AN APPROACH TO JDBC

Nethravathi H[1], Rakshitha H J[2], Priyanka G M[3], Nikitha S[4]

[1,2,3,4] Assistant Professor, Department of Computer Science, DVS College of Arts and Science, Shimoga, Karnataka, India.

Abstract : JDBC is an unofficial acronym for java database connectivity. JDBC helps us to connect to a database and execute SQL statements against a database. The **JDBC** API consists of classes and interfaces written in the Java programming language. Their main **purpose** is to provide a standard API for database developers and make it possible to write database applications using a pure Java API, which in turn ensures that your code will be portable.

Keywords : OAK, API, JVM, JDBC, Database.

I INTRODUCTION :

History of Java Programming Language

Java is an Object-Oriented programming language conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank in 1991 at Sun Microsystems. It took 18 months to build up the first functioning version. The team initiated this project to build up a language for digital devices such as set-top boxes, television, etc. James Gosling and his team called their project "**Greentalk**" and its file extension was **.gt** and later became to known as "**OAK**".

"Oak"

The name **Oak** was used by **Gosling** after an **oak tree** that remained exterior his office. Also, Oak is an image of unity and picked as a national tree of several nations like the U.S.A., France, Germany, Romania, etc. But they had to later rename it as "**JAVA**" as it was already a trademark by **Oak Technologies**.

"JAVA"

Gosling and his team did a inspirational session and after the session, they came up with numerous names such as **JAVA, DNA, SILK, RUBY, etc.**

Features of Java

The major reason behind construction of Java was to bring portability and security feature into a computer language. Beside these two key features, there were many other features that played an vital role in moulding out the final form of this excellent language. Those features are :

1) Simple

Java is easy to learn and its syntax is quite simple, clean and easy to realize. The confusing and indistinct concepts of C++ are either left out in Java or they have been re-implemented in a simpler way.

Example : Pointers and Operator Overloading are absent in java but were an important factors of C++.

2) Object Oriented

In java everything is Object which has some data and behaviour. Java can be easily comprehensive as it is based on Object Model.

3) Robust

Java makes an effort to remove error level codes by emphasizing mainly on compile time error checking and runtime checking. But the main areas which Java improved were Memory Management and mishandled Exceptions by introducing automatic **Garbage Collector** and **Exception Handling**.

4) Platform Independent

As compared with other programming languages such as C, C++ etc which are compiled into operating system specific machines. Java is assured to be write-once, run-anywhere language.

On compilation Java program is compiled into bytecode. This bytecode is platform independent and can be run on any machine, plus this bytecode collection also provide security. Any machine with Java Runtime Environment can run Java Programs.

5) Secure

When it comes to security, Java is the first choice in all time. With java secure features it allows us to develop virus free, temper free system. Java program always runs in Java runtime environment with almost null communication with system OS, hence it is more secure.

6) Multi Threading

Java multithreading aspect makes it feasible to write program that can do many jobs at the same time. Benefit of multithreading is that it utilizes same memory and other resources to execute multiple threads at the same time, like While typing, grammatical errors are checked along.

7) Architectural Neutral

Compiler generates bytecodes, which have nothing to do with particular computer design, hence a Java program is easy to interpret on any machine.

8) Portable

Java Byte code can be carried to any kind of OS. No performance dependent features. Everything related to storage is predefined, example: size of primitive data types

9) High Performance

Java is an interpreted language, so it will never be as fast as a compiled language like C or C++. But, Java allows high performance with the use of just-in-time compiler.

Features of JAVA 8

- Enhanced efficiency by providing Optional Classes feature, Lamda Expressions, Streams etc.
- Ease of Use
- Improved linguist programming. A **linguist** is a program or script, written in a form which is valid in various programming languages and it performs the same operations in various programming languages. So Java now chains such type of programming skill.
- Improved Security and performance.

II INTRODUCTION TO JDBC

Java Database Connectivity (JDBC) is an **Application Programming Interface (API)** used to connect Java application with Database. JDBC is used to relate with different type of Database such as Oracle, MS Access, My SQL and SQL Server. JDBC can also be defined as the platform-independent interface between a relational database and Java programming. It enables java program to execute SQL statement and retrieve result from database.

The JDBC classes are contained in the Java Package **java.sql** and **javax.sql**. JDBC helps to write Java applications that handle these three programming activities:

1. Connect to a data source, like a database.
2. Send queries and update statements to the database
3. Retrieve and process the results received from the database in answer to your query

