***************************
   Grants for Stefan@localhost: GRANT ALL PRIVILEGES ON
   `world_innodb`.`Country` TO 'Stefan'@'localhost'
   3 rows in set (0.00 sec)
   ```

   The user has privileges for all the `world_innodb` tables, except for `City`.

7. Allow `Stefan` to `SELECT` all columns in the `City` table. Log out of this `root` session.

   a. Enter the following at a mysql prompt, and receive the results shown below:

   ```
   mysql> GRANT SELECT (Name, Population) ON world_innodb.City
       -> TO 'Stefan'@'localhost';
   Query OK, 0 rows affected (0.00 sec)
   ```

b. Confirm the updated privileges:

```
mysql> SHOW GRANTS FOR 'Stefan'@'localhost'\G
*************************** 1. row ***************************
Grants for Stefan@localhost: GRANT USAGE ON *.* TO
'Stefan'@'localhost' IDENTIFIED BY PASSWORD
'*B6408F4D32E8BEC631EF224B6F743F3340E6E744'
*************************** 2. row ***************************
Grants for Stefan@localhost: GRANT ALL PRIVILEGES ON
`world_innodb`.`CountryLanguage` TO 'Stefan'@'localhost'
*************************** 3. row ***************************
Grants for Stefan@localhost: GRANT SELECT (Population, Name) ON
`world_innodb`.`City` TO 'Stefan'@'localhost'
*************************** 4. row ***************************
Grants for Stefan@localhost: GRANT ALL PRIVILEGES ON
`world_innodb`.`Country` TO 'Stefan'@'localhost'
4 rows in set (0.00 sec)

mysql> EXIT
Bye
```

The user now has privileges for all the `world_innodb` tables, including the `Name` and `Population` columns of the `City` table.

8. Using the second terminal window (opened earlier for the user `Stefan`), log in as `Stefan` again and verify that the account still works, by executing a `SELECT` on each `world_innodb` table.

   a. Enter the following at a mysql prompt, and receive the results shown below:

```
$ mysql -uStefan -pnew_pass
Welcome to the MySQL monitor.  Commands end with ; or \g.
...
```

b. Test the access to the `Country` table:

```
mysql> USE world_innodb
...
Database changed
mysql> SELECT * from Country\G
...
*************************** 239. row ***************************
          Code: ZWE
          Name: Zimbabwe
     Continent: Africa
        Region: Eastern Africa
   SurfaceArea: 390757.00
     IndepYear: 1980
    Population: 11669000
LifeExpectancy: 37.8
           GNP: 5951.00
        GNPOld: 8670.00
     LocalName: Zimbabwe
GovernmentForm: Republic
   HeadOfState: Robert G. Mugabe
       Capital: 4068
         Code2: ZW
239 rows in set (0.00 sec)
```

c. Test the access to the `CountryLanguage` table:

```
mysql> SELECT * from CountryLanguage;
...
| ZMB         | Bemba                      | F          |       29.7 |
| ZMB         | Chewa                      | F          |        5.7 |
| ZMB         | Lozi                       | F          |        6.4 |
| ZMB         | Nsenga                     | F          |        4.3 |
| ZMB         | Nyanja                     | F          |        7.8 |
| ZMB         | Tongan                     | F          |       11.0 |
| ZWE         | English                    | T          |        2.2 |
| ZWE         | Ndebele                    | F          |       16.2 |
| ZWE         | Nyanja                     | F          |        2.2 |
| ZWE         | Shona                      | F          |       72.1 |
+-------------+----------------------------+------------+------------+
984 rows in set (0.01 sec)
```

d. Test the access to the `City` table:

```
mysql> SELECT * FROM City;
ERROR 1143 (42000): SELECT command denied to user 'Stefan'@'localhost'
for column 'ID' in table 'city'
mysql> SELECT Name FROM City;
...
| Nablus                            |
| Rafah                             |
| New City                          |
+-----------------------------------+
4079 rows in set (0.00 sec)
```

Note that you cannot select all columns (`*`) from `City`, given the privileges granted.

e. Log out of the `Stefan` session:

```
mysql> EXIT
Bye
```

9. From the previous `root` session window, create a new user called `UserGroup4_01` who has the same privileges as `Stefan`, with a password of `0004nq2`.

   a. Enter the following at the `mysql` prompt logged in as `root`, and receive the results shown below:

```
mysql> CREATE USER 'UserGroup4_01'@'localhost'
    -> IDENTIFIED BY '0004nq2';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL ON world_innodb.Country
    -> TO 'UserGroup4_01'@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL ON world_innodb.CountryLanguage
    -> TO 'UserGroup4_01'@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT SELECT ON world_innodb.City
    -> TO 'UserGroup4_01'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

   b. Confirm that the new user exists:

```
mysql> SHOW GRANTS FOR 'UserGroup4_01'@'localhost'\G
*************************** 1. row ***************************
Grants for UserGroup4_01@localhost: GRANT USAGE ON *.* TO
'UserGroup4_01'@'localhost' IDENTIFIED BY PASSWORD
'*926ED64069CED09E0AAAED7677CAE9C44CF5C469'
*************************** 2. row ***************************
Grants for UserGroup4_01@localhost: GRANT ALL PRIVILEGES ON
`world_innodb`.`Country` TO 'UserGroup4_01'@'localhost'
*************************** 3. row ***************************
Grants for UserGroup4_01@localhost: GRANT SELECT ON
`world_innodb`.`City` TO 'UserGroup4_01'@'localhost'
*************************** 4. row ***************************
Grants for UserGroup4_01@localhost: GRANT ALL PRIVILEGES ON
`world_innodb`.`CountryLanguage` TO 'UserGroup4_01'@'localhost'
4 rows in set (0.00 sec)
```

10. Remove the new user `UserGroup4_01`.

   a. Enter the following at a `mysql` prompt, and receive the results shown below:

```
mysql> DROP USER 'UserGroup4_01'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

b. Confirm that the user no longer exists by attempting to show the grants for this user:

```
mysql> SHOW GRANTS FOR 'UserGroup4_01'@'localhost'\G
ERROR 1141 (42000): There is no such grant defined for user
'UserGroup4_01' on host 'localhost'
```

11. Start two separate queries in different terminal windows, and use a third to show the processes as they are running.

    Note: The order and timing of the instructions below must be followed exactly for this exercise to work properly. Read over the entire set of instructions and execute the instructions in quick succession.

    a. From the previous `root` session window, change the prompt to `t1` to distinguish it from the other two windows. Execute an initial `SHOW PROCESSLIST` statement that shows only the statement itself.

```
mysql> PROMPT t1> ;
PROMPT set to 't1> '
t1> SHOW PROCESSLIST;
+----+------+-----------+------+---------+------+-------+------------------+
| Id | User | Host      | db   | Command | Time | State | Info             |
+----+------+-----------+------+---------+------+-------+------------------+
|  4 | root | localhost | NULL | Query   |    0 | init  | SHOW PROCESSLIST |
+----+------+-----------+------+---------+------+-------+------------------+
1 row in set (0.00 sec)
```

    b. Open a second terminal window. Launch the `mysql` client by using the newly created user, `Stefan` (username: `Stefan`, password: `new_pass`). Change this `mysql` session prompt to `t2`. Change the database to use the `world_innodb` database.

```
$ mysql -uStefan -pnew_pass
...
mysql> PROMPT t2> ;
PROMPT set to 't2> '
t2> USE world_innodb;
...
Database changed
```

    c. Open a third terminal window. Launch the `mysql` client by using the `admin` login path. Change this `mysql` session prompt to `t3`. Change the database to use the `world_innodb` database. Enter the following, and receive the results shown below:

```
$ mysql --login-path=admin
...
mysql> PROMPT t3> ;
PROMPT set to 't3> '
t3> USE test;
...
Database changed
```

12. Start the queries from the separate windows. Enter the following, and receive the results shown below:

    a. In the `t1` session, execute a `SELECT SLEEP(60);` statement:

    ```
    t1> SELECT SLEEP(60);
    (pauses while processing for one minute)
    +-----------+
    | SLEEP(60) |
    +-----------+
    |         0 |
    +-----------+
    1 row in set (1 min 0.08 sec)
    ```

    b. In the `t2` session, execute the following statement:

    ```
    t2> SELECT Code FROM City, Country, CountryLanguage
        -> LIMIT 10000000;
    (The output pauses while processing for approximately 40 seconds)
    ...
    10000000 rows in set (41.91 sec)
    ```

    c. In the `t3` session, within a few seconds resend the `SHOW PROCESSLIST` statement and note the existence of the two new processes:

    ```
    t3> SHOW PROCESSLIST;
    +----+--------+-----------+--------------+---------+------+--------------+---------------+
    | Id | User   | Host      | db           | Command | Time | State        | Info          |
    +----+--------+-----------+--------------+---------+------+--------------+---------------+
    |  4 | root   | localhost | NULL         | Query   |    4 | User sleep   | SELECT SLEEP… |
    | 10 | Stefan | localhost | world_innodb | Query   |    2 | Sending data | SELECT Code F…|
    | 12 | root   | localhost | test         | Query   |    0 | init         | SHOW PROCESSL…|
    +----+--------+-----------+--------------+---------+------+--------------+---------------+
    3 rows in set (0.00 sec)
    ```

    – The preceding output is truncated to fit the page. Note that the `Time` column value depends upon when you execute this statement.

d. In the t3 session, within several more seconds resend SHOW PROCESSLIST again (with the **\G** terminator for easy-to-read results) and note the difference in the Time column, indicating the number of seconds the processes have been running:

```
t3> SHOW PROCESSLIST\G
*************************** 1. row ***************************
     Id: 8
   User: root
   Host: localhost
     db: NULL
Command: Query
   Time: 22
  State: User sleep
   Info: SELECT SLEEP(60)
*************************** 2. row ***************************
     Id: 10
   User: Stefan
   Host: localhost
     db: world_innodb
Command: Sleep
   Time: 20
  State:
   Info: NULL
*************************** 3. row ***************************
     Id: 12
   User: root
   Host: localhost
     db: test
Command: Query
   Time: 0
  State: init
   Info: SHOW PROCESSLIST
3 rows in set (0.00 sec)
```

You can now exit each of the terminal mysql clients and close the windows. Do not remove the Stefan user account, because you use it in a later practice.

Practices for Lesson 7: Security

Chapter 7

Practice 7-1: Quiz – MySQL Security

Overview

In this quiz, you answer questions about MySQL security.

Quiz Questions

Choose the best answer from those provided for each multiple choice or True/False question.

1. A `root` account has full privileges to perform any database operation, so you should not give it access to the `user` table in the `mysql` database.
    a. True
    b. False
2. The following are the types of MySQL server and data security risks.
    a. Eavesdropping
    b. Altering
    c. Playback
    d. Denial of service
    e. All of the above
3. MySQL uses security based on ____________________ for all connections, queries, and other operations.
    a. atomic control lines
    b. access control lists
    c. data access security
4. The most common MySQL installation security risks fall in the following categories:
    a. networks, operating systems, and file systems
    b. networks, secure connections, and users
    c. users and data
5. Protect your data from unauthorized access by using single quotation marks around numeric and string values.
    a. True
    b. False
6. Unencrypted connections are acceptable for moving data securely over a network, as long as users have passwords.
    a. True
    b. False

7. MySQL supports ______________________ connections between MySQL clients and the server for encryption of data.
   a. SSL-only
   b. SSL and SSH
   c. `--compress`
8. All MySQL installations support SSL connections.
   a. True
   b. False

Solutions 7-1: Quiz – MySQL Security

Quiz Solutions

1. **b**. False. Do not ever give anyone (except MySQL `root` accounts) access to the `user` table in the `mysql` database!
2. **e**. All of the above
3. **b**. access control lists
4. **a**. networks, operating systems, and file systems
5. **a**. True
6. **b**. False. If you use unencrypted connections between the client and the server, a user with access to the network can watch all the traffic and look at (and modify) the data being sent or received.
7. **b**. SSL (secure socket layer) and SSH (secure shell, specifically for Windows OS) protocols
8. **b**. False. To use SSL connections, your system must support OpenSSL or yaSSL (which comes bundled with MySQL standard installation), but not all forms of the MySQL binaries include the necessary setup.

Practice 7-2: Determining the Status of SSL Connectivity

Overview

In this practice, you use specific SQL statements to determine whether the MySQL server supports SSL connections.

Tasks

1. Within the `mysql` client, check whether a running `mysqld` server supports SSL; do this by displaying the value of the `have_ssl` system variable.
2. Check whether the current server connection uses SSL; do this by examining the value of the `Ssl_cipher` status variable. Then shut down the `mysql` client.

Solutions 7-2: Determining the Status of SSL Connectivity

Tasks

1. Within the `mysql` client, check whether your MySQL server supports SSL; do this by displaying the value of the `have_ssl` system variable. Enter the following in a terminal window, and receive the result shown below:

```
$ mysql --login-path=admin
Welcome to the MySQL monitor.  Commands end with ; or \g.
...

mysql> SHOW VARIABLES LIKE 'have_ssl';
+---------------+----------+
| Variable_name | Value    |
+---------------+----------+
| have_ssl      | DISABLED |
+---------------+----------+
```

   − The result shows that the SSL feature is available in your MySQL Server installation, but is not currently enabled.

2. Check whether your current server connection uses SSL; do this by examining the value of the `Ssl_cipher` status variable. Then shut down the `mysql` client. Enter the following in a terminal window (from the `mysql` client), and receive the result shown below:

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

mysql> EXIT
Bye
```

   − The result shows that no SSL connection cipher is assigned.

Practice 7-3: Additional Practice – Enabling MySQL Support for SSL Connections

Overview

In this practice, set the appropriate variables to permit the MySQL server and client to use an SSL connection.

Duration

This practice should take approximately 10 minutes to complete.

Tasks

1. Create the OpenSSL files needed to enable SSL connections by following the steps in the solution. Name the new folder `newcerts` and place it in the current `/etc/` directory.
2. Confirm the creation of the eight SSL files required for SSL connection:
   - `ca-cert.pem`
   - `client-cert.pem`
   - `client-req.pem`
   - `server-key.pem`
   - `ca-key.pem`
   - `client-key.pem`
   - `server-cert.pem`
   - `server-req.pem`
3. Use the `Stefan` user account (created in practices for the “User Management” lesson) to set up the SSL within the `mysql` client. From the `root` account, grant privileges to the `Stefan` account for `SELECT` on the `user` table of the `mysql` database, and require SSL-encrypted connections, within the same `GRANT` statement.
4. Restart the server with the option to enable the SSL certificate authority, server certificate, and server key.
5. Attempt a standard login to the **`mysql`** client as `Stefan`. What is the result?
6. Log in to the `Stefan` account, adding the option to enable the SSL connection.
7. List the current value of the `have_ssl` and `Ssl_ciper` variables. Does this confirm the SSL connection status?
8. Exit the `mysql` session.
9. In preparation for the next practice, stop and restart the server without the SSL connection options.

Solutions 7-3: Additional Practice – Enabling MySQL Support for SSL Connections

Tasks

1. Create the OpenSSL files needed to enable SSL connections by following the steps in the solution. Name the new folder `newcerts` and place it in the current `/etc/` directory. Enter the following in a terminal window, and receive the results shown below:

```
$ su -
Password: oracle
# cd /etc
# mkdir newcerts
# cd newcerts
# openssl genrsa 2048 > ca-key.pem
Generating RSA private key, 2048 bit long modulus
...........................+++
...........+++
e is 65537 (0x10001)
# openssl req -new -x509 -nodes -days 1000 \
        -key ca-key.pem > ca-cert.pem
You are about to be asked to enter information that will be
incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name or
a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]: US
State or Province Name (full name) []: Texas Locality
Name (eg, city) [Default City]: Dallas Organization
Name (eg, company) [Default Company Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:
Email Address []:
# openssl req -newkey rsa:2048 -days 1000 -nodes \
        -keyout server-key.pem > server-req.pem
Generating a 2048 bit RSA private key
.............+++
..+++
writing new private key to 'server-key.pem'
-----
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or
a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]: State
or Province Name (full name) []:
Locality Name (eg, city) [Default City]:
```

```
Organization Name (eg, company) [Default Company Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:
Email Address []:
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
# openssl x509 -req -in server-req.pem -days 1000 -CA ca-cert.pem \
         -CAkey ca-key.pem -set_serial 01 > server-cert.pem
Signature ok
subject=/C=XX/L=Default City/O=Default Company Ltd
Getting CA Private Key
# openssl req -newkey rsa:2048 -days 1000 -nodes \
     -keyout client-key.pem > client-req.pem
Generating a 2048 bit RSA private key
.....................................................................
......................................+++
.....................................................................
.................................................+++
writing new private key to 'client-key.pem'
-----
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or
a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]: State
or Province Name (full name) []:
Locality Name (eg, city) [Default City]:
Organization Name (eg, company) [Default Company Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:
Email Address []:
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
# openssl x509 -req -in client-req.pem -days 1000 -CA ca-cert.pem \
         -CAkey ca-key.pem -set_serial 01 > client-cert.pem
Signature ok
subject=/C=XX/L=Default City/O=Default Company Ltd
Getting CA Private Key
```

2. Confirm the creation of the eight SSL files required for SSL connection. Enter the following in a terminal window (at the same shell prompt as the previous step), and receive the result shown below:

```
# ls
ca-cert.pem  client-cert.pem  client-req.pem   server-key.pem
ca-key.pem   client-key.pem   server-cert.pem  server-req.pem
```

3. Use the `Stefan` user account (created in practices for the "User Management" lesson) to set up SSL within the `mysql` client. From the `root` account, grant privileges to the `Stefan` account for SELECT on the `user` table of the `mysql` database, and require SSL-encrypted connections, within the same GRANT statement. Confirm the privileges. Enter the following in a terminal window (at the `mysql` client prompt), and receive the result shown below:

```
$ mysql --login-path=admin
Welcome to the MySQL monitor.  Commands end with ; or \g.
...

mysql> GRANT SELECT ON mysql.user TO 'Stefan'@'localhost'
     > IDENTIFIED BY 'new_pass' REQUIRE SSL;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS FOR 'Stefan'@'localhost'\G
*************************** 1. row ***************************
Grants for Stefan@localhost: GRANT USAGE ON *.* TO
'Stefan'@'localhost' IDENTIFIED BY PASSWORD
'*B6408F4D32E8BEC631EF224B6F743F3340E6E744' REQUIRE SSL
*************************** 2. row ***************************
Grants for Stefan@localhost: GRANT SELECT ON `mysql`.`user` TO
'Stefan'@'localhost'
*************************** 3. row ***************************
Grants for Stefan@localhost: GRANT ALL PRIVILEGES ON
`world_innodb`.`Country` TO 'Stefan'@'localhost'
*************************** 4. row ***************************
Grants for Stefan@localhost: GRANT ALL PRIVILEGES ON
`world_innodb`.`CountryLanguage` TO 'Stefan'@'localhost'
*************************** 5. row ***************************
Grants for Stefan@localhost: GRANT SELECT (Population, Name) ON
`world_innodb`.`City` TO 'Stefan'@'localhost'
5 rows in set (0.00 sec)

mysql> EXIT
```

4. Restart the server with the option to enable the SSL certificate authority, server certificate, and server key. Enter the following in a terminal window (at a Linux terminal prompt logged in as `root`), and receive the results shown below:

```
# service mysql restart --ssl-ca=/etc/newcerts/ca-cert.pem \
     --ssl-cert=/etc/newcerts/server-cert.pem \
     --ssl-key=/etc/newcerts/server-key.pem
Shutting down MySQL..                                      [  OK  ]
Starting MySQL...                                          [  OK  ]
```

5. Attempt a standard login to the **mysql** client as Stefan. What is the result?

```
$ mysql -uStefan -pnew_pass
ERROR 1045 (28000): Access denied for user 'Stefan'@'localhost' (using
password: YES)
```

   You get an error for this user because it is now an SSL-required account, which requires an SSL option to log in to the account.

6. Log in to the Stefan account, adding the option to enable the SSL connection.

```
$ mysql -uStefan -pnew_pass --ssl-ca=/etc/newcerts/ca-cert.pem
Welcome to the MySQL monitor.  Commands end with ; or \g.
...
```

7. List the current value of the have_ssl and Ssl_ciper variables. Does this confirm the SSL connection status?

```
mysql> SHOW VARIABLES LIKE 'have_ssl';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| have_ssl      | YES   |
+---------------+-------+
1 row in set (0.01 sec)

mysql> SHOW STATUS LIKE 'Ssl_cipher';
+---------------+--------------------+
| Variable_name | Value              |
+---------------+--------------------+
| Ssl_cipher    | DHE-RSA-AES256-SHA |
+---------------+--------------------+
1 row in set (0.00 sec)
```

8. Exit the mysql session.

```
mysql> EXIT
Bye
```

9. In preparation for the next practice, stop and restart the server without the SSL connection options. Enter the following from a terminal logged in as root.

```
# service mysql restart
Shutting down MySQL...                                     [  OK  ]
Starting MySQL.                                            [  OK  ]
```

Practices for Lesson 8:

Table Maintenance and Exporting-Importing

Chapter 8

Practice 8-1: Quiz – Table Maintenance

Overview

In this quiz, you answer questions about table maintenance.

Quiz Questions

Choose the best answer from those provided for each multiple choice or True/False question.

1. You can use both `myisamchk` and `mysqlcheck` command line utilities to check InnoDB tables.
   a. True
   b. False
2. `myisamchk` and `mysqlcheck` both access tables directly, and do not communicate with the MySQL server.
   a. True
   b. False
3. Oracle recommends that you first run `mysqlcheck`:
   a. With the `––force` option
   b. With no options
   c. With the `--scan ––quick` options
      – Run `mysqlcheck` again if repairs are required.
4. Which of the following maintenance operations can be performed on InnoDB tables?
   a. `ANALYZE TABLE`
   b. `CHECK TABLE`
   c. `CHECKSUM TABLE`
   d. `REPAIR TABLE`
   e. `OPTIMIZE TABLE`
   f. All of the above
5. When you optimize a table, the unused space is reclaimed when rows are deleted from the table. To optimize a table, you can use the `OPTIMIZE TABLE` statement or the ______________ utility with the ______________ option.
   a. `mysqlcheck --optimize`
   b. `myisamchk --optimize`
   c. `mysqlcheck --reclaim`
   d. None of the above

6. Prior to running `myisamchk` on a MyISAM table, you should lock and flush the tables or stop the server to ensure that the server does not access the tables while other processes are using them.
    a. True
    b. False
7. To repair an InnoDB table after a crash from which InnoDB cannot automatically recover, restart the server by using the _____________ option.
    a. `--recover`
    b. `-recover -f`
    c. `--innodb_force_recovery`

Solutions 8-1: Quiz – Table Maintenance

Quiz Solutions

1. b. False. Of these two utilities, only `mysqlcheck` can be used to check InnoDB tables.
2. b. False. `mysqlcheck` does not connect directly to the tables, but communicates through the MySQL server while it is running.
3. b. No
4. a. `ANALYZE TABLE`, b. `CHECK TABLE`, c. `CHECKSUM TABLE`, and e. `OPTIMIZE TABLE`. `REPAIR TABLE` is not supported by InnoDB.
5. a. `mysqlcheck` with the `--optimize` option
6. a. True
7. c. `--innodb_force_recovery`. Use `innodb_force_recovery` when a corrupt tablespace prevents InnoDB from starting.

Practice 8-2: Using Table Maintenance SQL Statements

Overview

In this practice, you use SQL statements to check and optimize a table.

Tasks

1. In the world_innodb database, create a new table called City_temp, which is a copy of the City table. Insert the contents of the City table into the City_temp table:

   ```
   CREATE TABLE City_temp LIKE City;
   INSERT INTO City_temp SELECT * FROM City;
   ```

   – Show the current table status, for future comparison, using the SHOW TABLE STATUS statement.

2. Check the City_temp table for any problems. Does the table currently have a problem?
3. Create “holes” (gaps between rows resulting from deletes or updates) in the City_temp table data by removing several rows, using the ID column, as follows:

   ```
   DELETE FROM City_temp WHERE ID BETWEEN 1001 AND 2000;
   ```

   – Confirm that the rows were deleted, and the Data_Length and Index_Length statistics have changed.

