Lecture No. 43-45 – Database Connectivity

Database

A database (DB), in the most general sense, is an organized collection of data. More specifically, a database is an electronic system that allows data to be easily accessed, manipulated and updated.

In other words, a database is used by an organization as a method of storing, managing and retrieving information. Modern databases are managed using a database management system (DBMS).

Relational Database

A relational database at its simplest is a set of tables used for storing data. Each table has a unique name and may relate to one or more other tables in the database through common values.

- **Tables**

A table in a database is a collection of rows and columns. Tables are also known as entities or relations.

- **Rows**

A row contains data pertaining to a single item or record in a table. Rows are also known as records or tuples.

- **Columns**

A column contains data representing a specific characteristic of the records in the table. Columns are also known as fields or attributes.

- **Relationships**

A relationship is a link between two tables (i.e, relations). Relationships make it possible to find data in one table that pertains to a specific record in another table.

- **Datatypes**

Each of a table's columns has a defined datatype that specifies the type of data that can exist in that column. For example, the `FirstName` column might be defined as `varchar(20)`, indicating that it can contain a string of up to 20 characters. Unfortunately, datatypes vary widely between databases.

- **Primary Keys**

Most tables have a column or group of columns that can be used to identify records. For example, an `Employees` table might have a column called `EmployeeID` that is unique for every row. This makes it easy to keep track of a record over time and to associate a record with records in other tables.

- **Foreign Keys**

Foreign key columns are columns that link to primary key columns in other tables, thereby creating a relationship. For example, the `Customers` table might have a foreign key column called `SalesRep` that links to `EmployeeID`, the primary key in the `Employees` table.

- **Relational Database Management System**

A Relational Database Management System (RDBMS), commonly (but incorrectly) called a database, is software for creating, manipulating, and administering a database.

Popular Databases

- **Commercial Databases**
  - Oracle
  - SQL Server
  - DB2
- **Popular Open Source Databases**
  - MySQL
- PostgreSQL

SQL – Structured Query Languages

A high-level programming language, called *Structure Query Language* (SQL), is designed for interacting with the relational databases. SQL defines a set of commands, such as `SELECT`, `INSERT`, `UPDATE`, `DELETE`, `CREATE TABLE`, `DROP TABLE`, and etc.

Select Statement

The SELECT statement is used to select data from a database. The data returned is stored in a result table, called the result-set.

Syntax:

```
SELECT column1, column2, ...
FROM table_name;
```

If you want to select all the fields

```
SELECT * FROM table_name;
```

'*' refers to all columns

Examples:

```
SELECT CustomerName, City FROM Customers;
```

```
SELECT * FROM Customers;
```

Where Clause

The WHERE clause is used to filter records. The WHERE clause is used to extract only those records that fulfill a specified condition.

Syntax:

```
SELECT column1, column2, ...
FROM table_name
```

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
BETWEEN	Between a certain range

```
WHERE condition;
```

Examples

```
SELECT * FROM Customers
WHERE Country='Mexico';
```

```
SELECT * FROM Customers
WHERE Country='Germany' AND City='Berlin';
```

```
SELECT * FROM Customers
WHERE Country='Germany' OR City='Berlin';
```

```
SELECT * FROM Customers
WHERE NOT Country='Germany';
```

Insert Statement

The INSERT INTO statement is used to insert new records in a table.

Syntax:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

If you want to insert data in all the columns then you do not need to specify the column names:

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

Examples:

```
INSERT INTO Customers (CustomerName, ContactName, Address,
City, PostalCode, Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger',
'4006', 'Norway');
```

Update Statement

The UPDATE statement is used to modify the existing records in a table.

Syntax:

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Examples:

```
UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;
```

Delete Statement

The DELETE statement is used to delete existing records in a table.

Syntax

```
DELETE FROM table_name WHERE condition;
```

If you omit the **where clause** all records in the table will be deleted

DELETE FROM *table_name;*

JDBC

JDBC stands for **J**ava **D**ata**b**ase **C**onnectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.
The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage.

- Making a connection to a database.
- Creating SQL or MySQL statements.
- Executing SQL or MySQL queries in the database.
- Viewing & Modifying the resulting records.

JDBC is designed to make Java applications database agnostic. That is, a program written using JDBC will work with any JDBC compliant database. This means that a Java application that is tested with Apache Derby can confidently be deployed against an IBM DB2 database in production. However, there are differences between database vendors, and these differences must be abstracted away. The tool for abstracting away these differences is known as a JDBC driver.

Setting up MySql Database

The general MySQL Installer download is available at https://dev.mysql.com/downloads/windows/installer/. The MySQL Installer application can install, upgrade, and manage most MySQL products, including MySQL Workbench.

