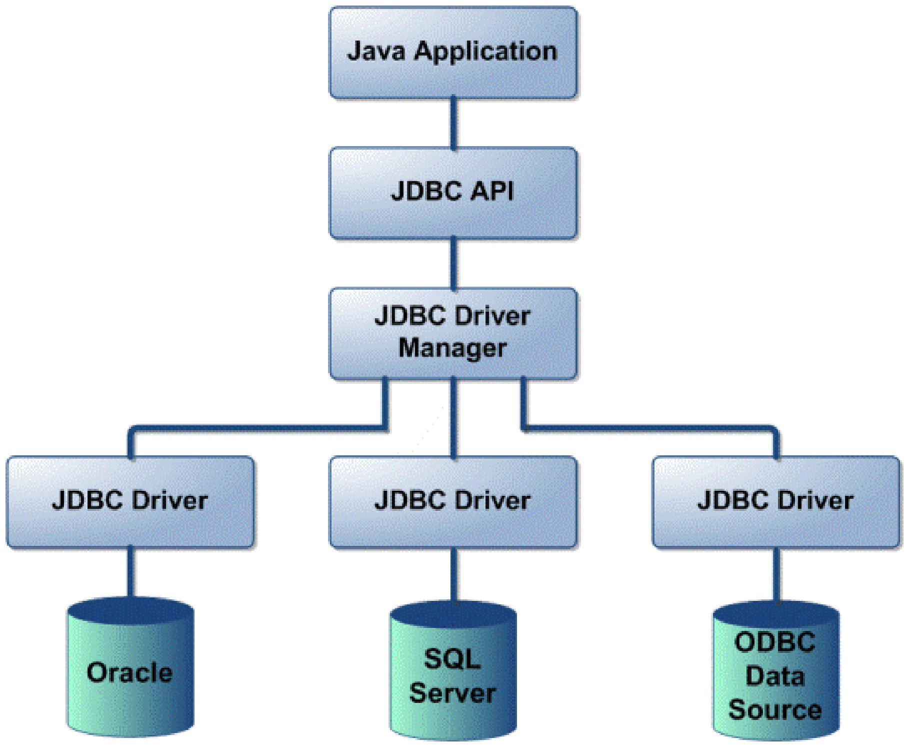
JDBC

Presented by,
B.Vijayalakshmi
Computer Centre
MIT Campus
Anna University

JDBC
(**J**ava **D**ata**b**ase **C**onnectivity)

- It is a **standard Java API for database-independent connectivity** between the java programming language and a wide range of databases(Oracle, MS Access, MySQL, SQL Server)
- The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage
  - ➢ Making connection to a database
  - ➢ Creating SQL (or) MYSQL Statements
  - ➢ Executing SQL or MYSQL queries in the database
  - ➢ Viewing or modifying the resulting records.
- **JDBC is a specification that provides a complete set of interfaces that allows for portable access to an underlying database**.
- All types of executable**, Java Application, Java Applets, Java Servlets, java server pages(JSP), Enterprise Java Beans (EJBs) are able to use a JDBC driver to access a database** and take advantage of the stored data

JDBC Architecture



- It has 2 layers,
  - **JDBC API**
    - → provides the application-to-JDBC Manager connection.
    - → It uses a driver manager and database-specific drivers to provide transparent connectivity to heterogeneous databases.
  - **JDBC Driver API**
    - → This supports the JDBC Manager-to-Driver Connection.
    - → It ensures that the correct driver is used to access each data source.
- The driver manager is capable of supporting multiple concurrent drivers connected to multiple heterogeneous databases.

Architecture of JDBC

- The JDBC API supports both two-tier and three- tier architecture for database access.
- **Two-tier Architecture**
  Two-tier Architecture provides direct communication between Java applications to the database. It requires a **JDBC driver** that can help to communicate with the particular database.

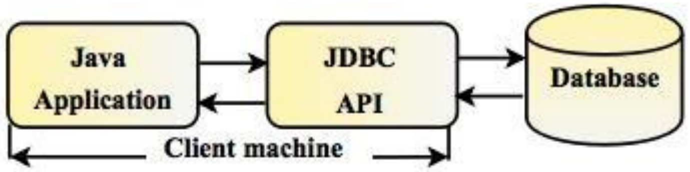


Fig: Two-tier Architecture of JDBC

- **Three-tier Architecture**

In the three-tier model, commands are sent by the HTML browser to middle services (Business Logic) which can send the commands to the particular database. It can also provide better performance.