4. Update the table’s statistics by running ANALYZE TABLE on the City_temp table. Confirm that the analysis took place. Then exit the mysql client session.

Solutions 8-2: Using Table Maintenance SQL Statements

Tasks

1. In the `world_innodb` database, create a new table called `City_temp`, which is a copy of the `City` table. Insert the contents of the `City` table into the `City_temp` table.
   a. Enter the following in a terminal window, and receive the results shown below:

```
$ mysql --login-path=admin
Welcome to the MySQL monitor.  Commands end with ; or \g.
...

mysql> USE world_innodb
Database changed

mysql> CREATE TABLE City_temp LIKE City;
Query OK, 0 rows affected (0.10 sec)
```

   b. Enter the following statement to populate the `City_temp` table, and receive the results shown below:

```
mysql> INSERT INTO City_temp SELECT * FROM City;
Query OK, 4079 rows affected, 1 warning (2.02 sec)
Records: 4079  Duplicates: 0  Warnings: 1
```

   The warning indicates that you are inserting rows from one table into another with an auto increment field while binary logging is enabled. You can safely ignore this warning in the current lesson.

   c. Show the current table status, for future comparison, using the `SHOW TABLE STATUS` statement:

```
mysql> SHOW TABLE STATUS LIKE 'City_temp' \G
*************************** 1. row ***************************
           Name: City_temp
         Engine: InnoDB
        Version: 10
     Row_format: Compact
           Rows: 4188
 Avg_row_length: 97
    Data_length: 409600
Max_data_length: 0
   Index_length: 131072
...
```

   – Take note of the current values for `Rows`, `Data_Length`, and `Index_Length`.

2. Check the `City_temp` table for any problems. Does the table currently have a problem?
   a. Enter the following in the current `mysql` client session, and receive the results shown below:

```
mysql> CHECK TABLE City_temp\G
*************************** 1. row ***************************
   Table: world_innodb.City_temp
      Op: check
Msg_type: status
Msg_text: OK
1 row in set (0.02 sec)
```

3. Create “holes” (gaps between rows resulting from deletes or updates) in the `City_temp` table data by removing several rows. Confirm that the rows were deleted and that the `Data_Length` and `Index_Length` statistics have changed.

   a. Enter the following in the current `mysql` client session, and receive the result shown below:

```
mysql> DELETE FROM City_temp WHERE Id BETWEEN 1001 AND 2000;
Query OK, 1000 rows affected (0.03 sec)
```

   b. Show the table status to confirm that the rows have been removed:

```
mysql> SHOW TABLE STATUS LIKE 'City_temp' \G
*************************** 1. row ***************************
           Name: City_temp
         Engine: InnoDB
        Version: 10
     Row_format: Compact
           Rows: 3109
 Avg_row_length: 131
    Data_length: 409600
Max_data_length: 0
   Index_length: 131072
...
```

   – Note the change in the number of rows. Although this number is an estimate only for InnoDB tables (and may change with each execution of the `SHOW TABLE STATUS` statement), it gives you the general idea that the number of rows has decreased.

   – The `Data_Length` and `Index_Length` numbers remain the same.

4. Update the table’s statistics by running `ANALYZE TABLE` on the `City_temp` table. Confirm that the analysis took place. Then exit the `mysql` client session.

   a. Enter the following in the current `mysql` client session, and receive the results shown below:

```
mysql> ANALYZE TABLE City_temp;
+------------------------+---------+----------+----------+
| Table                  | Op      | Msg_type | Msg_text |
+------------------------+---------+----------+----------+
| world_innodb.City_temp | analyze | status   | OK       |
+------------------------+---------+----------+----------+
1 row in set (1.13 sec)
```

b. Show the new table status to confirm the optimization:

```
mysql> SHOW TABLE STATUS LIKE 'City_temp' \G
*************************** 1. row ***************************
           Name: City_temp
         Engine: InnoDB
        Version: 10
     Row_format: Compact
           Rows: 3079
 Avg_row_length: 106
    Data_length: 327680
Max_data_length: 0
   Index_length: 98304
...
```

Note that the number of rows, average row length, and other values have changed after the analysis.

c. Exit the session:

```
mysql> EXIT
Bye
```

Practice 8-3: Using Table Maintenance Utilities

Overview

In this practice, you use MySQL command line utilities to check and repair a table.

Tasks

1. From a shell prompt, run a check on the `world_innodb` database by using the `mysqlcheck` client program (with the standard username (`-u`) and password (`-p`) options).
2. Run a similar check as in the previous step, but this time use a login path instead of -u and `-p`, and check on all current databases using `mysqlcheck`.
3. Modify the statement used in the previous step to analyze all tables in all databases. Note the messages given by each table, and the differences between those messages.

Solutions 8-3: Using Table Maintenance Utilities

Tasks

1. From a shell prompt, run a check on the `world_innodb` database by using the `mysqlcheck` client program (with the standard username (`-u`) and password (`-p`) options). Enter the following in a terminal window, and receive the results shown below:

```
$ mysqlcheck -uroot -p --databases world_innodb
Enter password: oracle
world_innodb.City                                   OK
world_innodb.Country                                OK
world_innodb.CountryLanguage                        OK
```

   – All eligible tables show that their status is **`OK`**.

2. Run a similar check as in the previous step, but this time use a login path instead of `-u` and `-p`, and check on all current databases using `mysqlcheck`.

   Enter the following in a terminal window, and receive the results shown below:

```
$ mysqlcheck --login-path=admin --all-databases
mysql.columns_priv                                  OK
mysql.db                                            OK
mysql.event                                         OK
...
world_innodb.City_temp                              OK
world_innodb.Country                                OK
world_innodb.CountryLanguage                        OK
```

   – All tables show that their status is **`OK`**.

3. Modify the statement used in the previous step to analyze all tables in all databases. Note the messages given by each table, and the differences between those messages.

   Enter the following in a terminal window, and receive the results shown below:

```
$ mysqlcheck --login-path=admin --all-databases --analyze
mysql.columns_priv                               Table is already up to
date
mysql.db                                         Table is already up to
date
mysql.event                                      Table is already up to
date
mysql.func                                       Table is already up to
date
mysql.general_log
note     : The storage engine for the table doesn't support analyze
mysql.help_category                              Table is already up to
date
...
mysql.slave_worker_info                          OK
mysql.slow_log
note     : The storage engine for the table doesn't support analyze
```

```
mysql.tables_priv                               Table is already up to
date
mysql.time_zone                                 Table is already up to
date
...
sakila.film_actor                               OK
sakila.film_category                            OK
sakila.film_text                                Table is already up to
date
sakila.inventory                                OK
sakila.language                                 OK
...
world_innodb.City_part                          OK
world_innodb.City_temp                          OK
world_innodb.Country                            OK
world_innodb.CountryLanguage                    OK
```

Analyzed tables return a status of `OK`. Tables analyzed in a previous practice (or that have not changed since an earlier `ANALYZE`) report that the `Table is already up to date`, and CSV tables such as `mysql.general_log` report that `The storage engine for the table doesn't support analyze`.

Practice 8-4: Exporting MySQL Data

Overview

In this practice, you use the `SELECT ... INTO OUTFILE` statement to export data. To accomplish this objective:

- Using the default export options, export the contents of the `CountryLanguage` table in the `world_innodb` database.
- Using a comma-separated value output, export the contents of the `CountryLanguage` table in the `world_innodb` database.

Tasks

1. Using the `mysql` client, in the `world_innodb` database, export the contents of the `CountryLanguage` table to a file called `CountryLanguage.txt` in the `/tmp` directory. Use the default export options.
2. In a second terminal window, view the contents of the `/tmp/countryLanguage.txt` file.
   - How are the different columns separated from each other?

     ____
3. In the first terminal window in the mysql client, in the `world_innodb` database, export the contents of the `CountryLanguage` table to a file called `CountryLanguage.csv` in the `/tmp` directory. Specify one row per line in the output file, enclose each value in double quotes and separate the values with commas.
4. In the second terminal window, view the contents of the `/tmp/countryLanguage.csv` file.
   - How are the different columns separated from each other?

     ____

Solutions 8-4: Exporting MySQL Data

Tasks

1. Using the `mysql` client, in the `world_innodb` database, export the contents of the `CountryLanguage` table to a file called `CountryLanguage.txt` in the `/tmp` directory. Use the default export options.

   At a Linux terminal prompt, issue the following commands and receive the results shown:

   ```
   $ mysql -uroot -poracle
   Welcome to the MySQL monitor.  Commands end with ; or \g.
   ...

   mysql> USE world_innodb;
   ...
   Database changed
   mysql> SELECT * INTO OUTFILE '/tmp/CountryLanguage.txt'
       -> FROM CountryLanguage;
   Query OK, 984 rows affected (0.00 sec)
   ```

   – The preceding command creates a file in the `/tmp` directory.

2. In a second terminal window, view the contents of the `/tmp/countryLanguage.txt` file.

   At a Linux terminal prompt, issue the following command and receive the results shown:

   ```
   $ more /tmp/CountryLanguage.txt
   ABW     Dutch   T       5.3
   ABW     English F       9.5
   ABW     Papiamento      F       76.7
   ABW     Spanish F       7.4
   AFG     Balochi F       0.9
   AFG     Dari    T       32.1
   AFG     Pashto  T       52.4
   AFG     Turkmenian      F       1.9
   AFG     Uzbek   F       8.8
   AGO     Ambo    F       2.4
   ...
   ```

   – Press `<space>` to page down, and q to exit the `more` window.

   – How are the different columns separated from each other? The columns are tab-delimited.

3. In the first terminal window in the mysql client, in the `world_innodb` database, export the contents of the `CountryLanguage` table to a file called `CountryLanguage.csv` in the `/tmp` directory. Specify one row per line in the output file, enclose each value in double quotes and separate the values with commas. At a `mysql` prompt, issue the following command and receive the results shown:

```
mysql> SELECT * INTO OUTFILE '/tmp/CountryLanguage.csv'
    -> FIELDS TERMINATED BY ',' ENCLOSED BY '"'
    -> LINES TERMINATED BY '\n'
    -> FROM CountryLanguage;
```

4. In the second terminal window, view the contents of the `/tmp/CountryLanguage.csv` file.

   At a Linux terminal prompt, issue the following command and receive the results shown:

```
$ more /tmp/CountryLanguage.csv
"ABW","Dutch","T","5.3"
"ABW","English","F","9.5"
"ABW","Papiamento","F","76.7"
"ABW","Spanish","F","7.4"
"AFG","Balochi","F","0.9"
"AFG","Dari","T","32.1"
"AFG","Pashto","T","52.4"
"AFG","Turkmenian","F","1.9"
"AFG","Uzbek","F","8.8"
"AGO","Ambo","F","2.4"
...
```

   – How are the different columns separated from each other? The columns are comma-separated.

Practice 8-5: Importing Data

Overview

In this practice, you use the `LOAD DATA INFILE` statement to import data. To accomplish this objective:

- Create a new table based on the `CountryLanguage` table in the `world_innodb` database.
- Import the contents of the text file created in the previous practice into this new table.

Tasks

1. Using the `mysql` client while connected to the `world_innodb` database, create a new table named `CountryLanguage2` that has the same structure as the original `CountryLanguage` table.
2. Show the tables list and confirm that the new table now exists.
3. Issue a `LOAD DATA INFILE` statement that loads the file `/tmp/CountryLanguage.txt` into the `CountryLanguage2` table.
4. Select all the data in the new `CountryLanguage2` table to confirm that the table is now populated. Then exit the `mysql` session.

Solutions 8-5: Importing Data

Tasks

1. Using the `mysql` client while connected to the `world_innodb` database, create a new table named `CountryLanguage2` that has the same structure as the original `CountryLanguage` table.

   At a `mysql` prompt, issue the following command:

```
mysql> CREATE TABLE CountryLanguage2 LIKE CountryLanguage;
Query OK, 0 rows affected (0.86 sec)
```

2. Show the tables list and confirm that the new table now exists.

   At a `mysql` prompt, issue the following command and receive the results shown:

```
mysql> SHOW TABLES;
+------------------------+
| Tables_in_world_innodb |
+------------------------+
| City                   |
| CityLanguage           |
| City_part              |
| City_temp              |
| Country                |
| CountryLanguage        |
| CountryLanguage2       |
+------------------------+
7 rows in set (0.00 sec)
```

3. Issue a `LOAD DATA INFILE` statement that loads the file `/tmp/CountryLanguage.txt` into the `CountryLanguage2` table.

   At a `mysql` prompt, issue the following command and receive the results shown:

```
mysql> LOAD DATA INFILE '/tmp/CountryLanguage.txt' INTO TABLE
    -> CountryLanguage2;
```

4. Select all the data in the new `CountryLanguage2` table to confirm that the table is now populated. Then exit the `mysql` session.

   At a `mysql` prompt, issue the following command and receive the results shown:

```
mysql> SELECT * FROM CountryLanguage2;
+-------------+---------------------------+------------+------------+
| CountryCode | Language                  | IsOfficial | Percentage |
+-------------+---------------------------+------------+------------+
| ABW         | Dutch                     | T          |        5.3 |
| ABW         | English                   | F          |        9.5 |
...
| ZWE         | Ndebele                   | F          |       16.2 |
| ZWE         | Nyanja                    | F          |        2.2 |
| ZWE         | Shona                     | F          |       72.1 |
+-------------+---------------------------+------------+------------+
984 rows in set (0.00 sec)

mysql> EXIT
Bye
```

Practices for Lesson 9: Programming Inside MySQL

Chapter 9

Practice 9-1: Creating Stored Routines

Overview

In this practice, you create and execute a stored procedure and a stored function. To accomplish this objective:

- Use the `world_innodb` database.
- Create and execute a stored procedure that displays the record count for the `City`, `Country`, and `CountryLanguage` tables in the `world_innodb` database.
- Create and execute a stored function that determines take-home pay based on inputting two values: base salary and tax rate.

Tasks

1. Using the `mysql` client, in the `world_innodb` database, enter the following SQL statements to create the `record_count` stored procedure:

```
DELIMITER //
CREATE PROCEDURE record_count ()
BEGIN
   SELECT 'Country count ', COUNT(*) FROM Country;
   SELECT 'City count ', COUNT(*) FROM City;
   SELECT 'CountryLanguage count', COUNT(*) FROM
    CountryLanguage;
END//
DELIMITER ;
```

2. Issue an appropriate command to execute the `record_count` stored procedure.
3. Change to the `test` database, and attempt to execute the `record_count` stored procedure.
   – Why did the `record_count` procedure fail?
   ____
   ____
4. Execute the `record_count` stored procedure, this time qualified with the name of the `world_innodb` database.

   Did the `record_count` stored procedure execute properly? ______________

5. Change to the `world_innodb` database and issue the following SQL statements to create the `pay_check` stored function:

```
DELIMITER //
CREATE FUNCTION pay_check (gross_pay FLOAT(9,2),
  tax_rate FLOAT(3,2))
RETURNS FLOAT(9,2)
NO SQL
BEGIN
   DECLARE net_pay FLOAT(9,2) DEFAULT 0;
   SET net_pay=gross_pay - gross_pay * tax_rate;
   RETURN net_pay;
END//
DELIMITER ;
```

6. Execute the `pay_check` stored function by issuing an appropriate SQL statement, passing in the arguments `100000` and `0.05`.

Solutions 9-1: Creating Stored Routines

Tasks

1. Using the mysql client, in the world_innodb database, enter the following SQL statements to create the record_count stored procedure.

   Enter the following at a Linux terminal and receive the results shown below:

```
$ mysql --login-path=admin
Welcome to the MySQL monitor.  Commands end with ; or \g.
...
mysql> USE world_innodb;
...
Database changed
mysql> DELIMITER //
mysql> CREATE PROCEDURE record_count ()
    -> BEGIN
    ->   SELECT 'Country count ', COUNT(*) FROM Country;
    ->   SELECT 'City count ', COUNT(*) FROM City;
    ->   SELECT 'CountryLanguage count', COUNT(*) FROM
    ->    CountryLanguage;
    -> END//
Query OK, 0 rows affected (0.06 sec)

mysql> DELIMITER ;
```

   The world_innodb database now contains a stored procedure called record_count.

2. Issue an appropriate command to execute the record_count stored procedure.

   Enter the following statement at a mysql prompt, and receive the results shown below:

```
mysql> CALL record_count();
+----------------+----------+
| Country count  | COUNT(*) |
+----------------+----------+
| Country count  |      239 |
+----------------+----------+
1 row in set (0.00 sec)

+-------------+----------+
| City count  | COUNT(*) |
+-------------+----------+
| City count  |     4079 |
+-------------+----------+
1 row in set (0.01 sec)

+-----------------------+----------+
| CountryLanguage count | COUNT(*) |
+-----------------------+----------+
| CountryLanguage count |      984 |
+-----------------------+----------+
1 row in set (0.01 sec)
```

```
Query OK, 0 rows affected (0.01 sec)
```

3. Change to the `test` database, and attempt to execute the `record_count` stored procedure.

   Enter the following in a terminal window, and receive the results shown:

```
mysql> USE test;
...
Database changed
mysql> CALL record_count();
ERROR 1305 (42000): PROCEDURE test.record_count does not exist.
```

   – Why did the `record_count` procedure fail?
     The stored procedure `record_count` is part of the `world_innodb` database. MySQL is looking for the `record_count` stored procedure in the `test` database.

4. Execute the `record_count` stored procedure, this time qualified with the name of the `world_innodb` database.

   Enter the following in a terminal window, and receive the results shown:

```
mysql> CALL world_innodb.record_count();
```

```
+---------------+----------+
| Country count | COUNT(*) |
+---------------+----------+
| Country count |      239 |
+---------------+----------+
1 row in set (0.01 sec)

+------------+----------+
| City count | COUNT(*) |
+------------+----------+
| City count |     4079 |
+------------+----------+
1 row in set (0.01 sec)

+---------------------+----------+
| CountryLanguage count | COUNT(*) |
+---------------------+----------+
| CountryLanguage count |      984 |
+---------------------+----------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)
```

   Did the `record_count` stored procedure execute properly? Yes

5. Change to the `world_innodb` database and issue the following SQL statements to create the `pay_check` stored function.

   Enter the following statements at a `mysql` prompt and receive the results shown:

```
mysql> USE world_innodb;
...
Database changed
mysql> DELIMITER //
mysql> CREATE FUNCTION pay_check (gross_pay FLOAT(9,2),
    ->  tax_rate FLOAT(3,2))
    -> RETURNS FLOAT(9,2)
    -> NO SQL
    -> BEGIN
    ->   DECLARE net_pay FLOAT(9,2) DEFAULT 0;
    ->   SET net_pay=gross_pay - gross_pay * tax_rate;
    ->   RETURN net_pay;
    -> END//
Query OK, 0 rows affected (0.06 sec)

mysql> DELIMITER ;
```

6. Execute the `pay_check` stored function by issuing an appropriate SQL statement, passing in the arguments `100000` and `0.05`.

   Enter the following statement at a `mysql` prompt and receive the results shown:

```
mysql> SELECT pay_check (100000, 0.05);
+-------------------------+
| pay_check (100000,0.05) |
+-------------------------+
|                95000.00 |
+-------------------------+
```

Practice 9-2: Reviewing Stored Routines

Overview

In this practice, you review the stored routines created in an earlier practice. To accomplish this objective:

- Review a specific stored routine located in the `world_innodb` database using the `SHOW CREATE PROCEDURE` command.
- Review stored routines located in the MySQL server using the `SHOW { PROCEDURE | FUNCTION } STATUS` commands.
- Review all the stored routines located in the MySQL server using the `INFORMATION_SCHEMA` database.

Tasks

1. Using the `mysql` client, in the `world_innodb` database, issue a `SHOW CREATE PROCEDURE` command to review the details of the `record_count` stored procedure.
2. Issue a `SHOW PROCEDURE STATUS` command to review the status of all stored procedures on the MySQL server that have a name beginning with `record`.
3. Issue a `SHOW FUNCTION STATUS` command to review the status of all stored functions located on the MySQL server.
4. Using the `INFORMATION_SCHEMA` database, issue a `SELECT` statement to review the details for all the stored routines located on the MySQL server.

Solutions 9-2: Reviewing Stored Routines

Tasks

1. Using the `mysql` client, in the `world_innodb` database, issue a `SHOW CREATE PROCEDURE` command to review the details of the `record_count` stored procedure.

   Enter the following statement at a `mysql` prompt connected to the `world_innodb` database, and receive the results shown:

```
mysql> SHOW CREATE PROCEDURE record_count\G
*************************** 1. row ***************************
           Procedure: record_count
            sql_mode: STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION
    Create Procedure: CREATE DEFINER=`root`@`localhost` PROCEDURE
`record_count`()
BEGIN
  SELECT 'Country count ', COUNT(*) FROM Country;
  SELECT 'City count ', COUNT(*) FROM City;
  SELECT 'CountryLanguage count', COUNT(*) FROM
   CountryLanguage;
END
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

2. Issue a `SHOW PROCEDURE STATUS` command to review the status of all stored procedures on the MySQL server that have a name beginning with `record`.

   Enter the following statement at a `mysql` prompt, and receive the results shown:

```
mysql> SHOW PROCEDURE STATUS LIKE 'record%'\G
*************************** 1. row ***************************
                  Db: world_innodb
                Name: record_count
                Type: PROCEDURE
             Definer: root@localhost
            Modified: 2013-02-10 13:48:11
             Created: 2013-02-10 13:48:11
       Security_type: DEFINER
             Comment:
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

3. Issue a `SHOW FUNCTION STATUS` command to review the status of all stored functions located on the MySQL server.

   Enter the following statement at a `mysql` prompt, and receive the results shown:

```
mysql> SHOW FUNCTION STATUS\G
*************************** 1. row ***************************
                  Db: sakila
                Name: get_customer_balance
                Type: FUNCTION
             Definer: root@localhost
            Modified: 2013-02-03 15:25:01
             Created: 2013-02-03 15:25:01
       Security_type: DEFINER
             Comment:
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
*************************** 2. row ***************************
                  Db: sakila
                Name: inventory_held_by_customer
                Type: FUNCTION
             Definer: root@localhost
            Modified: 2013-02-03 15:25:01
             Created: 2013-02-03 15:25:01
       Security_type: DEFINER
             Comment:
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
*************************** 3. row ***************************
                  Db: sakila
                Name: inventory_in_stock
                Type: FUNCTION
             Definer: root@localhost
            Modified: 2013-02-03 15:25:01
             Created: 2013-02-03 15:25:01
       Security_type: DEFINER
             Comment:
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
*************************** 4. row ***************************
                  Db: world_innodb
                Name: pay_check
                Type: FUNCTION
             Definer: root@localhost
            Modified: 2013-02-10 13:52:32
             Created: 2013-02-10 13:52:32
       Security_type: DEFINER
             Comment:
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
4 rows in set (0.00 sec)
```

4. Using the INFORMATION_SCHEMA database, issue a SELECT statement to review the details for all the stored routines located on the MySQL server.

