

Chapter 11

How to use a MySQL database

Objectives

Applied

1. Use MySQL Workbench to start and stop the MySQL server.
2. Use MySQL Workbench to run SQL statements.
3. Use MySQL Workbench to run SQL scripts.
4. Code simple SELECT, INSERT, UPDATE, and DELETE statements and use MySQL Workbench to test them.

Knowledge

1. Distinguish between SQL's Data Definition Language (DDL) and Data Manipulation Language (DML).
2. Describe the capabilities of a SELECT statement.
3. Describe the capabilities of INSERT, UPDATE, and DELETE statements.
4. Describe what a SQL script does.

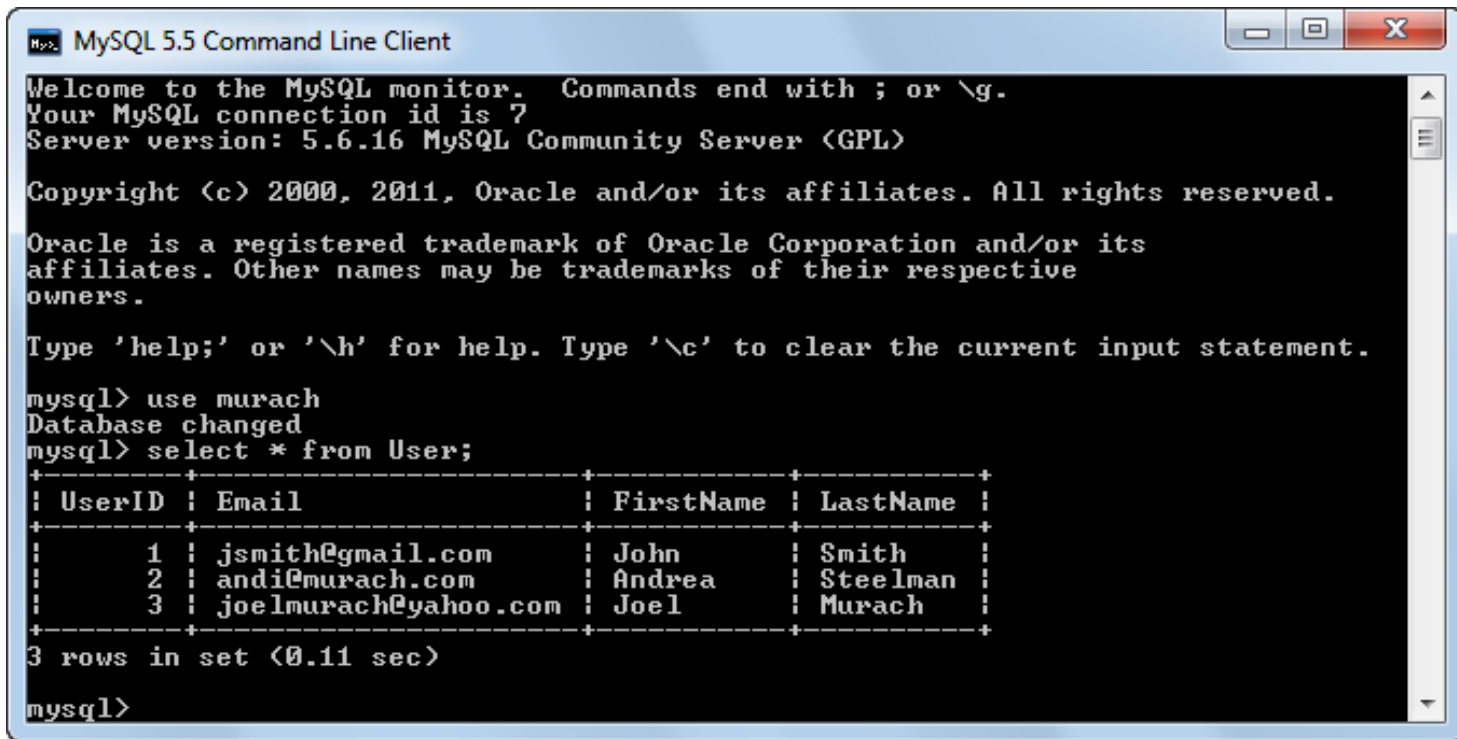
MySQL is...