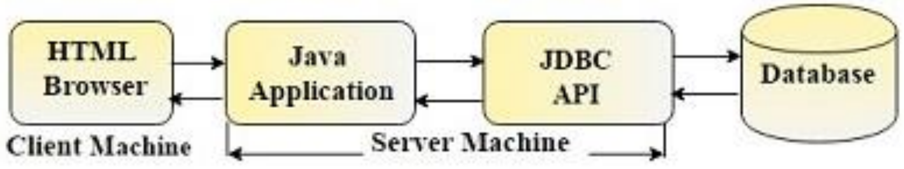
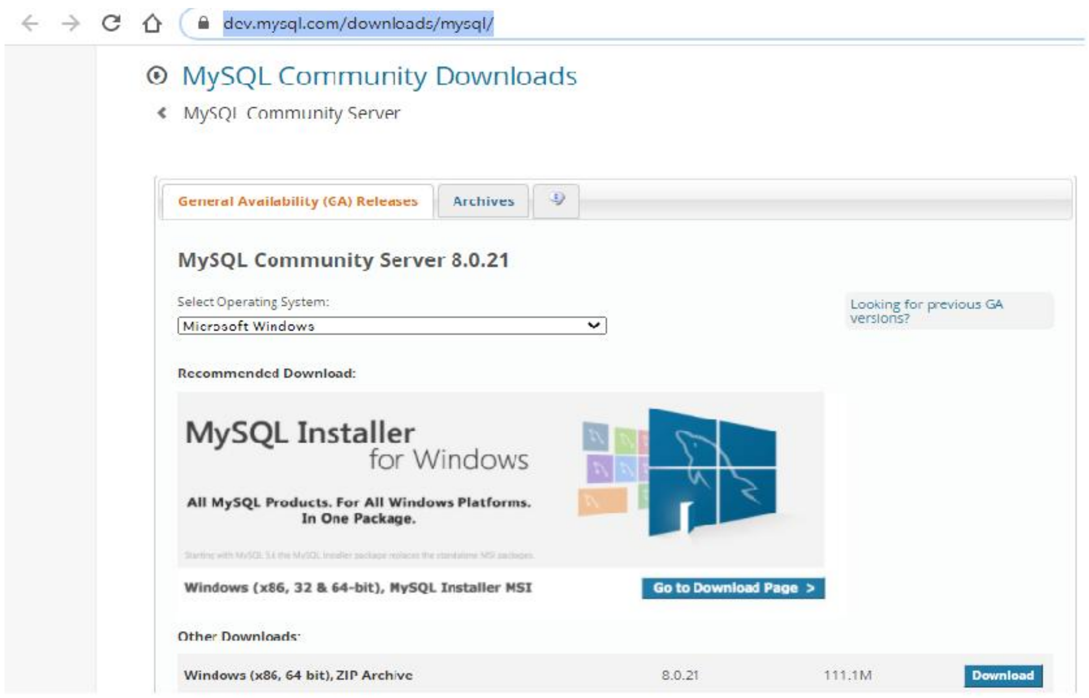


Fig: Three-tier Architecture of JDBC

The most important thing we will need, is an actual running database with a table that we can query and modify. So

Install a database that is most suitable for you

MySQL is an open source database:
You can download it from, https://dev.mysql.com/downloads/mysql/



dev.mysql.com/downloads/mysql/

MySQL Community Downloads

MySQL Community Server

MySQL Community Server 8.0.21

Select Operating System:
Microsoft Windows

Looking for previous GA versions?

Recommended Download:

MySQL Installer for Windows

All MySQL Products. For All Windows Platforms. In One Package.

Windows (x86, 32 & 64-bit), MySQL Installer MSI

Other Downloads:

Windows (x86, 64 bit), ZIP Archive	8.0.21	111.1M	Download

The full Windows installation is recommended for downloading

Basic SQL Commands

- **To Create Database:**

> create database database_name;

Example: create database student

- **To drop database (delete database):**

> drop database database_name;

Example: drop database student;

- **To Create Table:**

```
>create table table-name
(
column_name coulumn_dataType;
"                    "
);
```

Example:

```
create table details
( regno INT NOT NULL,
  age     INT NOT NULL,
name   VARCHAR (255),
PRIMARY KEY (regno)
);
```

Basic SQL Commands Cont’d

- **<u>To Drop Table (delete table):</u>**

  >drop table tablename;

  Example: drop table details;
- **<u>To Insert data into Table:</u>**

  >INSERT INTO table_name VALUES (column1,column2,…);

  Example: INSERT INTO details VALUES(111,18,’Ajay’);
- **<u>To Select Data:</u>**

  >SELECT column_name, column_name,…

  FROM table_name

  WHERE conditions;
- Where clause can use the comparison operators such as =,!=,<, >, <=, >= as well as BETWEEN and LIKE operations.

  Example:

  SELECT name,age FROM details WHERE regno=111;

  SELECT name,age FROM details WHERE name LIKE ‘%Ajay%’;

Basic SQL Commands Cont’d

- **To Update Data:**

  >UPDATE table_name

  SET column_name=value,column_name=value,...