   Enter the following statement at a mysql prompt, and receive the results shown:

```
mysql> USE INFORMATION_SCHEMA;
...
Database changed
mysql> SELECT * FROM ROUTINES\G
*************************** 1. row ***************************
           SPECIFIC_NAME: film_in_stock
         ROUTINE_CATALOG: def
          ROUTINE_SCHEMA: sakila
            ROUTINE_NAME: film_in_stock
...
*************************** 7. row ***************************
           SPECIFIC_NAME: pay_check
         ROUTINE_CATALOG: def
          ROUTINE_SCHEMA: world_innodb
            ROUTINE_NAME: pay_check
            ROUTINE_TYPE: FUNCTION
               DATA_TYPE: float
CHARACTER_MAXIMUM_LENGTH: NULL
  CHARACTER_OCTET_LENGTH: NULL
       NUMERIC_PRECISION: 9
           NUMERIC_SCALE: 2
      DATETIME_PRECISION: NULL
      CHARACTER_SET_NAME: NULL
          COLLATION_NAME: NULL
          DTD_IDENTIFIER: float(9,2)
            ROUTINE_BODY: SQL
      ROUTINE_DEFINITION: BEGIN
  DECLARE net_pay FLOAT(9,2) DEFAULT 0;
  SET net_pay=gross_pay - gross_pay * tax_rate;
  RETURN net_pay;
END
           EXTERNAL_NAME: NULL
       EXTERNAL_LANGUAGE: NULL
         PARAMETER_STYLE: SQL
        IS_DETERMINISTIC: NO
         SQL_DATA_ACCESS: NO SQL
                SQL_PATH: NULL
           SECURITY_TYPE: DEFINER
                 CREATED: 2013-02-10 13:52:32
            LAST_ALTERED: 2013-02-10 13:52:32
                SQL_MODE: STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION
         ROUTINE_COMMENT:
                 DEFINER: root@localhost
    CHARACTER_SET_CLIENT: utf8
    COLLATION_CONNECTION: utf8_general_ci
      DATABASE_COLLATION: latin1_swedish_ci
*************************** 8. row ***************************
           SPECIFIC_NAME: record_count
         ROUTINE_CATALOG: def
          ROUTINE_SCHEMA: world_innodb
            ROUTINE_NAME: record_count
            ROUTINE_TYPE: PROCEDURE
               DATA_TYPE:
```

```
CHARACTER_MAXIMUM_LENGTH: NULL
  CHARACTER_OCTET_LENGTH: NULL
       NUMERIC_PRECISION: NULL
           NUMERIC_SCALE: NULL
      DATETIME_PRECISION: NULL
      CHARACTER_SET_NAME: NULL
          COLLATION_NAME: NULL
          DTD_IDENTIFIER: NULL
            ROUTINE_BODY: SQL
      ROUTINE_DEFINITION: BEGIN
  SELECT 'Country count ', COUNT(*) FROM Country;
  SELECT 'City count ', COUNT(*) FROM City;
  SELECT 'CountryLanguage count', COUNT(*) FROM
   CountryLanguage;
END
           EXTERNAL_NAME: NULL
       EXTERNAL_LANGUAGE: NULL
         PARAMETER_STYLE: SQL
        IS_DETERMINISTIC: NO
         SQL_DATA_ACCESS: CONTAINS SQL
                SQL_PATH: NULL
           SECURITY_TYPE: DEFINER
                 CREATED: 2013-02-10 13:48:11
            LAST_ALTERED: 2013-02-10 13:48:11
                SQL_MODE: STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION
         ROUTINE_COMMENT:
                 DEFINER: root@localhost
    CHARACTER_SET_CLIENT: utf8
    COLLATION_CONNECTION: utf8_general_ci
      DATABASE_COLLATION: latin1_swedish_ci
8 rows in set (0.00 sec)
```

Practice 9-3: Creating a Trigger

Overview

In this practice, you create a trigger to capture data as it is deleted. To accomplish this objective:

- Create a table to store references to data that is deleted in the `City` table in the `world_innodb` database.
- Create a trigger that stores the reference to the deleted data after the delete has taken place in the created table.
- Delete data from the `City` table in the `world_innodb` database.
- Verify that the table created contains the references to the deleted data.

Tasks

1. Using the `mysql` client, in the `world_innodb` database, issue the following SQL statement to create a table to store references to records deleted in the `City` table:

```
CREATE TABLE DeletedCity
( ID INT UNSIGNED, Name
  VARCHAR(50),
  When_Deleted timestamp);
```

2. Issue the following SQL statement to create a trigger to capture data after a record is deleted from the `City` table and place the captured data in the `DeletedCity` table:

```
CREATE TRIGGER City_AD AFTER DELETE ON City
FOR EACH ROW
INSERT INTO DeletedCity (ID, Name)
   VALUES (OLD.ID, OLD.Name);
```

3. Issue an appropriate `SHOW` command to confirm that the trigger has been created.
4. Delete the records with the city name `Dallas` in the `City` table.
5. Attempt to select all the `City` table records with the name `Dallas` to verify that the delete operation worked properly.
6. View the contents of the `DeletedCity` table to determine whether the trigger captured the references to the record deleted in step 4.

Solutions 9-3: Creating a Trigger

Tasks

1. Using the `mysql` client, in the `world_innodb` database, issue the following SQL statement to create a table to store references to records deleted in the `City` table.

   Enter the following statements at a `mysql` prompt connected to the `world_innodb` database, and receive the results shown:

```
mysql> USE world_innodb;
...
Database changed
mysql> CREATE TABLE DeletedCity
    -> (ID INT UNSIGNED,
    ->  Name VARCHAR(50),
    ->  When_Deleted timestamp);
);
Query OK, 0 rows affected (0.06 sec)
```

2. Issue the following SQL statement to create a trigger to capture data after a record is deleted from the `City` table and place the captured data in the `DeletedCity` table.

   Enter the following statement at a `mysql` prompt, and receive the results shown:

```
mysql> CREATE TRIGGER City_AD AFTER DELETE ON City
    -> FOR EACH ROW
    -> INSERT INTO DeletedCity (ID, Name)
    ->  VALUES (OLD.ID, OLD.Name);
Query OK, 0 rows affected (0.06 sec)
```

3. Issue an appropriate `SHOW` command to confirm that the trigger has been created.

   Enter the following statement at a `mysql` prompt, and receive the results shown:

```
mysql> SHOW TRIGGERS\G
*************************** 1. row ***************************
             Trigger: City_AD
               Event: DELETE
               Table: City
           Statement: INSERT INTO DeletedCity (ID, Name)
   VALUES (OLD.ID, OLD.Name)
              Timing: AFTER
             Created: NULL
            sql_mode: STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION
             Definer: root@localhost
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.01 sec)
```

4. Delete the records with the city name `Dallas` in the `City` table.

   Enter the following statement at a `mysql` prompt, and receive the results shown:

```
mysql> DELETE FROM City WHERE Name = 'Dallas';
Query OK, 1 row affected (0.13 sec)
```

5. Attempt to select all the `City` table records with the name `Dallas` to verify that the delete operation worked properly.

   Enter the following statement at a `mysql` prompt, and receive the results shown:

```
mysql> SELECT * FROM City WHERE Name = 'Dallas';
Empty set (0.00 sec)
```

6. View the contents of the `DeletedCity` table to determine whether the trigger captured the references to the record deleted in step 4.

   Enter the following statement at a `mysql` prompt, and receive the results shown:

```
mysql> SELECT * FROM DeletedCity;
+------+--------+---------------------+
| ID   | Name   | When_Deleted        |
+------+--------+---------------------+
| 3800 | Dallas | 2013-02-10 14:01:51 |
+------+--------+---------------------+
1 row in set (0.00 sec)
```

   The deleted record appears in the result set, along with when it was deleted.

Practice 9-4: Creating and Testing an Event

Overview

In this practice, you create an event to locate processes that are taking too long to run on the MySQL server and terminate their connection. To accomplish this objective:

- Create an event that disconnects non-`root` users when queries run longer than 30 seconds.
- Turn on the event scheduler.
- Create a non-`root` user to test the created event.
- Test the created event.

Note: This practice uses an advanced level of programming in the event creation that is beyond the scope of this course. However, the example is intended to show the power of programming at an advanced level, and how beneficial learning more about this topic could be to your day-to-day tasks of managing a MySQL server.

Tasks

1. Using the `mysql` client, in the `world_innodb` database, issue the following SQL statement to create an event that disconnects non-`root` users when queries run longer than 30 seconds:

```
DELIMITER //
CREATE EVENT kill_user
  ON SCHEDULE EVERY 20 SECOND
DO
BEGIN
  DECLARE var_id INT;
  DECLARE procs CURSOR FOR SELECT id
  FROM INFORMATION_SCHEMA.PROCESSLIST WHERE TIME>30
    AND Command != 'Sleep' AND USER != 'root';
  OPEN procs;
  BEGIN
    DECLARE EXIT HANDLER FOR NOT FOUND
    BEGIN
      CLOSE procs;
    END;
    LOOP
      FETCH procs INTO var_id;
      KILL var_id;
    END LOOP;
  END;
END//
```

```
DELIMITER ;
```

2. Turn on the event scheduler.
3. Create a non-`root` user named `tester` logging in from the `localhost` with the password '`secret`'.
4. Grant the new user access to all databases.
5. In a second terminal window, log in to the `mysql` client as the new user.
6. In the second terminal window, enter the following command to create a `SELECT` statement that attempts to run longer than 30 seconds:

   ```
   mysql> SELECT SLEEP(60);
   ```

   – How long did it take the server to terminate the connections?
     ____.
7. Exit the `mysql` client in the second terminal window.

Solutions 9-4: Creating and Testing an Event

Tasks

1. Using the `mysql` client, in the `world_innodb` database, issue the following SQL statement to create an event that disconnects non-`root` users when queries run longer than 30 seconds:

```
mysql> DELIMITER //
mysql> CREATE EVENT kill_user
    -> ON SCHEDULE EVERY 20 SECOND
    -> DO
    -> BEGIN
    ->  DECLARE var_id INT;
    ->  DECLARE procs CURSOR FOR SELECT id
    ->  FROM INFORMATION_SCHEMA.PROCESSLIST WHERE TIME>30
    ->   AND Command != 'Sleep' AND USER != 'root';
    ->  OPEN procs;
    ->  BEGIN
    ->    DECLARE EXIT HANDLER FOR NOT FOUND
    ->    BEGIN
    ->      CLOSE procs;
    ->    END;
    ->    LOOP
    ->      FETCH procs INTO var_id;
    ->      KILL var_id;
    ->    END LOOP;
    ->  END;
    -> END//
Query OK, 0 rows affected (0.06 sec)

mysql> DELIMITER ;
```

2. Turn on the event scheduler.

```
mysql> SET GLOBAL event_scheduler = ON;
Query OK, 0 rows affected (0.06 sec)
```

3. Create a non-`root` user named tester logging in from the localhost with the password 'secret'.

```
mysql> CREATE USER tester@localhost IDENTIFIED BY 'secret';
Query OK, 0 rows affected (0.06 sec)
```

4. Grant the new user access to all databases.

```
mysql> GRANT SELECT ON *.* TO tester@localhost;
Query OK, 0 rows affected (0.06 sec)
```

5. In a second terminal window, log in to the `mysql` client as the new user.

```
$ mysql -utester -p
Enter password: secret
Welcome to the MySQL monitor.  Commands end with ; or \g.
...
```

6. In the second terminal window, enter the following command to create a `SELECT` statement that attempts to run longer than 30 seconds:

```
mysql> SELECT SLEEP(60);
ERROR 2013 (HY000): Lost connection to MySQL server during query
```

   - How long did it take the server to terminate the connection?
     Approximately 30 to 50 seconds after the query starts running, the server terminates the connection. Because this event is scheduled to run every 20 seconds, the event kills the process within 20 seconds of the 30-second limit.

7. Exit the `mysql` client in the second terminal window:

```
mysql> EXIT
Bye
```

Practices for Lesson 10: MySQL Backup and Recovery

Practice 10-1: MySQL Enterprise Backup

Overview

In this practice, you create a backup by using the `mysqlbackup` application. To accomplish this objective:

- Modify the `my.cnf` file.
- Create a load on the MySQL server.
- Execute the `mysqlbackup` application while there is a load on the MySQL server.
- Delete files from the `/var/lib/mysql` directory.
- Execute the backup to restore the files that were deleted.

Assumptions

- The MySQL server is installed and running.
- The `world_innodb` database is installed.
- The MySQL Enterprise Backup software is installed.

Tasks

1. Create the **`/backups`** directory, and change its owner and group to `mysql`.
2. In the `/etc/my.cnf` file, note the value of the `datadir` server parameter.
3. Set the following `my.cnf` options as shown.

```
 [mysqld]
log-bin=mybinlog innodb_log_files_in_group =
2 innodb_log_file_size = 256M
innodb_data_file_path=ibdata1:10M:autoextend
```

4. Add a **`[`**`mysqlbackup`**`]`** section to the `/etc/my.cnf file` with the following settings:
   - `backup-dir`: **`/backups/meb1`**
   - `user`: **`backupuser`**
   - `socket`: **`/var/lib/mysql/mysql.sock`**
5. Save the changes to `/etc/my.cnf` file and close the file.
6. Delete the InnoDB log files.
7. Restart the MySQL server to apply the updated `/etc/my.cnf`.
8. Open a second terminal window and start the `mysql` client.

9. In the second terminal window, create a MySQL user `backupuser` on `localhost`, identified by the password `oracle`, and grant that user all required permissions to use `mysqlbackup`.

10. In the second terminal window, use the `world_innodb` database and create a copy of the `City` table called `City_Large`.

11. In the second terminal window, issue the following command to insert all the records from the `City` table into the `City_Large` table.

    ```
    INSERT INTO City_Large (Name, CountryCode, District,
    Population) SELECT Name, CountryCode, District,
    Population FROM City;
    ```

12. In the second terminal window, issue the following command to double the records from the `City_Large` table:

    ```
    INSERT INTO City_Large (Name, CountryCode, District,
    Population) SELECT Name, CountryCode, District,
    Population FROM City_Large;
    ```

Note: In steps 13 to 16, you simulate a "hot backup" of a busy database. To do this, you perform many progressively larger (and slower) inserts, flush the logs to ensure those inserts are logged, execute a new long-running insert, and take a backup while it is still in progress.

13. In the second terminal window, execute the SQL statement from the preceding step seven times.

14. In the second terminal window, prior to running the backup, execute a `FLUSH LOGS` command.

15. In the second terminal window, issue the command from step 12 one more time. Do not wait for this step to finish; start the next step immediately.

16. In the first terminal window, issue the following commands (while the `mysql` command from the preceding step is continuing to execute) to run the `mysqlbackup` application as the `mysql` user.

    ```
    # su - mysql
    $ mysqlbackup --defaults-file=/etc/my.cnf -p backup-and-apply-log
    ```

17. In the first terminal window, list the files in the `/backups/meb1` directory.

18. In the first terminal window, exit from the `mysql` user shell back to the `root` shell. Then, in the `/var/lib/mysql` directory, delete the `ibdata1` file and the `world_innodb` directory.

19. In the second terminal window, display all the databases that the MySQL server contains.

20. In the second terminal window, exit the `mysql` client.

21. In the first terminal window, shut down the MySQL server.

22. In the first terminal window, execute the following commands to restore the backup created to the `/var/lib/mysql` directory.

    ```
    # su - mysql
    $ mysqlbackup --defaults-file=/etc/my.cnf copy-back
    ```

23. In the first terminal window, exit from the `mysql` shell and start the MySQL server.
24. In the second terminal window, start the `mysql` client and display all the databases that the MySQL server contains.
25. In the second terminal window, view all the tables that the `world_innodb` database contains.
26. In the second terminal window, exit the `mysql` client and close the terminal window.

Solutions 10-1: MySQL Enterprise Backup

Tasks

1. Create the **/backups** directory, and change its owner and group to `mysql`.

   Enter the following in a terminal window:

```
$ su -
Password: oracle
# mkdir /backups
# chown mysql:mysql /backups
```

2. In the `/etc/my.cnf` file, note the value of the `datadir` server parameter.

   Locate the heading `[mysqld]` and note the value of the `datadir` parameter:

```
[mysqld]
...
datadir = /var/lib/mysql
...
```

   Note: The `datadir` parameter must be in the file for cold backups to take place. The directory paths must be absolute. For hot or warm backups, `mysqlbackup` queries the value from the server; it does not assume any defaults for file locations.

3. Set the following `my.cnf` options as shown.

   Locate the heading `[mysqld]`, modify the value of log-bin, and add the remaining lines shown below:

```
[mysqld]
...
log-bin=mybinlog innodb_log_files_in_group =
2 innodb_log_file_size = 256M
innodb_data_file_path=ibdata1:12M:autoextend
...
```

4. Add a `[mysqlbackup]` section to the `/etc/my.cnf file` with the following settings:
   - `backup-dir`: **/backups/meb1**
   - `user`: **backupuser**
   - `socket`: **/var/lib/mysql/mysql.sock**

   Add the following lines to the end of `/etc/my.cnf`:

```
...
[mysqlbackup]
backup-dir=/backups/meb1
user=backupuser
socket=/var/lib/mysql/mysql.sock
```

5. Save the changes to the `/etc/my.cnf` file and close the file.

6. Delete the InnoDB log files.

   Enter the following in a terminal window:

```
# rm -f /var/lib/mysql/ib_logfile*
```

7. Restart the MySQL server to apply the updated `/etc/my.cnf`.

   Enter the following in a terminal window, and receive the result shown below:

```
# service mysql restart
Shutting down MySQL...                                     [  OK  ]
Starting MySQL.                                            [  OK  ]
```

8. Open a second terminal window and start the `mysql` client.

   Enter the following command in the second terminal window and receive the result shown below:

```
$ mysql --login-path=admin
Welcome to the MySQL monitor.  Commands end with ; or \g.
...
```

9. In the second terminal window, create a MySQL user `backupuser` on `localhost`, identified by the password `oracle`, and grant that user all required permissions to use `mysqlbackup`.

   Enter the following in the second terminal window, and receive the results shown below:

```
mysql> CREATE USER 'backupuser'@'localhost' IDENTIFIED BY 'oracle';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT RELOAD ON *.* TO 'backupuser'@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT CREATE, INSERT, DROP ON mysql.ibbackup_binlog_marker
     > TO 'backupuser'@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT CREATE, INSERT, DROP ON mysql.backup_progress
     > TO 'backupuser'@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT CREATE, INSERT, DROP ON mysql.backup_history
     > TO 'backupuser'@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT REPLICATION CLIENT ON *.* TO 'backupuser'@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT SUPER ON *.* TO 'backupuser'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> GRANT CREATE TEMPORARY TABLES ON mysql.*
     > TO 'backupuser'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

10. In the second terminal window, use the `world_innodb` database and create a copy of the `City` table called `City_Large`. Enter the following and receive the results shown below:

```
mysql> USE world_innodb;
...
Database changed
mysql> CREATE TABLE City_Large LIKE City;
```

11. In the second terminal window, issue the following command to insert all the records from the `City` table into the `City_Large` table and receive the result shown below:

```
mysql> INSERT INTO City_Large (Name, CountryCode, District,
    ->  Population) SELECT Name, CountryCode, District,
    ->  Population FROM City;
Query OK, 4078 rows affected, 1 warning (1.78 sec)
Records: 4078  Duplicates: 0  Warnings: 1
```

12. In the second terminal window, issue the following command to double the records from the `City_Large` table, and receive the result shown below:

```
mysql> INSERT INTO City_Large (Name, CountryCode, District,
    ->  Population) SELECT Name, CountryCode, District,
    ->  Population FROM City_Large;
Query OK, 4078 rows affected, 1 warning (1.68 sec)
Records: 4078  Duplicates: 0  Warnings: 1
```

Note: In steps 13 to 16, you simulate a "hot backup" of a busy database. To do this, you perform many progressively larger (and slower) inserts, flush the logs to ensure those inserts are logged, execute a new long-running insert, and take a backup while it is still in progress.

13. In the second terminal window, execute the SQL statement from the preceding step seven times. Enter the following and receive the result shown below:

    Note: The time for the operation to complete gets progressively longer due to the exponential doubling of the number of records that are being added to the `City_Large` table.

```
mysql> INSERT INTO City_Large (Name, CountryCode, District,
    ->  Population) SELECT Name, CountryCode, District,
    ->  Population FROM City_Large;
Query OK, 8156 rows affected, 1 warning (0.34 sec)
Records: 8156  Duplicates: 0  Warnings: 1

mysql> INSERT INTO City_Large (Name, CountryCode, District,
    ->  Population) SELECT Name, CountryCode, District,
    ->  Population FROM City_Large;
Query OK, 16312 rows affected, 1 warning (0.64 sec)
Records: 16312  Duplicates: 0  Warnings: 1

mysql> INSERT INTO City_Large (Name, CountryCode, District,
    ->  Population) SELECT Name, CountryCode, District,
    ->  Population FROM City_Large;
```

```
Query OK, 32624 rows affected, 1 warning (0.77 sec)
Records: 32624  Duplicates: 0  Warnings: 1

mysql> INSERT INTO City_Large (Name, CountryCode, District,
    ->  Population) SELECT Name, CountryCode, District,
    ->  Population FROM City_Large;
Query OK, 65248 rows affected, 1 warning (1.54 sec)
Records: 65248  Duplicates: 0  Warnings: 1

mysql> INSERT INTO City_Large (Name, CountryCode, District,
    ->  Population) SELECT Name, CountryCode, District,
    ->  Population FROM City_Large;
Query OK, 130496 rows affected, 1 warning (2.63 sec)
Records: 130496  Duplicates: 0  Warnings: 1

mysql> INSERT INTO City_Large (Name, CountryCode, District,
    ->  Population) SELECT Name, CountryCode, District,
    ->  Population FROM City_Large;
Query OK, 260992 rows affected, 1 warning (4.58 sec)
Records: 260992  Duplicates: 0  Warnings: 1

mysql> INSERT INTO City_Large (Name, CountryCode, District,
    ->  Population) SELECT Name, CountryCode, District,
    ->  Population FROM City_Large;
Query OK, 521984 rows affected, 1 warning (11.33 sec)
Records: 521984  Duplicates: 0  Warnings: 1
```

14. In the second terminal window, prior to running the backup, execute a `FLUSH LOGS` command. This forces the current binary log to close and the next incremental binary log to be opened.

```
mysql> FLUSH LOGS;
```

15. In the second terminal window, issue the command from step 12 one more time. Do not wait for this step to finish; start the next step immediately.
    Note: The operation should take 20–40 seconds to run. During this time, you return to the first terminal window and execute the **`mysqlbackup`** command. This provides a simulated "hot" backup of the InnoDB tables.

    Enter the following in the second terminal window, and receive the result shown below:

```
mysql> INSERT INTO City_Large (Name, CountryCode, District,
    ->  Population) SELECT Name, CountryCode, District,
    ->  Population FROM City_Large;
Query OK, 1043968 rows affected, 1 warning (29.49 sec)
Records: 1043968  Duplicates: 0  Warnings: 1
```

16. In the first terminal window, issue the following commands (while the `mysql` command from the preceding step is continuing to execute) to run the `mysqlbackup` application as the `mysql` user. Enter the following commands and receive the result shown below:

```
# su mysql
$ mysqlbackup --defaults-file=/etc/my.cnf -p backup-and-apply-log
MySQL Enterprise Backup version 3.8.1 [2013/01/28]
Copyright (c) 2003, 2012, Oracle and/or its affiliates. All Rights
Reserved.

 mysqlbackup: INFO: Starting with following command line ...
 mysqlbackup --defaults-file=/etc/my.cnf -p backup-and-apply-log

Enter password: oracle
 mysqlbackup: INFO: MySQL server version is '5.6.10-enterprise-
commercial-advanced-log'.
 mysqlbackup: INFO: Got some server configuration information from
running server.

IMPORTANT: Please check that mysqlbackup run completes successfully.
           At the end of a successful 'backup-and-apply-log' run
mysqlbackup
           prints "mysqlbackup completed OK!".

