CS205 Object Oriented Programming in Java

Module 5 - **Graphical User Interface and Database support of Java**

(Part 6)

Prepared by

Renetha J.B.

AP

Dept.of CSE,

Lourdes Matha College of Science and Technology

Topics

- ☑ **JDBC overview**
  - ☑ Creating and Executing Queries
    - ☑ create table
    - ☑ delete
    - ☑ Insert
    - ☑ select

Introduction

- Programming Language -**Java**
  - for coding, developing interfaces(**front end** of an application)
- Database – **MySQL , Oracle** , **PostgreSQL**
  - For storing data- (**back end** of an application)
    - Different types- relational database, object based database etc.
  - Relational database
    - Tabular structure
    - **E.g.** - Oracle, Microsoft Access, MySQL, PostgreSQL, MongoDB etc
- SQL- Structured Query Language
  - **SQL** statements are used to perform operations on a database. We can insert , delete, update and retrieve data in database using SQL statements.

JDBC overview

- JDBC stands for "**Java DataBase Connectivity**".
- JDBC API (*Application **Programming** **Interface***) is a Java API that can access any kind of **tabular data**, especially data stored in a ***Relational database*.**
- JDBC is used for executing SQL statements from Java program.
- With the help of JDBC API, we can insert, update, delete and fetch data from the database.

Why JDBC?

- Before JDBC, **ODBC**(**O**pen **D**ata**B**ase **C**onnectivity) API was the database API to connect and execute the query with the database.
  - But, ODBC API uses ODBC driver which is written in C language (i.e. *platform dependent and unsecure*).
  - That is why, Java has defined its own API (**JDBC API**) that uses JDBC drivers (*written in Java language*).
- If our application is using JDBC API to interact with the database, then we need not change much in our code even if we change the database of our application.

Advantage of JDBC

- JDBC **standardizes** how to do many of the operations like
  - connect to the database,
  - query the database,
  - update the database, and
  - call stored procedures.

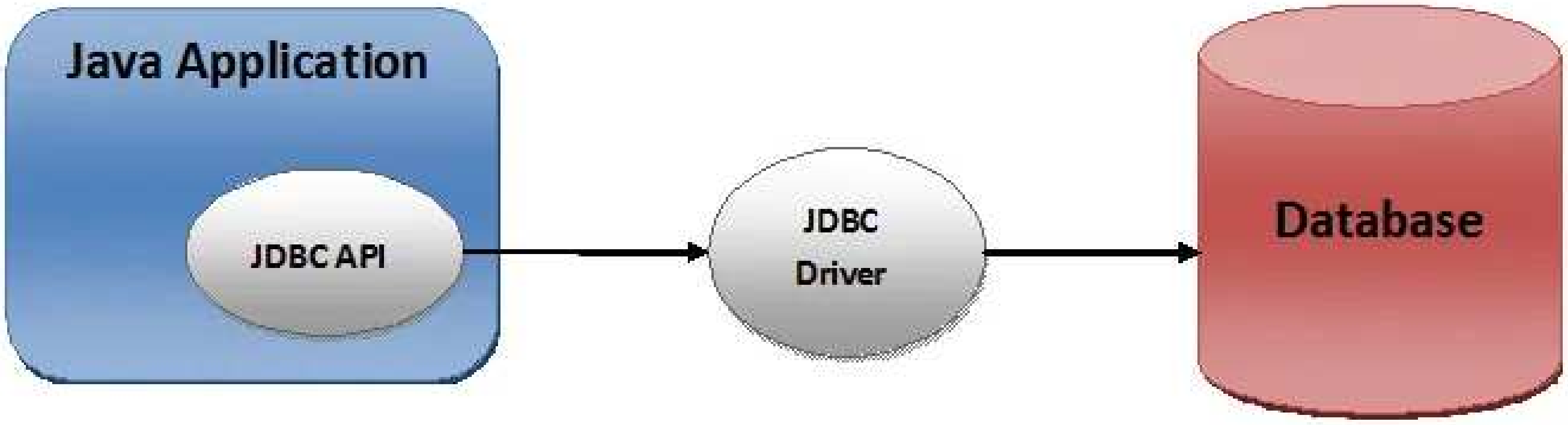
JDBC architecture

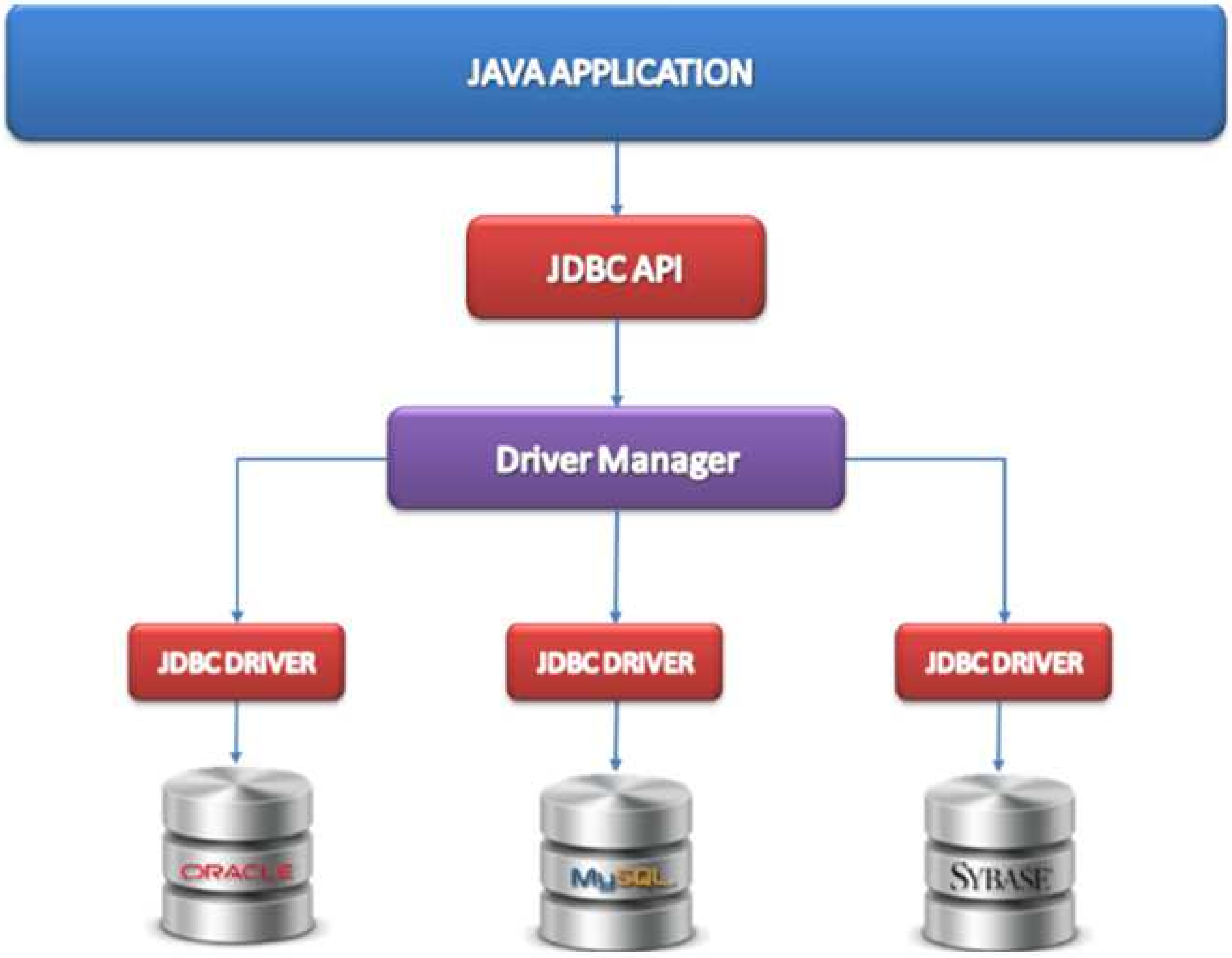
- **JDBC** architecture can be classified in 2 broad categories:-