  WHERE condition;

  Example: UPDATE details SET age=20 where id=111;

- **To Delete Data:**

  >DELETE FROM table_name WHERE conditions;

  Example: DELETE from details where id=111;

Steps to connect Java Application to Database

1. Import the package, java.sql.*;
2. Register the JDBC Driver
3. Create a connection (open a connection)
4. Create SQL statement
5. Execute SQL statement
6. Extract data from result set
7. Close connection

Register the JDBC Driver (Load)

- The program that needs database connection need to first load the driver.
- The driver is loaded with the help of static method,

public static void Class.forName(String classname) throws ClassNotFoundException

Drivername

Database, driver and URL with example

Relational Database	Driver Name (qualified class name)	Database URL & Example
MySQL	com.mysql.jdbc.Driver	jdbc:mysql://<server>:<port>/<databaseName> **Eg:** jdbc:mysql://localhost:3306/myDBName
Oracle	oracle.jdbc.driver.OracleDriver	jdbc:oracle:thin:@<server>:<port>:<databaseName> **Eg:** jdbc:oracle:thin:@localhost:1521:xe
BM DB2 App	com.ibm.db2.jdbc.app.DB2Driver	jdbc:db2:<databaseName> **Eg:** jdbc:db2:myDBName
IBM DB2 Net	com.ibm.db2.jdbc.net.DB2Driver	jdbc:db2//<server>:<port>/<databasebName> **Eg:** jdbc:db2://localhost:6789/myDBName

Database, driver and URL with example

Relational Database	Driver Name (qualified class name)	Database URL & Example
Sybase	com.sybase.jdbc.SybDriver	jdbc:sybase:Tds:<server> :<port>/<databaseName> **Eg:** jdbc:sybase:Tds:localhost :4100/myDBName
Teradata	com.teradata.jdbc.TeraDriver	jdbc:teradata://<server> /database=<databaseName> ,tmode=ANSI,charset=UTF8 **Eg:** jdbc:teradata://localhost /database=myDBName , tmode=ANSI, charset=UTF8
Microsoft SQL Server	com.microsoft.sqlserver .jdbc.SQLServerDriver	jdbc:sqlserver://<server> :<port>;databaseName=<database Name> **Eg:** jdbc:sqlserver://localhost :1433;databaseName=myDBNa me

Database, driver and URL with example

Relational Database	Driver Name (qualified class name)	Database URL & Example
Postgre	org.postgresql.Driver	jdbc:postgresql://<server> :<port>/<databaseName> **Eg:** jdbc:postgresql://local host :5432/myDBName
MS Access (JDBC-ODBC Bridge)	sun.jdbc.odbc.JdbcOdbcDriver	jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)}; DBQ=<myDBName.mdb>; **Eg:** jdbc:odbc:Driver={Mic rosoft Access Driver (*.mdb)}; DBQ=myDBName.mdb;

Create a connection

- Connection to the database is established using the static method getConnection of the DriverManager class
  1) **public static** Connection getConnection(String url)**throws** SQLException
  2) **public static** Connection getConnection(String url,String name,String password) **throws** SQLException
- If any problem occurs during accessing the database an SQL Exception is generated, else a connection object is returned which refers to a connection to a database
- **Connection is actually an interface in java.sql package**

Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/database_name", "root", “password");

Create statement

- The connection (after being established) is used to send SQL statements to the database.
- There are 3 interfaces,
  - **Statement**→ Used to execute normal SQL queries.
  - **Prepared Statement**→ Used to execute dynamic or parameterized SQL queries.
  - **Callable Statement**→ Used to execute the stored procedures.
- A statement object is used to send a simple SQL statement to the database with no parameters

public Statement createStatement()**throws** SQLException

Statement stmt=con.createStatement();

Execute the Query/Extract data from result set

- SQL statements are executed with the help of

1. ResultSet executeQuery(String sqlQuery)**throws** SQLException

   → It is used for executing SQL statements that return a single ResultSet.

2. int executeUpdate(String sqlQuery) **throws** SQLException

   →It is used for DDL (Data Definition Language) and DML (Data Manipulation Language) SQL statements like insert, update, delete, and create

   →This method returns an integer value of DML to indicate the number of rows affected/ inserted and 0 for DDL statements which do not return anything

Example: Select statement

```
ResultSet rs = s.executeQuery("select * from details");
while (rs.next())
{
     System.out.println(rs.getInt(1) + " " + rs.getString(2));
}
```

Close connection

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

```
public void close()throws SQLException
con.close();
```

- **<u>Note:</u>** Since Java 7, JDBC has ability to use try-with-resources statement to automatically close resources of type Connection, ResultSet, and Statement.
- It avoids explicit connection closing step.