- **Inexpensive.** Free for most uses and relatively inexpensive for other uses.
- **Fast.** One of the fastest relational databases currently available.
- **Easy to use.** Easy to install and use.
- **Portable.** Runs on most modern operating systems including Windows, OS X, and Linux.

MySQL provides...

- **Support for SQL.** Like any modern database product, MySQL supports SQL.
- **Support for multiple clients.** Supports access from multiple clients from a variety of interfaces and programming languages including Java, PHP, Python, Perl, and C.
- **Connectivity.** Provides access to data via an intranet or the Internet.
- **Security.** Protects access to your data so only authorized users can view the data.
- **Referential integrity.** With MySQL 5.5 and later, InnoDB tables are used by default, which support referential integrity.
- **Transaction processing.** With version 5.5, MySQL uses InnoDB tables by default, which provide support for transaction processing.

A command-line tool



```
MySQL 5.5 Command Line Client
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.6.16 MySQL Community Server (GPL)

Copyright (c) 2000, 2011, 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> use murach
Database changed
mysql> select * from User;
+-----+-----+-----+-----+
| UserID | Email                | FirstName | LastName |
+-----+-----+-----+-----+
|      1 | jsmith@gmail.com     | John     | Smith    |
|      2 | andi@murach.com      | Andrea   | Steelman |
|      3 | joelmurach@yahoo.com | Joel     | Murach   |
+-----+-----+-----+-----+
3 rows in set (0.11 sec)

mysql>
```

MySQL Workbench

The screenshot displays the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar contains a Navigator pane with sections for MANAGEMENT (Server Status, Client Connections, Users and Privileges, Status and System Variables, Data Export, Data Import/Restore), INSTANCE (Startup / Shutdown, Server Logs, Options File), and SCHEMAS (Filter objects, murach, Tables, user, Columns). The main workspace shows a query editor with the text `select * from User`. Below the editor is a table with the following data:

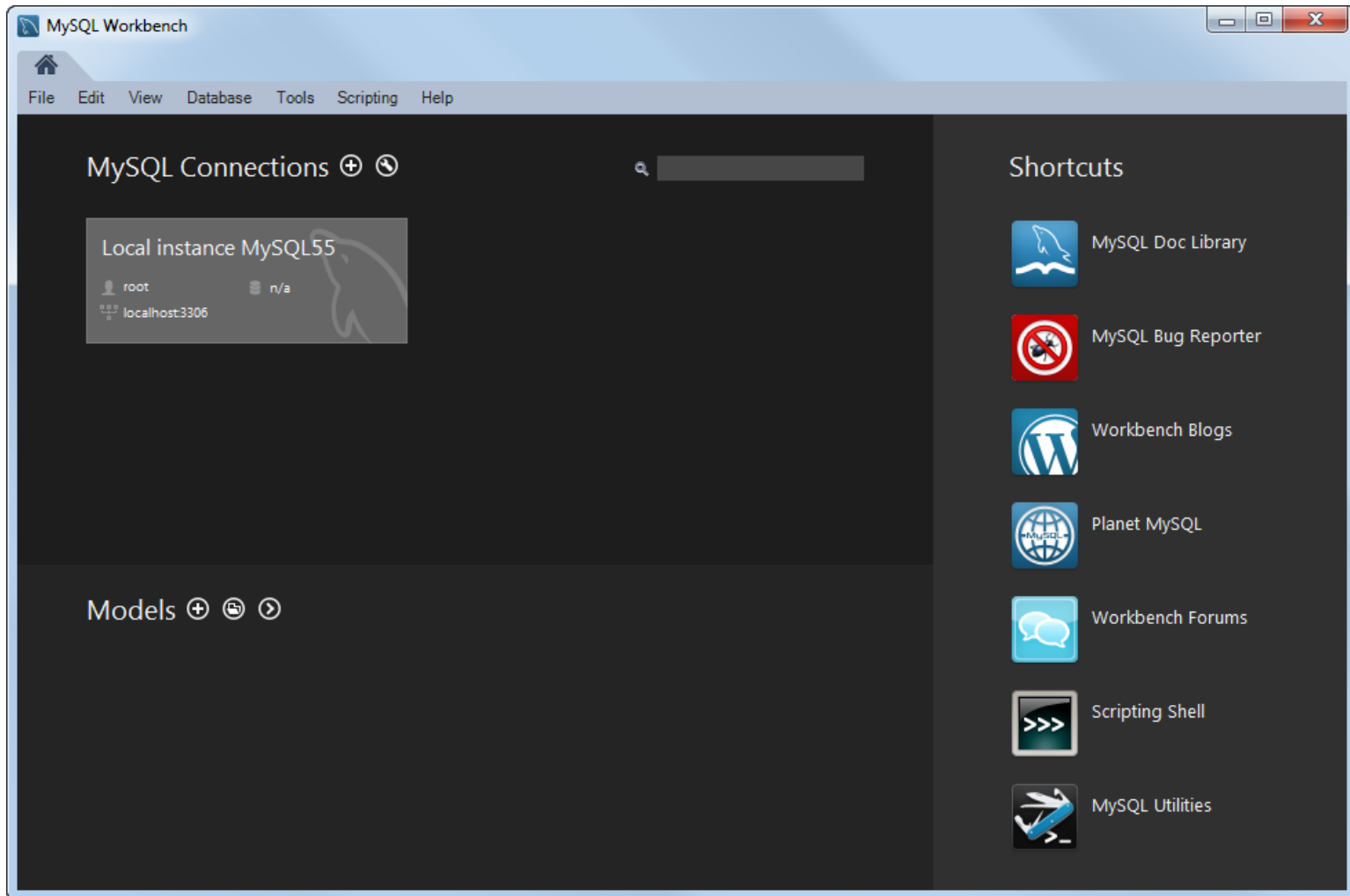
	UserID	Email	FirstName	LastName
▶	1	jsmith@gmail.com	John	Smith
	2	andi@murach.com	Andrea	Steelman
	3	joelmurach@yahoo.com	Joel	Murach
*	NULL	NULL	NULL	NULL

Below the table is an Output pane showing the Action Output table:

