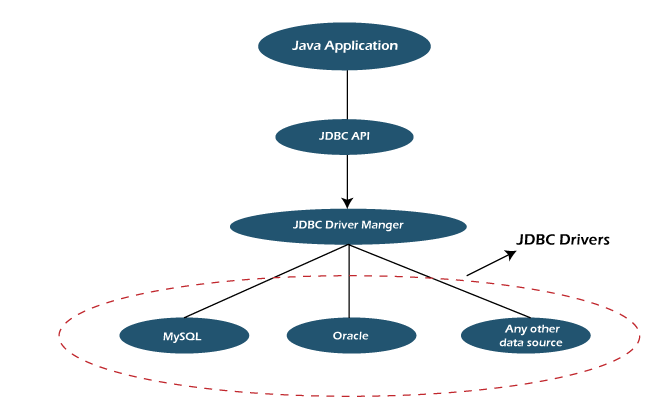
# Design of JDBC

****Java Database Connectivity**** (JDBC) is an Application Programming Interface (API), from Sun microsystem that is used by the Java application to communicate with the relational databases from different vendors. JDBC and database drivers work in tandem to access spreadsheets and databases. ****Design of JDBC**** defines the components of JDBC, which is used for connecting to the database.

## Components of JDBC



[JDBC](https://www.tpointtech.com/java-jdbc) has four major components that are used for the interaction with the database.

1. JDBC API
2. JDBC Test Suite
3. JDBC Driver Manger
4. JDBC ODBC Bridge Driver

****1) JDBC API:**** JDBC API provides various interfaces and methods to establish easy connection with different databases.

1. javax.sql.\*;
2. java.sql.\*;

****2) JDBC Test suite:**** JDBC Test suite facilitates the programmer to test the various operations such as deletion, updation, insertion that are being executed by the JDBC Drivers.

****3) JDBC Driver manager:**** JDBC Driver manager loads the database-specific driver into an application in order to establish the connection with the database. The JDBC Driver manager is also used to make the database-specific call to the database in order to do the processing of a user request.

****4) JDBC-ODBC Bridge Drivers:**** JDBC-ODBC Bridge Drivers are used to connect the database drivers to the database. The bridge does the translation of the JDBC method calls into the ODBC method call. It makes the usage of the sun.jdbc.odbc package that encompasses the native library in order to access the ODBC (Open Database Connectivity) characteristics.

#### *Note: Since Java 8, the JDBC-ODBC drivers have been removed. Oracle suggests using drivers provided by the vendor of the database.*

## Architecture of JDBC

****1) Application:**** It is the Java [servlet](https://www.tpointtech.com/servlet-tutorial) or an applet that communicates with the data source.

****2) The JDBC API:**** It allows the [Java programs](https://www.tpointtech.com/java-programs) to perform the execution of the [SQL](https://www.tpointtech.com/sql-tutorial) statements and then get the results.

A few of the crucial interfaces and classes defined in the JDBC API are the following:

* Drivers
* DriverManager
* Statement
* Connection
* CallableStatement
* PreparedStatement
* ResultSet
* SQL data

****3) DriverManager: DriverManager**** plays a crucial role in the architecture of JDBC.

It uses database-specific drivers to connect the enterprise applications to various databases.

****4) JDBC drivers:**** To interact with a data source with the help of the JDBC, one needs a JDBC driver which conveniently interacts with the respective data source.

## Different Types of Architecture of JDBC

The architecture of the JDBC consists of two and three tiers model in order to access the given database.

****Two-tier model:**** In this model, the application interacts directly with the source of data. The JDBC driver establishes the interaction between the data source and the application. When a query is sent by the user to the data source, the reply of those sent queries is sent directly to the user.

The source of data can be located on a different machine, and that machine is connected to the user machine following a client-server paradigm, where the machine which is sending the query is the client machine, and the machine that is sending the result of those queries is acting as the server.