1. JDBC API
2. JDBC Drivers

Java-JDBC API -JDBC Driver- Database

- **Java** – programming language – coding- Front end
- The **JDBC API** *defines a set of interfaces and classes that all major database providers* follow, so that using JDBC API, Java developers can connect to many Relational Database Management Systems (RDBMS).
  - **JDBC API** uses **JDBC drivers** to connect with the database.
- A **JDBC driver** is a software component that enables a Java application to interact with specific database.
- **Database** – store data – Back end


Java Application
JDBC API
JDBC
Driver
Database


JAVA APPLICATION
JDBC API
Driver Manager
JDBC DRIVER
JDBC DRIVER
JDBC DRIVER
ORACLE
MySQL
SYBASE

JDBC API

- The Java Database Connectivity (JDBC) API provides universal data access from the Java programming language
- Using the **JDBC API**, we can access virtually any data source like relational databases , spreadsheets etc. from Java.
- The JDBC API is comprised of two packages:

  java.sql

  javax.sql
- These packages contains classes and interfaces for JDBC API.

JDBC API

- Some classes and interfaces in **java.sql** package which support connectivity between Java and database are:
  - **DriverManager** :-"DriverManager class" **manages** all the Drivers found in JDBC environment, *load the most appropriate driver* for connectivity.
  - **Connection**: Connection interface objects which represents connection and it's object also helps in *creating object of Statement,* PreparedStatement etc.
  - **Statement** : :-Statement interface object is used to *execute query* and also store it's value to "ResultSet" object.

JDBC API(contd.)

- **PreparedStatement**:- represents a *precompiled SQL statement* .
- **Callable Statement**:-Callable statement *support stored procedure* .
- **ResultSet**: it is used to store the result of SQL query. Java application get the result of database from this ResultSet.
- **SQLException**:- SQLException class is used to represent *error or warning* during access from database or during connectivity.

JDBC Drivers

- **JDBC API** uses **JDBC drivers** to connect with the database.
- JDBC drivers .All major vendors provide their own JDBC drivers .
- A **JDBC driver** is a software component that enables a Java application to interact with a database.
- JDBC drivers contain a set of java classes that enables to connect to that particular database.

JDBC Drivers

Types Of JDBC Drivers:

JDBC drivers are divided into four types or levels.

Type 1:

- JDBC-ODBC Bridge driver (**Bridge**)

Type 2:

- Native-API/partly Java driver (**Native**)

Type 3:

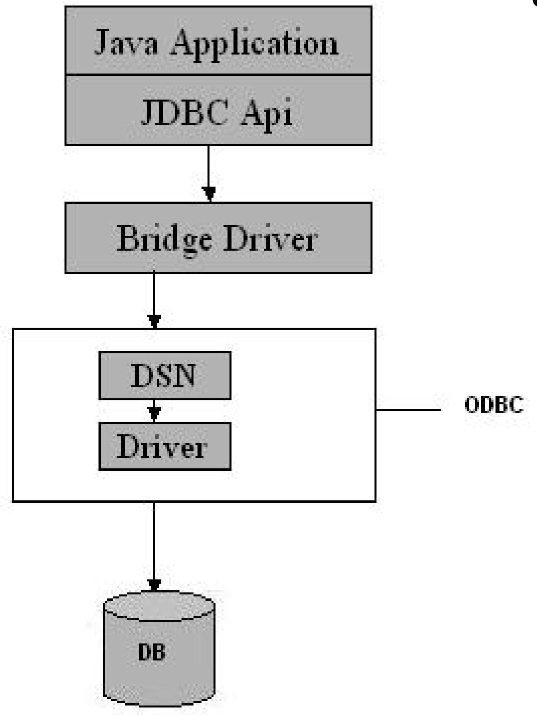
- All Java/Net-protocol driver (**Middleware**)

Type 4:

- All Java/Native-protocol driver /Thin Driver (**Pure**)

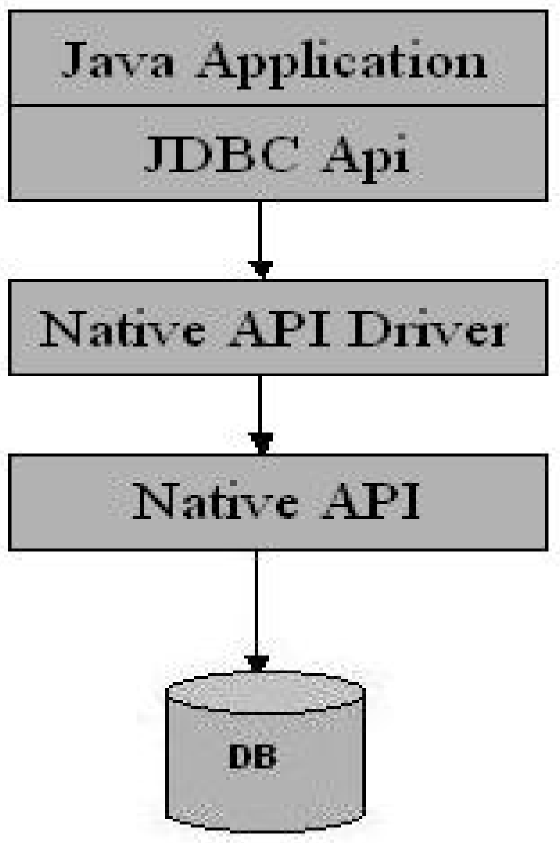
Type 1 JDBC Driver

- The Type 1(**<u>JDBC-ODBC Bridge driver)</u>** driver translates all JDBC calls into ODBC calls and sends them to the ODBC driver
- **Advantage**
  - The JDBC-ODBC Bridge allows access to almost any database, since the database's ODBC drivers are already available.