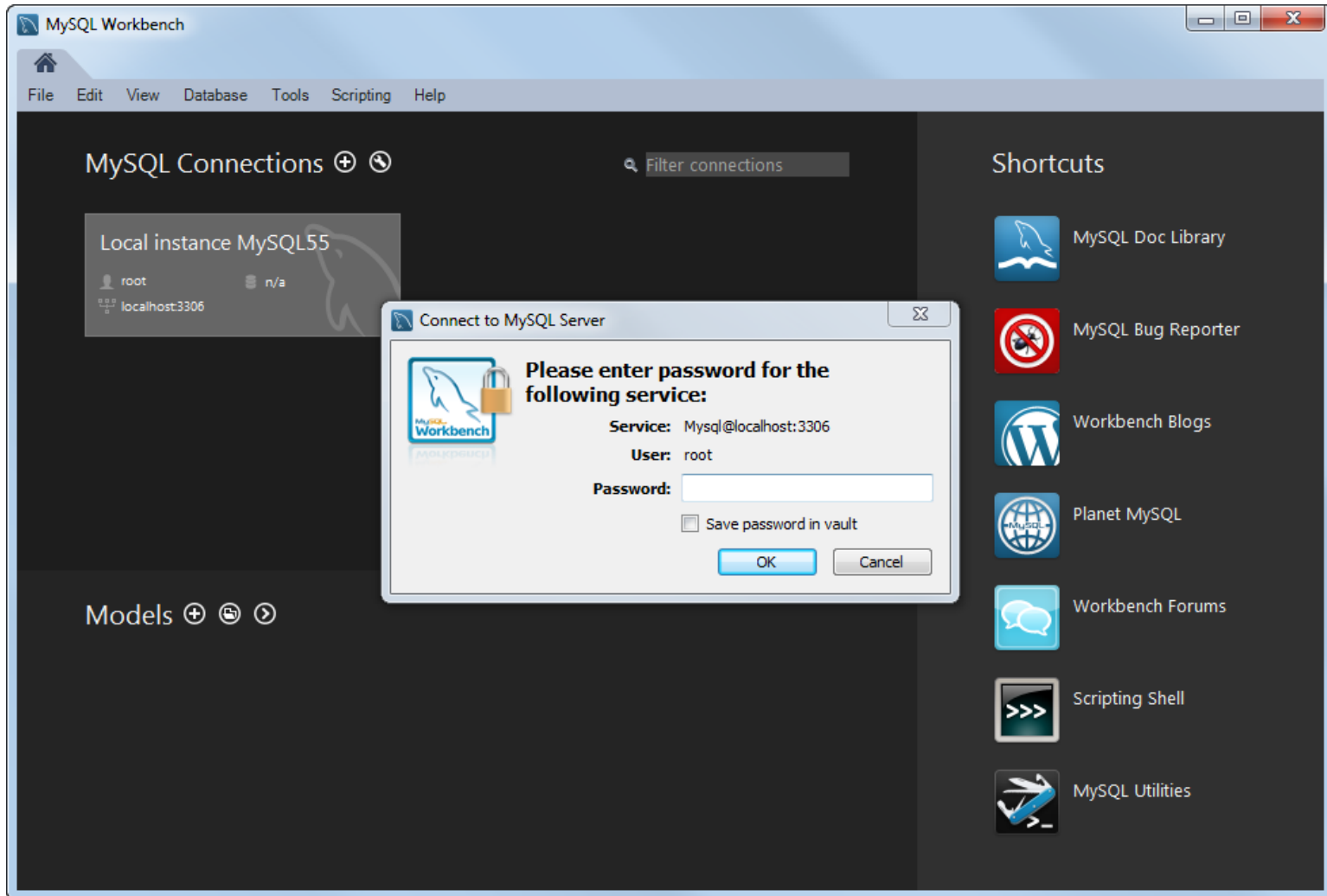
	Time	Action	Message	Duration / Fetch
✓	1 13:56:07	select * from User LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

The bottom status bar indicates "Query Completed".