****Three-tier model:**** In this model, the queries of the user are being sent to the middle-tier services, from where the commands are sent again to the source of data. The answers to those queries are reverted to the middle tier, and from there, it is again sent to the user.

## JDBC Working

Any Java application that needs to interact with a database needs to be programmed using the JDBC API. The JDBC driver that supports the data sources like Oracle or MySql needs to be added; then, only the interaction happens with the data source.

****FileName:**** JDBCExample.java

1. // required import statements
2. import java.sql.\*;
3. public class JDBCExample
4. {
5. // URL for establishing the connection to the database
6. //  TCP port number is 3306
7. // Name of the database is mydb
8. final static String DB\_URL = "jdbc:mysql://localhost:3306/mydb";
9. // Mysql driver class
10. final static String DB\_DRIVER = "com.mysql.jdbc.Driver";
11. // Password and User name for using the database
12. static String uName = "root";
13. static String psd = "root";
14. // main method
15. public static void main(String argvs[])
16. {
17. Connection conn = null; // for establishing connection
18. String query = null; // for storing the queries
19. Statement sttment = null; // for executing the query
20. ResultSet resultSet = null; // for storing the response of query
21. try
22. {
23. // Registering the database driver
24. Class.forName(DB\_DRIVER);
25. System.out.println("Database connection established");
26. // Creating a connection to the database
27. conn = DriverManager.getConnection(DB\_URL, uName, psd);
28. // the query be executed
29. // EmployeeId, EmployeeName, Department are the fields or column names
30. // mytable is the table name available in the mydb database
31. query = "select EmployeeId, EmployeeName, Department from mytable";
32. // for query execution
33. sttment = conn.createStatement();
34. // the query is executed and the result is stored
35. resultSet = sttment.executeQuery(query);
36. while(resultSet.next() )
37. {
38. // receiving the results using the table column name
39. int eId  = resultSet.getInt("EmployeeId");
40. String eName = resultSet.getString("EmployeeName");
41. String department = resultSet.getString("Department");
43. // printing the values
44. System.out.print("Employee ID: " + eId);
45. System.out.print(", Employee Name: " + eName);
46. System.out.println(", Department: " + department);
47. }
48. }
49. catch(SQLException sqlExp)
50. {
51. // For handling the exception raised from JDBC
52. sqlExp.printStackTrace();
53. }
54. catch(Exception e)
55. {
56. // For handing the issues raised from Class.forName
57. e.printStackTrace();
58. }
59. finally
60. {
61. try
62. {
63. // performing the clean-up work
64. // terminating the connection
65. resultSet.close();
66. sttment.close();
67. conn.close();
68. System.out.println("The Connection is closed.");
69. }
70. catch(SQLException sqlExp)
71. {
72. sqlExp.printStackTrace();
73. }
74. }
75. }
76. }

****Output:****

*Database connection established*

*Employee ID: 100, Employee Name: Nitesh Singh, Department: Project Management*

*Employee ID: 104, Employee Name: Amit Kumar, Department: Game Development*

*Employee ID: 105, Employee Name: Amrit Kumar, Department: Database Management*

*Employee ID: 109, Employee Name: Rohit Kumar, Department: Software Testing*

*Employee ID: 120, Employee Name: Ajeet Chouhan, Department: Software Design*

*Employee ID: 155, Employee Name: Aman Jatt, Department: Art Integration*

*The Connection is closed.*

****Explanation:**** The above Java application connects to the [MySQL](https://www.tpointtech.com/mysql-tutorial) Database System. Therefore, we need the driver for MySQL in order to access the data. That driver (com.mysql.jdbc.Driver) is provided in the [mysqlconnector.jar](https://images.tpointtech.com/src/jdbc/mysql-connector.jar) file, which must be included in the classpath when the above program is executed. Similarly, if instead of MySQL had we used Oracle, then the drivers corresponding to Oracle must be used.