Type 2 JDBC Driver

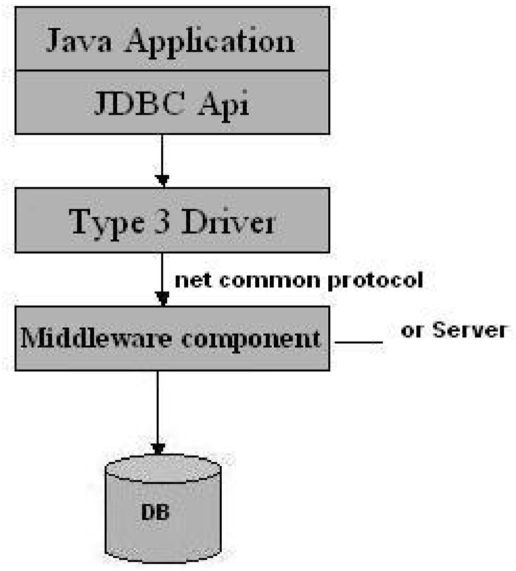
- Type 2 drivers(**Native-API/partly Java driver**) convert JDBC calls into database-specific calls.



Type 3 JDBC Driver

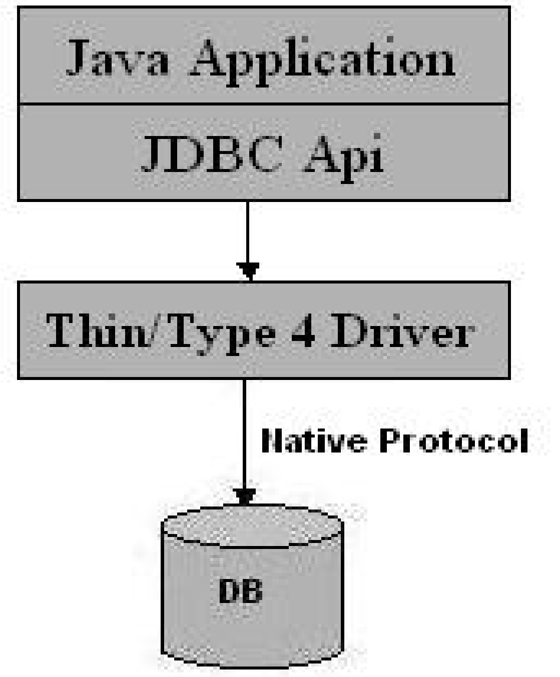
- Type 3 database driver(<u>**All Java/Net-protocol driver**</u>) requests are passed through the network to the middle-tier server.
- The middle-tier then translates the request to the database.

Advantage
This driver is *fully written in Java* and hence *portable*.
It is *suitable for the web*.



Type 4 JDBC Driver

- Type 4 drivers(**Native-protocol/all-Java driver)** uses java networking libraries to communicate directly with the database server.
- Advantage
  - They are completely written in Java-platform independent.
  - It is most suitable for the web.



Java-Database connectivity

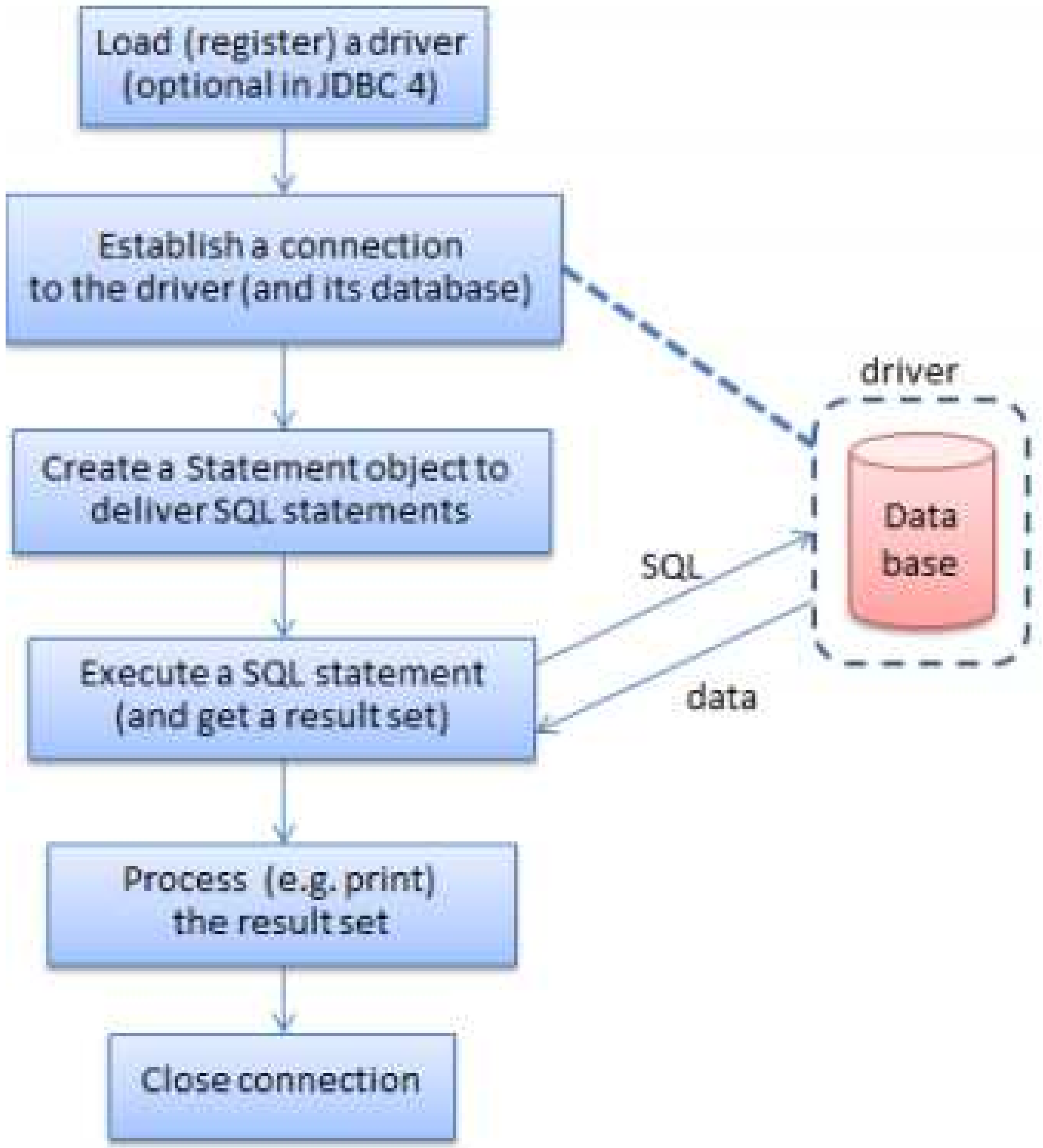
Basic Steps fo connecting Java and database

- Before we can create a java JDBC connection to the database, we must first import the **java.sql package** using:-

import java.sql.*;

Steps for connecting Java and database are:

- Load and register a database driver
- Establish(create) a connection to the database
- Create Statement object
- Execute the SQL Statements
- Process the result
- Close the connection and Statement


Load (register) a driver (optional in JDBC 4)
Establish a connection to the driver (and its database)
Create a Statement object to deliver SQL statements
Execute a SQL statement (and get a result set)
Process (e.g. print) the result set
Close connection
driver
Data base
SQL
data

- **Steps to develop JDBC application:**
  1. Load and Register Driver
  2. Establish the connection between java application and database
  3. Creation of statement object
  4. Send and execute SQL query
  5. Process Result from ResultSet
  6. Close the connection and statement.