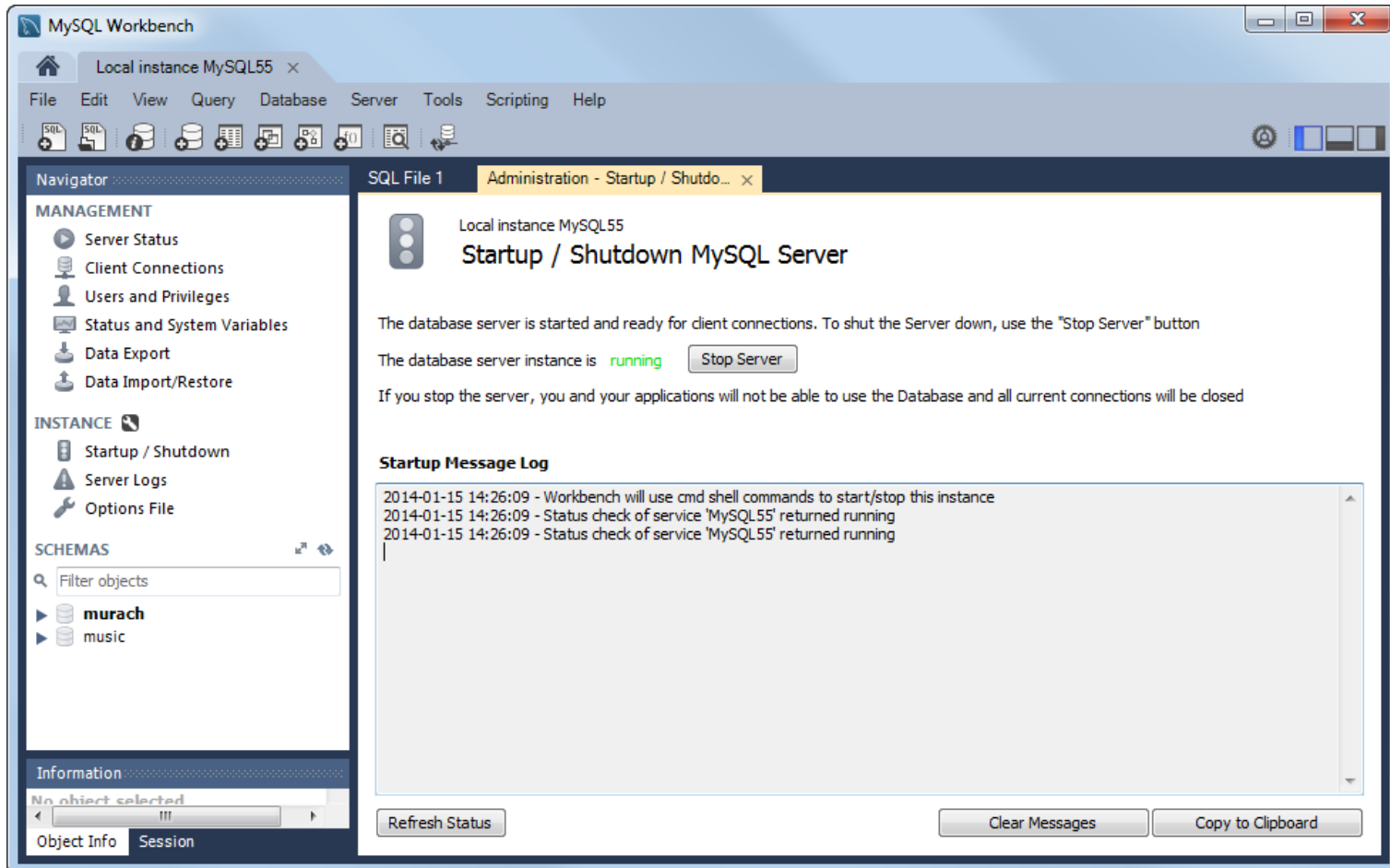
The Home tab of MySQL Workbench



The dialog box for opening database connections



The Startup/Shutdown option



A SELECT statement and its results

Create New SQL Tab button

Execute Current Statement button

SQL tab

Results tab

The screenshot shows the MySQL Workbench interface. The main window is titled 'Query 1' and contains the following SQL query:

```
1 SELECT UserID, LastName, FirstName
2 FROM User
3 ORDER BY UserID
```

Below the query editor, the 'Result Set Filter' is empty. The results are displayed in a table with the following data:

UserID	LastName	FirstName
1	Smith	John
2	Steelman	Andrea
3	Murach	Joel
* NULL	NULL	NULL

Below the table, the 'Output' section shows the 'Action Output' for the query:

Time	Action	Message	Duration / Fetch
14:19:29	SELECT UserID, LastName, FirstName FROM user O...	3 row(s) returned	0.000 sec / 0.000 sec

The interface also shows a 'Navigator' pane on the left with 'MANAGEMENT', 'INSTANCE', and 'SCHEMAS' sections. The 'SCHEMAS' section shows the 'murach' schema with tables 'download', 'sequence', and 'user', and columns 'UserID', 'Email', and 'FirstName'. The 'Information' pane at the bottom shows 'Schema: murach' and 'Object Info' for 'Session'. The status bar at the bottom indicates 'Query Completed'.

How to enter and execute a SQL statement

- To open a new SQL tab, press Ctrl+T or click the Create New SQL Tab button in the SQL editor toolbar.
- To select the current database, double-click it in the Schemas section of the Navigator window. This displays the selected database in bold.
- To enter a SQL statement, type it into the SQL tab.
- As you enter the text for a statement, the SQL tab applies color to various elements, such as SQL keywords, to make them easy to identify.
- To execute a SQL statement, press Ctrl+Enter, or click the Execute Current Statement button in the SQL editor toolbar. If the statement retrieves data, the data is displayed in a Results tab below the SQL tab.

