Hands-on Lab: MySQL Storage Engines and System Tables

Estimated time needed: 25 minutes

In this lab, you will use the MySQL Command Line Interface (CLI) to carry out a variety of functions related to selecting and understanding some of the alternative storage engines available in MySQL. You will then continue on to explore the system tables in MySQL which contain meta data about the objects in the server.

Objectives

After completing this lab, you will be able to use the MySQL command line interface to:

- Create tables using alternative storage engines.
- Query MySQL system tables to retrieve meta data about objects in the database.

Software and Database Used in this lab

Software Used in this Lab

In this lab, you will use MySQL. MySQL is a Relational Database Management System (RDBMS) designed to efficiently store, manipulate, and retrieve data.

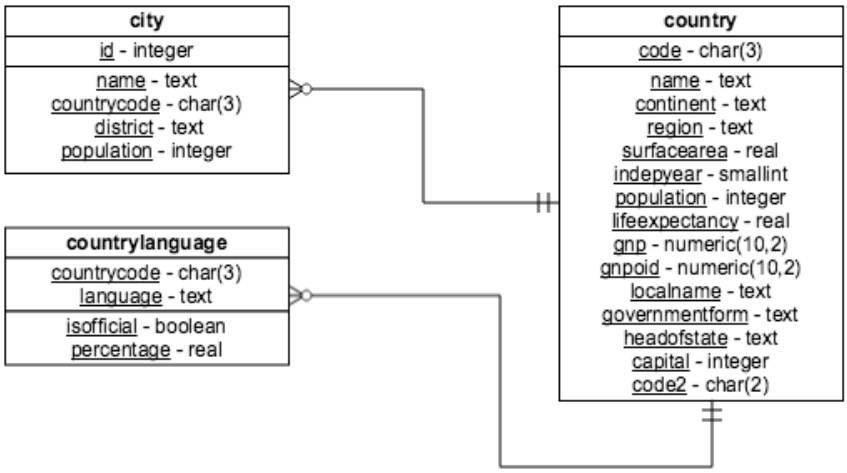
To complete this lab you will utilize the MySQL relational database service available as part of the IBM Skills Network Labs (SN Labs) Cloud IDE. SN Labs is a virtual lab environment used in this course.

Database Used in this Lab

The World database used in this lab comes from the following source: https://dev.mysql.com/doc/world-setup/en/ under CC BY 4.0 License with Copyright 2021 - Statistics Finland.

You will use a modified version of the database for the lab, so to follow the lab instructions successfully please use the database provided with the lab, rather than the database from the original source.

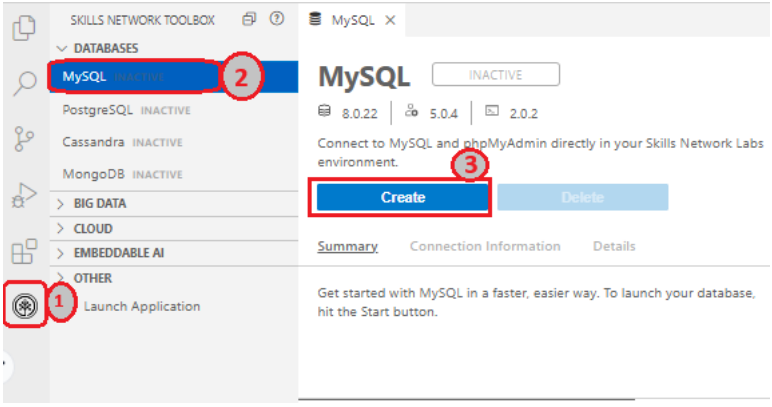
The following ERD diagram shows the schema of the World database:



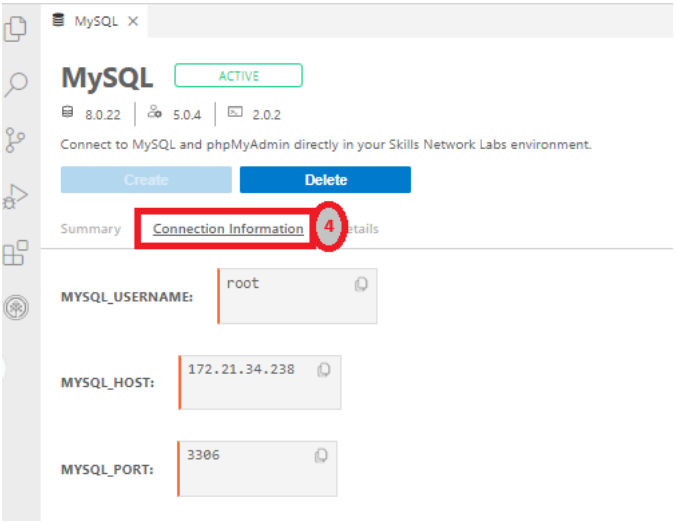
The first row is the table name, the second is the primary key, and the remaining items are any additional attributes.

Exercise 1: Create your database

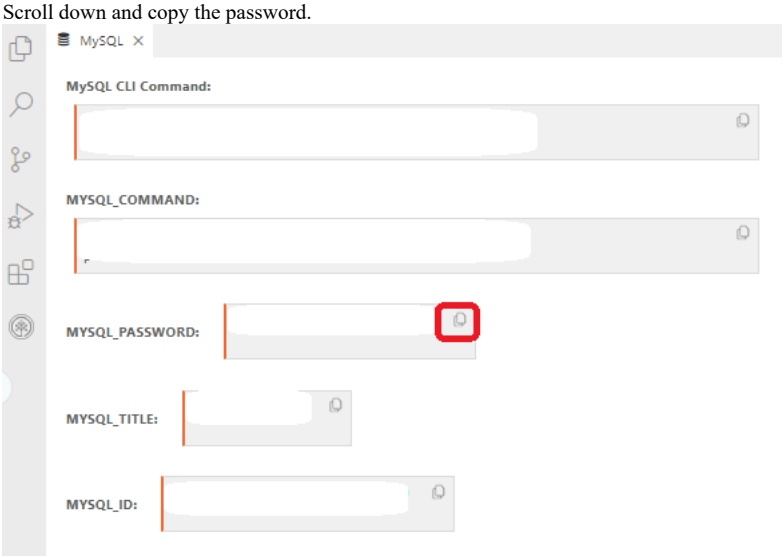
1. Go to **Skills Network Toolbox** by clicking the icon shown below from the side by side launched Cloud IDE.
2. From the **Databases** drop down menu, click **MySQL** to open the MySQL service session tab.
3. Click the **Createe** button and wait until MySQL service session gets launched.



The MySQL server will take a few moments to start. Once it is ready, you will see the green “Active” label near the top of the window.



Scroll down and copy the password.



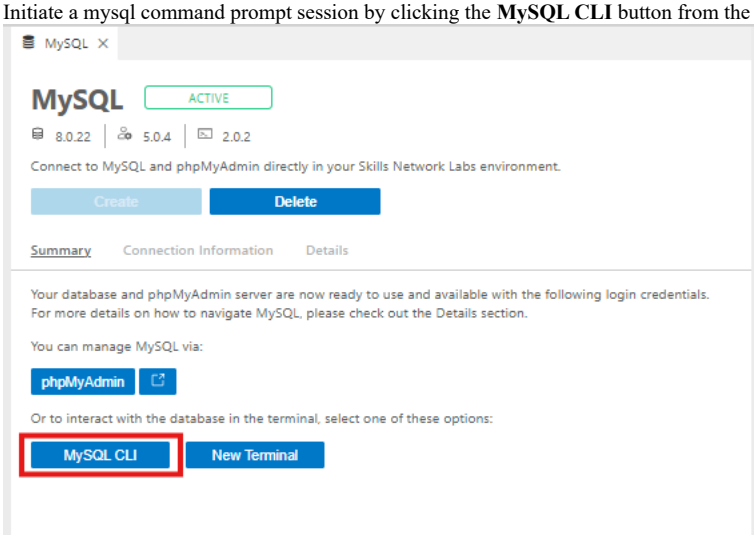
- **NOTE:** Whenever you are required to enter your MySQL service session password from the MySQL service session tab at any step of the lab, copy the password by clicking on the small copy button on the right of the password block. Paste the password into the terminal using **Ctrl + V** (Mac: ⌘ + V), and press **Enter** on the keyboard. For security reasons, you will not see the password as it is entered on the terminal.