Java Database Connectivity steps

1. **Load or register a database driver**

- We can load /register a driver in Java in one of two ways:
  - ❑ **Class.forName(String *driver*)**
  - ❑ **DriverManager.registerDriver(new *constructor of the driver*)**
- We can load the driver class by calling **Class.forName()** with the Driver class name as an *argument* or **DriverManager.registerDriver()** with *constructor of the driver class as argument*
  - Once loaded, the Driver class creates an instance of itself.
  - JDBC-ODBC Bridge driver is commonly used.
- Each database has its own driver.
- The JDBC Driver class for MySQL database are

  com.mysql.jdbc.Driver

  com.mysql.cj.jdbc.Driver

Java Database Connectivity

- The code for **loading MySQL database driver from Java**

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

or

```
DriverManager.registerDriver(new com.mysql.cj.jdbc.Driver());
```

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

Or

```
DriverManager.registerDriver(new com.mysql.jdbc.Driver());
```

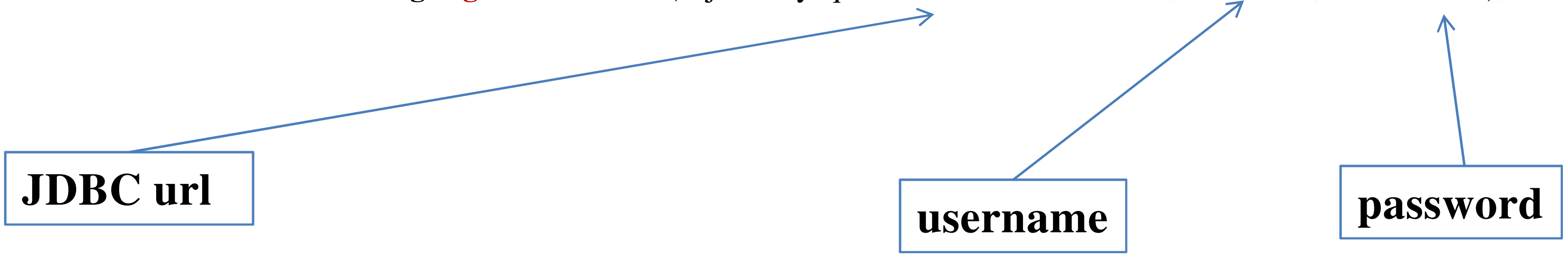
Java Database Connectivity(contd.)

2. Establish the connection between Java application and database

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

public static Connection getConnection(String ***url*** ,String ***name***, String ***password***) **throws SQLException**

```
Connection con=DriverManager.getConnection ( "jdbc:mysql://localhost:3306/ABC" , "root" , "root123" );
```



Java Database Connectivity(contd.)

- A JDBC URL provides a way of identifying a database so that the appropriate driver will recognize it and establish a connection with it.

The standard syntax for JDBC URLs is:

jdbc:<***subprotocol***>:<***subname***>

- A JDBC URL has three parts, which are separated by colons:
  - **jdbc** is the protocol.
  - <**subprotocol**> is usually the **driver** or the database connectivity mechanism, which may be supported by one or more drivers.
  - <**subname**> is the database.
  - For example, to access a MySQL database through a JDBC-ODBC bridge, one might use a URL such as the following:

jdbc:mysql://localhost:3306/ABC

Java Database Connectivity(contd.)

Connection con=**DriverManager**.**getConnection**(
"**jdbc:mysql://localhost:3306/databasename**" , "**username**", "**password**");

- ❑ The JDBC url used is

jdbc:mysql://localhost:3306/databasename

- Here **jdbc** is the API,
- **mysql** is the database,
- **localhost** is the server name on which **mysql** is running(we can give IP address here)
- **3306** is the port number
- **databasename** is the name of the database created in MySQL

- ❑ **username** Give the username of MySQL(root is the default username. Other MySQL username also we can give here).
- ❑ **password** Give the password of MySQL login given during installation.

Java Database Connectivity(contd.)

3. Creating a Statement object

- Once a connection is established, we can interact with the database.
- To execute SQL statements, we need to create a Statement object from the Connection object by using the createStatement() method.
- A Statement object is used to send and execute SQL statements to a database.

Statement object= connectionobject. **createStatement();**

E.g.

Connection con=**DriverManager**.**getConnection**("jdbc:mysql://localhost:3306/ABC" , "**root**" , "**root123**");

Statement st = con.createStatement();

- Statement object **st** is used to **send and execute SQL statements** to a database.

Java Database Connectivity(contd.)

4. Execute the SQL Statements

- To execute a SQL statement we use **executeQuery()** or **executeUpdate() method** on **Statement** object**.**
  - ❑ The **executeUpdate()** method <u>executes the **CREATE, INSERT, DELETE** and **UPDATE**</u> statements.
    - E.g.:

    st.executeUpdate("CREATE TABLE stud1(roll int, name varchar(15))");

    st.executeUpdate(“INSERT INTO student values(1, ’Anu’) ”);

  - ❑ The ***executeQuery***() method <u>executes a **SELECT** query</u>
    - it takes an SQL SELECT query string as an argument and returns the result(output) as a ***ResultSet*** object.
    - *E.g.:*

    ResultSet rs=st***.executeQuery***(“SELECT rollno, name from student”);

Java Database Connectivity(contd.)

5. Process the Result (in the case of SELECT query only -To Retrieve the Result)

- The result of SELECT query (retrieves data from database) is stored in **ResultSet** object.
- To retrieve the data from the ResultSet object, we have to use ResultSet's getxxx() method (where xxx is the data type.)

  public int getxxx(int columnIndex):

  public int getxxx(Stringt columnIndex):

  - getInt() to retrieve a integer value.
  - getString() method can be used to retrieve a string value.
  - getFloat() method can be used to retrieve a floating point value.

Java Database Connectivity(contd.)

Eg:

```
ResultSet   rs=st.executeQuery("SELECT   rollno,   name
                        from   student");
while(rs.next())
{
        System.out.println(rs.getInt(1));
        System.out.println(rs.getString(2));
}
```

– Here rs contains all rollno and name rows in studenttable

– **ResultSet cursor** is <u>initially positioned **before the first row**</u>**;**

- **the first call to the method next()** (rs.next()) <u>moves the cursor forward and makes the first row the current row)</u>
- the second call to next() <u>moves the cursor forward and makes the second row as the current row</u>, and so on.

Java Database Connectivity(contd.)