A SQL script and its results

Execute SQL Script
button

The screenshot displays the MySQL Workbench interface. The main window shows a SQL script with the following content:

```
1 SELECT UserID, LastName, FirstName
2 FROM User
3 ORDER BY UserID;
4
5 SELECT *
6 FROM User
7 WHERE UserID = 1;
```

The results of the execution are shown in a table below the script:

UserID	Email	FirstName	LastName
1	jsmith@gmail.com	John	Smith
*	NULL	NULL	NULL

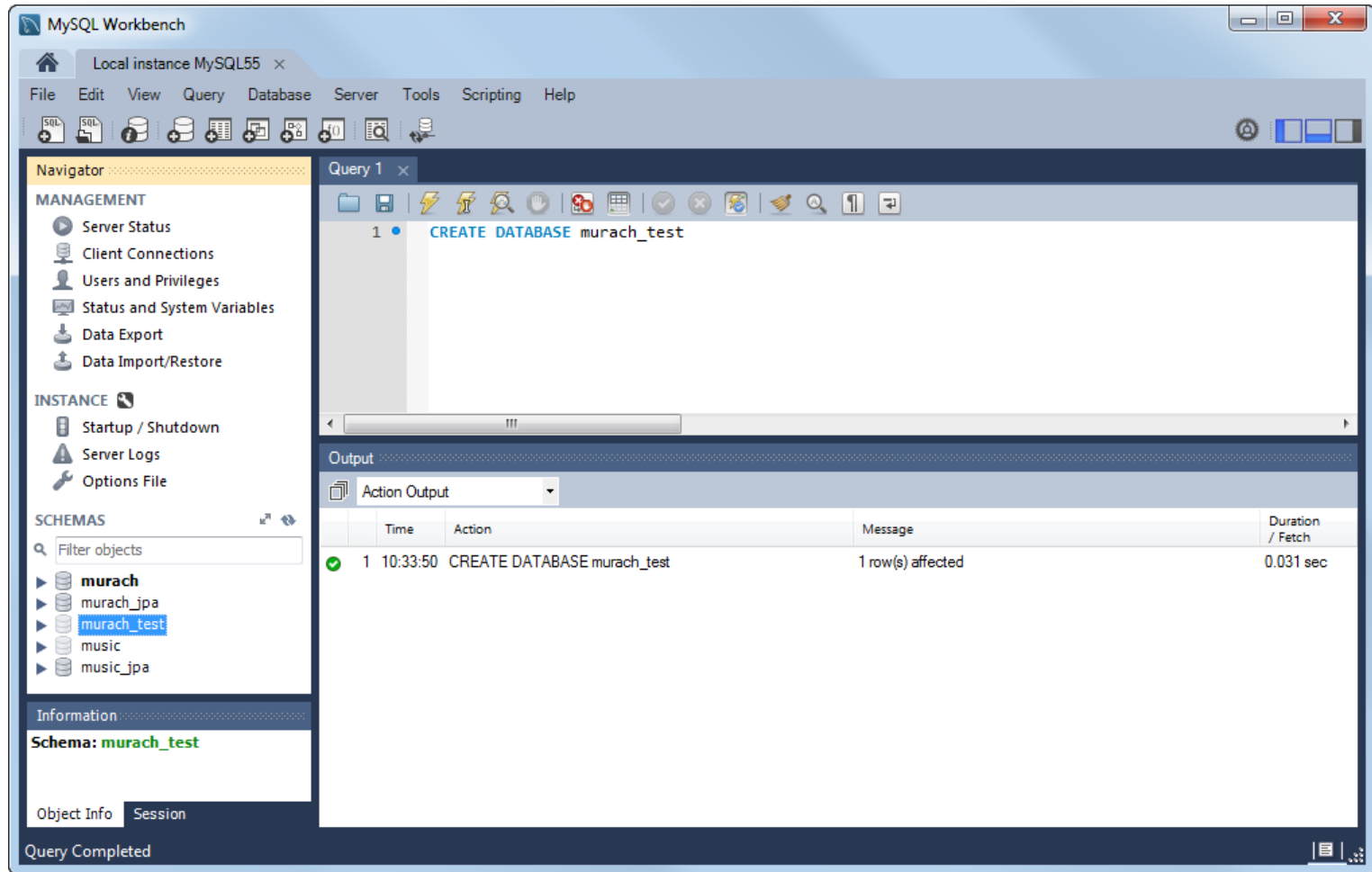
The Action Output table at the bottom provides details on the execution:

	Time	Action	Message	Duration / Fetch
✓	1 11:37:24	SELECT UserID, LastName, FirstName FROM User ORD...	3 row(s) returned	0.000 sec / 0.000 sec
✓	2 11:37:24	SELECT * FROM User WHERE UserID = 1 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

How to enter and execute a SQL script

- When you code a script that contains more than one statement, code a semicolon at the end of each statement.
- To run an entire SQL script, press the Ctrl+Shift+Enter keys or click the Execute SQL Script button that's located just to the left of the Execute Current Statement button in the SQL editor toolbar.
- When you run a SQL script, the results of each statement that returns data are displayed in a separate Results tab.
- To execute one SQL statement within a script, move the insertion point into that statement and press Ctrl+Enter or click the Execute Current Statement button. If the statement retrieves data, the data is displayed in a Results tab.
- To execute two or more statements within a script, select them in the editor and then press Ctrl+Shift+Enter or click the Execute SQL Script button.

After a statement has been executed



How to create a database

```
CREATE DATABASE murach_test
```

How to select a database for use

```
USE murach_test
```

How to drop a database

```
DROP DATABASE murach_test
```

How to create, select, and drop a database

- Use the CREATE DATABASE statement to create a database and the DROP DATABASE statement to delete a database. These are *SQL statements*.
- Use the USE command to select the database that you want to work with. This is a *MySQL command*.
- You can also select a database by double-clicking on it in the Schemas section in MySQL Workbench.
- The selected database will appear in bold in the Schemas section.

How to create a table

```
CREATE TABLE User (  
    UserID INT NOT NULL AUTO_INCREMENT,  
    Email VARCHAR(50),  
    FirstName VARCHAR(50),  
    LastName VARCHAR(50),  
    PRIMARY KEY (UserID)  
)
```

How to drop a table

```
DROP TABLE User
```

How to drop a table only if it exists

```
DROP TABLE IF EXISTS User
```

Tables, rows, columns, etc.

- A *relational database* consists of one or more *tables* that consist of *rows (records)* and *columns (fields)*.
- The *primary key* in a table is the one that uniquely identifies each of the rows in the table.
- A *foreign key* is used to relate the rows in one table to the rows in another table.
- When you create a table, you define each of its columns and you identify its primary key.
- To define a column, you must supply the name and the data type, whether it's automatically generated for new rows, and so on.
- On Unix systems, the table and column names are case-sensitive.

An INSERT statement that inserts multiple rows

```
INSERT INTO User
  (FirstName, LastName, Email)
VALUES
  ('John', 'Smith', 'jsmith@gmail.com'),
  ('Andrea', 'Steelman', 'andi@murach.com'),
  ('Joel', 'Murach', 'joelmurach@yahoo.com')
```

How to insert multiple rows into a table

- The INSERT statement lets you insert one or more rows into one table of a database. When you code it, you need to include data for all columns that aren't defined with default values or aren't automatically generated.
- On a Unix system, table and column names are case-sensitive.

A SELECT statement that gets all columns

Syntax

```
SELECT *  
FROM table-1  
[WHERE selection-criteria]  
[ORDER BY column-1 [ASC|DESC] [, column-2 [ASC|DESC] ...]]
```

A statement that selects all rows and columns

```
SELECT * FROM User
```

Result set

	UserID	Email	FirstName	LastName
▶	1	jsmith@gmail.com	John	Smith
	2	andi@murach.com	Andrea	Steelman
	3	joelmurach@yahoo.com	Joel	Murach
*	NULL	NULL	NULL	NULL

A SELECT statement that gets selected columns

Syntax

```
SELECT column-1 [,column-2] ...  
FROM table-1  
[WHERE selection-criteria]  
[ORDER BY column-1 [ASC|DESC] [,column-2 [ASC|DESC] ...]]
```

