09

DATABASE IN JAVA

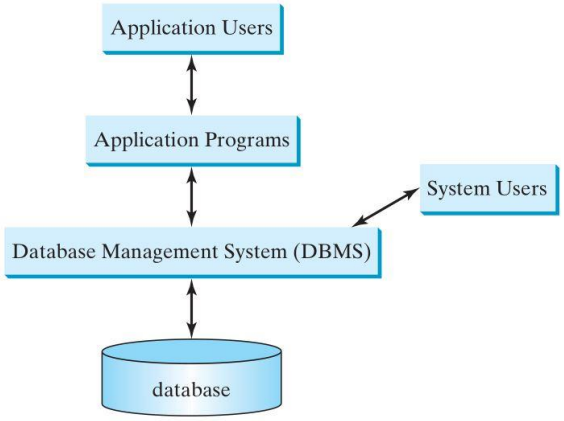
OBJECT ORIENTED PROGRAMMING LAB

SESSION - 09

CSE215 | DAFFODIL INTERNATIONAL UNIVERSITY

Relational Database Systems

A database is a repository of data that form information. A database system consists of a database, the software that stores and manages data in the database, and the application programs that present data and enable the user to interact with the database system.



What is API

API (Application programming interface) is a document that contains description of all the features of a product or software. It represents classes and interfaces that software programs can follow to communicate with each other. An API can be created for applications, libraries, operating systems, etc

What is 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.

SQL Syntax

Structured Query Language (SQL) is a standardized language that allows you to perform operations on a database, such as creating entries, reading content, updating content, and deleting entries.

SQL is supported by almost any database you will likely use, and it allows you to write database code independently of the underlying database.

SQL COMMANDS:

1. **Create Database**

```
SQL> CREATE DATABASE DATABASE_NAME;
```

2. **Drop Database**

```
SQL> DROP DATABASE DATABASE_NAME;
```

3. **Create Table**

```
SQL> CREATE TABLE table_name
(
   column_name column_data_type,
   column_name column_data_type,
   column_name column_data_type
   ...
);
```

4. **INSERT Data**

```
SQL> INSERT INTO table_name VALUES (column1, column2, ...);
```

5. **SELECT Data**

```
SQL> SELECT column_name, column_name, ...
     FROM table_name
     WHERE conditions;
```

6. **UPDATE Data**

```
SQL> UPDATE table_name
     SET column_name = value, column_name = value, ...
     WHERE conditions;
```

7. **DELETE Data**

```
SQL> DELETE FROM table_name WHERE conditions;
```

Creating Database

1. Make sure XAMPP is installed and Running MySQL server.
2. Change the directory to 'mysql' folder through commandline:
    a. Open command line with adminstritive privilege.
    b. Type the following commands:

```
C:\Windows\system32> cd C:?
C:\Windows\System32> cd C:/
C:\>cd xampp
C:\xampp>cd mysql
C:\xampp\mysql\bin>mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 16
Server version: 5.6.24 MySQL Community Server (GPL)
Copyright (c) 2000, 2015, 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.
```

3. Get MySQL root access:

```
C:\xampp\mysql\bin>mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 17
Server version: 5.6.24 MySQL Community Server (GPL)
Copyright (c) 2000, 2015, 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.+--------------------+
| information_schema |
+--------------------+
1 row in set (0.00 sec)
```

4. Show Default Databases:

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| cdcol              |
| mysql              |
| performance_schema |
| phpmyadmin         |
| webauth            |
+--------------------+
6 rows in set (0.00 sec)
```

5. Create a database named student:

```
mysql> create database student;
Query OK, 1 row affected (0.00 sec)
```

6. Show the databases again and check if students is created:

```
SQL> mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| cdcol              |
| mysql              |
| performance_schema |
| phpmyadmin         |
| student            |
| webauth            |
+--------------------+
```

7. Select `student database:`

```
mysql> use student
Database changed
```

8. Create a table named ‘Student’.