- rs.getInt(1) will retrieve the value of first attribute in the current row(getInt() is used because first attribute is rollno which is integer)
- rs.getString(2) will retrieve the value of second attribute in the current row(getString() is used because second attribute is name which is character string)

Java Database Connectivity(contd.)

```
ResultSet rs=st.executeQuery("SELECT rollno, name from   student");
          while(rs.next())
          {
                    System.out.println(rs.getInt(1));
          System.out.println(rs.getString(2));
          }
```

Working:

- Here rollno and name in all rows in student table are retrieved.
- **ResultSet** object **rs** maintains a cursor that is initially positioned **before the first row.**
- When while(rs.**next()**) is first executed the cursor moves forward to first row of result and prints the value of first attribute (rollno) and second attribute(name)in first row in the result
- Next time while loop is executed, if second row is there, then rs.nex() moves cursor to second row and prints the values in thar row.
- This continues until there is no more row in the result.

Java Database Connectivity(contd.)

6. Close the connection and Statement

- Finally open connections need to be closed using **close()** method as:

con.**close()**

st. **close**()

MySQL

- MySQL is an open source database software.
  - We can create databases, tables etc for storing data and we can perform various database operations.
- Take mysqlshell and type:

\sql

\connect root@localhost

Enter password **root123** **(password given during installation)**

- Create database

CREATE DATABASE ABC;

- Use the database for creating tables

USE ABC;

- Now you can do database operations in MySQL

SQL Commands- CREATE TABLE

- **CREATE TABLE** tablename (atribute1 type, attribute2 type, attribute3 type Primary Key,….. atributen type,***constraints***);
  - Type can be

    int

    char(size)

    varchar(size)

    real

    date
- E.g. CREATE TABLE person(name varchar(15) , age int);

SQL Commands- INSERT

- Insert rows(values of attribute) into table.
- **INSERT INTO** tablename (attribute,attribute..) **VALUES**(value1,value2,….);
- If type of attribute is varchar or char its value should be enclose in single quotes.
- E.g. Insert the following details into Person table
- Name is Anu Age 20, Name is Smith Age 10 ,

  Name is Roy Age 70

INSERT INTO Person(name,age) VALUES(‘Anu’, 20);

INSERT INTO Person(name,age) VALUES(‘Smith’, 10);

INSERT INTO Person(name,age) VALUES(‘Roy’, 70);

SQL Commands- DELETE

- Delete from table.

DELETE FROM tablename **WHERE** condition;

- Condition can be of the form:

  Attribute=value   Attribute<value   Attribute>value

  Attribute<=value   Attribute>=value   Attribute!=value

  Attribute BETWEEN value 1AND value2

  – If more than one condition is there, then they can be combined using AND , OR as required.

E.g. . *Remove details of persons having age 20*

DELETE FROM Person WHERE age=20;

Remove details of persons having name Smith or age more than 60

DELETE FROM Person where age>60 OR name='Smith';

SQL Commands - UPDATE

- To update or modify the contents in table.

  UPDATE tablename **SET** attribute=value , attribute=value **WHERE** condition;

- Condition can be of the form:

  Attribute**=**value Attribute**<**value Attribute**>**value

  Attribute**<=**value Attribute**>=**value Attribute**!=**value

  Attribute BETWEEN value 1AND value2

  – If more than one condition is there, then they can be combined using AND , OR as required.

E.g. *Change the age of Roy to 25*

UPDATE Person SET age=25 WHERE name='Roy';

SQL Commands - SELECT

- To select or retrieve content from table.

  SELECT attribute1, attribute2,….. attributen **FROM** tablename **WHERE** condition;

- Condition can be of the form:

  Attribute**=**value Attribute**<**value Attribute**>**value

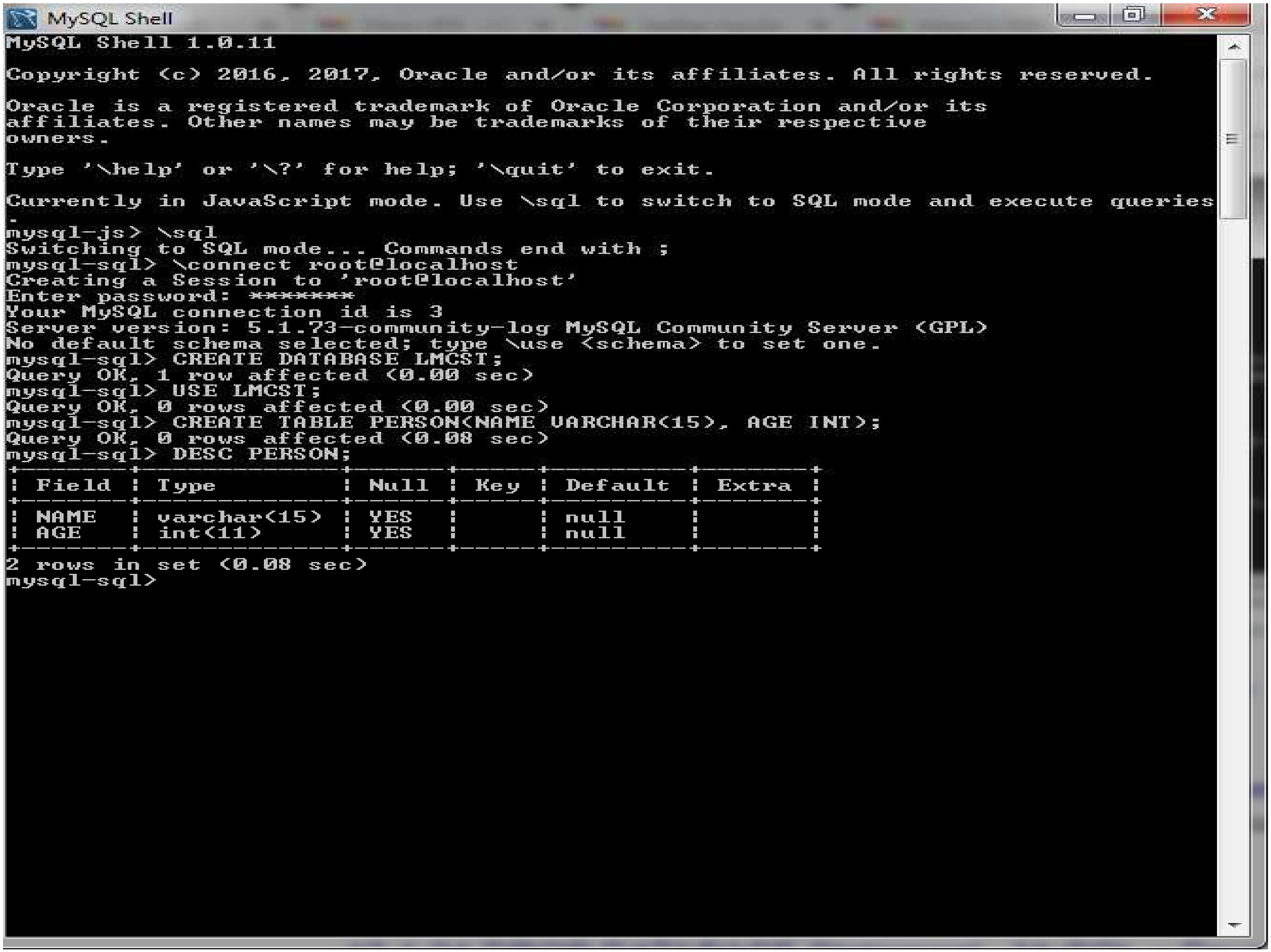
  Attribute**<=**value Attribute**>=**value Attribute**!=**value

  Attribute BETWEEN value 1AND value2

  – If more than one condition is there, then they can be combined using AND , OR as required.

