David Ebert
October 25, 2016

Tutorial: Connecting R and MySQL

Here are the steps I used to combine R and MySQL together happily.

1. Get MySQL and the R package RMySQL.

2. Create a database in MySQL. Notice that while the syntax for `CREATE TABLE` is the same as Oracle, the data types are different. In particular, MySQL uses `TINYINT`, `SMALLINT`, etc. for integers.

```
$ mysql -u root -p -h localhost

mysql> CREATE DATABASE shop;

mysql> CREATE TABLE inventory
        (id SMALLINT PRIMARY KEY,
        name VARCHAR(50),
        quantity SMALLINT);

mysql> INSERT INTO inventory VALUES(1, 'Tomato', '10');
mysql> INSERT INTO inventory VALUES(2, 'Potato', '20');
mysql> INSERT INTO inventory VALUES(3, 'Rhubarb', '0');
mysql> INSERT INTO inventory VALUES(4, 'Eggplant', '2');
mysql> INSERT INTO inventory VALUES(5, 'Brussels Spouts', '15');
mysql> INSERT INTO inventory VALUES(6, 'Onion', '10');

mysql> SELECT * from inventory;

mysql> DESCRIBE inventory; -- This is a lot like the schema tab in Oracle SQL live
```

3. Next, we need to create a user with permission to access the database. To do that, I created a new user in MySQL.[1]

```
$ mysql -u root -p -h localhost

mysql> CREATE USER 'dangle'@'localhost' IDENTIFIED BY 'dongle';
mysql>   -- username is dangle. password is dongle.
mysql> GRANT ALL PRIVILEGES ON shop.* TO 'dangle'@'localhost';
mysql> quit;
```

[1]I'm sure it's possible to skip this step and use root user. But I had trouble with this in R. I guess this workaround is good DBA practice.

We can now access the database directly through R. But if you want to log in to MySQL as dangle, you would use the following command before entering the password:

```
$ mysql -u dangle -p -h localhost
```

4. We're now ready to use R. First, we need to create a database connection, which we'll name `dbcon`:

```
> library(RMySQL)
> dbcon = dbConnect(MySQL(), user = 'dangle',
+                  password = 'dongle',
+                  dbname = 'shop',
+                  host = '127.0.0.1')
```

5. Now we're ready. First, let's look at the tables in the database, and then the columns in the inventory.

```
> dbListTables(dbcon)

[1] "employee"  "inventory"

> dbListFields(dbcon, 'inventory')

[1] "id"       "name"     "quantity"
```

Next, let's bring the table data into R.

```
> inventory_table = dbReadTable(dbcon, "inventory")
> inventory_table

  id            name quantity
1  1          Tomato       10
2  2          Potato       20
3  3         Rhubarb        0
4  4        Eggplant        2
5  5 Brussels Spouts       15
6  6           Onion       10
```

We can go the other way, too. Let's create a new table in R, called `employee` and send that table to the `shop` database.

```
> fname = c("Alice", "Bob", "Charlie", "Dave")
> lname = c("Alvarez", "Brown", "Chaplin", "Dangle")
> employee = data.frame(lname, fname)
> employee

    lname   fname
1 Alvarez   Alice
2   Brown     Bob
3 Chaplin Charlie
4  Dangle    Dave

> dbWriteTable(dbcon, "employee",
+              employee,
+              overwrite = TRUE,
+              append = FALSE)

[1] TRUE
```

Finally, let's send queries to the database from within R. First we send a query to SQL. Then the `fetch` command imports the table into R.

```
> sql_query = dbSendQuery(dbcon,
+                         'SELECT name, quantity
+                         FROM inventory
+                         WHERE quantity > 8')
> sql_query

<MySQLResult:8,0,8>

> inventory_table = fetch(sql_query, n=-1)
> inventory_table

            name quantity
1         Tomato       10
2         Potato       20
3 Brussels Spouts      15
4          Onion       10

> on.exit(dbDisconnect(dbcon))
```