Things you need to install:

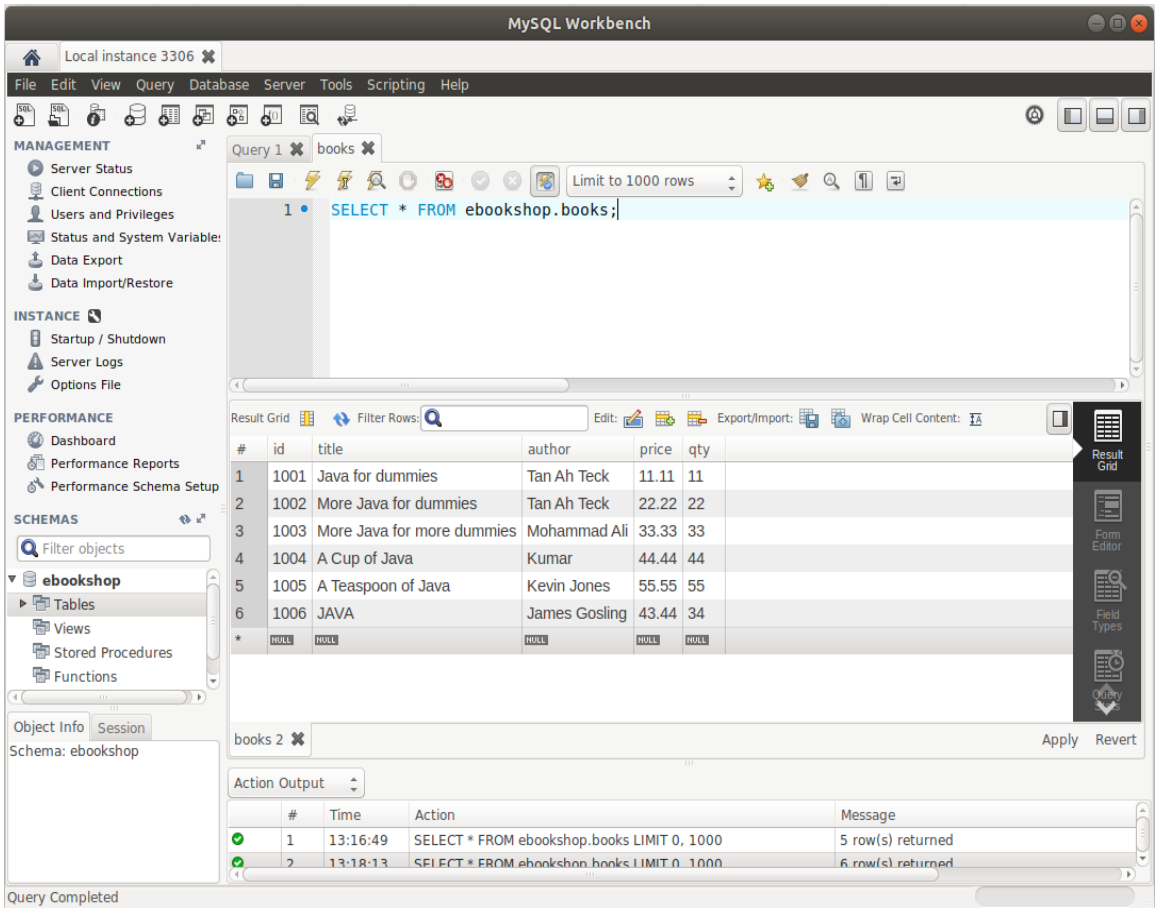
1. MySQL Server

2. Connector/J

   Standardized database driver for Java platforms and development.