```
mysql> create table student(
    -> id char(13) not null,
    -> name char(60),
    -> cgpa float);
Query OK, 0 rows affected (0.47 sec)
```

9. Check if the table was created:

```
mysql> show tables;
+------------------+
| Tables_in_student |
+------------------+
| student           |
+------------------+
1 row in set (0.00 sec)
```

10. Show ‘Students’ table:

```
mysql> desc student;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| id    | char(13) | NO   |     | NULL    |       |
| name  | char(60) | YES  |     | NULL    |       |
| cgpa  | float    | YES  |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

11. Insert data into table:

```
mysql> insert into student(id,name,cgpa) values('101-15-007','James Bond',4.0);
Query OK, 1 row affected (0.12 sec)
```

12. Show all the data from the table;

```
mysql> select * from student;
+------------+------------+------+
| id         | name       | cgpa |
+------------+------------+------+
| 101-15-007 | James Bond |    4 |
+------------+------------+------+
1 row in set (0.00 sec)
```

13. Insert 3 more data:

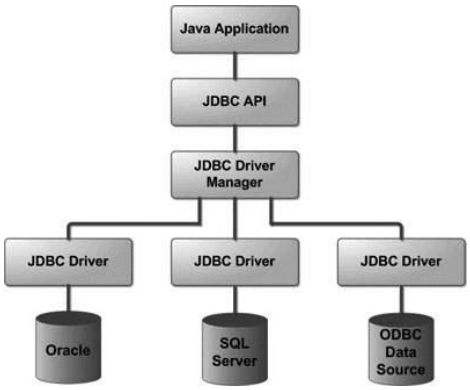
```
mysql> insert into student(id,name,cgpa) values('101-15-008','Rustom Ali', 3.8);
Query OK, 1 row affected (0.06 sec)
mysql> insert into student(id,name,cgpa) values('101-15-009','Johora Begum', 3.9
9);
Query OK, 1 row affected (0.06 sec)
mysql> insert into student(id,name,cgpa) values('101-15-010','Mohd. Ali Clay', 3
.7);
Query OK, 1 row affected (0.08 sec)
```

14. Show the whole table:

```
mysql> select * from student;
+------------+----------------+------+
| id         | name           | cgpa |
+------------+----------------+------+
| 101-15-007 | James Bond     |    4 |
| 101-15-008 | Rustom Ali     |  3.8 |
| 101-15-009 | Johora Begum   | 3.99 |
| 101-15-010 | Mohd. Ali Clay |  3.7 |
+------------+----------------+------+
4 rows in set (0.00 sec)
```

Common JDBC Components

The JDBC API provides the following interfaces and classes –



- **DriverManager:** This class manages a list of database drivers. Matches connection requests from the java application with the proper database driver using communication sub protocol. The first driver that recognizes a certain subprotocol under JDBC will be used to establish a database Connection.

- **Driver:** This interface handles the communications with the database server. You will interact directly with Driver objects very rarely. Instead, you use DriverManager objects, which manages objects of this type. It also abstracts the details associated with working with Driver objects.

- **Connection:** This interface with all methods for contacting a database. The connection object represents communication context, i.e., all communication with database is through connection object only.

- **Statement:** You use objects created from this interface to submit the SQL statements to the database. Some derived interfaces accept parameters in addition to executing stored procedures.

- **ResultSet:** These objects hold data retrieved from a database after you execute an SQL query using Statement objects. It acts as an iterator to allow you to move through its data.

- **SQLException:** This class handles any errors that occur in a database application.

Connecting Database:

The JDBC API consists of classes and interfaces for establishing connections with data- bases, sending SQL statements to databases, processing the results of the SQL statements, and obtaining database metadata. Four key interfaces are needed to develop any database application using Java:

- Driver
- Connection
- Statement
- ResultSet

These interfaces define a framework for generic SQL database access. The JDBC API defines these interfaces. The JDBC driver vendors provide implementation for them. Programmers use the interfaces. The relationship of these interfaces is shown in Figure 3. A JDBC application loads an appropriate driver using the Driver interface, connects to the database using the Connection interface, creates and executes SQL statements using the Statement interface, and processes the result using the ResultSet interface if the statements return results.