A statement that selects two rows and two columns

```
SELECT FirstName, LastName  
FROM User  
WHERE UserID < 3  
ORDER BY LastName ASC
```

Result set

	FirstName	LastName
▶	John	Smith
	Andrea	Steelman

How to select data from a single table

- A **SELECT** statement is a SQL DML statement that returns a *result set* (or *result table*) that consists of the specified rows and columns.
- To specify the columns, use the **SELECT** clause.
- To specify the rows, use the **WHERE** clause.
- To specify the table that the data should be retrieved from, use the **FROM** clause.
- To specify how the result set should be sorted, use the **ORDER BY** clause.

A SELECT statement that joins two tables

Syntax

```
SELECT column-1 [,column-2] ...  
FROM table-1  
    {INNER | LEFT OUTER | RIGHT OUTER} JOIN table-2  
    ON table-1.column-1 {=<|>|<=|>=|<>} table-2.column-2  
[WHERE selection-criteria]  
[ORDER BY column-1 [ASC|DESC] [,column-2 [ASC|DESC] ...]]
```

A SELECT statement that joins two tables (cont.)

A statement that joins the User and Download tables

```
SELECT Email, DownloadFilename, DownloadDate
FROM User
    INNER JOIN Download
    ON User.UserID = Download.UserID
WHERE DownloadDate > '2014-01-01'
ORDER BY Email ASC
```

Result set

	Email	DownloadFilename	DownloadDate
▶	andi@murach.com	jr01_filter.mp3	2014-03-27 13:05:39
	joelmurach@yahoo.com	jr01_so_long.mp3	2014-03-27 13:05:39
	jsmith@gmail.com	jr01_so_long.mp3	2014-02-01 00:00:00
	jsmith@gmail.com	jr01_filter.mp3	2014-03-27 13:05:39

How to select data from multiple tables

- To return a result set that contains data from two tables, join the tables. To do that, use a JOIN clause. Most of the time, you'll want to code an *inner join* so that rows are only included when the key of a row in the first table matches the key of a row in the second table.
- In a *left outer join*, the data for all of the rows in the first table (the one on the left) are included in the table, but only the data for matching rows in the second table are included. In a *right outer join*, the reverse is true.
- An inner join is the default type of join. As a result, it's common to omit the INNER keyword from a SELECT statement for an inner join.

The INSERT statement

Syntax

```
INSERT INTO table-name [(column-list)]  
VALUES (value-list)
```

A statement that adds one row to the Download table

```
INSERT INTO Download  
    (UserID, DownloadDate, DownloadFilename, ProductCode)  
VALUES  
    (1, '2014-05-01', 'jr01_so_long.mp3', 'jr01')
```

A statement that uses MySQL's NOW function to get the current date

```
INSERT INTO Download  
    (UserID, DownloadDate, DownloadFilename, ProductCode)  
VALUES  
    (1, NOW(), 'jr01_filter.mp3', 'jr01')
```

The UPDATE statement

Syntax

```
UPDATE table-name  
SET expression-1 [, expression-2] ...  
WHERE selection-criteria
```

A statement that updates the FirstName column in one row

```
UPDATE User  
SET FirstName = 'Jack'  
WHERE Email = 'jsmith@gmail.com'
```

A statement that updates the ProductPrice column in selected rows

```
UPDATE Product  
SET ProductPrice = 36.95  
WHERE ProductPrice = 36.50
```

The DELETE statement

Syntax

```
DELETE FROM table-name  
WHERE selection-criteria
```

A statement that deletes one row from the User table

```
DELETE FROM User WHERE Email = 'jsmith@gmail.com'
```

A statement that deletes selected rows from the Downloads table

```
DELETE FROM Download WHERE DownloadDate < '2014-06-01'
```

How to insert, update, and delete data

- INSERT, UPDATE, and DELETE statements modify the data that's stored in a database, but they don't return a result set. Instead, they return the number of rows that were affected by the query.
- These statements are sometimes referred to as *action queries*.