4. Click **New Terminal** button from the mysql service session tab. Now you need to fetch two mysql script files to the Cloud IDE user session storage. Copy the command below by clicking on the little copy button on the bottom right of the codeblock. Then paste it into the terminal at the command line prompt using **Ctrl + V** (Mac: ⌘ + V), and **Enter** on keyboard. Do this for each of the commands below one at a time.

    - world_mysql_script.sql

```
wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0231EN-SkillsNetwork/datasets/World/world_mysql_script.
```

5. Initiate a mysql command prompt session by clicking the **MySQL CLI** button from the mysql service session tab.



6. Create a new database **world** using the command below in the terminal:

```
CREATE DATABASE world;
```

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3039
Server version: 8.0.22 MySQL Community Server - GPL

Copyright (c) 2000, 2020, 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> create database world;
Query OK, 1 row affected (0.01 sec)

mysql>
```

7. To use the newly created world database, use the command below in the terminal:

```
USE world;
```

8. Execute the world mysql script (world_mysql.sql) to complete the world database creation process using the command below in the terminal:

```
SOURCE world_mysql_script.sql;
```

9. To list all the table names from the world database, use the command below in the terminal:

```
SHOW TABLES;
```

```
mysql> SHOW TABLES;
+-----------------+
| Tables_in_world |
+-----------------+
| city            |
| country         |
| countrylanguage |
+-----------------+
3 rows in set (0.00 sec)

mysql>
```

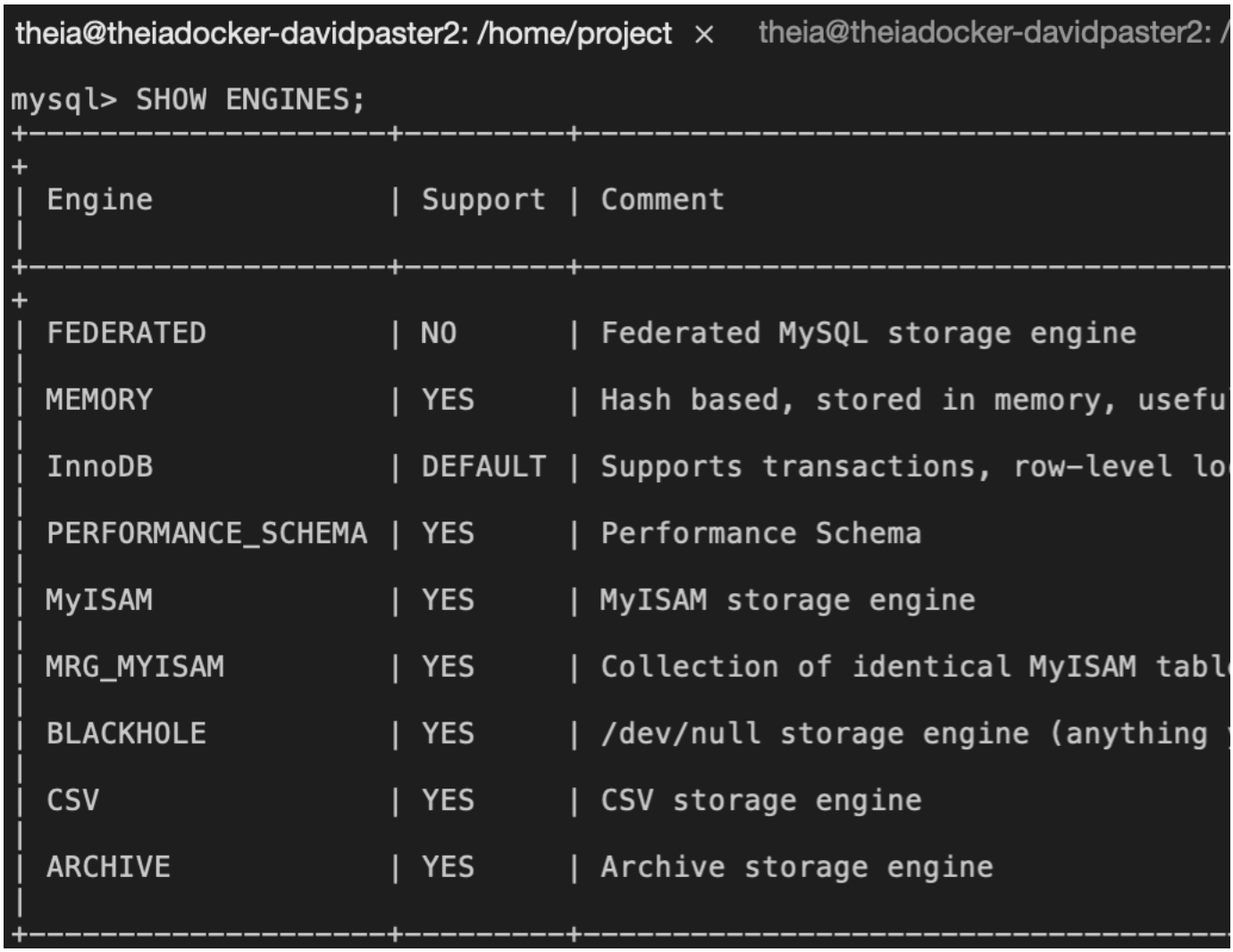
Exercise 2: Manage MySQL Storage Engines

In MySQL, ***Storage Engines*** are components that handle SQL operations for different table types. The default and most general purpose storage engine in MySQL is **InnoDB**. When you create a new table in MySQL using the `CREATE TABLE` command in the Command Line Interface, it creates a InnoDB table by default. This is the most widely useful one and is recommended for most general applications except for a few specialized cases.

As detailed in the MySQL documentation, MySQL is built with a pluggable storage engine architecture that allows storage engines to be easily loaded into and unloaded from a running MySQL server.

1. To see a list of the Storage Engines supported on your MySQL server, enter the following command into the MySQL Command Line Interface:

```
SHOW ENGINES;
```



As you can see, there are several columns. The first column gives the name of the Storage Engine and the next column tells us whether that engine is supported on your MySQL server. As you can see, the InnoDB engine is listed as the DEFAULT.

The CSV Storage Engine

From the `SHOW ENGINES;` command you ran earlier, you can see that the **CSV** Engine is supported on the current running MySQL server. CSV files, short for *Comma Separated Values*, are delimited text files that uses a comma to separate values.

Let's go ahead and try making a table in our database using the CSV storage engine.

2. To create a new table with a storage engine other than the default InnoDB database, we specify the storage engine we wish to use inside the `CREATE TABLE` command. Let's create a new table called "test_csv" using the CSV engine by entering the following command into the CLI:

```
CREATE TABLE csv_test (i INT NOT NULL, c CHAR(10) NOT NULL) ENGINE = CSV;
```

3. Let's confirm that the table was successfully created with the following command:

```
SHOW TABLES;
```

```
mysql> CREATE TABLE csv_test (i INT NOT NULL, c CHAR(10) NOT NULL) ENGI
Query OK, 0 rows affected (0.01 sec)

mysql> SHOW TABLES;
+-----------------+
| Tables_in_world |
+-----------------+
| city            |
| country         |
| countrylanguage |
| csv_test        |
+-----------------+
4 rows in set (0.00 sec)

mysql>
```

4. let's add some sample data into our table. We will add three entries with the following command:

```
INSERT INTO csv_test VALUES(1,'data one'),(2,'data two'),(2,'data three');
```

5. Take a look at the new values you entered with the CLI:

