COP 2251 – Java Programming II - Databases – Chapter 34

Relational Database Systems

* A relational database stores data in tables (relations). A database is a collection of tables.
* A table consists of a list of records (think rows).
* Each record in a table has the same structure, with a fixed number of fields (think columns) of a given type.

Structured Query Language (SQL) pronounced ess-cue-ell or sequel

* SQL is the standard language for managing relational database management systems.
* SQL is used to create and manage databases and their tables.
* There are SQL commands for adding, retrieving, deleting, and updating data.
* All database products such as Oracle, Sybase, DB2, SQL Server, and MySQL support SQL.

MySQL

* MySQL is a popular open source database with millions of important installations, including Yahoo, NASA, AP, and HP. It is updated regularly, very speedy, and it is free.
* There is a great tutorial and fine documentation on the MySQL web site.
* Oracle bought MySQL. The people who made MySQL were concerned.
* They created Maria DB, a clone of MySQL, that will always be free.
* Try learning SQl at <http://www.tutorialspoint.com/mysql/index.htm>
* Also good is <https://www.w3schools.com/sql/default.asp>

Text

Description automatically generated

If you don’t see these databases, you need to watch the YouTube video to setup MariaDB.

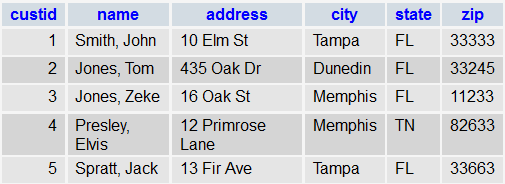
Database example has just one table, product.

Table: product



Database pcparts has several tables as below.

Table: customers



The database tables have these records just after importing.

Table: parts

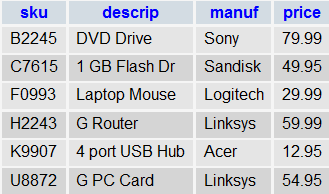
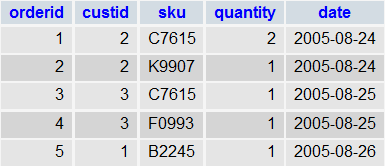


Table: orders



* Poke around but don’t make any changes.
* Try some queries. Avoid DROP because that deletes a table or database.

JDBC

* JDBCTM is a core of libraries in **java.sql** that allows you to make Structured Query Language (**SQL**) queries and updates of relational databases.
* Windows uses the Open Database Connectivity (ODBC) API for database access.
* Think of JDBC as ODBC for the Java platform.

Database Access with Java

* A database **driver,** software that handles the communication between a particular database and a Java program, is needed.
* We will use **MySQL Connector/J**, the official JDBC driver for MySQL.
* It’s in our zipped project in the lib folder.
* The driver name is **mysql-connector-java-8.0.13.jar**. (works fine with Maria)
* To add the jar file to the build path in Eclipse:
  + Create a new **source folder** called **lib** in your project folder.
  + Drag the jar file you need into this folder.
  + Refresh your project in eclipse.
  + Select the jar files, then right click and select **Build Path** -> **Add to Build Path**.

JDBC Classes

Every JDBC application uses objects from four JDBC interfaces:

1. **Driver**

Try Connect.java.

If it outputs “Connected OK”, all is well.

* Each database vendor (MySQL, Oracle) has its driver.

1. **Connection** class object

* This object provides a direct link to the database.
* The DriverManager class looks through a list of drivers for one that recognizes the specified URL.
* If it finds one, it creates a connection and passes a Connection object to the program.
* If not, the DriverManager throws a **SQLException** so try – catch blocks are needed.

Connection conn = DriverManager.getConnection( url, user, pwd );

1. **Statement** object

* The Connection object created earlier is used to create a Statement object.
* Later, a SQL statement will be assigned to this object.

Statement st = conn.createStatement();

1. **ResultSet** object

* A resultSet is one or more rows returned (in memory) by a database query.

ResultSet rSet = st.executeQuery( “SELECT \* FROM tablename” );

* A ResultSet object’s **next( )** method is used for referencing each row, one row at a time, typically in a loop.
* The ResultSet objects’ s **getString( )** method is used to access a particular field, or column, in the row.

Try SQLVehicle.java. It should output the product table’s records.

Updates, Deletes, and Inserts

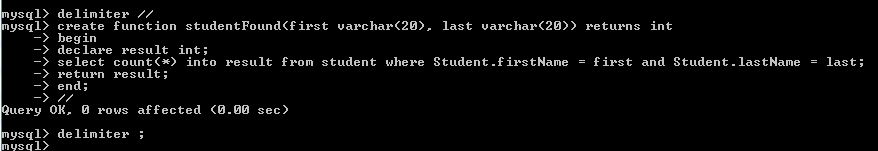
* To execute these SQL statements, use **executeUpdate( )**, not executeQuery( ).
* Whereas executeQuery( ) returns a ResultSet, executeUpdate( ) returns an **int** equal to the number of rows changed by the command.
* NOTE the nested single quotes within double quotes in the following insert statement and the **executeUpdate()** method, **not executeQuery()**.

String sql = "insert into vehicle values('" + vin + "','" + make + "','" + model + "'," + year + "," + price + )";

st.executeUpdate(sql);

Interface CallableStatement

* This interface is for executing SQL stored procedures.
* The example here requires you to create the function named **studentFound** as shown on the bottom of page 1200 and continued on page 1201. It looks like this in a command window:



Database MetaData

Try TestStoredFunction.java.

Two interfaces in the JDBC provide access to metadata:

1. **DatabaseMetaData**:
   * Access metadata regarding the database such as driver name, driver version etc.
2. **ResultSetMetaData**

* Access metadata for a result set such as count, column names etc.
* The statements below determine the number of records in the result set and output the column names:

int columnCount = resultSet.getMetaData().getColumnCount();

String row = "";

// Display column names

for (int i = 1; i <= columnCount; i++) {

row += resultSet.getMetaData().getColumnName(i) + "\t";

}

Try TestDatabaseMetaData.java, FindUserTables.java, and TestResultSetMetaData.java.