--------------------------------------------------------------------
                       Server Repository Options:
--------------------------------------------------------------------
  datadir = /var/lib/mysql/
  innodb_data_home_dir =
  innodb_data_file_path = ibdata1:12M:autoextend
  innodb_log_group_home_dir = /var/lib/mysql/
  innodb_log_files_in_group = 2
  innodb_log_file_size = 268435456
  innodb_page_size = 16384
  innodb_checksum_algorithm = innodb
  innodb_undo_directory = /var/lib/mysql/
  innodb_undo_tablespaces = 0
  innodb_undo_logs = 128

--------------------------------------------------------------------
                       Backup Config Options:
--------------------------------------------------------------------
  datadir = /backups/meb1/datadir
  innodb_data_home_dir = /backups/meb1/datadir
  innodb_data_file_path = ibdata1:12M:autoextend
  innodb_log_group_home_dir = /backups/meb1/datadir
  innodb_log_files_in_group = 2
  innodb_log_file_size = 268435456
  innodb_page_size = 16384
  innodb_checksum_algorithm = innodb
  innodb_undo_directory = /backups/meb1/datadir
  innodb_undo_tablespaces = 0
  innodb_undo_logs = 128

 mysqlbackup: INFO: Unique generated backup id for this is
13605079305675972

 mysqlbackup: INFO: Uses posix_fadvise() for performance optimization.
```

```
InnoDB: Progress in percent: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62
63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85
86 87 88 89 90 91 92 93 94 95 96 97 98 99 Setting log file size to
268435456
InnoDB: Progress in MB: 100 200
Setting log file size to 268435456
InnoDB: Progress in MB: 100 200
130210 14:52:30 mysqlbackup: INFO: We were able to parse
ibbackup_logfile up to
          lsn 309947815.
 mysqlbackup: INFO: Last MySQL binlog file position 0 2671, file name
mybinlog.000003
130210 14:52:30 mysqlbackup: INFO: The first data file is
'/backups/meb1/datadir/ibdata1'
          and the new created log files are at '/backups/meb1/datadir'
130210 14:52:30 mysqlbackup: INFO: Apply-log operation completed
successfully.
130210 14:52:30 mysqlbackup: INFO: Full backup prepared for recovery
successfully.

mysqlbackup completed OK!
```

17. In the first terminal window, list the files in the `/backups/meb1` directory, and receive the result shown below:

```
$ ls /backups/meb1/
backup-my.cnf   datadir   meta
```

18. In the first terminal window, exit from the `mysql` user shell back to the `root` shell. Then, in the `/var/lib/mysql` directory, delete the `ibdata1` file and the `world_innodb` directory.

Enter the following in a terminal window, and receive the result shown below:

```
$ exit
# cd /var/lib/mysql
# rm -f ibdata1
# rm -rf world_innodb
```

19. In the second terminal window, display all the databases that the `MySQL` server contains.

Enter the following at the `mysql` prompt, and receive the result shown below:

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| test               |
| world2             |
+--------------------+
6 rows in set (0.00 sec)
```

- Does the listing of databases include the `world_innodb` database? No

20. In the second terminal window, exit the `mysql` client.

```
mysql> EXIT;
```

21. In the first terminal window, shut down the MySQL server.

```
# service mysql stop
Shutting down MySQL..                                      [  OK  ]
```

22. In the first terminal window, execute the following commands to restore the backup created to the `/var/lib/mysql` directory, and receive the result shown below:

```
# su mysql
$ mysqlbackup --defaults-file=/etc/my.cnf copy-back
MySQL Enterprise Backup version 3.6.0 [2011/07/01]
Copyright (c) 2003, 2011, Oracle and/or its affiliates. All Rights
Reserved.

INFO: Starting with following command line ...
./mysqlbackup --defaults-file=/etc/my.cnf --backup-dir=/backups/meb1
       copy-back

IMPORTANT: Please check that mysqlbackup run completes successfully.
           At the end of a successful 'copy-back' run mysqlbackup
           prints "mysqlbackup completed OK!".

mysqlbackup: INFO: Server repository configuration: datadir
                                  =  /var/lib/mysql
 innodb_data_home_dir             =  /var/lib/mysql
 innodb_data_file_path            =  ibdata1:10M:autoextend
 innodb_log_group_home_dir        =  /var/lib/mysql
 innodb_log_files_in_group        =  2
 innodb_log_file_size             =  256M

mysqlbackup: INFO: Backup repository configuration:
 datadir                          =  /backups/meb1/datadir
 innodb_data_home_dir             =  /backups/meb1/datadir
 innodb_data_file_path            =  ibdata1:10M:autoextend
 innodb_log_group_home_dir        =  /backups/meb1/datadir
 innodb_log_files_in_group        =  2
 innodb_log_file_size             =  268435456

mysqlbackup: INFO: Starting to copy back files
mysqlbackup: INFO: in '/backups/meb1/datadir' directory
mysqlbackup: INFO: back to original data directory '/var/lib/mysql'
mysqlbackup: INFO: Copying back directory
'/backups/meb1/datadir/mysql'
```

```
mysqlbackup: INFO: Copying back directory
'/backups/meb1/datadir/performance_schema'
mysqlbackup: INFO: Copying back directory '/backups/meb1/datadir/test'
mysqlbackup: INFO: Copying back directory
'/backups/meb1/datadir/world_innodb'
mysqlbackup: INFO: Starting to copy back InnoDB tables and indexes
in '/backups/meb1' back to original InnoDB data directory:
/var/lib/mysql
mysqlbackup: INFO: Copying back file '/backups/meb1/datadir/ibdata1'
mysqlbackup: INFO: Starting to copy back InnoDB log files
in '/backups/meb1/datadir' back to original InnoDB log directory
'/var/lib/mysql'
mysqlbackup: INFO: Copying back file
'/backups/meb1/datadir/ib_logfile0'
mysqlbackup: INFO: Copying back file
'/backups/meb1/datadir/ib_logfile1'
mysqlbackup: INFO: Finished copying backup files.
```

23. In the first terminal window, exit from the `mysql` shell and start the MySQL server.

```
$ exit
# service mysql start
```

24. In the second terminal window, start the `mysql` client and display all the databases that the MySQL server contains. Enter the following in the second terminal window, and receive the results shown below:

```
$ mysql --login-path=admin
...

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| test               |
| world2             |
| world_innodb       |
+--------------------+
7 rows in set (0.00 sec)
```

25. In the second terminal window, view all the tables that the `world_innodb` database contains. Enter the following in the second terminal window, and receive the results shown below:

```
mysql> USE world_innodb
Database changed.

mysql> SHOW TABLES;
+------------------------+
| Tables_in_world_innodb |
+------------------------+
| City                   |
| CityLanguage           |
| City_Large             |
| City_part              |
| City_temp              |
| Country                |
| CountryLanguage        |
| CountryLanguage2       |
| DeletedCity            |
+------------------------+
9 rows in set (0.00 sec)
```

26. In the second terminal window, exit the **`mysql`** client and close the terminal window.

```
mysql> EXIT
Bye
$ exit
```

Practice 10-2: `mysqldump`

Overview

In this practice, you use the `mysqldump` application to create a backup copy of the `world_innodb` database and then restore most of the contents to another database. To accomplish this objective:

- Back up the `world_innodb` database by using the `mysqldump` application.
- Create a new database called `world3`.
- Use the `mysql` client to create most of the tables that were backed up.
- Use the `mysqlimport` command to load the backed up data into the created tables.
- Verify that the tables that were backed up are the same as those that were restored from the backup into a new database.

Assumptions

- The MySQL server is installed and running.
- The `world_innodb` database is installed.

Tasks

1. Using the `mysqldump` application, make a tab-delimited backup of the `world_innodb` database to `/backups`.
2. Review the contents of the `/backups` directory.
   - How many `*.sql` files are located in this directory?

     ____

   - How many `*.txt` files are located in this directory?

     ____

3. Review the contents of the `Country.sql` file.
4. Review the first few lines of the `/backups/Country.txt` file.
   - Reviewing the data, what is the file format?

     ____

5. Using the `mysqladmin` client, create a new database called `world3`.
6. Using the `mysql` application, load each of the `*.sql` files for the `Country`, `City`, and `CountryLanguage` tables created in step 1 into the `world3` database.
7. Using the `mysqlimport` application, load each of the `*.txt` files for the `Country`, `City`, and `CountryLanguage` tables created in step 1.
8. Using the `mysql` client, review the tables in the `world3` database.

9. Review the record count for the `world3.City` table and `world_innodb.City` table to verify that they both contain the same row counts.
   - Is the number of rows the same for both of the tables? ____________________
10. Review the record count for the `world3.Country`, `world_innodb.Country`, `world3.CountryLanguage`, and `world_innodb.CountryLanguage` tables to verify that the respective tables contain the same row counts.
    - Is the number of rows the same for the `Country` tables? ____________________
    - Is the number of rows the same for the `CountryLanguage` tables?

      ____
11. Exit the `mysql` client session.

Solutions 10-2: mysqldump

Tasks

1. Using the mysqldump application, make a tab-delimited backup of the world_innodb database to /backups.

```
# mysqldump -uroot -p --tab=/backups world_innodb
Enter password: oracle
```

2. Review the contents of the /backups directory. Enter the following in a terminal window, and receive the result shown below:

```
# ls /backups -l
total 77828
-rw-r--r-- 1 root  root      1469 Feb 10 16:50 CityLanguage.sql
-rw-rw-rw- 1 mysql mysql        0 Feb 10 16:50 CityLanguage.txt
-rw-r--r-- 1 root  root      1616 Feb 10 16:50 City_Large.sql
-rw-rw-rw- 1 mysql mysql 79207921 Feb 10 16:51 City_Large.txt
-rw-r--r-- 1 root  root      1610 Feb 10 16:51 City_part.sql
-rw-rw-rw- 1 mysql mysql   107645 Feb 10 16:51 City_part.txt
-rw-r--r-- 1 root  root      2620 Feb 10 16:50 City.sql
-rw-r--r-- 1 root  root      1610 Feb 10 16:51 City_temp.sql
-rw-rw-rw- 1 mysql mysql   108797 Feb 10 16:51 City_temp.txt
-rw-rw-rw- 1 mysql mysql   143528 Feb 10 16:50 City.txt
-rw-r--r-- 1 root  root      1608 Feb 10 16:51 CountryLanguage2.sql
-rw-rw-rw- 1 mysql mysql    18234 Feb 10 16:51 CountryLanguage2.txt
-rw-r--r-- 1 root  root      1702 Feb 10 16:51 CountryLanguage.sql
-rw-rw-rw- 1 mysql mysql    18234 Feb 10 16:51 CountryLanguage.txt
-rw-r--r-- 1 root  root      2038 Feb 10 16:51 Country.sql
-rw-rw-rw- 1 mysql mysql    31755 Feb 10 16:51 Country.txt
-rw-r--r-- 1 root  root      1489 Feb 10 16:51 DeletedCity.sql
-rw-rw-rw- 1 mysql mysql       32 Feb 10 16:51 DeletedCity.txt
drwx------ 4 mysql mysql     4096 Feb 10 14:52 meb1
```

   – How many *.sql files are located in this directory? Nine, one for each table in the world_innodb database
   – How many *.txt files are located in this directory? Nine, one for each table in the world_innodb database

3. Review the contents of the Country.sql file.

   Enter the following in a terminal window, and receive the result shown below:

```
# more /backups/Country.sql
-- MySQL dump 10.13  Distrib 5.6.10, for Linux (x86_64)
--
-- Host: localhost    Database: world_innodb
-- ------------------------------------------------------
-- Server version       5.6.10-enterprise-commercial-advanced-log

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `Country`
--

DROP TABLE IF EXISTS `Country`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `Country` (
  `Code` char(3) NOT NULL DEFAULT '',
  `Name` char(52) NOT NULL DEFAULT '',
  `Continent` enum('Asia','Europe','North
America','Africa','Oceania','Antarctica','South America') NOT NULL
DEFAULT 'Asia',
  `Region` char(26) NOT NULL DEFAULT '',
  `SurfaceArea` float(10,2) NOT NULL DEFAULT '0.00',
  `IndepYear` smallint(6) DEFAULT NULL,
  `Population` int(11) NOT NULL DEFAULT '0',
  `LifeExpectancy` float(3,1) DEFAULT NULL,
  `GNP` float(10,2) DEFAULT NULL,
  `GNPOld` float(10,2) DEFAULT NULL,
  `LocalName` char(45) NOT NULL DEFAULT '',
  `GovernmentForm` char(45) NOT NULL DEFAULT '',
  `HeadOfState` char(60) DEFAULT NULL,
  `Capital` int(11) DEFAULT NULL,
  `Code2` char(2) NOT NULL DEFAULT '',
  PRIMARY KEY (`Code`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2013-02-10 16:51:01
```

4. Review the first few lines of the `/backups/Country.txt` file.

   Enter the following in a terminal window, and receive the result shown below:

```
# head /backups/Country.txt
ABW     Aruba   North America   Caribbean       193.00  \N      103000
78.4      828.00   793.00   Aruba    Nonmetropolitan Territory of The
Netherlands    Beatrix 129     AW
AFG         Afghanistan         Asia       Southern  and  Central  Asia
652090.00           1919      22720000            45.9      5976.00  \N
Afganistan/Afqanestan    Islamic Emirate Mohammad Omar   1       AF
...
```

   – Reviewing the data, what is the file format? It is a tab-delimited text file

5. Using the `mysqladmin` client, create a new database called `world3`.

```
# mysqladmin -uroot -p create world3
Enter password: oracle
```

6. Using the `mysql` application, load each of the `*.sql` files for the `Country`, `City`, and `CountryLanguage` tables created in step 1 into the `world3` database.

   Enter the following in a terminal window, and receive the result shown below:

```
# cd /backups
# mysql -uroot -poracle world3 < Country.sql
Warning: Using a password on the command line interface can be
insecure.
# mysql -uroot -poracle world3 < CountryLanguage.sql
Warning: Using a password on the command line interface can be
insecure.
# mysql -uroot -poracle world3 < City.sql
Warning: Using a password on the command line interface can be
insecure.
```

7. Using the `mysqlimport` application, load in each of the `*.txt` files for the `Country`, `City`, and `CountryLanguage` tables created in step 1.

   Enter the following in a terminal window, and receive the result shown below:

```
# mysqlimport -uroot -poracle world3 /backups/Country.txt
Warning: Using a password on the command line interface can be
insecure.
world3.Country: Records: 239  Deleted: 0  Skipped: 0  Warnings: 0
# mysqlimport -uroot -poracle world3 /backups/CountryLanguage.txt
Warning: Using a password on the command line interface can be
insecure.
world3.CountryLanguage: Records: 984  Deleted: 0  Skipped: 0
Warnings: 0
# mysqlimport -uroot -poracle world3 /backups/City.txt
Warning: Using a password on the command line interface can be
insecure.
world3.City: Records: 4078  Deleted: 0  Skipped: 0  Warnings: 0
```

8. Using the `mysql` client, review the tables in the `world3` database.

   Enter the following in a terminal window, and receive the result shown below:

```
# mysql -uroot -poracle
Welcome to the MySQL monitor.  Commands end with ; or \g.
...
mysql> USE world3
Database changed
mysql> SHOW TABLES;
+-------------------+
| Tables_in_world3 |
+-------------------+
| City              |
| Country           |
| CountryLanguage  |
+-------------------+
3 rows in set (0.01 sec)
```

9. Review the record count for the `world3.City` table and `world_innodb.City` table to verify that they both contain the same row counts.

   Enter the following in a terminal window, and receive the result shown below:

```
mysql> SELECT COUNT(*) FROM world3.City;
+----------+
| COUNT(*) |
+----------+
|     4078 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM world_innodb.City;
+----------+
| COUNT(*) |
+----------+
|     4078 |
+----------+
1 row in set (0.00 sec)
```

   − Is the number of rows the same for both of the tables? Yes

10. Review the record count for the `world3.Country`, `world_innodb.Country`, `world3.CountryLanguage`, and `world_innodb.CountryLanguage` tables to verify that the respective tables contain the same row counts.

    Enter the following in a terminal window, and receive the result shown below:

```
mysql> SELECT COUNT(*) FROM world3.Country;
+----------+
| COUNT(*) |
+----------+
|      239 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM world_innodb.Country;
+----------+
| COUNT(*) |
+----------+
|      239 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM world3.CountryLanguage;
+----------+
| COUNT(*) |
+----------+
|      984 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM world_innodb.CountryLanguage;
+----------+
| COUNT(*) |
+----------+
|      984 |
+----------+
1 row in set (0.00 sec)
```

    – Is the number of rows the same for the `Country` tables? Yes
    – Is the number of rows the same for the `CountryLanguage` tables? Yes

11. Exit the `mysql` client session.

```
mysql> EXIT
Bye
```

Practice 10-3: Backup and Recovery Using the Binary Log

Assumptions

- The MySQL server is installed and running.
- The `world_innodb.sql` file is located in the `/labs` directory.

Tasks

1. Stop the MySQL server.
2. Execute the following commands in a terminal window logged in as `root` to create a copy of the MySQL data directory on a loop device LVM logical volume.

```
dd if=/dev/zero of=/var/local/mysqldisk bs=1M count=2000
losetup /dev/loop2 /var/local/mysqldisk
vgcreate VG_MYSQL /dev/loop2
lvcreate -l50%VG -n lv_datadir VG_MYSQL
mkfs.ext4 /dev/VG_MYSQL/lv_datadir
mkdir -p /datadir
mount /dev/VG_MYSQL/lv_datadir /datadir
rmdir /datadir/lost+found
cp -a /var/lib/mysql/* /datadir/
chown mysql:mysql /datadir
```

3. Create a directory called `/binlogs` and change the owner and group to `mysql`.
4. Change MySQL's data directory setting to point to the new `/datadir` data directory, and set binary logging to log to the `/binlogs/mysql-bin`.

5. Restart MySQL.
6. Review the contents of the `/binlogs` directory. Is the MySQL server writing the binlogs to this directory?
7. Using the `mysql` client from a new terminal window, drop the `world_innodb` database and re-create it using the `/labs/world_innodb.sql` file, as done in the practices for the “System Administration” lesson.
8. View the number of tables that the `world_innodb` database contains. How many tables does the `world_innodb` database contain?
9. View the processes that are running against the MySQL server by issuing the following command:

   ```
   SHOW PROCESSLIST;
   ```

   – Are there any queries running against the server?
10. Simulate heavy traffic on the server by executing steps 9–12 in the “MySQL Enterprise Backup” practice to re-create the table `City_Large`. Ensure that a large slow `INSERT` operation is executing while you create the snapshot in the following step. Set `autocommit` to 0 during the insert operations to extend the duration of the transaction.
11. While the previous step is still executing, flush the MySQL logs and create an LVM snapshot of the MySQL data directory by issuing the following commands at the terminal window logged in as `root`:

    ```
    mysqladmin -uroot -poracle flush-logs
    lvcreate -s -n lv_datadirbackup -L 500M /dev/VG_MYSQL/lv_datadir
    ```

12. Commit the transaction started in step 10 by setting `autocommit` back to `1`.
13. Add a new row to the `City` table, with the country code `SWE` and the name `Sakila`. Confirm the insert with a suitable query. Then drop the `City` table.

    Note: Steps 12 and 13 are performed *after* the snapshot but *before* the backup operation. The backup in the following step is done on the *snapshot*, not the live server.
14. Make a backup of the snapshot contents using the **`tar`** command, by mounting the snapshot to a local directory. When you complete the backup, unmount the snapshot.
15. Remove the snapshot by using the following command:

    ```
    lvremove VG_MYSQL/lv_datadirbackup -f
    ```

16. To simulate a disaster, execute the following command:

    ```
    lvresize -fL 1M VG_MYSQL/lv_datadir
    ```

    – This command resizes the logical volume to 1 MB, effectively destroying all data in its file system.

17. Execute some statements at a `mysql` prompt to see whether MySQL is still running, and then exit the client.
18. Attempt to shut down MySQL using the service command and using `mysqladmin`.
19. Verify that MySQL is running by issuing a `ps` command.
20. Kill the `mysqld_safe` process.
21. Re-execute the command in step 19 to verify that MySQL is no longer running.
22. To simulate fixing the faulty file system, execute the following commands in a terminal window logged in as `root` to remove the broken LVM logical volume and create a new one:

    ```
    umount /datadir
    lvremove -f VG_MYSQL/lv_datadir
    lvcreate -l50%VG -n lv_datadir VG_MYSQL
    mkfs.ext4 /dev/VG_MYSQL/lv_datadir
    mount /dev/VG_MYSQL/lv_datadir /datadir
    rmdir /datadir/lost+found
    chown mysql:mysql /datadir
    ```

    Note: These commands are similar to those in step 2, when you initially created the logical volume. You are not re-creating the volume group, the `/datadir` directory, or the copy of the old data directory. At the end of this step, the data directory is again empty.
23. Restore the contents of the backup taken in step 14.
24. Restart MySQL.
25. Log in to the `mysql` client from the terminal logged in as `oracle` and view the tables in the `world_innodb` database. Ensure that the `City` table exists, and establish whether the `Sakila` row added in step 13 exists.
26. In a terminal window, view the contents of the `/binlogs` directory.
    - Which binlog was started after the backup was made?
27. Starting with the binlog that was started after the backup was made, view the contents of each binlog to locate the binary log that contained the statement that drops the `City` table in the `world_innodb` database?
    - Which binlog contains the `DROP TABLE` command?
    - What is the position containing the `DROP TABLE` command?
28. Using the data recorded in the preceding step, create a SQL script and execute it with the `mysql` client that restores the `world_innodb` database up to the position of the `DROP TABLE` command.
29. Using the `mysql` client, view all the records in the `City` table of the `world_innodb` database that have an `ID` greater than 4070.
    - Is the record that you added in step 13 in the `City` table?
30. Change MySQL's data directory setting back to the original value, and restart MySQL to apply the changes.

Solutions 10-3: Backup and Recovery Using LVM Snapshots and the Binary Log

Tasks

1. Stop the MySQL server.

   Execute the following in a terminal window logged in as `root` and receive the results shown:

```
# service mysql stop
Shutting down MySQL..                                       [  OK  ]
```

2. Execute the following commands in a terminal window logged in as `root` to create a copy of the MySQL data directory on a loop device LVM logical volume.

```
dd if=/dev/zero of=/var/local/mysqldisk bs=1M count=2000
losetup /dev/loop2 /var/local/mysqldisk
vgcreate VG_MYSQL /dev/loop2
lvcreate -l50%VG -n lv_datadir VG_MYSQL
mkfs.ext4 /dev/VG_MYSQL/lv_datadir
mkdir -p /datadir
mount /dev/VG_MYSQL/lv_datadir /datadir
rmdir /datadir/lost+found
cp -a /var/lib/mysql/* /datadir/
chown mysql:mysql /datadir
```

3. Create a directory called `/binlogs` and change the owner and group to `mysql`.

```
# mkdir /binlogs
# chown mysql:mysql /binlogs
```

4. Change MySQL's data directory setting to point to the new `/datadir` data directory, and set binary logging to log to the `/binlogs/mysql-bin`.

   Open the `/etc/my.cnf` file as `root`, and make the following changes:

```
[mysqld]
...
log-bin=/binlogs/mysql-bin
...
datadir=/datadir
...
```

5. Restart MySQL.

   Execute the following in a terminal window logged in as `root` and receive the results shown:

```
# service mysql start
Starting MySQL.                                             [  OK  ]
```

6. Review the contents of the `/binlogs` directory.

   Enter the following in a terminal window, and receive the result shown below:

```
# ls /binlogs
mysql-bin.000001  mysql-bin.index
```

   – Is the MySQL server writing the binlogs to this directory? Yes

7. Using the `mysql` client from a new terminal window, drop the `world_innodb` database and re-create it using the `/labs/world_innodb.sql` file, as done in the practices for the "System Administration" lesson.

```
$ mysql --login-path=admin
Welcome to the MySQL monitor.  Commands end with ; or \g.
...
mysql> DROP DATABASE world_innodb;
Query OK, 4 rows affected (1.45 sec)

mysql> CREATE DATABASE world_innodb;
Query OK, 1 row affected (0.00 sec)

mysql> USE world_innodb
Database changed
mysql> SET autocommit=0;
Query OK, 0 rows affected (0.00 sec)

mysql> SOURCE /labs/world_innodb.sql
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
...
mysql> SET autocommit=1;
Query OK, 0 rows affected (0.04 sec)
```

8. View the number of tables that the `world_innodb` database contains.

   Enter the following in a `mysql` window, and receive the result shown below:

```
mysql> SHOW TABLES;
+------------------------+
| Tables_in_world_innodb |
+------------------------+
| City                   |
| Country                |
| CountryLanguage        |
+------------------------+
3 rows in set (0.00 sec)
```

   – How many tables does the `world_innodb` database contain? Three—the correct number of tables in a newly created `world_innodb` database.