In all my Examples, in MYSQL

- I have used the following commands to create database and table in MySQL

```
> create database student;
>use student;
>create table details(
            reg_no    int not null,
            name      varchar(30),
            age       int,
            gender    varchar(11),
            dob       date,
            address   varchar(50),
            primary key(reg_no);
            );
>insert into details values(111, 'Ajay',20,'male','1996-02-17','no-7 Gandhi
street,porur,chennai');
>insert into details values(111, 'Balu',20,'male','1996-04-25','no-9 kk
nagar,madurai');
>select * from details
```

Example 1

```
import java.sql.*;
public class JDBCExample
{
   public static void main(String[] args) throws Exception
    {
        Class.forName("com.mysql.jdbc.Driver");
        Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/student",
"root", "12345");
        Statement stmt=con.createStatement();
        ResultSet rs=stmt.executeQuery("select * from details");
        while(rs.next())
        {
            System.out.print(rs.getString(1)+"\t");
            System.out.print(rs.getString(2)+"\t");
            System.out.print(rs.getString(3)+"\t");
            System.out.print(rs.getString(4)+"\t");
            System.out.print(rs.getString(5)+"\t");
            System.out.print(rs.getString(6)+"\t");
            System.out.println("\n");
        }
        con.close();
  }
```

O/P:

```
111     Ajay    20      male    1996-02-17      no 7 gandhi street,porur,chennai
112     Balu    20      male    1996-04-25      no 9 kk nagar,madurai
```

Example 2

```
import java.sql.*;
public class JDBCExample
{
   public static void main(String[] args) throws Exception
    {
        Class.forName("com.mysql.jdbc.Driver");
        Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/student",
"root", "12345");
        Statement stmt=con.createStatement();
        ResultSet rs=stmt.executeQuery("select * from details");
        while(rs.next())
        {
            System.out.print(rs.getString(1)+"\t");
            System.out.print(rs.getString(2)+"\t");
            System.out.print(rs.getString(3)+"\t");
            System.out.print(rs.getString(4)+"\t");
            System.out.print(rs.getString(5)+"\t");
            System.out.print(rs.getString(6)+"\t");
            System.out.println("\n");
        }
```

```
       System.out.println(" After Prepared statement \n  ");
        PreparedStatement pstmt=con.prepareStatement("insert into  details values (?, ?, ?, ?, ? ,
?)");
        pstmt.setInt(1, 113);
        pstmt.setString(2, "Swetha");
        pstmt.setInt(3, 19);
        pstmt.setString(4, "Female");
        pstmt.setDate(5, new java.sql.Date(1997, 02, 11));
        pstmt.setString(6, "33-4 nehru stree coimbatore");
        pstmt.executeUpdate();
        rs=stmt.executeQuery("select * from details");
        while(rs.next())
        {
            System.out.print(rs.getString(1)+"\t");
            System.out.print(rs.getString(2)+"\t");
            System.out.print(rs.getString(3)+"\t");
            System.out.print(rs.getString(4)+"\t");
            System.out.print(rs.getString(5)+"\t");
            System.out.print(rs.getString(6)+"\t");
            System.out.println("\n");
          }
        con.close();
    }
  }
```

Output:

```
111    Ajay     20    male      1996-02-17      no 7 gandhi street,porur,chennai
112    Balu     20    male      1996-04-25      no 9 kk nagar,madurai
 After Prepared statement

111    Ajay     20    male      1996-02-17      no 7 gandhi street,porur,chennai
112    Balu     20    male      1996-04-25      no 9 kk nagar,madurai
113    Swetha   19    Female    3897-03-11      33-4 nehru stree coimbatore
```

Example 3

```
    System.out.println(" After delete tatement \n  ");
    pstmt = con.prepareStatement("delete from details where reg_no= ? ; ");
    pstmt.setInt(1, 113);
    pstmt.executeUpdate();
    rs=stmt.executeQuery("select * from details");
    while(rs.next())
    {
        System.out.print(rs.getString(1)+"\t");
        System.out.print(rs.getString(2)+"\t");
        System.out.print(rs.getString(3)+"\t");
        System.out.print(rs.getString(4)+"\t");
        System.out.print(rs.getString(5)+"\t");
        System.out.print(rs.getString(6)+"\t");
        System.out.println("\n");

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

Output:

```
111        Ajay      20        male      1996-02-17           no 7 gandhi street,porur,chennai
112        Balu      20        male      1996-04-25           no 9 kk nagar,madurai
 After Prepared statement

111        Ajay      20        male      1996-02-17           no 7 gandhi street,porur,chennai
112        Balu      20        male      1996-04-25           no 9 kk nagar,madurai
113        Swetha    19        Female    3897-03-11           33-4 nehru stree coimbatore
 After delete tatement

111        Ajay      20        male      1996-02-17           no 7 gandhi street,porur,chennai
112        Balu      20        male      1996-04-25           no 9 kk nagar,madurai
```