E.g. *Display name and age of persons with age more than 10.*

SELECT name, age FROM Person WHERE age>10;



MySQL Shell

```
MySQL Shell 1.0.11

Copyright (c) 2016, 2017, 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 '\?' for help; '\quit' to exit.

Currently in JavaScript mode. Use \sql to switch to SQL mode and execute queries
.
mysql-js> \sql
Switching to SQL mode... Commands end with ;
mysql-sql> \connect root@localhost
Creating a Session to 'root@localhost'
Enter password: ********
Your MySQL connection id is 3
Server version: 5.1.73-community-log MySQL Community Server (GPL)
No default schema selected; type \use <schema> to set one.
mysql-sql> CREATE DATABASE LMCST;
Query OK, 1 row affected (0.00 sec)
mysql-sql> USE LMCST;
Query OK, 0 rows affected (0.00 sec)
mysql-sql> CREATE TABLE PERSON(NAME VARCHAR(15), AGE INT);
Query OK, 0 rows affected (0.08 sec)
mysql-sql> DESC PERSON;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| NAME  | varchar(15) | YES  |     | null    |       |
| AGE   | int(11)     | YES  |     | null    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.08 sec)
mysql-sql>
```

- Creating and Executing Queries

✓ **CREATE TABLE**

✓ **DELETE**

✓**INSERT**

✓**SELECT.**

CREATE TABLE in MySQL from Java

- Write a Java program to create a table named **Student** with fields **rollno** and **name** in the database ABC

Attribute	**Domain type**
rollno	int
name	Varchar(15)

Create table from Java

```
import java.sql.*;
public class CreateTableEg
{    public static void main(String args[]) throws ClassNotFoundException
     {
     try {
 // Register JDBC driver
      Class.forName("com.mysql.jdbc.Driver");
//Open a connection
     Connection con=DriverManager.getConnection(
           "jdbc:mysql://localhost:3306/ABC" , "root","root123");
 //Create a statement object
     Statement st=con.createStatement();
     st.executeUpdate("CREATE TABLE Student (rollno int, name varchar(15))");
     System.out.println("Table created");
     con.close();                              //Close the connection
           }catch(SQLException e) {      System.out.println("Error is " +e);    }

     }
```

INSERT row using Java

- Write a Java program to insert the following row into **Student** table with fields **rollno** and **name** in the database ABC

1	**Anu**

INSERT row using Java

```
import java.sql.*;
public class InsertEg
{    public static void main(String args[]) throws ClassNotFoundException
     {
     try {
 // Register JDBC driver
      Class.forName("com.mysql.jdbc.Driver");
//Open a connection
     Connection con=DriverManager.getConnection(
             "jdbc:mysql://localhost:3306/ABC" , "root","root123");
 //Create a statement object
     Statement st=con.createStatement();
      st.executeUpdate("INSERT INTO Student(rollno,name) VALUES (1,'Anu')");
     System.out.println(“Data inserted successfully");
     con.close();                                   //Close the connection
             }catch(SQLException e) {      System.out.println("Error is " +e);     }
     }
}
```

- To access value of java **variable** inside the SQL command use : *Singlequote*Doublequote+**varaiable+**Doublequote*Singlequote*
- E.g.

```
int roll=2;
String nam="Anu";
st.executeUpdate("INSERT INTO Student(rollno,name)
                          VALUES('"+ roll +"'  , '"+ nam +"'  )");
```

INSERT **row using Java**

- Write a Java program to insert details about n students into **Student** table with fields **rollno** and **name** in the database ABC

INSERT n rows from Java

```
import java.sql.*;      import java.util.*;
public class CreateTableEg
{   public static void main(String args[]) throws ClassNotFoundException
    { try {            Scanner sc=new Scanner(System.in);
                       int roll,n,i;
                       String nam;
     Class.forName("com.mysql.jdbc.Driver");
    Connection con=DriverManager.getConnection(
          "jdbc:mysql://localhost:3306/ABC" , "root","root123");
    Statement st=con.createStatement();
     System.out.println("Enter how many students detail to be inserted");
    n=sc.nextInt();
    for(i=0;i<n;i++)
    {      System.out.println("Enter the rollno");
           roll=sc.nextInt();
           System.out.println("Enter the name");
           sc.nextLine();
           nam=sc.nextLine();
st.executeUpdate("INSERT INTO Student(rollno,name)
                       values('"+ roll +"'  , '"+ nam +"' )");
    }      System.out.println("Data inserted successfully");
```

```
     con.close();
     }
catch(SQLException e)
      {
      System.out.println(e);
      }
     }
}
```

DELETE rows using Java

- Write a Java program to delete details about students into **Student** table with given roll number.

DELETE rows using Java

```
import java.sql.*;      import java.util.*;
public class InsertEg
{   public static void main(String args[]) throws ClassNotFoundException
    {       int roll;
            Scanner sc=new Scanner(System.in);
    try {   Class.forName("com.mysql.jdbc.Driver");
            Connection con=DriverManager.getConnection(
                        "jdbc:mysql://localhost:3306/ABC" , "root","root123");
            Statement st=con.createStatement();


            System.out.println("Enter the rollno of student to be deleted");
            roll=sc.nextInt();
    st.executeUpdate("DELETE FROM Student WHERE rollno='"+ roll +"' ");
            System.out.println("Data deleted succesfully.");
            con.close();
            }catch(SQLException e) {      System.out.println("Error is " +e);    }
    }
}
```

UPDATE rows using Java

- Write a Java program to update the name of roll number 1 to John in **Student** table

UPDATE rows using Java

```
import java.sql.*;    import java.util.*;
public classUpdateEg
{    public static void main(String args[]) throws ClassNotFoundException
     {        Scanner sc=new Scanner(System.in);
              int roll;                  String nam;
              try {       Class.forName("com.mysql.jdbc.Driver");
              Connection con=DriverManager.getConnection(
                          "jdbc:mysql://localhost:3306/ABC" , "root","root123");
              Statement st=con.createStatement();
               System.out.println("Enter the rollno of student ");
              roll=sc.nextInt();                          sc.nextLine();
              System.out.println("Enter the name of new student ");
              nam=sc.nextLine();
st.executeUpdate("UPDATE  Student SET name='"+ nam +"'  WHERE
                          rollno='"+ roll +"' ");
              con.close();
              }catch(SQLException e) {      System.out.println("Error is " +e);    }
     }
}
```

SELECT rows using Java

- Write a Java program to **list all names and roll numbers** in **Student** table

SELECT rows Using Java

```
import java.sql.*;
public class InsertEg
{   public static void main(String args[]) throws ClassNotFoundException
    {
    try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection con=DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/ABC" , "root","root123");
            Statement st=con.createStatement();
            ResultSet rs=st.executeQuery("SELECT rollno,name FROM student");
            while(rs.next())
            {               System.out.println(rs.getInt(1)+"  "+rs.getString(2));
            }
            con.close();
      }catch(SQLException e) {      System.out.println("Error is " +e);    }
    }
}
```