9. View the processes that are running against the MySQL server by issuing the following command, and receive the result shown below:

```
mysql> SHOW PROCESSLIST;
+----+------+-----------+-----+---------+------+-------+--------------
---+
| Id | User | Host      | db  | Command | Time | State | Info
|
+----+------+-----------+-----+---------+------+-------+--------------
---+
|  2 | root | localhost | NULL| Query   |    0 | NULL  | SHOW
PROCESSLIST|
+----+------+-----------+-----+---------+------+-------+--------------
---+
1 row in set (0.00 sec)
```

   – Are there any queries running against the server? No

10. Simulate heavy traffic on the server by executing steps 9–12 in the "MySQL Enterprise Backup" practice to re-create the table `City_Large`. Ensure that a large slow `INSERT` operation is executing while you create the snapshot in the following step. Set **`autocommit`** to 0 during the insert operations to extend the duration of the transaction.

```
mysql> CREATE TABLE City_Large LIKE City;
Query OK, 0 rows affected (2.82 sec)

mysql> INSERT INTO City_Large (Name, CountryCode, District,
    ->  Population) SELECT Name, CountryCode, District,
    ->  Population FROM City;
Query OK, 4079 rows affected (0.14 sec)
Records: 4079  Duplicates: 0  Warnings: 0

mysql> SET autocommit=0;
```

   Issue the following command 6–10 times to vastly increase the number of records in the `City_Large` table and to cause traffic on the server while the snapshot is being taken. Use the up arrow on your keyboard to quickly repeat the commands.

```
mysql> INSERT INTO City_Large (Name, CountryCode, District,
    ->  Population) SELECT Name, CountryCode, District,
    ->  Population FROM City_Large;
```

11. While the previous step is still executing, flush the MySQL logs and create an LVM snapshot of the MySQL data directory by issuing the following commands at the terminal window logged in as `root`.

   Enter the following commands and receive the results shown:

```
# mysqladmin -uroot -poracle flush-logs ; \
  lvcreate -s -n lv_datadirbackup -L 500M /dev/VG_MYSQL/lv_datadir
  Logical volume "lv_datadirbackup" created
```

   Note: This technique does not permit you to get the binary log position, and is therefore not suited to creating backups in some situations such as when creating a replication slave.

12. Commit the transaction started in step 10 by setting `autocommit` back to `1`.

```
mysql> SET autocommit=1;
Query OK, 0 rows affected (0.66 sec)
```

13. Add a new row to the `City` table, with the country code `SWE` and the name `Sakila`. Confirm the insert with a suitable query. Then drop the `City` table.

    Enter the following in a `mysql` window, and receive the result shown below:

```
mysql> INSERT INTO City(Name, CountryCode) VALUES ('Sakila', 'SWE');
Query OK, 1 row affected (0.11 sec)

mysql> SELECT * FROM City ORDER BY ID DESC LIMIT 1;
+------+--------+-------------+----------+------------+
| ID   | Name   | CountryCode | District | Population |
+------+--------+-------------+----------+------------+
| 4080 | Sakila | SWE         |          |          0 |
+------+--------+-------------+----------+------------+
1 row in set (0.00 sec)

mysql> DROP TABLE City;
Query OK, 0 rows affected (0.31 sec)
```

Note: Steps 12 and 13 are performed *after* the snapshot but *before* the backup operation. The backup in the following step is done on the *snapshot*, not the live server.

14. Make a backup of the snapshot contents using the **`tar`** command, by mounting the snapshot to a local directory. When you complete the backup, unmount the snapshot.

    Enter the following commands in a terminal window logged in as `root`:

```
mkdir /root/snapshot
mount /dev/VG_MYSQL/lv_datadirbackup /root/snapshot
cd /root/snapshot
tar -czf /root/mysqldatadir.tgz .
cd
umount /root/snapshot
```

15. Remove the snapshot by using the following command:

    Enter the following commands in a terminal window logged in as `root` and receive the results shown:

```
# lvremove VG_MYSQL/lv_datadirbackup -f
  Logical volume "lv_datadirbackup" successfully removed
```

16. To simulate a disaster, execute the following command.

    Enter the following command in a terminal window logged in as `root`:

```
# lvresize -fL 1M VG_MYSQL/lv_datadir
```

    – This command resizes the logical volume to the smallest physical extent (4 MB, rounded up from the specified 1 MB), effectively destroying all data in its file system.

17. Execute some statements at a `mysql` prompt to see whether MySQL is still running, and then exit the client.

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| world3             |
| world_innodb       |
+--------------------+
5 rows in set (0.00 sec)

mysql> DROP DATABASE world3;
No connection. Trying to reconnect...
ERROR 2002 (HY000): Can't connect to local MySQL server through socket
'/var/lib/mysql/mysql.sock' (2)

mysql> EXIT
Bye
```

Note: Depending on how much information MySQL has cached, it can crash sooner. Try commands like `STATUS` and `SHOW DATABASES` and several **`USE`** statements, before trying to modify a database. As soon as MySQL cannot log or record changes, it crashes.

Note: You might see different errors, such as "ERROR 2013 (HY000): Lost connection to MySQL server during query" or "ERROR 2006 (HY000): MySQL server has gone away." You might also see operating system messages such as "kernel:journal commit I/O error", referring to the serious file system problems caused by shrinking the volume.

18. Attempt to shut down MySQL using the service command and using `mysqladmin`.

    Enter the following commands in a terminal window logged in as `root`:

```
# service mysql stop
MySQL server process #17951 is not running!                [FAILED]
rm: cannot remove `/datadir/<hostname>.pid': Read-only file system
# mysqladmin -uroot -poracle shutdown
mysqladmin: connect to server at 'localhost' failed
error: 'Can't connect to local MySQL server through socket
'/var/lib/mysql/mysql.sock' (2)'
Check that mysqld is running and that the socket:
'/var/lib/mysql/mysql.sock' exists!
```

Note: You have simulated a serious hard-disk failure, so the error message can differ from that shown.

19. Verify that MySQL is running by issuing a `ps` command.

    Enter the following commands in a terminal window logged in as `root`:

    ```
    # ps ax | grep mysqld
    17738 pts/2    S      2:11 /bin/sh /usr/bin/mysqld_safe --
    datadir=/datadir --pid-file=/datadir/<hostname>.pid
    17846 pts/2    R      0:00 /usr/sbin/mysqld --basedir=/usr --
    datadir=/datadir --plugin-dir=/usr/lib64/mysql/plugin --user=mysql --
    log-error=/var/log/mysqld.log --pid-file=/datadir/<hostname>.pid --
    socket=/var/lib/mysql/mysql.sock
    17849 pts/2    S+     0:00 grep mysqld
    ```

    MySQL is still running because it is restarted by the `mysqld_safe` script every time it crashes. If you run this statement multiple times, you will note that sometimes `mysqld` is running and sometimes it is not.

20. Kill the `mysqld_safe` process.

    Enter the following commands in a terminal window logged in as `root`:

    ```
    # killall -9 mysqld_safe
    ```

21. Re-execute the command in step 19 to verify that MySQL is no longer running.

    Enter the following commands in a terminal window logged in as `root`:

    ```
    # ps ax | grep mysqld
    11528 pts/2    S+     0:00 grep mysqld
    ```

    – MySQL is no longer running.

22. To simulate fixing the faulty file system, execute the following commands in a terminal window logged in as `root` to remove the broken LVM logical volume and create a new one:

    ```
    umount /datadir
    lvremove -f VG_MYSQL/lv_datadir
    lvcreate -l50%VG -n lv_datadir VG_MYSQL
    mkfs.ext4 /dev/VG_MYSQL/lv_datadir
    mount /dev/VG_MYSQL/lv_datadir /datadir
    rmdir /datadir/lost+found
    chown mysql:mysql /datadir
    ```

    Note: These commands are similar to those in step 2, when you initially created the logical volume. You are not re-creating the volume group, the `/datadir` directory, or the copy of the old data directory. At the end of this step, the data directory is again empty.

23. Restore the contents of the backup taken in step 14.

    Enter the following commands in a terminal window logged in as `root`:

    ```
    # cd /datadir
    # tar -xzf /root/mysqldatadir.tgz
    ```

24. Restart MySQL.

    Execute the following in a terminal window logged in as `root` and receive the results shown:

```
# service mysql start
Starting MySQL....                                                 [  OK  ]
```

    – Were you able to start the MySQL server? Yes

Note: The start operation can take some time if you created the snapshot during a period of heavy activity, because InnoDB needs to perform recovery when starting after an abnormal shutdown, and a snapshot of a running system is indistinguishable from a power cut or other crash.

25. Log in to the `mysql` client from the terminal logged in as `oracle` and view the tables in the `world_innodb` database. Ensure that the `City` table exists, and establish whether the `Sakila` row added in step 12 exists.

    Enter the following in a terminal window, and receive the result shown below:

```
$ mysql --login-path=admin
Welcome to the MySQL monitor.  Commands end with ; or \g.
...
mysql> USE world_innodb
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT * FROM City ORDER BY ID DESC LIMIT 1;
+------+-------+-------------+----------+------------+
| ID   | Name  | CountryCode | District | Population |
+------+-------+-------------+----------+------------+
| 4079 | Rafah | PSE         | Rafah    |      92020 |
+------+-------+-------------+----------+------------+
1 row in set (0.00 sec)
```

    The `City` table exists, but the `Sakila` row does not, because the backup came from a snapshot that was taken before you created the row and dropped the `City` table. You can use the binary log to recover changes made since taking the backup.

26. In a terminal window, view the contents of the `/binlogs` directory.

    Enter the following in a terminal window, and receive the result shown below:

```
# ls /binlogs -l
total 856
-rw-rw---- 1 mysql mysql 863061 Feb 11 05:09 mysql-bin.000001
-rw-rw---- 1 mysql mysql    940 Feb 11 05:42 mysql-bin.000002
-rw-rw---- 1 mysql mysql    120 Feb 11 05:42 mysql-bin.000003
-rw-rw---- 1 mysql mysql     78 Feb 11 05:42 mysql-bin.index
```

    – Which binlog was started after the backup was made? mysql-bin.000002

27. Starting with the first binlog created after the snapshot, view the contents of each binlog to locate the binary log containing the statement that drops the `City` table in the `world_innodb` database?

    Enter the following in a terminal window, and receive the result shown below:

```
# mysqlbinlog /binlogs/mysql-bin.000002 | more
/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=1*/;
/*!40019 SET @@session.max_insert_delayed_threads=0*/;
/*!50003 SET
@OLD_COMPLETION_TYPE=@@COMPLETION_TYPE,COMPLETION_TYPE=0*/;
DELIMITER /*!*/;
# at 4
#130211  5:09:50 server id 1  end_log_pos 120 CRC32 0xc405eb90  Start:
binlog v 4, server v 5.6
.10-enterprise-commercial-advanced-log created 130211  5:09:50
BINLOG '
Hn0YUQ8BAAAAdAAAAHgAAAAAAAQANS42LjEwLWVudGVycHJpc2UtY29tbWVyY2lhbC1hZH
ZhbmNl
ZC1sb2cAAAAAAAAAAAAAAAAAEzgNAAgAEgAEBAQEEgAAXAAEGggAAAAICAgCAAAACgoKGR
kAAZDr
BcQ=
'/*!*/;
# at 120
#130211  5:09:36 server id 1  end_log_pos 215 CRC32 0xa312ddd4  Query
thread_id=3     exec_ti
me=28   error_code=0
SET TIMESTAMP=1360559376/*!*/;
SET @@session.pseudo_thread_id=3/*!*/;
SET @@session.foreign_key_checks=1, @@session.sql_auto_is_null=0,
@@session.unique_checks=1, @@
session.autocommit=1/*!*/;
SET @@session.sql_mode=1075838976/*!*/;
SET @@session.auto_increment_increment=1,
@@session.auto_increment_offset=1/*!*/;
/*!\C utf8 *//*!*/;
SET
@@session.character_set_client=33,@@session.collation_connection=33,@@
session.collation_ser
ver=8/*!*/;
SET @@session.lc_time_names=0/*!*/;
SET @@session.collation_database=DEFAULT/*!*/;
BEGIN
/*!*/;
# at 215
# at 247
#130211  5:09:36 server id 1  end_log_pos 247 CRC32 0x3fd40695  Intvar
SET INSERT_ID=524277/*!*/;
```

```
#130211  5:09:36 server id 1  end_log_pos 468 CRC32 0xccb3fceb  Query
thread_id=3     exec_ti
me=28   error_code=0
use `world_innodb`/*!*/;
SET TIMESTAMP=1360559376/*!*/;
INSERT INTO City_Large (Name, CountryCode, District,
 Population) SELECT Name, CountryCode, District,
 Population FROM City_Large
/*!*/;
# at 468
#130211  5:09:36 server id 1  end_log_pos 499 CRC32 0x5c1a7c2e  Xid =
5572
COMMIT/*!*/;
# at 499
#130211  5:12:46 server id 1  end_log_pos 594 CRC32 0xc74b18b0  Query
thread_id=3     exec_ti
me=0    error_code=0
SET TIMESTAMP=1360559566/*!*/;
BEGIN
/*!*/;
# at 594
# at 626
#130211  5:12:46 server id 1  end_log_pos 626 CRC32 0x36ec76bb  Intvar
SET INSERT_ID=4080/*!*/;
#130211  5:12:46 server id 1  end_log_pos 776 CRC32 0xdaffdd6e  Query
thread_id=3     exec_ti
me=0    error_code=0
SET TIMESTAMP=1360559566/*!*/;
INSERT INTO City(Name, CountryCode) VALUES ('Sakila', 'SWE')
/*!*/;
# at 776
#130211  5:12:46 server id 1  end_log_pos 807 CRC32 0x0deff828  Xid =
5626
COMMIT/*!*/;
# at 807
#130211  5:13:01 server id 1  end_log_pos 940 CRC32 0xce845fd6  Query
thread_id=3     exec_ti
me=2    error_code=0
SET TIMESTAMP=1360559581/*!*/;
DROP TABLE `City` /* generated by server */
/*!*/;
DELIMITER ;
# End of log file
ROLLBACK /* added by mysqlbinlog */;
/*!50003 SET COMPLETION_TYPE=@OLD_COMPLETION_TYPE*/;
/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=0*/;
```

– Which binlog contains the `DROP TABLE` command? mysql-bin.000002

– What is the position containing the `DROP TABLE` command? The `DROP TABLE` command happens at position 807.

Note: These values may differ on your system.

28. Using the data recorded in the preceding step, create a SQL script and execute it with the `mysql` client that restores the `world_innodb` database up to the position of the `DROP TABLE` command.

```
# cd /binlogs
# mysqlbinlog --disable-log-bin --stop-position=807 \
    mysql-bin.000002 | mysql -uroot -poracle
```

29. Using the `mysql` client, view all the records in the `City` table of the `world_innodb` database that have an **ID** greater than 4070.

Enter the following in a terminal window, and receive the result shown below:

```
mysql> SELECT * FROM world_innodb.City WHERE ID > 4070;
+------+-------------+-------------+-----------+------------+
| ID   | Name        | CountryCode | District  | Population |
+------+-------------+-------------+-----------+------------+
| 4071 | Mount Darwin | ZWE        | Harare    |     164362 |
| 4072 | Mutare      | ZWE         | Manicaland |    131367 |
| 4073 | Gweru       | ZWE         | Midlands  |     128037 |
| 4074 | Gaza        | PSE         | Gaza      |     353632 |
| 4075 | Khan Yunis  | PSE         | Khan Yunis |    123175 |
| 4076 | Hebron      | PSE         | Hebron    |     119401 |
| 4077 | Jabaliya    | PSE         | North Gaza |    113901 |
| 4078 | Nablus      | PSE         | Nablus    |     100231 |
| 4079 | Rafah       | PSE         | Rafah     |      92020 |
| 4080 | Sakila      | SWE         |           |          1 |
+------+-------------+-------------+-----------+------------+
10 rows in set (0.00 sec)
```

Is the record that you added in step 13 in the `City` table? Yes.

30. Change MySQL's data directory setting back to the original value, and restart MySQL to apply the changes.

a. Execute the following in a terminal window logged in as root and receive the results shown:

```
# service mysql stop
Shutting down MySQL..                                      [  OK  ]
```

b. Open the **/etc/my.cnf** file as `root`, and make the following change:

```
[mysqld]
...
datadir=/var/lib/mysql
...
```

c. After saving the preceding changes, enter the following in a terminal window, and receive the result shown below:

```
# service mysql start
Starting MySQL.                                            [  OK  ]
```

Practices for Lesson 11: Replication

Chapter 11

Practice 11-1: Quiz – Replication

Overview

In this practice, you answer questions about replication.

Duration

This practice should take approximately five minutes to complete.

Quiz Questions

Choose the best answer from those provided for each multiple choice or True/False question.

1. Which of the following is true concerning MySQL master servers?
   a. There is no limit on the number of slaves a single master can have.
   b. It is possible for a slave to have a different MySQL version from the master.
   c. It is common to limit the number of slaves to less than 30 in most production setups.
   d. All of the above
2. At its simplest, MySQL replication works through a one-way, log-shipping, asynchronous mechanism, making it a master-slave relationship.
   a. True
   b. False
3. Which of the following is a common use for replication?
   a. Scale-out solutions
   b. High availability
   c. Analytics
   d. All of the above
4. MySQL replication uses a log-shipping system in which all data changes that occur on the master are stored in a log and then retrieved by the slave and executed from these received log files. What is the name of this log file in MySQL?
   a. Slave log
   b. Master log
   c. Binary log
   d. Error log
5. Slaves need to be connected permanently to receive updates from the master.
   a. True
   b. False
6. What is a disadvantage to using statement-based replication?
   a. Disk space usage and bandwidth requirements for replication are larger.
   b. Replication occurs on the row level.
   c. Some functions might not replicate correctly to a remote server of a different version.

d. None of the above

7. Which thread is responsible for downloading the binary logs from the master into a local file set called the relay logs?

   a. `BINARY_THREAD`
   b. `IO_THREAD`
   c. `SQL_THREAD`
   d. `MASTER_THREAD`

Solutions 11-1: Quiz – Replication

Quiz Solutions

1. **d.** All of the above
2. **a.** True
3. **d.** All of the above
4. **c.** Binary log
5. **b.** False. Slaves do not need to be connected permanently to receive updates from the master.
6. **c.** Some functions might not replicate correctly to a remote server of a different version.
7. **b.** `IO_THREAD`

Practice 11-2: Configuring Replication

Overview

In this practice, you start four server instances of MySQL, configure one server as a slave of another, create some data on the master, and see that it replicates to the slave.

Assumptions

The MySQL server is installed, and the file `/labs/repl.cnf` exists and has contents as shown in the solution to step 2 of this practice.

Tasks

1. Stop the MySQL server.
2. View the contents of the file `/labs/repl.cnf`.
3. Using `mysqld_multi`, start the four servers defined in `/labs/repl.cnf`.
4. In a new terminal window, use the `mysql` client to connect to the first server as `root`, and set the prompt to "`1> `". **Note:** For the sake of simplicity and brevity in a training environment, the four servers do not have a `root` password.
5. Execute a query to find the log coordinates of the first server.
6. On the first server, create a user called `repl`, with the password `oracle`, and grant that user the `REPLICATION SLAVE` permission.
7. On the first server, create the `world_innodb` database from script, as you did in the practices for the "System Administration" lesson.
8. In a new terminal window, use the `mysql` client to connect to the second server as `root`, and set the prompt to "`2> `".
9. In the second `mysql` window, issue a `CHANGE MASTER TO`… command to configure the second server as a slave of the first, using the log coordinates noted in step 5 and the user created in step 6. Display the text of any warnings generated.
10. Display the databases that exist on the second server.
11. Start the slave threads on the second server.
12. Show the processes on the first and second server.
13. Display the databases on the second server again. Note the differences.

Solutions 11-2: Configuring Replication

Tasks

1. Stop the MySQL server.
   a. Enter the following at a terminal logged in as root, and receive the results shown:

```
# service mysql stop
Shutting down MySQL..                                      [  OK  ]
```

2. View the contents of the file `/labs/repl.cnf`.
   a. Enter the following at a terminal logged in as root, and receive the results shown:

```
# cat /labs/repl.cnf
[mysqld1]
datadir=/var/lib/mysql1
port=3311
socket=/var/lib/mysql1/mysql.sock
server-id=1
user=mysql
log-bin=mysql1-bin
relay-log=mysql1-relay-bin
log-slave-updates
log-error=mysql1
report-host=localhost
report-port=3311
relay-log-recovery=1
master-info-repository=TABLE
relay-log-info-repository=TABLE
# gtid-mode=ON
# enforce-gtid-consistency

[mysqld2]
datadir=/var/lib/mysql2
port=3312
socket=/var/lib/mysql2/mysql.sock
server-id=2
user=mysql
log-bin=mysql2-bin
relay-log=mysql2-relay-bin
log-slave-updates
log-error=mysql2
report-host=localhost
report-port=3312
relay-log-recovery=1
master-info-repository=TABLE
relay-log-info-repository=TABLE
```

```
# gtid-mode=ON
# enforce-gtid-consistency

[mysqld3]
datadir=/var/lib/mysql3
port=3313
socket=/var/lib/mysql3/mysql.sock
server-id=3
user=mysql
log-bin=mysql3-bin
relay-log=mysql3-relay-bin
log-slave-updates
log-error=mysql3
report-host=localhost
report-port=3313
relay-log-recovery=1
master-info-repository=TABLE
relay-log-info-repository=TABLE
# gtid-mode=ON
# enforce-gtid-consistency

[mysqld4]
datadir=/var/lib/mysql4
port=3314
socket=/var/lib/mysql4/mysql.sock
server-id=4
user=mysql
log-bin=mysql4-bin
relay-log=mysql4-relay-bin
log-slave-updates
log-error=mysql4
report-host=localhost
report-port=3314
relay-log-recovery=1
master-info-repository=TABLE
relay-log-info-repository=TABLE
# gtid-mode=ON
# enforce-gtid-consistency
```

3. Using `mysqld_multi`, start the four servers defined in `/labs/repl.cnf`.
   a. Enter the following at a terminal logged in as root, and receive the results shown:

```
# mysqld_multi --defaults-file=/labs/repl.cnf start 1-4

Installing new database in /var/lib/mysql1

Installing MySQL system tables...2012-12-16 13:08:17 0 [Warning]
TIMESTAMP with implicit DEFAULT value is deprecated. Please use --
explicit_defaults_for_timestamp server option (see documentation for
more details).
2012-12-16 13:08:17 6876 [Note] InnoDB: The InnoDB memory heap is
disabled
2012-12-16 13:08:17 6876 [Note] InnoDB: Mutexes and rw_locks use GCC
atomic builtins
2012-12-16 13:08:17 6876 [Note] InnoDB: Compressed tables use zlib
1.2.3
2012-12-16 13:08:17 6876 [Note] InnoDB: CPU does not support crc32
instructions
2012-12-16 13:08:17 6876 [Note] InnoDB: Using Linux native AIO
2012-12-16 13:08:17 6876 [Note] InnoDB: Initializing buffer pool, size
= 128.0M
2012-12-16 13:08:17 6876 [Note] InnoDB: Completed initialization of
buffer pool
2012-12-16 13:08:17 6876 [Note] InnoDB: The first specified data file
./ibdata1 did not exist: a new database to be created!
...
```

   – This command starts four server instances with the settings stored in `/labs/repl.cnf`. The first time the servers start, the process creates data directories as specified in the options for each server.

4. In a new terminal window, use the `mysql` client to connect to the first server as `root`, and set the prompt to “`1> `”. **Note:** For the sake of simplicity and brevity in a training environment, the four servers do not have a `root` password.
   a. Enter the following at a new terminal, and receive the results shown:

```
$ mysql -uroot -h127.0.0.1 -P3311
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
...
mysql> PROMPT 1> ;
PROMPT set to '1> '
```

5. Execute a query to find the log coordinates of the first server.
   a. In the `mysql` prompt created in the preceding step, enter the following command and receive the results shown:

```
1> SHOW MASTER STATUS \G
*************************** 1. row ***************************
             File: mysql-bin.000001
         Position: 120
     Binlog_Do_DB:
 Binlog_Ignore_DB:
Executed_Gtid_Set:
1 row in set (0.00 sec)
```

   – Note that the log file is `mysql-bin.000001` and the log position is `120`.

6. On the first server, create a user called `repl`, with the password `oracle`, and grant that user the `REPLICATION SLAVE` permission.
   a. In the `mysql` prompt used in the preceding step, enter the following command and receive the results shown:

```
1> CREATE USER 'repl'@'127.0.0.1' IDENTIFIED BY 'oracle';
Query OK, 0 rows affected (0.00 sec)

1> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'127.0.0.1';
Query OK, 0 rows affected (0.00 sec)
```

7. On the first server, create the `world_innodb` database from script, as you did in the practices for the “System Administration” lesson.
   a. In the `mysql` prompt used in the preceding step, enter the following command and receive the results shown:

```
1> CREATE DATABASE world_innodb;
Query OK, 1 row affected (0.00 sec)

