INTRODUCTION TO JDBC

JDBC is an PI for accessing data in relational databases such as Oracle, MySQL etc. from Java in a standard way using the SQL language. JDBC actually lets you access any tabular data sources like relational databases, spreadsheets or even flat files. You pass SQL to java methods in the JDBC classes and get back JDBC objects that represent the results of your query. JDBC is portable since Java is portable across platforms. JDBC also allow us to access an ODBC-based database using a JDBC-ODBC bridge.

The JDBC API is defined by two packages in Java:

- The core API **java.sql** provides the API for accessing and processing the data stored in a database (usually a relational database) using Java. This package provides the foundation and most commonly used objects such as Connection, ResultSet, Statement, and PreparedStatement. The DriverManager class is the main component here that loads drivers and retrieves a connection.
- The extension API **javax.sql** provides the API for server-side data source access and processing from Java. This package provides services for Java EE such as DataSource and RowSet. The DataSource interface is an alternative to DriverManager for establishing a connection with a data source. JDBC extension API also provides extension services such as connection pooling, using XA enabled connection for distributed transactions etc.

JDBC architecture overview

Java application code calls a JDBC library; JDBC (DriverManager class) then loads a database driver and the driver then talks to a particular database.

JDBC architecture can be **2-tier or 3-tier**. In the 2-tier architecture, java apps directly communicate with the data source using JDBC API. In the 3-tier architecture, the client delivers the command to the middle tier; this then sends the command to the database or other data source. Here the middle tier implements the JDBC driver to interact with the particular database.

JDBC driver

A **JDBC driver** is a set of classes/interfaces that implements the interfaces that are provided in the JDBC API such as java.sql.Driver, in database specific way. Usually, each vendor such as MySQL, Oracle etc.

provides these drivers. A Java application uses JDBC to interact with relational databases without knowing about the underlying JDBC driver implementations.
JDBC drivers can be classified as Type-1, Type-2, Type-3 and Type-4:

- **Type-1** drivers are bridge drivers (like JDBC-ODBC Bridge).
- **Type-2** drivers are partly java and partly native drivers (like Type-2 driver for sybase) that uses JNI calls for DB specific native client APIs.
- **Type-3** (like Weblogic RMI driver) uses a middleware to connect to a database and
- **Type-4** (like thin driver for oracle) directly connects to a database. We will be using type-4 drivers for our examples. Type-3 and Type-4 drivers are pure java drivers.

Basic steps for using JDBC in an application

1. Obtaining the connection
2. Creating a JDBC statement object that can hold an SQL statement
3. Executing the statement and retrieving the result.
4. Closing the connection

We have seen the terms Database, relational database, SQL, ODBC etc. in our discussion. We will quickly see what they are.

Database, DBMS and RDBMS

A **database** is a system to store, retrieve, and organize information. A telephone directory or attendance register in class room are examples of a database. A database management system (**DBMS**) is a computer application that enables a user to store, manage, and retrieve data. A file or excel used to store some data online can be considered a DBMS. A relational database management system (**RDBMS**) is a database management system (DBMS) that is based on the relational model introduced by E. F. Codd, of IBM. In the**relational model** of a database, all data is represented in terms of tuples, grouped into relations. A tuple in the relational model is formally defined as a finite function that maps attributes to values like (name : "Heartin", age : 25). In simple terms, an **RDBMS** is a persistent system that organizes data into tables of related rows and columns as specified by the relational model.

SQL

SQL (Structured Query Language) is a special-purpose programming language designed for managing data held in a relational database management system (RDBMS).

ODBC and JDBC-ODBC Bridge

ODBC (Open Database Connectivity) is a C-based interface to SQL-based database systems originally developed by Microsoft in early 90s to access various DBMSs in a consistent way. ODBC accomplishes DBMS independence by using an ODBC driver as a translation layer between the application and the DBMS. Vendors need to provide specific drivers or bridges to their particular DBMS. To access an ODBC-based database from a Java client, you can use a JDBC-ODBC bridge. The JDBC-ODBC Bridge implements JDBC for any database for which an ODBC driver is available and act as an interface between JDBC and ODBC. The bridge is implemented as the sun.jdbc.odbc package and contains a native library used to access ODBC. The sun.jdbc.odbc package is defined in the jre/lib/rt.jar file, which contains the sun.jdbc.odbc.JdbcOdbcDriver class. The JDBC-ODBC Bridge is designed to work with any database that supports ODBC.

ODBC and JDBC-ODBC Bridge has some drawbacks over using JDBC with native client libraries. C++ as a language is not portable because C++ is not completely specified like java. JDBC is portable since Java is portable across platforms. Also, ODBC interface is more complicated compared to JDBC. The JDBC-ODBC Bridge is designed to work with any database that supports ODBC, it may be slower than other JDBC drivers that use protocols specific to an individual database. Because of its poor performance and lack of transaction support, the JDBC-ODBC bridge driver is recommended only when no other alternative is available for a platform; however it is very rare that a database or platform doesn't have a pure java native driver.

Also, the rise of thin client computing using HTML as an intermediate format has reduced the need for ODBC. Many web development platforms contain direct links to target databases. In these scenarios, there is no direct client-side access nor multiple client software systems to support, everything goes through the programmer-supplied HTML application. The virtualization that ODBC offers is no longer a strong requirement, and development of ODBC is no longer as active as it once was. The JDBC-ODBC Bridge will be removed in JDK 8. Oracle recommends that you use JDBC drivers provided by the vendor of your database instead of the JDBC-ODBC Bridge.

Source : http://javajee.com/introduction-to-jdbc