Import classes and interfaces from java.sql package

Load the Driver for MySQL

Establish connection

Create statement

Execute Query and represent result as result set

First column

Second column

Close the connection

Load the Driver for MySQL

SELECT rowsUsing Java

```
import java.sql.*;
public class InsertEg
{   public static void main(String args[]) throws ClassNotFoundException
    {
    try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection con=DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/ABC" , "root","root123");
            Statement st=con.createStatement();
            ResultSet rs=st.executeQuery("select rollno,name from student");
            while(rs.next())
            {               System.out.println(rs.getInt(1)+"  "+rs.getString(2));
            }
            con.close();
        }catch(SQLException e) {    System.out.println("Error is " +e);   }
    }
}
```

WORKING

- In this example we used MySQL as the database.
- Class.forName() is used for loading the Driver class
- **Driver class:** The driver class for the mysql database is **com.mysql.jdbc.Driver**.
- DriverManager.getConnection() helps to establish the connection
  - **Connection URL:** The connection URL for the mysql database is **jdbc:mysql://localhost:3306/ABC** 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 **ABC**is the database name. We may use any database in MySQL here.
  - **Username:** The default username for the mysql database is **root**.
  - **Password:** It is the password given by the user at the time of installing the mysql database.

WORKING(contd.)

Connection con=**DriverManager**.**getConnection**(
"**jdbc:mysql://localhost:3306/ABC**" , "**root**" , "**root123**");

❑ The JDBC url used is

jdbc:mysql://localhost:3306/ABC

- Here **jdbc** is the API,
- **mysql** is the database,
- **localhost** is the server name on which **mysql** is running(we can also give IP address here)
- **3306** is the port number
- **ABC** is the database name created in MySQL *(give your database name here)*

❑ **root** is the username of MySQL(root is the default username . Other MySQL username we can give here).

❑ **root123** is the password of MySQL given during installation(give password you gave during installation)

WORKING(contd.)

Statement **st**=**con**.createStatement(); // con is the Connection object

- Statement object **st** is created from the Connection object **con** using the method createStatement(). **st** can be used to **send and execute SQL statements** to a database.

- To execute a SQL statement **SELECT** we use **executeQuery()** method on **Statement** object **.**
- (Note:To execute a SQL commands CREATE TABLE, INSERT, DELETE , UPDATE we use executeUpdate() method on **Statement** object**.**)

ResultSet rs=**st**.**executeQuery**("SELECT rollno, name FROM Student");

- SELECT command is executed using **executeQuery()** method on **Statement** object st.
- Result of this SELECT is the rows containing all rollno and name from the table Student in the form of ResultSet.
- **ResultSet** object **rs** maintains a cursor that is initially positioned **before the first row** in the result of SELECT command.

Prepared by Renetha J.B.

WORKING(contd.)

```
ResultSet rs=st.executeQuery("SELECT rollno, name FROM   student");
        while(rs.next())
        {
                System.out.println(rs.getInt(1));
                System.out.println(rs.getString(2));
        }
```

- **ResultSet** object **rs maintains a cursor** that is initially positioned **before the first row in the result.**
- When **while(rs.next()) is first executed** the cursor moves forward to first row of result
  - System.out.println(rs.getInt(1)); prints the value of first attribute (rollno)
  - System.out.println(rs.getString(2)); prints the value of second attribute (name) in the result
- Next time **while** loop is executed, if second row is there, then rs.nex() moves cursor to second row and prints the values in that row.
- This continues until there is no more row in the result.

con.close(); closes the connection

WORKING(contd.)

```
ResultSet rs=st.executeQuery("SELECT rollno, name FROM   student");
        while(rs.next())
        {
                System.out.println(rs.getInt(1));
                System.out.println(rs.getString(2));
        }
```

Working:

- Here rollno and name in all rows in student table are retrieved.
- **ResultSet** object **rs** maintains a cursor that is initially positioned **before the first row in the result.**
- When **while(rs.next()) is first executed** the cursor moves forward to first row of result and prints the value of first attribute (rollno) and second attribute(name)in first row in the result
- Next time while loop is executed, if second row is there, then rs.nex() moves cursor to second row and prints the values in that row.
- This continues until there is no more row in the result.

con.close(); closes the connection

WORKING(contd.)

- ClassNotFoundException , SQLException and other exceptions may occur during these steps.
- ClassNotFoundException is thrown from main function

public static void main(String args[]) **throws** **ClassNotFoundException**

- Using try catch other exceptions like SQLException can be handled

```
try{


    }catch(SQLException se)
    {     }
catch(Exception e)
    {     }
```

SELECT rows based on conditionusing Java

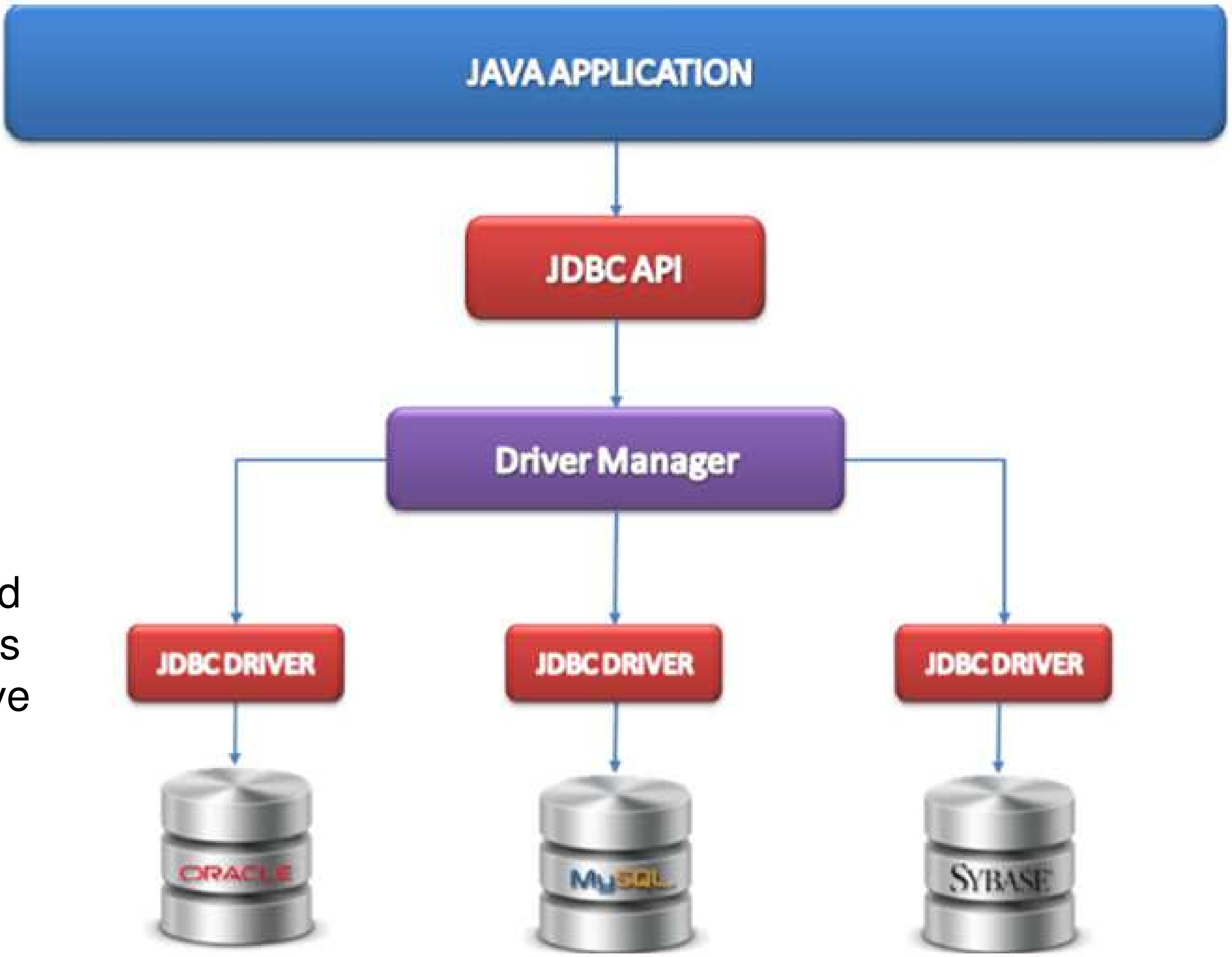
- Write a Java program to **list name of the student** in **Student** table **having given roll number**

