Building MySQL Applications

Introduction

This guide briefly describes the requirements for building applications that utilize MySQL databases. During this course, you will be provided access to several databases that reside on the Computer Science department’s MySQL server, located on sunapee. You will be asked to write client applications to access these databases. **You should not code or test your applications on the sunapee system; it may adversely affect the performance of the machine for other users.** Instead, you will have access, either directly or through SSH, to other CS systems, including those in Lab 101. A full list of available machines can be found at [http://www.cs.dartmouth.edu/~wbc/auditour/index.html](http://www.cs.dartmouth.edu/~wbc/auditour/index.html) (only accessible on campus or via VPN).

Using CS Systems

MySQL provides APIs for many languages, including Java, C, and Python. The CS systems’ machines have all the necessary libraries to begin writing applications in these languages. The general workflow is the same for each language: connect to a database, execute a query, and then process the results.

Java

To begin writing Java applications, you will need the MySQL Java Connector JAR file, located at `/usr/share/java/mysql-connector-java.jar`. You will need to include this JAR file on your classpath each time you compile and run your application. There are two methods to do this:

1. Use the `-cp` or `-classpath` command line option.
   
   ```bash
   javac -cp .:/usr/share/java/mysql-connector-java.jar <java-files>
   ```

2. Define the `CLASSPATH` environment variable.
   
   ```bash
   export CLASSPATH=".:/usr/share/java/mysql-connector-java.jar"
   ```

   If you add the `CLASSPATH` variable to your startup scripts—`.bash_profile`, `.bashrc`, or `.cshrc`—then it will be present for all your Java compilations.

   In any Java class in which you want to access a MySQL database, you will need to import the Java SQL package and load the MySQL driver. This is shown in the Listing 1.

   Java uses a Statement for queries and a ResultSet for the query results. The ResultSet can be seen as an iterator through the rows of the result.
import java.sql.*;

public class dbexample {
    public static final String SERVER = "jdbc:mysql://sunapee.cs.dartmouth.edu/";
    public static final String USERNAME = "user";
    public static final String PASSWORD = "pass";
    public static final String DATABASE = "db";
    public static final String QUERY = "SELECT * FROM instructor;"

    public static void main(String[] args) {
        Connection con = null;
        Statement stmt = null;
        ResultSet res = null;
        int numColumns = 0;

        try {
            // attempt to connect to db
            try {
                // load mysql driver
                Class.forName("com.mysql.jdbc.Driver").newInstance();
            } catch (Exception e) {
                // catch SQL errors
                System.err.format("SQL Error: %s", e.getMessage());
            }
            // initialize connection
            con = DriverManager.getConnection(SERVER+DATABASE, USERNAME, PASSWORD);
            System.out.println("Connection established.");
            // initialize a query statement
            stmt = con.createStatement();
            // query db and save results
            res = stmt.executeQuery(QUERY);
            System.out.format("Query executed: '%s'\nResults:\n" , QUERY);
            // the result set contains metadata
            numColumns = res.getMetaData().getColumnCount();
            // print table header
            for(int i = 1; i <= numColumns; i++) {
                System.out.format("%-12s", res.getMetaData().getColumnLabel(i));
            }
            System.out.println("\n−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−");
            // iterate through results
            while(res.next()) {
                for(int i = 1; i <= numColumns; i++) {
                    System.out.format("%-12s", res.getObject(i));
                }
                System.out.println("\n");
            }
        } catch (SQLException e) {
            // catch SQL errors
            System.err.format("SQL Error: %s", e.getMessage());
        } catch (Exception e) {
            // anything else
            e.printStackTrace();
        }

        finally {
            try {
                con.close();
                stmt.close();
                res.close();
                System.out.println("\nConnection terminated.");
            } catch (Exception e) { /* ignore cleanup errors */ }
        }
    }
}

Listing 1: Java connecting to MySQL database.

C

The header mysql.h contains the necessary definitions for connecting to MySQL databases, as seen in Listing 2. When compiling C applications, you will need certain compiler flags and libraries. These can be obtained using the following command line:

gcc -o executable `mysql_config --cflags` source.c `mysql_config --libs`
Listing 2: C connecting to MySQL database.
Python can be used to access a MySQL database by simply importing the MySQL Python Connector, as shown in Listing 3. A cursor might be a new concept to you, but it can be seen as an extension of the iterator concept. You will utilize a cursor to execute queries and access result sets.

```python
from __future__ import print_function  # make print a function
import mysql.connector  # mysql functionality
import sys  # for misc errors

SERVER = "sunapee.cs.dartmouth.edu"  # db server to connect to
USERNAME = "user"  # user to connect as
PASSWORD = "pass"  # user’s password
DATABASE = "db"  # db to user
QUERY = "SELECT * FROM instructor;"  # query statement

if __name__ == "__main__":
    try:
        # initialize db connection
        con = mysql.connector.connect(host=SERVER, user=USERNAME, password=PASSWORD, database=DATABASE)
        print("Connection established.")
        # initialize a cursor
        cursor = con.cursor()
        # query db
        cursor.execute(QUERY)
        print("Query executed: '{0}'\nResults: "{0}".format(QUERY))
        # print table header
        print("---".join(["{:<12}".format(col) for col in cursor.column_names]))
        print("---".join(["{:<12}".format(col) for col in row]))

        except mysql.connector.Error as e:
            # catch SQL errors
            print("SQL Error: {0}".format(e.msg))
        except:
            # anything else
            print("Unexpected error: {0}".format(sys.exc_info()[0]))

    except:
        print("Unexpected error: {0}".format(sys.exc_info()[0]))

    # cleanup
    con.close()
    cursor.close()

    print("\nConnection terminated.", end="")

Listing 3: Python connecting to MySQL database.

Resources