1> USE world_innodb
Database changed
1> SET autocommit=0;
Query OK, 0 rows affected (0.00 sec)

1> SOURCE /labs/world_innodb.sql
Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)
...
1> SET autocommit=1;
Query OK, 0 rows affected (0.06 sec)
```

8. In a new terminal window, use the `mysql` client to connect to the second server as `root`, and set the prompt to "`2> `".

   a. Enter the following at a new terminal, and receive the results shown:

```
$ mysql -uroot -h127.0.0.1 -P3312
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
...
mysql> PROMPT 2> ;
PROMPT set to '2> '
```

9. In the second `mysql` window, issue a `CHANGE MASTER TO`… command to configure the second server as a slave of the first, using the log coordinates noted in step 5 and the user created in step 6. Display the text of any warnings generated.

   a. In the `mysql` prompt created in the preceding step, enter the following command and receive the results shown:

```
2> CHANGE MASTER TO
    -> MASTER_HOST='127.0.0.1',
    -> MASTER_PORT=3311,
    -> MASTER_LOG_FILE='mysql1-bin.000001',
    -> MASTER_LOG_POS=120,
    -> MASTER_USER='repl',
    -> MASTER_PASSWORD='oracle';
Query OK, 0 rows affected, 2 warnings (0.51 sec)

2> SHOW WARNINGS\G
*************************** 1. row ***************************
  Level: Note
   Code: 1759
Message: Sending passwords in plain text without SSL/TLS is extremely
insecure.
*************************** 2. row ***************************
  Level: Note
   Code: 1760
Message: Storing MySQL user name or password information in the
master.info repository is not secure and is therefore not recommended.
Please see the MySQL Manual for more about this issue and possible
alternatives.
2 rows in set (0.00 sec)
```

10. Display the databases that exist on the second server.

    a. In the `mysql` prompt used in the preceding step, enter the following command and receive the results shown:

```
2> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
4 rows in set (0.00 sec)
```

11. Start the slave threads on the second server.

    a. In the `mysql` prompt used in the preceding step, enter the following command and receive the results shown:

```
2> START SLAVE;
Query OK, 0 rows affected (0.05 sec)
```

12. Show the processes on the first and second server.

    a. At the first `mysql` prompt, issue the following command and receive the results shown:

```
1> SHOW PROCESSLIST\G
*************************** 1. row ***************************
     Id: 1
   User: root
   Host: localhost:35112
     db: world_innodb
Command: Query
   Time: 0
  State: init
   Info: SHOW PROCESSLIST
*************************** 2. row ***************************
     Id: 2
   User: repl
   Host: localhost:35115
     db: NULL
Command: Binlog Dump
   Time: 1944
  State: Master has sent all binlog to slave; waiting for binlog to be
updated
   Info: NULL
2 rows in set (0.00 sec)
```

b. At the second `mysql` prompt, issue the following command and receive the results shown:

```
2> SHOW PROCESSLIST\G
*************************** 1. row ***************************
     Id: 1
   User: root
   Host: localhost:39075
     db: NULL
Command: Query
   Time: 0
  State: init
   Info: SHOW PROCESSLIST
*************************** 2. row ***************************
     Id: 2
   User: system user
   Host:
     db: NULL
Command: Connect
   Time: 1811
  State: Waiting for master to send event
   Info: NULL
*************************** 3. row ***************************
     Id: 3
   User: system user
   Host:
     db: NULL
Command: Connect
   Time: 2311
  State: Slave has read all relay log; waiting for the slave I/O
thread to update it
   Info: NULL
3 rows in set (0.00 sec)
```

13. Display the databases on the second server again. Note the differences.

    a. In the `mysql` prompt used in the preceding step, enter the following command and receive the results shown:

```
2> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
| world_innodb       |
+--------------------+
5 rows in set (0.00 sec)
```

    – The `world_innodb` database is in the list, because it was created on the first server after you noted the log coordinates in step 5. The slave process on the second server replicated every change after those coordinates, including all data in the `world_innodb` database, and the user created in step 6.

Practice 11-3: Adding a New Slave

Overview

In this practice, you provision a new server as a new slave of the existing slave (using `mysqldump`), change some data on the master, and see that it replicates to both slaves.

Assumptions

You have successfully carried out the steps in Practice 17-2.

Tasks

1. Using `mysqldump`, take a backup of the `world_innodb` database on the second server, including the information needed to create and use the database, and configure a slave.
2. Edit the backup file created in the preceding step to change the `CHANGE MASTER TO …` line so that it points to the second server, and is uncommented.
3. Connect to the third server using the `mysql` client, and set the prompt to "`3> `".
4. Apply the backup taken in step 1 and altered in step 2 to the third server.
5. Start the slave process on the third server, and display the slave status.
6. Delete all rows where the ID is greater than 4070 from the `City` table on the first server, and ensure that the changes replicate to the second and third servers.
7. To prepare for the following practice, create a user on the third server called `repl`, with the password `oracle`, and grant that user the `REPLICATION SLAVE` permission.

Solutions 11-3: Adding a New Slave

Tasks

1. Using mysqldump, take a backup of the world_innodb database on the second server, including the information needed to create and use the database, and configure a slave.
   a. Enter the following at a terminal window, and receive the results shown:

```
$ mysqldump -uroot -h127.0.0.1 -P3312 --master-data=2 \
     -B world_innodb > /tmp/server2.sql
```

2. Edit the backup file created in the preceding step to change the CHANGE MASTER TO … line so that it points to the second server, and is uncommented.
   a. Using an editor such as gedit, open the /tmp/server2.sql file and locate the following line:

```
-- CHANGE MASTER TO MASTER_LOG_FILE='mysql2-bin.000001',
MASTER_LOG_POS=860071;
```

   b. Change that line as follows:

```
CHANGE MASTER TO MASTER_HOST='127.0.0.1', MASTER_PORT=3312,
MASTER_USER='repl', MASTER_PASSWORD='oracle',
MASTER_LOG_FILE='mysql2-bin.000001', MASTER_LOG_POS=860071;
```

   – Note that the log coordinates may differ from that shown in your output, and that the replication user on the second server was originally created on the first server and replicated in the preceding practice.

3. Connect to the third server using the mysql client, and set the prompt to "3> ".
   a. Enter the following at a new terminal window, and receive the results shown:

```
$ mysql -uroot -h127.0.0.1 -P3313
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
...
mysql> PROMPT 3> ;
PROMPT set to '3> '
```

4. Apply the backup taken in step 1 and altered in step 2 to the third server.
   a. In the mysql prompt created in the preceding step, enter the following command and receive the results shown:

```
3> SOURCE /tmp/server2.sql
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
...
```

5. Start the slave process on the third server, and display the slave status.

   a. In the `mysql` prompt used in the preceding step, enter the following commands and receive the results shown:

```
3> START SLAVE;
Query OK, 0 rows affected (0.05 sec)

3> SHOW SLAVE STATUS\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 127.0.0.1
                  Master_User: repl
                  Master_Port: 3312
                Connect_Retry: 60
              Master_Log_File: mysql2-bin.000001
          Read_Master_Log_Pos: 860071
               Relay_Log_File: mysql3-relay-bin.000002
                Relay_Log_Pos: 284
        Relay_Master_Log_File: mysql2-bin.000001
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
...
```

6. Delete all rows where the ID is greater than 4070 from the `City` table on the first server, and ensure that the changes replicate to the second and third servers.

   a. On the first server, enter the following command and receive the results shown:

```
1> DELETE FROM world_innodb.City WHERE ID > 4070;
Query OK, 9 rows affected (0.05 sec)
```

   b. On the second server, enter the following command and receive the results shown:

```
2> SELECT * FROM world_innodb.City ORDER BY Id DESC LIMIT 5;
+------+------------------+-------------+----------------+------------+
| ID   | Name             | CountryCode | District       | Population |
+------+------------------+-------------+----------------+------------+
| 4070 | Chitungwiza      | ZWE         | Harare         |     274912 |
| 4069 | Bulawayo         | ZWE         | Bulawayo       |     621742 |
| 4068 | Harare           | ZWE         | Harare         |    1410000 |
| 4067 | Charlotte Amalie | VIR         | St Thomas      |      13000 |
| 4066 | Charleston       | USA         | South Carolina |      89063 |
+------+------------------+-------------+----------------+------------+
5 rows in set (0.00 sec)
```

c. On the third server, enter the following command and receive the results shown:

```
3> SELECT * FROM world_innodb.City ORDER BY Id DESC LIMIT 5;
+------+------------------+-------------+----------------+------------+
| ID   | Name             | CountryCode | District       | Population |
+------+------------------+-------------+----------------+------------+
| 4070 | Chitungwiza      | ZWE         | Harare         |     274912 |
| 4069 | Bulawayo         | ZWE         | Bulawayo       |     621742 |
| 4068 | Harare           | ZWE         | Harare         |    1410000 |
| 4067 | Charlotte Amalie | VIR         | St Thomas      |      13000 |
| 4066 | Charleston       | USA         | South Carolina |      89063 |
+------+------------------+-------------+----------------+------------+
5 rows in set (0.00 sec)
```

7. To prepare for the following practice, create a user on the third server called `repl`, with the password `oracle`, and grant that user the `REPLICATION SLAVE` permission.

   a. In the `mysql` prompt used in the preceding step, enter the following command and receive the results shown:

```
3> CREATE USER 'repl'@'127.0.0.1' IDENTIFIED BY 'oracle';
Query OK, 0 rows affected (0.00 sec)

3> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'127.0.0.1';
Query OK, 0 rows affected (0.00 sec)
```

   At this point, the `repl` user exists on three servers, having been created on the first server, replicated to the second, and finally created on the third.

Practice 11-4: Enabling GTID and Configuring Circular Replication

Overview

In this practice, you enable GTID on the three servers, connect the master so that it becomes a slave of the second slave, and test the newly created circular topology by changing some data.

Tasks

1. Using `mysqld_multi`, stop all four servers.
2. Edit the `/labs/repl.cnf` file to uncomment all commented lines.
3. Using `mysqld_multi`, start all four servers.
4. Stop the slaves.
5. Issue a `RESET MASTER` command on each server so that the log files contain only events that use GTIDs.
6. Issue an appropriate `CHANGE MASTER TO…` command on the slaves to use the GTID replication protocol.
7. Restart the slaves.
8. Delete all rows where the ID is greater than 4060 from the `City` table on the first server, and ensure that the changes replicate to the second and third servers.
9. Note the server UUIDs for the first, second, and third servers.
10. On the third server, view the slave status.
11. Issue an appropriate `CHANGE MASTER TO…` statement on the first server, configuring it as a slave to the third server, and start the slave threads.
12. Delete all rows where the ID is greater than 4050 from the `City` table on the second server, and ensure that the changes replicate to the first and third servers.

Solutions 11-4: Enabling GTID and Configuring Circular Replication

Tasks

Note: Execute all Linux terminal commands in this practice at a prompt logged in as `root`.

1. Using `mysqld_multi`, stop all four servers.
   a. Enter the following at a terminal window:

```
# mysqld_multi --defaults-file=/labs/repl.cnf --user=root stop 1-4
```

2. Edit the `/labs/repl.cnf` file to uncomment all commented lines.
   a. Using an editor such as `gedit`, open the `/labs/repl.cnf` file and find all commented lines:

```
# gtid-mode=ON
# enforce-gtid-consistency
```

   b. Remove the comments, as follows:

```
gtid-mode=ON
enforce-gtid-consistency
```

   c. Repeat for all such commented lines (one pair for each server option group) and save and close the file.

3. Using `mysqld_multi`, start all four servers.
   a. Enter the following at a terminal window:

```
# mysqld_multi --defaults-file=/labs/repl.cnf start 1-4
```

   − Note that the command does not create the data directories (as in a previous practice), because they already exist.

4. Stop the slaves.
   a. On the second server, enter the following command and receive the results shown:

```
2> STATUS
ERROR 2013 (HY000): Lost connection to MySQL server during query
2> STOP SLAVE;
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id:    2
Current database: world_innodb

Query OK, 0 rows affected (0.15 sec)
```

   b. On the third server, enter the following command and receive the results shown:

```
3> STATUS
ERROR 2013 (HY000): Lost connection to MySQL server during query
3> STOP SLAVE;
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id:    2
```

```
Current database: world_innodb

Query OK, 0 rows affected (0.12 sec)
```

5. Issue a `RESET MASTER` command on each server so that the log files contain only events that use GTIDs.

   a. In the first server connection, issue any command (for example, `STATUS`) to reconnect, and then reset the master settings, as follows:

```
1> STATUS
ERROR 2013 (HY000): Lost connection to MySQL server during query
1> RESET MASTER;
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id:    2
Current database: world_innodb

Query OK, 0 rows affected (0.21 sec)
```

   b. In the second server connection, issue the following command and receive the result shown:

```
2> RESET MASTER;
Query OK, 0 rows affected (0.21 sec)
```

   c. In the third server connection, issue the following command and receive the result shown:

```
3> RESET MASTER;
Query OK, 0 rows affected (0.22 sec)
```

6. Issue an appropriate `CHANGE MASTER TO`… command on the slaves to use the GTID replication protocol.

   a. In the second server connection, issue the following command and receive the result shown:

```
2> CHANGE MASTER TO MASTER_AUTO_POSITION=1;
Query OK, 0 rows affected (0.24 sec)
```

   b. In the third server connection, issue the following command and receive the result shown:

```
3> CHANGE MASTER TO MASTER_AUTO_POSITION=1;
Query OK, 0 rows affected (0.25 sec)
```

7. Restart the slaves.

   a. In the `mysql` prompt connected to the second server, enter the following command and receive the results shown:

```
2> START SLAVE;
Query OK, 0 rows affected (0.05 sec)
```

b. In the `mysql` prompt connected to the third server, enter the following command and receive the results shown:

```
3> START SLAVE;
Query OK, 0 rows affected (0.04 sec)
```

8. Delete all rows where the ID is greater than 4060 from the `City` table on the first server, and ensure that the changes replicate to the second and third servers.

   a. On the first server, enter the following command and receive the results shown:

```
1> DELETE FROM world_innodb.City WHERE ID > 4060;
```

   b. On the second server, enter the following command and receive the results shown:

```
2> SELECT * FROM world_innodb.City ORDER BY Id DESC LIMIT 5;
+------+--------------+-------------+----------------+------------+
| ID   | Name         | CountryCode | District       | Population |
+------+--------------+-------------+----------------+------------+
| 4060 | Santa Monica | USA         | California     |      91084 |
| 4059 | Cary         | USA         | North Carolina |      91213 |
| 4058 | Boulder      | USA         | Colorado       |      91238 |
| 4057 | Visalia      | USA         | California     |      91762 |
| 4056 | San Mateo    | USA         | California     |      91799 |
+------+--------------+-------------+----------------+------------+
5 rows in set (0.00 sec)
```

   c. On the third server, enter the following command and receive the results shown:

```
3> SELECT * FROM world_innodb.City ORDER BY Id DESC LIMIT 5;
+------+--------------+-------------+----------------+------------+
| ID   | Name         | CountryCode | District       | Population |
+------+--------------+-------------+----------------+------------+
| 4060 | Santa Monica | USA         | California     |      91084 |
| 4059 | Cary         | USA         | North Carolina |      91213 |
| 4058 | Boulder      | USA         | Colorado       |      91238 |
| 4057 | Visalia      | USA         | California     |      91762 |
| 4056 | San Mateo    | USA         | California     |      91799 |
+------+--------------+-------------+----------------+------------+
5 rows in set (0.00 sec)
```

9. Note the server UUIDs for the first, second, and third servers.

   a. On the first server, enter the following command and receive a result similar to the following:

```
1> SELECT @@server_uuid;
+--------------------------------------+
| @@server_uuid                        |
+--------------------------------------+
| ba1e7829-7416-11e2-b285-0019b944b7f7 |
+--------------------------------------+
1 row in set (0.00 sec)
```

b. On the second server, enter the following command and receive a result similar to the following:

```
2> SELECT @@server_uuid;
+--------------------------------------+
| @@server_uuid                        |
+--------------------------------------+
| c701d6c5-7416-11e2-b285-0019b944b7f7 |
+--------------------------------------+
1 row in set (0.00 sec)
```

c. On the third server, enter the following command and receive a result similar to the following:

```
3> SELECT @@server_uuid;
+--------------------------------------+
| @@server_uuid                        |
+--------------------------------------+
| d3f1fc98-7416-11e2-b286-0019b944b7f7 |
+--------------------------------------+
1 row in set (0.00 sec)
```

Note: Your UUIDs differ from those shown; the server UUID is unique by design.

10. On the third server, view the slave status.

a. On the third server, enter the following command and receive a result similar to that shown:

```
3> SHOW SLAVE STATUS\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 127.0.0.1
                  Master_User: repl
                  Master_Port: 3312
                Connect_Retry: 60
              Master_Log_File: mysql2-bin.000001
          Read_Master_Log_Pos: 460
               Relay_Log_File: mysql3-relay-bin.000002
                Relay_Log_Pos: 672
        Relay_Master_Log_File: mysql2-bin.000001
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
```

```
                  Skip_Counter: 0
           Exec_Master_Log_Pos: 460
               Relay_Log_Space: 877
               Until_Condition: None
                Until_Log_File:
                 Until_Log_Pos: 0
            Master_SSL_Allowed: No
            Master_SSL_CA_File:
            Master_SSL_CA_Path:
               Master_SSL_Cert:
             Master_SSL_Cipher:
                Master_SSL_Key:
         Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
                 Last_IO_Errno: 0
                 Last_IO_Error:
                Last_SQL_Errno: 0
                Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
              Master_Server_Id: 2
                   Master_UUID: c701d6c5-7416-11e2-b285-0019b944b7f7
              Master_Info_File: mysql.slave_master_info
                     SQL_Delay: 0
           SQL_Remaining_Delay: NULL
       Slave_SQL_Running_State: Slave has read all relay log; waiting
for the slave I/O thread to update it
            Master_Retry_Count: 86400
                   Master_Bind:
       Last_IO_Error_Timestamp:
      Last_SQL_Error_Timestamp:
                Master_SSL_Crl:
            Master_SSL_Crlpath:
            Retrieved_Gtid_Set: ba1e7829-7416-11e2-b285-0019b944b7f7:1
             Executed_Gtid_Set: ba1e7829-7416-11e2-b285-0019b944b7f7:1
                 Auto_Position: 1
1 row in set (0.00 sec)
```

– Note that the master UUID identifies the second server, but the GTID for the data modification in step 8 identifies the first server; the transaction is globally identified.

11. Issue an appropriate `CHANGE MASTER TO`… statement on the first server, configuring it as a slave to the third server, and start the slave threads.

    a. On the first server, enter the following command and receive a result similar to that shown:

```
1> CHANGE MASTER TO MASTER_HOST='127.0.0.1', MASTER_PORT=3313,
    > MASTER_USER='repl', MASTER_PASSWORD='oracle',
    > MASTER_AUTO_POSITION=1;
Query OK, 0 rows affected, 2 warnings (0.59 sec)
1> START SLAVE;
Query OK, 0 rows affected (0.04 sec)
```

12. Delete all rows where the ID is greater than 4050 from the `City` table on the second server, and ensure that the changes replicate to the first and third servers.

    a. On the second server, enter the following command and receive the results shown:

```
2> DELETE FROM world_innodb.City WHERE ID > 4050;
Query OK, 10 rows affected (0.24 sec)
```

    b. On the first server, enter the following command and receive the results shown:

```
1> SELECT * FROM world_innodb.City ORDER BY Id DESC LIMIT 5;
+------+----------+-------------+---------------+------------+
| ID   | Name     | CountryCode | District      | Population |
+------+----------+-------------+---------------+------------+
| 4050 | Roanoke  | USA         | Virginia      |      93357 |
| 4049 | Brockton | USA         | Massachusetts |      93653 |
| 4048 | Albany   | USA         | New York      |      93994 |
| 4047 | Richmond | USA         | California    |      94100 |
| 4046 | Norman   | USA         | Oklahoma      |      94193 |
+------+----------+-------------+---------------+------------+
5 rows in set (0.00 sec)
```

    c. On the third server, enter the following command and receive the results shown:

```
3> SELECT * FROM world_innodb.City ORDER BY Id DESC LIMIT 5;
+------+----------+-------------+---------------+------------+
| ID   | Name     | CountryCode | District      | Population |
+------+----------+-------------+---------------+------------+
| 4050 | Roanoke  | USA         | Virginia      |      93357 |
| 4049 | Brockton | USA         | Massachusetts |      93653 |
| 4048 | Albany   | USA         | New York      |      93994 |
| 4047 | Richmond | USA         | California    |      94100 |
| 4046 | Norman   | USA         | Oklahoma      |      94193 |
+------+----------+-------------+---------------+------------+
5 rows in set (0.00 sec)
```

    – Circular replication is in effect, and a change on the second server replicated back to the first via the third.

Practices 11-5: Using MySQL Utilities and Performing a Failover

Overview

In this practice, you use the MySQL Utilities to provision a new slave, simulate a server failure, and perform an automated failover.

Tasks

1. On the first server, stop and reset the slave, removing any stored connection details.
2. Connect to the fourth server using the `mysql` client, and set the prompt to “`4> `”.
3. Apply the backup taken in the preceding practice (saved to `/tmp/server2.sql`) to the fourth server.
4. Issue the following command to compare the databases on servers one and four, and save the output to `/tmp/diff.sql`.

```
/usr/share/mysql-workbench/python/mysqldbcompare \
     --server1=root@127.0.0.1:3311 \
     --server2=root@127.0.0.1:3314 --changes-for=server2 \
     --difftype=sql -a world_innodb:world_innodb > /tmp/diff.sql
```

5. View the contents of `/tmp/diff.sql`.
6. Source the file `/tmp/diff.sql` on the fourth server to bring it up to date with the first.
7. On the fourth server, change the master settings to point to the first server and to use GTID, and start the slave.
8. In a terminal window, issue the following command to display the replication topology.

```
/usr/share/mysql-workbench/python/mysqlrplshow \
     --master=root@127.0.0.1:3311 \
     --discover-slaves-login=root --recurse
```

9. To prepare for automatic failover, create a `repl` user on the fourth server, with the password `oracle`, and grant that user the `REPLICATION SLAVE` permission.
10. Repeat the preceding step using the host `localhost` instead of the loopback address.
11. Launch `mysqlfailover` interactively in `auto` mode by using the following command.