SELECT row based on condition Using Java

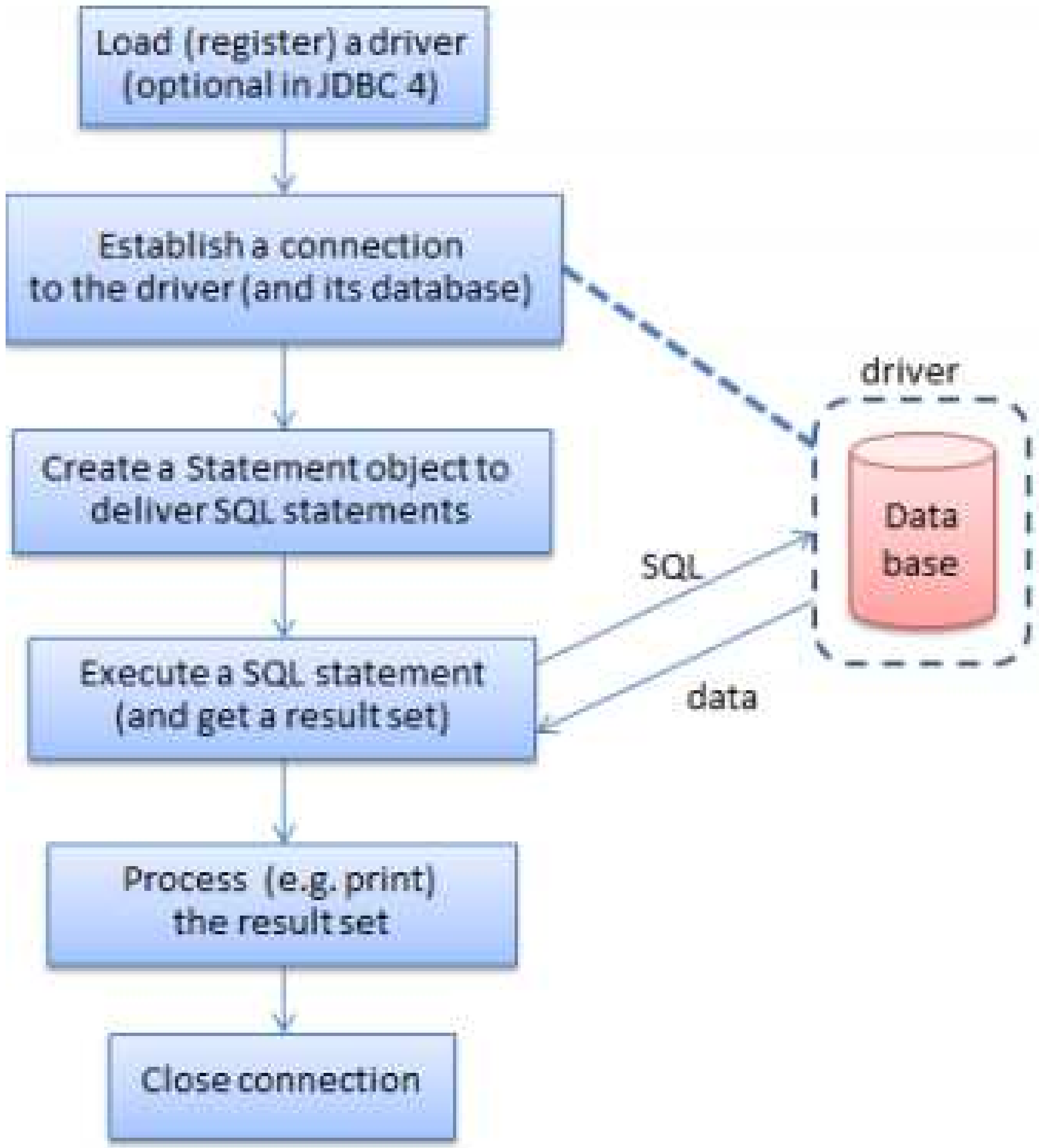
```
import java.sql.*;
import java.util.*
public class InsertEg
{    public static void main(String args[]) throws ClassNotFoundException
     { try { int roll;
              Scanner sc=new Scanner(System.in);
             Class.forName("com.mysql.jdbc.Driver");
             Connection con=DriverManager.getConnection(
                         "jdbc:mysql://localhost:3306/ABC" , "root","root123");
             Statement st=con.createStatement();
             System.out.println("Enter the rollno of student ");
             roll=sc.nextInt();
ResultSet   rs=st.executeQuery("SELECT   name   FROM   student   WHERE
     rollno='"+ roll +"' ");
             System.out.println("Name of student is "+rs.getString(1));

             con.close();
               }catch(SQLException e) {       System.out.println("Error is " +e);    }
     }
}
```

Summary

- Programming Language -**Java**
  - for coding, developing interfaces(**front end** of an application)
- Database – **MySQL ,** Oracle , PostgreSQL
  - For storing data- (**back end** of an application)
  - Relational database -Tabular structure
    - **E.g.** - Oracle, Microsoft Access, **MySQL**, PostgreSQL, MongoDB etc
- SQL- Structured Query Language
  - **SQL** statements are used to perform operations on a database. CREATE TABLE, INSERT, DELETE, UPDATE, SELECT



Com.jd
bc.mys
ql.Drive
r


Load (register) a driver (optional in JDBC 4)
Establish a connection to the driver (and its database)
Create a Statement object to deliver SQL statements
Execute a SQL statement (and get a result set)
Process (e.g. print) the result set
Close connection
driver
Data base
SQL
data

Thank You