5 Steps to connect to the database in java

There are 5 steps to connect any java application with the database in java using JDBC. They are as follows:

- Register the driver class
- Creating connection
- Creating statement
- Executing queries
- Closing connection

1) Register the driver class

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

Syntax of forName() method

```
public static void forName(String className)throws ClassNotFoundException
```

Example to register the OracleDriver class

```
Class.forName("com.mysql.jdbc.Driver");
```

---

2) Create the connection object

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

Syntax of getConnection() method

```
1) public static Connection getConnection(String url)throws SQLException
2) public static Connection getConnection(String url,String name,String password)
throws SQLException
```

Example to establish connection with the Oracle database

```
Connection con = DriverManager.getConnection(
"jdbc:mysql://localhost:3306/sonoo","root","root");
```

---

3) Create the Statement object

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

Syntax of createStatement() method

```
public Statement createStatement()throws SQLException
```

Example to create the statement object

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

---

4) Execute the 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 of executeQuery() method

```
public ResultSet executeQuery(String sql)throws SQLException
```

Example to execute query

```
ResultSet rs=stmt.executeQuery("select * from emp");
while(rs.next())
System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));
```

---

5) Close the connection object

By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection.

Syntax of close() method

```
public void close()throws SQLException
```

Example to close connection

```
con.close();
```

Example to connect to the mysql database in java

For connecting java application with the mysql database, you need to follow 5 steps to perform database connectivity.

In this example we are using MySql as the database. So we need to know following informations for the mysql database:

1. **Driver class:** The driver class for the mysql database is **com.mysql.jdbc.Driver**.
2. **Connection URL:** The connection URL for the mysql database is **jdbc:mysql://localhost:3306/sonoo** where jdbc is the API, mysql is the database, localhost is the server name on which mysql is running, we may also use IP address, 3306 is the port number and sonoo is the database name. We may use any database, in such case, you need to replace the sonoo with your database name.
3. **Username:** The default username for the mysql database is **root**.
4. **Password:** Password is given by the user at the time of installing the mysql database. In this example, we are going to use root as the password.

Let's first create a table in the mysql database, but before creating table, we need to create database first.

```
1. create database sonoo;
2. use sonoo;
3. create table emp(id int(10),name varchar(40),age int(3));
```

Example to Connect Java Application with mysql database

In this example, sonoo is the database name, root is the username and password.

```
1. con. import java.sql.*;
2. class MysqlCon{
3. public static void main(String args[]){
4. try{
5. Class.forName("com.mysql.jdbc.Driver");
6. Connection con=DriverManager.getConnection(
7. "jdbc:mysql://localhost:3306/sonoo","root","root");
8. //here sonoo is database name, root is username and password
9. Statement stmt=con.createStatement();
10. ResultSet rs=stmt.executeQuery("select * from emp");
11. while(rs.next())
12. System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));
13. con.close();
14. }catch(Exception e){ System.out.println(e);}
15. }
16. }
```

"To connect java application with the mysql database mysqlconnector.jar file is required to be loaded."

download the jar file mysql-connector.jar

Two ways to load the jar file:

1. paste the mysqlconnector.jar file in jre/lib/ext folder
2. set classpath

1) paste the mysqlconnector.jar file in JRE/lib/ext folder:

Download the mysqlconnector.jar file. Go to jre/lib/ext folder and paste the jar file here.

2) set classpath:

There are two ways to set the classpath:

- temporary
- permanent

How to set the temporary classpath

open command prompt and write:

1. C:>set classpath=c:\folder\mysql-connector-java-5.0.8-bin.jar;.;

How to set the permanent classpath

Go to environment variable then click on new tab. In variable name write **classpath** and in variable value paste the path to the mysqlconnector.jar file by appending mysqlconnector.jar;.; as C:\folder\mysql-connector-java-5.0.8-bin.jar;.;