```
SELECT * FROM csv_test;
```

```
mysql>  INSERT INTO csv_test VALUES(1,'data one'),(2,'data two'),(2,'da
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM csv_test;
+---+------------+
| i | c          |
+---+------------+
| 1 | data one   |
| 2 | data two   |
| 2 | data three |
+---+------------+
3 rows in set (0.00 sec)
```

As you can see, CSV storage engines function in many of the same ways as the default InnoDB engines, however, there are a few limitations. These include not supporting indexing or partitioning.

As you saw for yourself from the output of the `SHOW ENGINES;` command there are many different storage engines. We encourage you to explore and experiment with them yourself! You can read more about their specific use cases and limitations in the MySQL documentation.

Exercise 3: Navigate the MySQL System Tables

The MySQL server contains a database called `mysql`. This is the system database that contains information *required* for the server to run, such as meta data on all the other tables in the database. This database is one of the special cases where the default InnoDB storage engine is *not* used. Instead, the tables in the `mysql` database used the MyISAM storage engine. In general, we mostly query the system tables and rarely modify them directly.

The tables in the `mysql` database fall into several categories, some of which include:

- Grant System Tables
- Object Information System Tables
- Log System Tables
- Server-Side Help System Tables

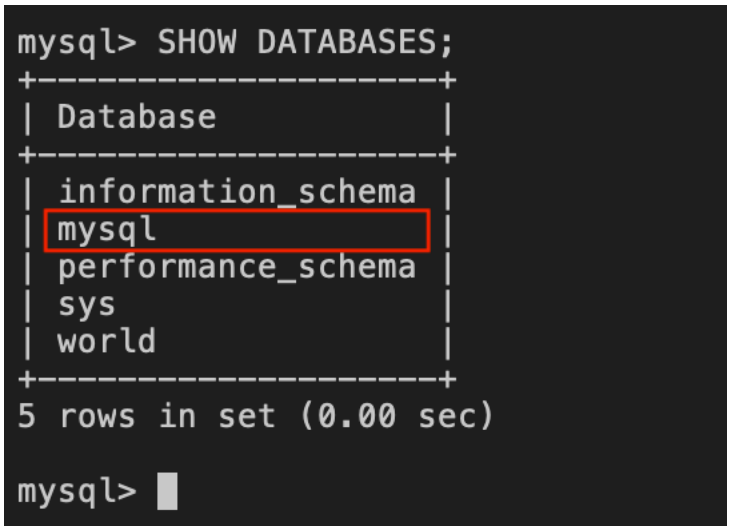
For your reference, an exhaustive list of the categories can be found in Section 5.7 of the MySQL documentation.

Grant System Table Category

Let's take a deeper look at the **Grant System Table** category. They contain information about the user accounts and the privileges granted to them.

1. First, let's see all the databases on the MySQL server by entering the following command into the CLI:

```
SHOW DATABASES;
```



2. Now connect to the **mysql** data by entering:

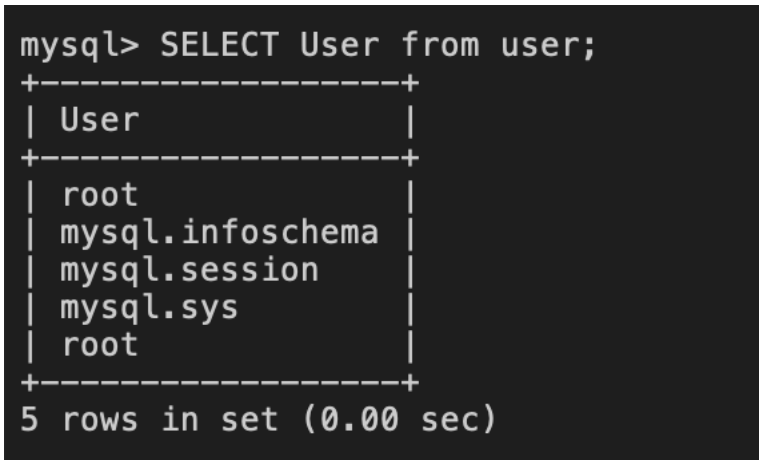
```
USE mysql;
```

3. Take a look at all the tables in the database by entering the following in the CLI:

```
SHOW TABLES;
```

4. The **user** table contains user accounts, global privileges, and other nonprivilege columns. There are many columns in this table and is a little unwieldy to look at so let's take a look at just the first column which lists the names of the users in the database. Enter the following into the CLI:

```
SELECT User from user;
```



5. Let's add a new user to the database and see if the change is reflected in the user table. We will go into depth about adding users to a database later in the course in the *Hands-on Lab: MySQL User Management, Access Control, and Encryption* but for now, we'll just execute a simple command in the CLI that will create a new user named "test_user":

```
CREATE USER test_user;
```

6. **Try it yourself:** Confirm that the *user* table in the *mysql* database was automatically updated to reflect the addition of the new user you created. *Enter the appropriate command into the CLI to show the the User column in the user table.*

▼ **Hint** (Click Here)
Take a peek at the command you entered in step 4.
▼ **Solution** (Click Here)