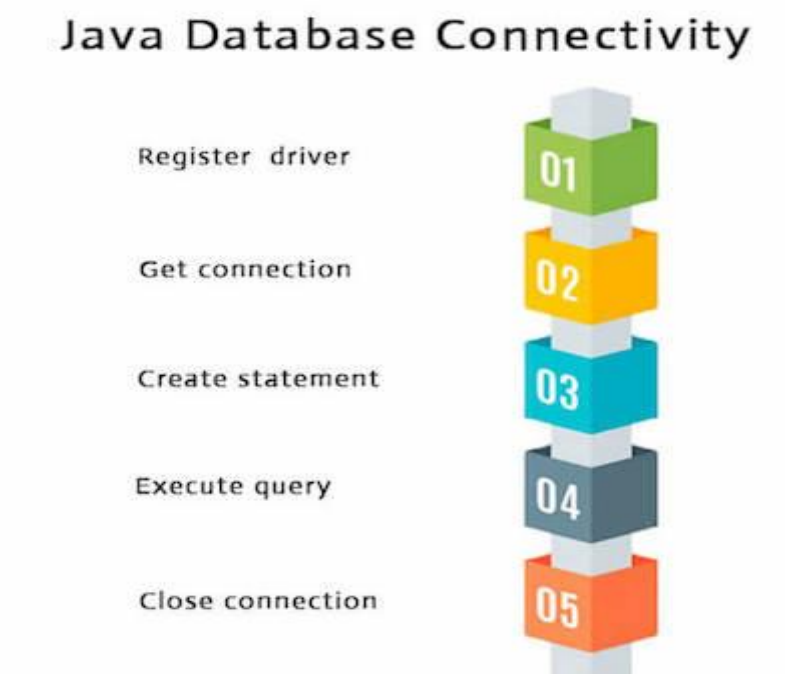
3. MySQL Workbench

MySQL Workbench provides DBAs and developers an integrated tools environment for Database Administration.



How to connect to a database with JDBC

The basic steps to connect to a JDBC database are:



1. **Register Driver**

The **forName()** method of Class class is used to register the driver class. This method is used to dynamically load the driver class.

Syntax:

```
Class.forName(String DriverClass);
```

Example:

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

2. **Get Connection**

The **getConnection()** method of DriverManager class is used to establish connection with the database.

Syntax:

```
Connection conn = DriverManager.getConnection( String url,
                                  String user, String password);
```

Example:

```
Connection conn = DriverManager.getConnection(
 "jdbc:mysql://localhost:3306/ebookshop", "root", "javadb");
```

3. **Create Statement**

The createStatement() method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database.

Syntax:

```
Statement stmt=con.createStatement();
```

4. **Execute Query**

The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.

Syntax:

```
stmt.executeQuery(String sql)
```

Example:

```
ResultSet rs=stmt.executeQuery("select * from emp");

while(rs.next()){
```

System.out.println(rs.getInt(1)+" "+rs.getString(2));}

5. **Close Connection**

conn.close()

Example Code

```
import java.sql.*;

public class JdbcSelectTest {
   public static void main(String[] args) {
      try {
         //Step 1: Register Driver
         Class.forName("com.mysql.jdbc.Driver");

         // Step 2: Allocate a database 'Connection' object
         Connection conn = DriverManager.getConnection(
               "jdbc:mysql://localhost:3306/ebookshop",
               "root", "javadb");   // For MySQL only
               // The format is: "jdbc:mysql://hostname:port/databaseName", "username", "passw

         // Step 3: Allocate a 'Statement' object in the Connection
         Statement stmt = conn.createStatement();
         // Step 3: Execute a SQL SELECT query.
         //The query result is returned in a 'ResultSet' object.
         String strSelect = "select title, price, qty from books";
         System.out.println("The SQL statement is: " + strSelect + "\n");
          // Step 4: Execute Query.
         ResultSet rset = stmt.executeQuery(strSelect);

         //  For each row, retrieve the contents of the cells with getXxx(columnName).
         System.out.println("The records selected are:");
         int rowCount = 0;
         while(rset.next()) { // Move the cursor to the next row, return false if no more row
            String title = rset.getString("title");
            double price = rset.getDouble("price");
            int    qty   = rset.getInt("qty");
            System.out.println(title + ", " + price + ", " + qty);
            ++rowCount;
         }
         System.out.println("Total number of records = " + rowCount);

      } catch(Exception ex) {
            System.out.println(ex);
            ex.printStackTrace();
      }  // Step 5: Close conn and stmt - Done automatically by try-with-resources (JDK 7)
   }
}
```

Compile

javac classpath path-to-jdbc-jar-file;path-to-java-file javafile.java

javac -classpath D:/java/mysql-connector-java-5.1.48-bin.jar;.
JdbcSelectTest.java

Run

java classpath path-to-jdbc-jar-file;path-to-java-file javafile

java -classpath D:/java/mysql-connector-java-5.1.48-bin.jar;. JdbcSelectTest.java

Output:

```
The SQL statement is: select title, price, qty from books

The records selected are:
Java for dummies, 11.11, 11
More Java for dummies, 22.22, 22
More Java for more dummies, 33.33, 33
A Cup of Java, 44.44, 44
A Teaspoon of Java, 55.55, 55
JAVA, 43.44, 34
Total number of records = 6
```

Class Tasks

Task # 01: Create a database StudentDB and a table Students with columns ROLL_NUM, NAME, BATCH, DEPT, YEAR SEMESTER using MySQL Workbench

Task # 02: Write Java code that connects to the StudentDB and inserts 5 records the Student Table.

Task # 03: Now add code for displaying records of students that are enrolled in semester 2

Task # 04: Write java code to update the record of a student whose ROLL_NUM is equal to your roll number

Task # 05: Write java code to delete the records of all students of first year