```
/usr/share/mysql-workbench/python/mysqlfailover \
     --master=root@127.0.0.1:3311 \
     --discover-slaves-login=root --rpl-user=repl:oracle@127.0.0.1
```

12. View the different screens by pressing the `G, H` and `U` keys.

13. Shut down the first server from a separate terminal window.
14. Return to the terminal containing the `mysqlfailover` process, and observe the tool performing an automatic failover.
15. Issue a `mysqlrplshow` command to display the replication topology with the fourth server as master.
16. Stop all four servers used in this lesson, and start the normal `mysql` service.

Solutions 11-5: Using MySQL Utilities and Performing a Failover

Tasks

1. On the first server, stop and reset the slave, removing any stored connection details.

   a. Enter the following at the first `mysql` prompt, and receive the results shown:

```
1> STOP SLAVE;
Query OK, 0 rows affected (0.10 sec)

1> RESET SLAVE ALL;
Query OK, 0 rows affected (0.32 sec)
```

2. Connect to the fourth server using the `mysql` client, and set the prompt to “4> ”.

   a. Enter the following at a new terminal window, and receive the results shown:

```
$ mysql -uroot -h127.0.0.1 -P3314
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
...
mysql> PROMPT 4> ;
PROMPT set to '4> '
```

3. Apply the backup taken in the preceding practice (saved to `/tmp/server2.sql`) to the fourth server.

   a. In the `mysql` prompt created in the preceding step, enter the following command and receive the results shown:

```
4> SOURCE /tmp/server2.sql
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
...
```

Note: This backup includes a `CHANGE MASTER TO`… statement that specifies log coordinates rather than GTIDs.

4. Issue the following command to compare the databases on servers one and four, and save the output to `/tmp/diff.sql`.

   a. Enter the following at a terminal window:

```
$ /usr/share/mysql-workbench/python/mysqldbcompare \
     --server1=root@127.0.0.1:3311 \
     --server2=root@127.0.0.1:3314 --changes-for=server2 \
     --difftype=sql -a world_innodb:world_innodb > /tmp/diff.sql
```

5. View the contents of `/tmp/diff.sql`.

   a. Enter the following at a terminal window and receive the results shown:

```
$ cat /tmp/diff.sql
# server1 on 127.0.0.1: ... connected.
# server2 on 127.0.0.1: ... connected.
# Checking databases world_innodb on server1 and world_innodb on
server2
#
#                                                    Defn    Row
Data
# Type      Object Name                              Diff    Count
Check
# ---------------------------------------------------------------------
-----
# TABLE     City                                     FAIL    FAIL
FAIL
#
# Transformation for --changes-for=server2:
#

ALTER TABLE world_innodb.City
  DROP INDEX CountryCode,
  DROP PRIMARY KEY,
  ADD PRIMARY KEY(ID),
  ADD INDEX CountryCode (CountryCode),
AUTO_INCREMENT=4071;

# Row counts are not the same among world_innodb.City and
world_innodb.City.
#
# Transformation for --changes-for=server2:
#

# Data differences found among rows:
UPDATE world_innodb.City  WHERE ID = '694';
UPDATE world_innodb.City SET ID = '785', Name = 'Pasay', CountryCode =
'PHL', District = 'National Capital Reg', Population = '354908' WHERE
ID = '785';
DELETE FROM world_innodb.City WHERE ID = '4079';
...
DELETE FROM world_innodb.City WHERE ID = '4068';


# TABLE     Country                                  pass    pass
pass
# TABLE     CountryLanguage                          pass    pass
pass
```

```

# Database consistency check failed.
#
# ...done
```

6. Source the file `/tmp/diff.sql` on the fourth server to bring it up to date with the first.
   a. In the `mysql` prompt connected to the fourth server, enter the following command and receive the results shown:

```
4> SOURCE /tmp/diff.sql
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

ERROR 1064 (42000): You have an error in your SQL syntax; check the
manual that corresponds to your MySQL server version for the right
syntax to use near 'WHERE ID = '694'' at line 1
Query OK, 0 rows affected (0.06 sec)
Rows matched: 1  Changed: 0  Warnings: 0

Query OK, 1 row affected (0.05 sec)

...
Query OK, 1 row affected (0.05 sec)
```

   - The error in the preceding output is due to an incorrect syntax in the `mysqldbcompare` output.

7. On the fourth server, change the master settings to point to the first server and to use GTID, and start the slave.
   a. In the `mysql` prompt connected to the fourth server, enter the following command and receive the results shown:

```
4> CHANGE MASTER TO MASTER_PORT=3311, MASTER_AUTO_POSITION=1;
Query OK, 0 rows affected (0.34 sec)
4> START SLAVE;
Query OK, 0 rows affected (0.05 sec)
```

8. In a terminal window, issue the following command to display the replication topology.
   a. Enter the following at a terminal window and receive the results shown:

```
$ /usr/share/mysql-workbench/python/mysqlrplshow \
     --master=root@127.0.0.1:3311 \
     --discover-slaves-login=root --recurse
# master on 127.0.0.1: ... connected.
# Finding slaves for master: 127.0.0.1:3311
# master on localhost: ... connected.
# Finding slaves for master: localhost:3312
# master on localhost: ... connected.
# Finding slaves for master: localhost:3313
# master on localhost: ... connected.
# Finding slaves for master: localhost:3314

# Replication Topology Graph
127.0.0.1:3311 (MASTER)
   |
   +--- localhost:3312 - (SLAVE + MASTER)
   |   |
   |   +--- localhost:3313 - (SLAVE)
   |
   +--- localhost:3314 - (SLAVE)
```

   – The output shows a diagram representing the relationship of slaves to masters.

9. To prepare for automatic failover, create a `repl` user on the fourth server, with the password `oracle`, and grant that user the `REPLICATION SLAVE` permission.
   a. In the `mysql` prompt used in the preceding step, enter the following command and receive the results shown:

```
4> CREATE USER 'repl'@'127.0.0.1' IDENTIFIED BY 'oracle';
Query OK, 0 rows affected (0.00 sec)

4> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'127.0.0.1';
Query OK, 0 rows affected (0.00 sec)
```

   – At this point, the `repl` user exists on all four servers.

10. Repeat the preceding step using the host `localhost` instead of the loopback address.
   a. In the `mysql` prompt used in the preceding step, enter the following command and receive the results shown:

```
4> CREATE USER 'repl'@'localhost' IDENTIFIED BY 'oracle';
Query OK, 0 rows affected (0.00 sec)

4> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

11. Launch `mysqlfailover` interactively in `auto` mode by using the following command.
    a. Enter the following at a terminal window and receive the results shown:

```
$ /usr/share/mysql-workbench/python/mysqlfailover \
     --master=root@127.0.0.1:3311 \
     --discover-slaves-login=root --rpl-user=repl:oracle
# Discovering slaves for master at 127.0.0.1:3311
# Checking privileges.
# WARNING: You may be mixing host names and IP addresses. This may
result in negative status reporting if your DNS services do not
support reverse name lookup.
#
# Failover console will start in 10 seconds.
```

    b. When the console appears, it looks somewhat similar to the following:

```
MySQL Replication Failover Utility
Failover Mode = auto     Next Interval = Mon Feb 11 07:55:33 2013

Master Information
------------------
Binary Log File    Position  Binlog_Do_DB  Binlog_Ignore_DB
mysql1-bin.000002  679

GTID Executed Set
ba1e7829-7416-11e2-b285-0019b944b7f7:1-4 [...]

Replication Health Status
+------------+-------+---------+--------+------------+---------+
| host       | port  | role    | state  | gtid_mode  | health  |
+------------+-------+---------+--------+------------+---------+
| 127.0.0.1  | 3311  | MASTER  | UP     | ON         | OK      |
| localhost  | 3312  | SLAVE   | UP     | ON         | OK      |
| localhost  | 3314  | SLAVE   | UP     | ON         | OK      |
+------------+-------+---------+--------+------------+---------+
Q-quit R-refresh H-health G-GTID Lists U-UUIDs
```

    – Note that the display does not show circularity, nor does it recurse. It shows only the slaves of the master provided on the command line.

12. View the different screens by pressing the `G, H` and `U` keys.
    a. Press the `G` key to view the "Master GTID Executed Set," and receive the results shown:

```
MySQL Replication Failover Utility
Failover Mode = auto     Next Interval = Mon Feb 11 07:57:03 2013

Master Information
------------------
Binary Log File    Position  Binlog_Do_DB  Binlog_Ignore_DB
```

```
mysql1-bin.000002  679

GTID Executed Set
ba1e7829-7416-11e2-b285-0019b944b7f7:1-4 [...]

Master GTID Executed Set
+-------------------------------------------+
| gtid                                      |
+-------------------------------------------+
| ba1e7829-7416-11e2-b285-0019b944b7f7:1-4  |
| c701d6c5-7416-11e2-b285-0019b944b7f7:1    |
+-------------------------------------------+
Q-quit R-refresh H-health G-GTID Lists U-UUIDs
```

b. Press the G key again to view the “Transactions executed on the servers,” and receive the results shown:

```
MySQL Replication Failover Utility
Failover Mode = auto     Next Interval = Mon Feb 11 07:57:21 2013

Master Information
------------------
Binary Log File    Position  Binlog_Do_DB  Binlog_Ignore_DB
mysql1-bin.000002  679

GTID Executed Set
ba1e7829-7416-11e2-b285-0019b944b7f7:1-4 [...]

Transactions executed on the servers:
+------------+-------+---------+-------------------------------------------+
| host       | port  | role    | gtid                                      |
+------------+-------+---------+-------------------------------------------+
| 127.0.0.1  | 3311  | MASTER  | ba1e7829-7416-11e2-b285-0019b944b7f7:1-4   |
| 127.0.0.1  | 3311  | MASTER  | c701d6c5-7416-11e2-b285-0019b944b7f7:1     |
| localhost  | 3312  | SLAVE   | ba1e7829-7416-11e2-b285-0019b944b7f7:1-4   |
| localhost  | 3312  | SLAVE   | c701d6c5-7416-11e2-b285-0019b944b7f7:1     |
| localhost  | 3314  | SLAVE   | 85bae58f-741f-11e2-b2be-0019b944b7f7:1-49  |
| localhost  | 3314  | SLAVE   | ba1e7829-7416-11e2-b285-0019b944b7f7:1-4   |
| localhost  | 3314  | SLAVE   | c701d6c5-7416-11e2-b285-0019b944b7f7:1     |
+------------+-------+---------+-------------------------------------------+
Q-quit R-refresh H-health G-GTID Lists U-UUIDs Up|Down-scroll
```

c. Press the G key again to view the “Transactions purged from the servers,” and receive the results shown:

```
MySQL Replication Failover Utility
Failover Mode = auto     Next Interval = Mon Feb 11 07:57:57 2013

Master Information
------------------
Binary Log File    Position  Binlog_Do_DB  Binlog_Ignore_DB
mysql1-bin.000002  679
```

```
GTID Executed Set
ba1e7829-7416-11e2-b285-0019b944b7f7:1-4 [...]

Transactions purged from the servers:
0 Rows Found.
Q-quit R-refresh H-health G-GTID Lists U-UUIDs
```

d. Press the G key again to view the "Transactions owned by another server," and receive the results shown:

```
MySQL Replication Failover Utility
Failover Mode = auto     Next Interval = Mon Feb 11 07:58:33 2013

Master Information
------------------
Binary Log File    Position  Binlog_Do_DB  Binlog_Ignore_DB
mysql1-bin.000002  679

GTID Executed Set
ba1e7829-7416-11e2-b285-0019b944b7f7:1-4 [...]

Transactions owned by another server:
0 Rows Found.
Q-quit R-refresh H-health G-GTID Lists U-UUIDs
```

e. Press the U key to view the "UUIDs," and receive the results shown:

```
MySQL Replication Failover Utility
Failover Mode = auto     Next Interval = Mon Feb 11 07:58:51 2013

Master Information
------------------
Binary Log File    Position  Binlog_Do_DB  Binlog_Ignore_DB
mysql1-bin.000002  679

GTID Executed Set
ba1e7829-7416-11e2-b285-0019b944b7f7:1-4 [...]

UUIDs
+------------+-------+---------+---------------------------------------+
| host       | port  | role    | uuid                                  |
+------------+-------+---------+---------------------------------------+
| 127.0.0.1  | 3311  | MASTER  | ba1e7829-7416-11e2-b285-0019b944b7f7  |
| localhost  | 3312  | SLAVE   | c701d6c5-7416-11e2-b285-0019b944b7f7  |
| localhost  | 3314  | SLAVE   | 85bae58f-741f-11e2-b2be-0019b944b7f7  |
+------------+-------+---------+---------------------------------------+
Q-quit R-refresh H-health G-GTID Lists U-UUIDs
```

f. Press the H key to return to the `Health` screen.

13. Shut down the second server from a separate terminal window.

    a. Enter the following at a terminal window.

```
$ mysqladmin -uroot -h127.0.0.1 -P3311 shutdown
```

14. Return to the terminal containing the `mysqlfailover` process, and observe the tool performing an automatic failover.

    The output should resemble the following:

```
Failover starting in 'auto' mode...
# Candidate slave localhost:3314 will become the new master.
# Preparing candidate for failover.
# Creating replication user if it does not exist.
# Stopping slaves.
# Performing STOP on all slaves.
# Switching slaves to new master.
# Starting slaves.
# Performing START on all slaves.
# Checking slaves for errors.
# Failover complete.
# Discovering slaves for master at localhost:3314

Failover console will restart in 5 seconds.
```

15. Issue a `mysqlrplshow` command to display the replication topology with the fourth server as master.

    a. Enter the following at a terminal window, and receive the results shown.

```
$ /usr/share/mysql-workbench/python/mysqlrplshow \
     --master=root@127.0.0.1:3314 \
     --discover-slaves-login=root --recurse
# master on 127.0.0.1: ... connected.
# Finding slaves for master: 127.0.0.1:3314
# master on localhost: ... connected.
# Finding slaves for master: localhost:3312
# master on localhost: ... connected.
# Finding slaves for master: localhost:3313

# Replication Topology Graph
127.0.0.1:3314 (MASTER)
   |
   +--- localhost:3312 - (SLAVE + MASTER)
       |
       +--- localhost:3313 - (SLAVE)
```

16. Stop all four servers used in this lesson, and start the normal `mysql` service.

    a. Enter the following at a terminal window, and receive the results shown.

```
# mysqld_multi --defaults-file=/labs/repl.cnf --user=root stop 1-4
# service mysql start
Starting MySQL..                                          [  OK  ]
```

Practices for Lesson 12: Introduction to Performance Tuning

Practice 12-1: Quiz – Introduction to Performance Tuning

Overview

In this practice, you answer questions about performance tuning.

Quiz Questions

Choose the best answer from those provided for each multiple choice or True/False question.

1. Which of the following is *not* a benefit of normalization?
   a. Makes it easier to read queries
   b. Eliminates redundant data
   c. Minimizes data becoming inconsistent
   d. Provides flexible access to data
2. Choosing a data type that is larger than the space required can become a large problem when the data grows.
   a. True
   b. False
3. Querying a table with no indexes or insufficient indexes results in ______________.
   a. Corrupt data
   b. Full table scans
   c. Improved performance
   d. All of the above
4. The ________________ command describes how MySQL intends to execute a specified SQL statement, but it does not return any data from the data sets.
   a. `ANALYZE`
   b. `DESCRIBE`
   c. `EXPLAIN`
   d. `SHOW`
5. `PROCEDURE ANALYZE` analyzes the stored procedures in your application.
   a. True
   b. False

6. Which status variable displays the number of currently open connections?
   a. `Key_reads`
   b. `Max_used_connections`
   c. `Open_tables`
   d. `Threads_connected`
7. Which server system variable defines how often the InnoDB log buffer is written out to the log file and how often the flush-to-disk operation is performed on the log file?
   a. `innodb_buffer_pool_size`
   b. `innodb_flush_log_at_trx_commit`
   c. `innodb_log_buffer_size`
   d. `innodb_log_file_size`

Solutions 12-1: Quiz – Introduction to Performance Tuning

Quiz Solutions

1. **a.** Makes it easier to read queries
2. **a.** True
3. **b.** Full table scans
4. **c.** `EXPLAIN`
5. **b.** False. `PROCEDURE ANALYZE` analyzes the columns in a given query and provides tuning feedback on each field.
6. **d.** `Threads_connected`
7. **b.** `innodb_flush_log_at_trx_commit`

Practice 12-2: EXPLAIN

Overview

In this practice, you view the execution plan of several SELECT statements by using the EXPLAIN command. To accomplish this objective, do the following:

- Use the world_innodb database to view the execution plans of multiple SELECT statements.
- Alter an existing table to improve the execution of a query.

Assumptions

- The MySQL server is installed and running.
- The world_innodb database is installed.

Tasks

1. Open a terminal window, start the mysql client with the admin login path, and use the world_innodb database.
2. Start over with a clean copy of the world_innodb database by re-creating it using the world_innodb.sql file.
3. Execute the following command to view the execution plan for a query against the City table in the world_innodb database:

```
EXPLAIN SELECT Name FROM City WHERE Name LIKE 'A%'\G
```

   - What is the select type for this query? ________________
   - Are there any possible indexes that this query can use? ______________
   - How many rows must be examined to complete this query? ___________

4. Execute the following command to view the execution plan for another query against the City table in the world_innodb database:

```
EXPLAIN SELECT Name FROM City WHERE ID > 4070\G
```

   - What is the select type for this query? _______________
   - Are there any possible indexes that this query can use? ______________
   - How many rows must be examined to complete this query? ____________

5. Comparing the outputs from the first and second EXPLAIN statements, determine which is a better performing execution plan.

   ____

   ____

6. Improve the performance of the first query by making modifications to the `City` table and then rerun the `EXPLAIN` statement.
   - What is the select type for this query? ________________
   - Are there any possible indexes that this query can use?
     ____
   - How many rows must be examined to complete this query? _____________

Solutions 12-2: EXPLAIN

Tasks

1. Open a terminal window, start the `mysql` client with the `admin` login path, and use the `world_innodb` database.

   Enter the following at a terminal window and receive the results shown:

```
$ mysql --login-path=admin
Welcome to the MySQL monitor.  Commands end with ; or \g.
...
mysql> USE world_innodb
...
Database changed
```

2. Start over with a clean copy of the `world_innodb` database by re-creating it using the `world_innodb.sql` file.

   Enter the following at a `mysql` prompt and receive the results shown:

```
mysql> DROP DATABASE world_innodb;
Query OK, 4 rows affected (0.00 sec)

mysql> CREATE DATABASE world_innodb;
Query OK, 1 row affected (0.00 sec)

mysql> USE world_innodb;
Database changed
mysql> SET AUTOCOMMIT=0;
Query OK, 0 rows affected (0.00 sec)

mysql> SOURCE /labs/world_innodb.sql
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
...
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> SET AUTOCOMMIT=1;
Query OK, 0 rows affected (0.12 sec)
```

3. Execute the following command to view the execution plan for a query against the `City` table in the `world_innodb` database.

   Enter the following at a `mysql` prompt and receive the results shown:

```
mysql> EXPLAIN SELECT Name FROM City WHERE Name LIKE 'A%'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: City
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 4188
        Extra: Using where
1 row in set (0.00 sec)
```

   – What is the select type for this query? ALL
   – Are there any possible indexes that this query can use? No
   – How many rows must be examined to complete this query? Approximately 3982. Your output may differ. There are 4079 rows in the table, so `EXPLAIN` performs an estimate.

4. Execute the following command to view the execution plan for another query against the `City` table in the `world_innodb` database.

   Enter the following at a `mysql` prompt and receive the results shown:

```
mysql> EXPLAIN SELECT Name FROM City WHERE ID > 4070\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: City
         type: range
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: NULL
         rows: 9
        Extra: Using where
1 row in set (0.00 sec)
```

   – What is the select type for this query? `SIMPLE`
   – Are there any possible indexes that this query can use? Yes, the primary key for this table can be used.
   – How many rows must be examined to complete this query? Nine. This is the actual number of rows returned by the query.

5. Comparing the outputs from the first and second `EXPLAIN` statements, determine which is a better performing execution plan.
   - The second query takes advantage of an index and does not need to perform a full table scan.
6. Improve the performance of the first query by making modifications to the `City` table and then rerun the `EXPLAIN` statement.

   Enter the following statements at a `mysql` prompt and receive the results shown:

```
mysql> CREATE INDEX iName ON City (Name) USING BTREE;
Query OK, 0 rows affected (0.12 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN SELECT Name FROM City WHERE Name LIKE 'A%'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: City
         type: range
possible_keys: iName
          key: iName
      key_len: 35
          ref: NULL
         rows: 258
        Extra: Using where; Using index
1 row in set (0.00 sec)
```

   - What is the select type for this query? `SIMPLE`
   - Are there any possible indexes that this query can use?
     Yes. The execution plan uses the index created on the Name column.
   - How many rows must be examined to complete this query? 258

Practice 12-3: PROCEDURE ANALYSE

Overview

In this practice, you execute several SELECT statements with the PROCEDURE ANALYSE option. To accomplish this objective, do the following:

- Use the world_innodb database and execute multiple SELECT statements.
- Evaluate existing tables to improve the table design.

Assumptions

- The MySQL server is installed and running.
- The world_innodb database is installed.

Tasks

1. Using the terminal from the previous practice (which is already running the mysql client and using the world_innodb database), execute the following command:

   ```
   SELECT Name, CountryCode, Population FROM City PROCEDURE ANALYSE()\G
   ```

   – What does the PROCEDURE ANALYSE option state is the optimal field type for the Name column in the City table based on the existing data in the table?

   ____

   – What does the PROCEDURE ANALYSE option state is the optimal field type for the CountryCode column in the City table based on the existing data in the table?

   ____

   – What does the PROCEDURE ANALYSE option state is the optimal field type for the Population column in the City table based on the existing data in the table?

   ____

2. Execute the following command to view the City table design:

   ```
   DESC City;
   ```

   Comparing the design of the City table and the recommendations from the PROCEDURE ANALYSE option run in step 1, evaluate each recommendation.

   ____

   ____

   ____

   ____

3. Execute the following command to change how `PROCEDURE ANALYSE()` recommends `ENUM` types, and compare the recommendations with those in the preceding steps:

```
SELECT Name, CountryCode, Population FROM City
PROCEDURE ANALYSE(256,1024)\G
```

4. Execute the following command to evaluate the `CountryLanguage` table design:

```
SELECT * FROM CountryLanguage PROCEDURE ANALYSE(256,1024)\G
```

5. Execute the following command to view the `CountryLanguage` table design:

```
DESC CountryLanguage;
```

   Comparing the design of the `CountryLanguage` table and the recommendations from the `PROCEDURE ANALYSE` option run in step 3, which recommendation will you implement?

   ____

   ____

   ____

   ____

6. Exit the `mysql` client session.

Solutions 12-3: PROCEDURE ANALYSE

Tasks

1. Using the terminal from the previous practice (which is already running the `mysql` client and using the `world_innodb` database), execute the following command.

   Enter the following statement at a `mysql` prompt and receive the results shown:

```
mysql> SELECT Name, CountryCode, Population FROM City
    -> PROCEDURE ANALYSE()\G
*************************** 1. row ***************************
             Field_name: world_innodb.City.Name
              Min_value: A Coruña (La Coruña)
              Max_value: ´s-Hertogenbosch
             Min_length: 3
             Max_length: 33
       Empties_or_zeros: 0
                  Nulls: 0
Avg_value_or_avg_length: 8.5295
                    Std: NULL
      Optimal_fieldtype: VARCHAR(33) NOT NULL
*************************** 2. row ***************************
             Field_name: world_innodb.City.CountryCode
              Min_value: ABW
              Max_value: ZWE
             Min_length: 3
             Max_length: 3
       Empties_or_zeros: 0
                  Nulls: 0
Avg_value_or_avg_length: 3.0000
                    Std: NULL
      Optimal_fieldtype: ENUM('ABW','AFG',...,'ZMB','ZWE') NOT NULL
*************************** 3. row ***************************
             Field_name: world_innodb.City.Population
              Min_value: 42
              Max_value: 10500000
             Min_length: 2
             Max_length: 8
       Empties_or_zeros: 0
                  Nulls: 0
Avg_value_or_avg_length: 350468.2236
                    Std: 723686.9870
      Optimal_fieldtype: MEDIUMINT(8) UNSIGNED NOT NULL
3 rows in set (0.00 sec)
```

   - What does the `PROCEDURE ANALYSE` option state is the optimal field type for the `Name` column in the `City` table based on the existing data in the table?
     `VARCHAR(33) NOT NULL`
   - What does the `PROCEDURE ANALYSE` option state is the optimal field type for the `CountryCode` column in the `City` table based on the existing data in the table?
     `ENUM('ABW','AFG',...,'ZMB','ZWE') NOT NULL`
   - What does the `PROCEDURE ANALYSE` option state is the optimal field type for the `Population` column in the `City` table based on the existing data in the table?
     `MEDIUMINT(8) UNSIGNED NOT NULL`

2. Execute the following command to view the `City` table design. Comparing the design of the `City` table and the recommendations from the `PROCEDURE ANALYSE` option run in step 1, evaluate each recommendation.

   Enter the following statement at a `mysql` prompt and receive the results shown:

```
mysql> DESC City;
+-------------+------------+------+-----+---------+------------+
| Field       | Type       | Null | Key | Default | Extra      |
+-------------+------------+------+-----+---------+------------+
| ID          | int(11)    | NO   | PRI | NULL    | auto_i ... |
| Name        | char(35)   | NO   | MUL |         |            |
| CountryCode | char(3)    | NO   | MUL |         |            |
| District    | char(20)   | NO   |     |         |            |
| Population  | int(11)    | NO   |     | 0       |            |
+-------------+------------+------+-----+---------+------------+
5 rows in set (0.00 sec)
```

   – Comparing the design of the `City` table and the recommendations from the `PROCEDURE ANALYSE` option run in step 1, which recommendation will you implement?