```
SELECT User from user;
```

```
mysql> CREATE USER test_user;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT User from user;
+------------------+
| User             |
+------------------+
| root             |
| test_user        |
| mysql.infoschema |
| mysql.session    |
| mysql.sys        |
| root             |
+------------------+
6 rows in set (0.01 sec)
```

As you can see, “test_user” was added to this table automatically, you didn’t have to use any `INSERT` commands. This is, in general, the case for system tables - you rarely should have to edit them directly. Instead, they are typically queried.

Query the `INFORMATION_SCHEMA` Database Tables

The `INFORMATION_SCHEMA` is a database found inside every MySQL server. It contains meta data about the MySQL server such as the name of a database or table, the data type of a column, or access privileges. Note that this database contains *read-only* tables, so you cannot directly use any `INSERT`, `UPDATE`, or `DELETE` commands on them. Let’s go ahead and connect to the database.

1. **TRY IT YOURSELF:** Using the MySQL CLI, view all the databases on the server.

   ▼ **Hint** (Click Here)
   You already did this is Step 1 of the previous section about tbe Grant System Table category. Feel free to refer back to that for some guidance.
   ▼ **Solution** (Click Here)

   ```
   SHOW DATABASES;
   ```

   ```
   mysql> SHOW DATABASES;
   +--------------------+
   | Database           |
   +--------------------+
   | information_schema |
   | mysql              |
   | performance_schema |
   | sys                |
   | world              |
   +--------------------+
   5 rows in set (0.00 sec)
   ```

   As you can see, the `information_schema` database is there by default.

2. **Try it yourself:** Enter the relevant command in the CLI to connect to the `information_schema` database.

   ▼ **Hint** (Click Here)
   Recall the command you entered in the previous section to connect to the `mysql` database. How can it be modified to connect to the `information_schema` database?
   ▼ **Solution** (Click Here)

   ```
   USE information_schema;
   ```

3. In the `information_schema` database, there exists a table called `COLUMNS` which contains meta data about the columns for all tables and views in the server. One of the columns in this table contains the names of all the other columns in every table. Let’s go ahead and look at the names of the columns in the `country` table in the `world` database by entering the following command in the CLI:

   ```
   SELECT COLUMN_NAME FROM COLUMNS WHERE TABLE_NAME = 'country';
   ```

```
mysql> SELECT COLUMN_NAME FROM COLUMNS WHERE TABLE_NAME = 'country';
+----------------+
| COLUMN_NAME    |
+----------------+
| Capital        |
| Code           |
| Code2          |
| Continent      |
| GNP            |
| GNPOld         |
| GovernmentForm |
| HeadOfState    |
| IndepYear      |
| LifeExpectancy |
| LocalName      |
| Name           |
| Population     |
| Region         |
| SurfaceArea    |
+----------------+
15 rows in set (0.01 sec)
```

4. Another point of interest in the `information_schema` database is the `TABLES` table which contains meta data about all the tables in the server. One of the columns in this table contains information about a table's storage engine type. To tie this back to our earlier discussion about storage engines, run the following command in the CLI to view the storage engine type for the 'country', 'city', 'countrylanguage', and finally the 'csv_test' table you created:

```
SELECT table_name, engine FROM INFORMATION_SCHEMA.TABLES
WHERE table_name = 'country' OR table_name = 'city'
OR table_name = 'countrylanguage' OR table_name = 'csv_test';
```

```
mysql>     SELECT table_name, engine FROM INFORMATION_SCHEMA.TABLES
    ->     WHERE table_name = 'country' OR table_name = 'city'
    ->     OR table_name = 'countrylanguage' OR table_name = 'csv_te
+-----------------+--------+
| TABLE_NAME      | ENGINE |
+-----------------+--------+
| city            | InnoDB |
| country         | InnoDB |
| countrylanguage | InnoDB |
| csv_test        | CSV    |
+-----------------+--------+
4 rows in set (0.00 sec)
```

As expected, the first three tables mentioned use the default InnoDB storage engine, while the 'csv_test' table uses the CSV storage engine.

5. Finally, the `TABLES` table in the `information_schema` database contains information on the the size of a given table in bytes. This information is stored in two columns: **data_length** and **index_length** which stores the size of the data in the table and the size of the index file for that table, respectively. Therefore, the total size of the table is the sum of the values in these two columns. This value would be given in bytes, however, if you wish to use a more convenient unit, the sum can be converted to kB by dividing by 1024. You can find the size of the tables (in kB) you queried in the previous step with the following command in the CLI:

```
SELECT table_name, (data_length + index_length)/1024 FROM INFORMATION_SCHEMA.TABLES
WHERE table_name = 'country' OR table_name = 'city'
OR table_name = 'countrylanguage' OR table_name = 'csv_test';
```

```
mysql> SELECT table_name, (data_length + index_length)/1024 FROM INF
    ->     WHERE table_name = 'country' OR table_name = 'city'
    ->     OR table_name = 'countrylanguage' OR table_name = 'csv_te
+-----------------+-----------------------------------+
| TABLE_NAME      | (data_length + index_length)/1024 |
+-----------------+-----------------------------------+
| city            |                          496.0000 |
| country         |                           96.0000 |
| countrylanguage |                          160.0000 |
| csv_test        |                            0.0000 |
+-----------------+-----------------------------------+
4 rows in set (0.00 sec)
```

Exercise 4: Try it Yourself!

In this practice exercise, you will put together what you learned about storage engines in MySQL to first create a new table using the non-default MyISAM storage engine. You will then apply what you learned about MySQL System Tables to fetch metadata about your newly created table.

1. **Try it yourself:** First, connect to the `world` database using the CLI:

   ▼ **Hint** (Click Here)
   The command to connect to a database takes on the following general form:

   ```
   USE <database name>;
   ```

   ▼ **Solution** (Click Here)

   ```
   USE world;
   ```

2. **Try it yourself:** Create a new table called `MyISAM_test` that uses the `MYISAM` storage engine.

   ▼ **Hint** (Click Here)
   Recall that you did something very similar in Exercise 2 when you created a new table using th csv storage engine. What can you change in the command you used to accomplish this task?

   To save you the scrolling, here's the command you used in Exercise 2 to create `csv_test`:

   ```
   CREATE TABLE csv_test (i INT NOT NULL, c CHAR(10) NOT NULL) ENGINE = CSV;
   ```

   ▼ **Solution** (Click Here)

   ```
   CREATE TABLE MyISAM_test (i INT NOT NULL, c CHAR(10) NOT NULL) ENGINE = MYISAM;
   ```

   This will create an empty table called `MyISAM_test` where the first column, named `i` will eventually contain an integer, and the second column named `c` will contain up to 10 characters.

3. **Try it yourself:** Next, you'll want to query a table in the `information_schema` database, but before that, you'll have to connect to the database first. Use the CLI to connect to the `information_schema` database.

   ▼ **Hint** (Click Here)

   Recall what you did in Step 1 of this practice exercise. This step is very similar.

   ▼ **Solution** (Click Here)

   ```
   USE information_schema;
   ```

4. **Try it yourself:** Using the CLI, query the `TABLES` table in the `information_schema` database to display the `table_name` and `engine` columns of all tables that have `table_schema = 'world'`. Confirm that the table you created in this exercise is there and it has the correct storage engine.

   ▼ **Hint** (Click Here)

   To query specific columns in a table based on some condition, use a command of the form:

   ```
   SELECT <column_1>, <column_2>, ... FROM <table name> WHERE <condition>;
   ```

   Bonus hint: In this case, the table name should be `INFORMATION_SCHEMA.TABLES`.

   ▼ **Solution** (Click Here)

   ```
   SELECT table_name, engine FROM INFORMATION_SCHEMA.TABLES WHERE table_schema = 'world';
   ```

```
mysql> SELECT table_name, engine FROM INFORMATION_SCHEMA.TABLES WHER
+-----------------+--------+
| TABLE_NAME      | ENGINE |
+-----------------+--------+
| MyISAM_test     | MyISAM |
| city            | InnoDB |
| country         | InnoDB |
| countrylanguage | InnoDB |
| csv_test        | CSV    |
+-----------------+--------+
5 rows in set (0.01 sec)
```

As you can see, the `MyISAM_test` table is there and it uses the MyISAM storage engine as expected.

Conclusion

Congratulations! You have completed this hands on lab and now understand the basics of MySQL storage engines and have the skills to create new tables with alternate storage engines for specialized use cases. Furthermore, you have gained some familiarity with system tables in MySQL and how to query them to retrieve meta data about other objects in your database.

Author

- David Pasternak

Other Contributors

- Sandip Saha Joy