     – Altering the `Population` column to a `MEDIUMINT UNSIGNED` field is a recommendation that suits the current data. The largest number that the `MEDIUMINT UNSIGNED` field can handle is 16,777,215, and it takes up only 3 bytes. For comparison, the largest number that the `BIGINT UNSIGNED` field can handle is 18,446,744,073,709,555,615 and it takes up 8 bytes. However, the best field type for this column to allow for future values is `INT UNSIGNED` because it can handle up to 4,294,967,295 and takes up 4 bytes. With cities such as Tokyo with 35 million people (as of 2013), the `MEDIUMINT UNSIGNED` field does not cater to the population of all cities now and into the future.

     – The recommendation to change the `Name` column does not save much space, but monitor the type as the table grows.

     – You do not need to implement the recommendation to change the `CountryCode` column to an `ENUM` field type; `ENUM` fields with a large number of values add complexity to applications, and are difficult to maintain.

3. Execute the following command to change how `PROCEDURE ANALYSE()` recommends `ENUM` types, and compare the recommendations with those in the preceding steps.

   Enter the following statement at a `mysql` prompt and receive the results shown:

```
mysql> SELECT Name, CountryCode, Population FROM City
     > PROCEDURE ANALYSE(256,1024)\G
*************************** 1. row ***************************
             Field_name: world_innodb.City.Name
              Min_value: A Coruña (La Coruña)
              Max_value: ´s-Hertogenbosch
             Min_length: 3
             Max_length: 33
       Empties_or_zeros: 0
                  Nulls: 0
Avg_value_or_avg_length: 8.5295
                    Std: NULL
      Optimal_fieldtype: VARCHAR(33) NOT NULL
*************************** 2. row ***************************
             Field_name: world_innodb.City.CountryCode
              Min_value: ABW
              Max_value: ZWE
             Min_length: 3
             Max_length: 3
       Empties_or_zeros: 0
                  Nulls: 0
Avg_value_or_avg_length: 3.0000
                    Std: NULL
      Optimal_fieldtype: CHAR(3) NOT NULL
*************************** 3. row ***************************
             Field_name: world_innodb.City.Population
              Min_value: 42
              Max_value: 10500000
             Min_length: 2
             Max_length: 8
       Empties_or_zeros: 0
                  Nulls: 0
Avg_value_or_avg_length: 350468.2236
                    Std: 723686.9870
      Optimal_fieldtype: MEDIUMINT(8) UNSIGNED NOT NULL
3 rows in set (0.01 sec)
```

   – The recommendation for `CountryCode` has changed to `CHAR(3)`, the most appropriate type.

4. Execute the following command to evaluate the `CountryLanguage` table design:

```
mysql> SELECT * FROM CountryLanguage PROCEDURE ANALYSE(256,1024)\G
*************************** 1. row ***************************
             Field_name: world_innodb.CountryLanguage.CountryCode
              Min_value: ABW
              Max_value: ZWE
             Min_length: 3
             Max_length: 3
       Empties_or_zeros: 0
                  Nulls: 0
Avg_value_or_avg_length: 3.0000
                    Std: NULL
      Optimal_fieldtype: CHAR(3) NOT NULL
*************************** 2. row ***************************
             Field_name: world_innodb.CountryLanguage.Language
              Min_value: Abhyasi
              Max_value: [South]Mande
             Min_length: 2
             Max_length: 25
       Empties_or_zeros: 0
                  Nulls: 0
Avg_value_or_avg_length: 7.1606
                    Std: NULL
      Optimal_fieldtype: VARCHAR(25) NOT NULL
*************************** 3. row ***************************
             Field_name: world_innodb.CountryLanguage.IsOfficial
              Min_value: F
              Max_value: T
             Min_length: 1
             Max_length: 1
       Empties_or_zeros: 0
                  Nulls: 0
Avg_value_or_avg_length: 1.0000
                    Std: NULL
      Optimal_fieldtype: ENUM('F','T') NOT NULL
*************************** 4. row ***************************
             Field_name: world_innodb.CountryLanguage.Percentage
              Min_value: 0.1
              Max_value: 100.0
             Min_length: 3
             Max_length: 5
       Empties_or_zeros: 65
                  Nulls: 0
Avg_value_or_avg_length: 20.4
                    Std: 30.8
      Optimal_fieldtype: FLOAT(4,1) NOT NULL
4 rows in set (0.00 sec)
```

5. Execute the following command to view the `CountryLanguage` table design:

```
mysql> DESC CountryLanguage;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| CountryCode | char(3)      | NO   | PRI |         |       |
| Language    | char(30)     | NO   | PRI |         |       |
| IsOfficial  | enum('T','F') | NO  |     | F       |       |
| Percentage  | float(4,1)   | NO   |     | 0.0     |       |
+-------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

   Comparing the design of the `CountryLanguage` table and the recommendations from the `PROCEDURE ANALYSE` option run in step 3, which recommendation will you implement?

   – <u>The only changed recommendation is to set the language name to `VARCHAR(25)` from `CHAR(30)`. Because `VARCHAR` is more space-efficient, this is a good recommendation, although you must ensure that any future values for the column fit within 25 characters.</u>

6. Exit the `mysql` client session:

```
mysql> EXIT;
Bye
```

Practices for Lesson 13: Introduction to MySQL Cluster

Chapter13

Practice 13-1: Quiz–Introduction to MySQL Cluster

Overview

In this practice, you answer questions that test your introductory knowledge of MySQL Cluster.

Quiz Questions

Choose the best answer from those provided for each multiple choice or True/False question.

1. True or false: Sharding is the division of queries between management nodes.
2. Which of the following best describes MySQL Cluster's architecture?
    a) Shared disk
    b) Shared memory
    c) Shared nothing
    d) Shared resource
3. True or false: Using a Fibre Channel SAN improves communication between MySQL Cluster nodes.
4. True or false: Applications connecting to MySQL Cluster can connect through MySQL Server, but they can also bypass MySQL Server by using a direct NoSQL API.
5. Which of the following describes a requirement for a system that is not a good application for MySQL Cluster?
    a) Automatic failover and recovery
    b) High-volume OLTP
    c) Large databases over 3 TB
    d) Online schema alteration and scalability
    e) Real-time performance
6. True or false: MySQL Cluster supports both transactions and foreign keys.
7. True or false: You can start a cluster with only data nodes.
8. Which of the following is not a valid way to connect an application to MySQL Cluster?
    a) ClusterJPA
    b) Connector/J
    c) iSCSI
    d) Memcached API
    e) NDB API
9. True or false: At a minimum, you need two data nodes in a cluster.
10. True or false: Each node group must have the same number of data nodes.
11. Which of the following node types participate in node groups?
    a) API nodes
    b) Data nodes
    c) Management nodes
    d) SQL nodes

12. Which type of partitioning does MySQL Cluster automatic sharding use?
    a) Horizontal
    b) Mirrored
    c) Striped
    d) Vertical

Solutions 13-1: Quiz–Introduction to MySQL Cluster

1. **False**. Sharding is the partitioning of data between data nodes.
2. **c**. Shared nothing.
3. **False**. Fibre Channel SAN is used for shared storage between cluster hosts in a shared-disk architecture.
4. **True**
5. **c**. Large databases over 3 TB. MySQL Cluster is better suited to data sets that can fit in memory across a commodity cluster.
6. **True**
7. **False**. You need at least one management node to start other nodes.
8. **c**. iSCSI
9. **False**. A cluster can run with only one data node, but this architecture has no redundancy or performance benefits.
10. **True**
11. **b**. Data nodes.
12. **a**. Horizontal.

Practice 13-2: Installing MySQL Cluster from a Binary Package

Overview

In this practice, you install the MySQL Cluster binaries from a Linux binary RPM package.

Duration

This practice should take approximately 5 minutes.

Tasks

1. Install the MySQL Cluster server and client RPM packages on **host1**.
2. Reset the MySQL Server `root` password to `oracle` by starting the MySQL service and running the `/labs/resetroot.sh` script.

Solutions 13-2: Installing MySQL Cluster from a Binary Package

Tasks

1. Install the MySQL Cluster server and client RPM packages on **host1**.

   Run the following commands at the **host1** prompt and receive the results shown:

```
# rpm -Uhv --replacefiles \
    /labs/MySQL-Cluster-server-advanced.rpm
Preparing...                ################################# [100%]
   1:MySQL-Cluster-server-ad################################# [100%]
-date and time- 0 [Warning] TIMESTAMP with implicit DEFAULT value is
deprecated. Please use --explicit_defaults_for_timestamp server option
(see documentation for more details).
...
A RANDOM PASSWORD HAS BEEN SET FOR THE MySQL root USER !
You will find that password in '/root/.mysql_secret'.

You must change that password on your first connect,
no other statement but 'SET PASSWORD' will be accepted.
See the manual for the semantics of the 'password expired' flag.
...
WARNING: Default config file /etc/my.cnf exists on the system
This file will be read by default by the MySQL server
If you do not want to use this, either remove it, or use the
--defaults-file argument to mysqld_safe when starting the server

# rpm -Uhv /labs/MySQL-Cluster-client-advanced.rpm
Preparing...                ################################# [100%]
   1:MySQL-Cluster-client-ad################################# [100%]
```

   The MySQL Cluster binaries are now installed in the correct locations but not configured.

   Note that the MySQL Server installation creates a random `root` password.

2. Reset the MySQL Server `root` password to `oracle` by starting the MySQL service and running the `/labs/resetroot.sh` script.

   Run the following commands at the **host1** prompt and receive the results shown:

```
# service mysql start
Starting MySQL.............................. SUCCESS!
# /labs/resetroot.sh
Warning: Using a password on the command line interface can be
insecure.
MySQL root password reset to "oracle".
```

Practice 13-3: Starting a Single-Machine Cluster

Overview

In this practice, you configure a single-machine cluster on **host1**.

Tasks

1. Create a new file named `/etc/config.ini` with the following contents:

```
[ndb_mgmd]
Hostname=host1

[ndbd default]
NoOfReplicas=1

[ndbd]
Hostname=host1

[mysqld]
```

2. Perform an initial startup of the management daemon and multi-threaded data node daemon, pointing to the configuration file created in the preceding step.
3. Launch the management console and issue a `SHOW` command to view the cluster status.
4. Exit the management console.

Solutions 13-3: Starting a Single-Machine Cluster

Tasks

1. Create a new file named `/etc/config.ini` with the following contents.

   At the command line, use an editor such as `nano` or `vim` to edit the file with the following command:

   ```
   # nano /etc/config.ini
   ```

   Create the file contents as follows:

   ```
   [ndb_mgmd]
   Hostname=host1

   [ndbd default]
   NoOfReplicas=1

   [ndbd]
   Hostname=host1

   [mysqld]
   ```

   The configuration file specifies a cluster where each node group has a single node (`NoOfReplicas=1`) and the management node and data node are both on **host1**.

2. Perform an initial startup of the management daemon and multi-threaded data node daemon, pointing to the configuration file created in the preceding step.

   In the terminal window, issue the following commands and receive the results shown:

   ```
   # ndb_mgmd -f /etc/config.ini --initial
   MySQL Cluster Management Server mysql-5.6.17 ndb-7.3.5
   -date and time- [MgmtSrvr] INFO     -- The default config directory
   '/usr/mysql-cluster' does not exist. Trying to create it...
   -date and time- [MgmtSrvr] INFO     -- Sucessfully created config
   directory
   # ndbmtd -c host1 --initial
   -date and time- [ndbd] INFO     -- Angel connected to 'host1:1186'
   -date and time- [ndbd] INFO     -- Angel allocated nodeid: 2
   ```

3. Launch the management console and issue a `SHOW` command to view the cluster status.

   In the same terminal window, issue the following commands and receive the results shown:

```
# ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> SHOW
Connected to Management Server at: localhost:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     1 node(s)
id=2    @192.168.100.201  (mysql-5.6.17 ndb-7.3.5, Nodegroup: 0, *)

[ndb_mgmd(MGM)] 1 node(s)
id=1    @192.168.100.201  (mysql-5.6.17 ndb-7.3.5)

[mysqld(API)]   1 node(s)
id=3 (not connected, accepting connect from any host)
```

   The cluster is now running and ready to accept connections from an API node such as MySQL Server. Note that the management node has been allocated node ID 1, and the data node has been allocated node ID 2. Because data nodes can only occupy node IDs in the range 1-48, it is good practice to configure other node types (including management and MySQL Server nodes) on node IDs greater than 48. You do this in the next practice.

4. Exit the management console.

   In the same terminal window, issue the following command:

```
ndb_mgm> EXIT
```

Practice 13-4: Adding a Node to a Cluster

Overview

In this practice, you add a new node to the cluster, thereby restarting the cluster.

Tasks

1. Edit the `/etc/config.ini` file with the following contents:

```
[ndb_mgmd]
Hostname=host1
NodeId=49

[ndbd default]
NoOfReplicas=2

[ndbd]
Hostname=host1
NodeId=1

[ndbd]
Hostname=host2
NodeId=2

[mysqld]
NodeId=50
```

2. Launch the management console and shut down the cluster.
3. Delete the contents of the configuration cache directory `/usr/mysql-cluster`.
4. Restart the management and data node daemons on **host1**, reloading the configuration.
5. Use `scp` to copy the multi-threaded data node binary from `/usr/sbin/ndbmtd` to the same location on **host2**.
6. Connect to **host2** as `root` and launch a data node connected to the management daemon on **host1**.
7. Use the management console to view the cluster status.
8. Exit the management console.

Solutions 13-4: Adding a Node to a Cluster

Tasks

1. Edit the `/etc/config.ini` file with the following contents.

```
[ndb_mgmd]
Hostname=host1
NodeId=49

[ndbd default]
NoOfReplicas=2

[ndbd]
Hostname=host1
NodeId=1

[ndbd]
Hostname=host2
NodeId=2

[mysqld]
NodeId=50
```

   The configuration file specifies a cluster where each node group has two data nodes (`NoOfReplicas=2`) and the data nodes are on **host1** and **host2**. This enables high availability between the nodes. Each node also has an explicit `NodeId` that follows best practice: Data nodes have node IDs in the range 1–48, and other node types have node IDs greater than 48.

2. Launch the management console and shut down the cluster.

   At a **host1** terminal, issue the following commands and receive the results shown:

```
# ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> SHUTDOWN
Connected to Management Server at: localhost:1186
Node 2: Cluster shutdown initiated
Node 2: Node shutdown completed.
2 NDB Cluster node(s) have shutdown.
Disconnecting to allow management server to shutdown.
ndb_mgm> EXIT
```

   Note that if the cluster had been started with `NoOfReplicas=2`, you could add a node without shutting down the whole cluster. Your cluster started with only a single data node, so such an online operation is not possible.

3. Delete the contents of the configuration cache directory `/usr/mysql-cluster`.

   At a **host1** terminal, issue the following command and receive the results shown:

```
# rm /usr/mysql-cluster/*
rm: remove regular file `/usr/mysql-cluster/ndb_1_config.bin.1'? y
```

   The configuration cache contains information about node ID 1 which was previously a management node but is now a data node. You must remove this cache to avoid problems starting the management node.

4. Restart the management and data node daemons on **host1**, reloading the configuration.

   At the **host1** terminal, issue the following commands and receive the results shown:

```
# ndb_mgmd -f /etc/config.ini --initial
MySQL Cluster Management Server mysql-5.6.17 ndb-7.3.5
-date and time- [MgmtSrvr] WARNING  -- at line 13: Cluster
configuration warning:
  arbitrator with id 49 and db node with id 1 on same host host1
  Running arbitrator on the same host as a database node may
  cause complete cluster shutdown in case of host failure.
# ndbmtd -c host1 --initial
-date and time- [ndbd] INFO     -- Angel connected to 'host1:1186'
-date and time- [ndbd] INFO     -- Angel allocated nodeid: 1
```

   Note that you need to use the `--initial` option when changing the `NoOfReplicas` option. During a normal cluster restart, do not use `--initial` when restarting data nodes.

   The arbitrator role becomes relevant when you have multiple data nodes. In this case, the management node is on the same host as one of the data nodes; this is not an optimal configuration, so the management node displays an error. The arbitrator role is further described in the lesson titled “Designing a MySQL Cluster.”

5. Use `scp` to copy the multi-threaded data node binary from `/usr/sbin/ndbmtd` to the same location on **host2**.

   At the **host1** terminal, issue the following commands and receive the results shown:

```
# scp /usr/sbin/ndbmtd host2:/usr/bin/ndbmtd
The authenticity of host 'host2 (192.168.100.202)' can't be
established.
RSA key fingerprint is e6:47:68:53:2a:8c:ae:d6:77:96:a0:a3:10:21:29:34.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'host2,192.168.100.202' (RSA) to the list of
known hosts.
ndbmtd                                    100%   34MB  33.5MB/s   00:01
```

6. Connect to **host2** as `root`, and launch a data node connected to the management daemon on **host1**.

   In a new terminal window, issue the following commands and receive the results shown:

```
# ssh host2
Last login: -date and time- from 192.168.100.201
# ndbmtd -c host1 --initial
-date and time- [ndbd] INFO     -- Angel connected to 'host1:1186'
-date and time- [ndbd] INFO     -- Angel allocated nodeid: 2
```

7. Use the management console to view the cluster status.

   In a terminal window connected to **host1**, issue the following commands and receive the results shown:

```
# ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> SHOW
Connected to Management Server at: localhost:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=1    @192.168.100.201  (mysql-5.6.17 ndb-7.3.5, Nodegroup: 0, *)
id=2    @192.168.100.202  (mysql-5.6.17 ndb-7.3.5, Nodegroup: 0)

[ndb_mgmd(MGM)] 1 node(s)
id=49    @192.168.100.201  (mysql-5.6.17 ndb-7.3.5)

[mysqld(API)]   1 node(s)
id=50 (not connected, accepting connect from any host)
```

8. Exit the management console.

   In the same terminal window, issue the following command:

```
ndb_mgm> EXIT
```

Practice 13-5: Configuring and Running a Cluster with MySQL Cluster Auto-Installer

Overview

In this practice, you extract the server binaries to **host3**, and you use the MySQL Cluster Auto-Installer to configure and run a basic cluster.

Note: In Oracle classrooms, **host3** and **host4** both have a `mysql` user configured, and the `mysql` user on **host3** can connect to **host4** using SSH key-based authentication. In non-Oracle classrooms you might need to create the `mysql` user and provide a username and password in the SSH configuration textboxes in the Auto-Installer to enable a connection between the Auto-Installer on **host3** and the shell on **host4**.

Tasks

1. Open an SSH connection to **host3**.
2. Extract the MySQL Cluster compressed archive to `/usr/share`.
3. Create a symlink to the newly extracted directory, named `mysql-cluster`.
4. Create symlinks to the management daemon and multi-threaded data node daemon server binaries in the extracted directory, placing them in `/usr/sbin`.
5. Using the `scp` program, copy the multi-threaded data node daemon server binary to **host4**'s `/usr/sbin` directory.
6. Start the MySQL Cluster Auto-Installer using the following command:

```
su - mysql -c \
   '/usr/share/mysql-cluster/bin/ndb_setup.py -n -N host3 -p8080'
```

7. Launch Firefox on the course image and navigate to **http://host3:8080**. In the browser window, use the following steps to configure a MySQL Cluster:
   - Click “Create New MySQL Cluster.”
   - On the “Define cluster” page, In the “Host list” text box, enter `host3,host4`. Click Next.
   - On the “Define hosts” page, view (but do not modify) the default values in the “MySQL Cluster data directory” fields for both hosts.
   - Ensure that the “MySQL Cluster install directory” field for **host3** contains `/usr/share/mysql-cluster`.
   - Edit the “MySQL Cluster install directory” field for **host4**, and change its value to `/usr`. Click Next.

- On the Define Processes page, modify the cluster topology, deleting nodes so that only the following nodes remain. Then click Next.
  - host3:
    - Management node 1
    - SQL node 1
    - Multi threaded data node 1
  - host4:
    - API node 3
    - Multi threaded data node 2
- On the “Define parameters” page, click “Show advanced configuration options” and browse through the settings of the different nodes. Click Next when you have viewed all settings.
- On the “Deploy configuration” page, select “Management node 1” in the “MyCluster processes” pane, and view the details of the configuration file.
- Click “Deploy and start cluster” to start the cluster processes.

8. Close the browser window.
9. In the terminal window running `ndb_setup.py`, press Ctrl + C to close the Auto-Installer.

Solutions 13-5: Configuring and Running a Cluster with MySQL Cluster Auto-Installer

Tasks

1. Open an SSH connection to **host3**.

   In a terminal window logged in as `root`, issue the following command:

   ```
   # ssh host3
   The authenticity of host 'host3 (192.168.100.203)' can't be
   established.
   RSA key fingerprint is e6:47:68:53:2a:8c:ae:d6:77:96:a0:a3:10:21:29:34.
   Are you sure you want to continue connecting (yes/no)? yes
   Warning: Permanently added 'host3,192.168.100.203' (RSA) to the list of
   known hosts.
   Last login: -date and time- from 192.168.100.201
   ```

2. Extract the MySQL Cluster compressed archive to `/usr/share`.

   At the same terminal, now logged in as `root` on **host3**, issue the following commands to extract the compressed archive to `/usr/share`:

   ```
   # cd /usr/share
   # tar xf /labs/mysql-cluster-advanced.tar.gz
   ```

3. Create a symlink to the newly extracted directory, named `mysql-cluster`.

   In the same terminal, issue the following command:

   ```
   # ln -s mysql-cluster-advanced* mysql-cluster
   ```

4. Create symlinks to the management daemon and multi-threaded data node daemon server binaries in the extracted directory, placing them in `/usr/sbin`.

   At the same terminal, issue the following commands:

   ```
   # ln -s /usr/share/mysql-cluster/bin/ndb_mgmd /usr/sbin/
   # ln -s /usr/share/mysql-cluster/bin/ndbmtd /usr/sbin/
   ```

5. Using the `scp` program, copy the multi-threaded data node daemon server binary to **host4**'s `/usr/sbin` directory.

   At the same terminal, issue the following command:

   ```
   # scp /usr/share/mysql-cluster/bin/ndbmtd host4:/usr/sbin/
   The authenticity of host 'host4 (192.168.100.204)' can't be
   established.
   RSA key fingerprint is e6:47:68:53:2a:8c:ae:d6:77:96:a0:a3:10:21:29:34.
   Are you sure you want to continue connecting (yes/no)? yes
   Warning: Permanently added 'host4,192.168.100.204' (RSA) to the list of
   known hosts.
   ndbmtd                                    100%   27MB  26.8MB/s   00:01
   ```

6. Start the MySQL Cluster Auto-Installer using the following command.

   At the same terminal, issue the following command and receive results similar to those shown:

   ```
   # su - mysql -c \
      '/usr/share/mysql-cluster/bin/ndb_setup.py -n -N host3 -p8080'
   Running out of install dir: /usr/share/mysql-cluster/bin
   Starting web server on port 8080
   deathkey=366696
   Navigate to http://host3:8080/welcome.html to launch the application.
   ```

   This command launches the MySQL Cluster Auto-Installer back-end web server under the identity of the `mysql` user, without launching a browser on the local machine. The web server responds to requests to **host3** on port 8080.

7. Launch Firefox on the course image and navigate to **http://host3:8080**. In the browser window, use the following steps to configure a MySQL Cluster:
   - Click "Create New MySQL Cluster."
   - On the "Define cluster" page, in the "Host list" text box, enter `host3,host4`. Click Next.
   - On the "Define hosts" page, view (but do not modify) the default values in the "MySQL Cluster data directory" fields for both hosts.
   - Ensure the "MySQL Cluster install directory" field for **host3** contains `/usr/share/mysql-cluster`.
   - Edit the "MySQL Cluster install directory" field for **host4**, and change its value to `/usr`. Click Next.
   - On the Define Processes page, modify the cluster topology, deleting nodes so that only the following nodes remain. Then click Next.
     - host3:
       - Management node 1
       - SQL node 1
       - Multi threaded data node 1
     - host4:
       - API node 3
       - Multi threaded data node 2
   - On the "Define parameters" page, click "Show advanced configuration options" and browse through the settings of the different nodes. Click Next when you have viewed all settings.
   - On the "Deploy configuration" page, select "Management node 1" in the "MyCluster processes" pane, and view the details of the configuration file.
   - Click "Deploy and start cluster" to start the cluster processes.

8. Close the browser window.
9. In the terminal window running `ndb_setup.py`, press Ctrl + C to close the Auto-Installer.