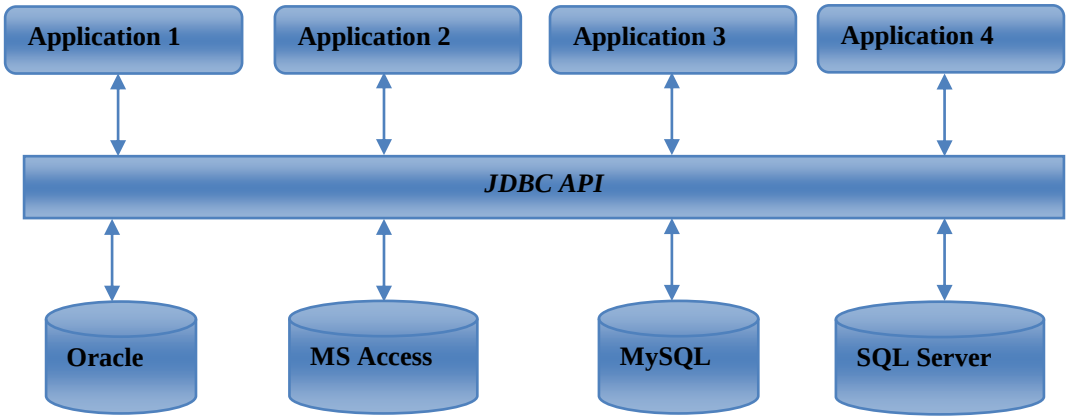

Figure (a) : JDBC Structure

JDBC Drivers

JDBC drivers are client-side adapters (installed on the client machine, not on the server) that convert requests from Java programs to a protocol that the DBMS can understand. There are 4 types of JDBC drivers:

1. Type-1 driver or JDBC-ODBC bridge driver
2. Type-2 driver or Native-API driver
3. Type-3 driver or Network Protocol driver
4. Type-4 driver or Thin driver

1. Type-1 driver

Type-1 driver or JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls. Type-1 driver is also called Universal driver because it can be used to connect to any of the databases.

- As a universal driver is used in order to relate with different databases, the data transferred through this driver is not so protected.
- The ODBC bridge driver is required to be installed in individual client machines.
- Type-1 driver isn't coded in java, that's why it isn't a portable driver.

2. Type-2 driver

The Native API driver uses the client -side libraries of the database. This driver converts JDBC method calls into native calls of the database API. In order to relate with different database, this driver requires their local API, that's why data transfer is much more protected as compared to type-1 driver.

- Driver needs to be installed independently in individual client machines
- The Vendor client library requires to be installed on client machine.
- Type-2 driver isn't coded in java, that's why it isn't a portable driver

3. Type-3 driver

The Network Protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol. Here all the database connectivity drivers are present in a single server, hence no need of individual client-side installation.

- Type-3 drivers are completely coded in Java, hence they are portable drivers.

- No client side library is needed because of application server that can perform many jobs like auditing, load balancing, logging etc.
- Network support is needed on client machine.
- Maintenance of Network Protocol driver becomes expensive because it needs database-specific coding to be done in the middle tier.

4. Type-4 driver

Type-4 driver is also a native protocol driver. These drivers cooperate directly with database. It does not need any native database library that is why it is also known as Thin Driver.

- Does not need any native library and Middleware server, so no client-side or server-side installation.
- It is completely coded in Java language, hence they are portable drivers.

JDBC Process

1. Load the JDBC Driver
2. Establish the Database Connection
3. Create a Statement Object
4. Execute a Query
5. Process the Results
6. Close the Connection

1) Loading the JDBC Driver

```
Class.forName("jdbc.DriverXYZ");
```

2) Establish the Connection

```
Connection con = DriverManager.getConnection(url, "myLogin", "myPassword");
```

3) Create a Statement Object

The JDBC Statement object sends SQL statements to the database

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

4) Execute a Query

executeQuery() : Executes the SQL query and returns the data in a table (ResultSet)

```
ResultSet results =  statement.executeQuery("SELECT a, b FROM table");
```

executeUpdate() : Used to execute for INSERT, UPDATE, or DELETE SQL statements . The return is the number of rows that were affected in the database.

5) Process the Results

A ResultSet contains the results of the SQL query.

next **:** Attempts to move to the next row in the ResultSet . If successful true is returned; otherwise, false
The first call to next positions the cursor a the first row

6) Close the Connection

To close the database connection:
stmt.close();
connection.close();

Example :

```
import java.sql.*;
public class SelectCoffees {
  public static void main(String args[ ]) throws SQLException
 {
            ResultSet  rs = null;
            PreparedStatement  ps = null;
            String url = "jdbc:odbc:coffeecon";
            Connection con;
            Statement stmt;
            try {
                  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
             }
              catch(java.lang.ClassNotFoundException e)
             {
                  System.err.print("ClassNotFoundException: ");
                  System.err.println(e.getMessage());
             }
              try
             {
               con = DriverManager.getConnection(url, "system", "tiger");
              stmt = con.createStatement();
                  ResultSet  uprs = stmt.executeQuery("SELECT * FROM COFFEES");
                  System.out.println("Table COFFEES:");
                  while (uprs.next( ))
                  {
                        String name = uprs.getString("COF_NAME");
                            int id = uprs.getInt("SUP_ID");
                            float price = uprs.getFloat("PRICE");
                            int sales = uprs.getInt("SALES");
                            int total = uprs.getInt("TOTAL");
                            System.out.print(name + "    " + id + "    " + price);
                            System.out.println("    " + sales + "    " + total);
                  }
                              uprs.close();
                              stmt.close();
                              con.close();
            }
              catch(SQLException ex)
            {
                  System.err.println("-----SQLException-----");
                  System.err.println("SQLState:  " + ex.getSQLState());
                  System.err.println("Message:  " + ex.getMessage());
                  System.err.println("Vendor:  " + ex.getErrorCode());
            }       }      }
```

New in JDBC 4.0

JDBC 4.0 is new and advance specification of JDBC. It provides the following advance features

- Connection Management
- Auto loading of Driver Interface.
- Better exception handling
- Support for large object
- Annotation in SQL query.

III LIMITATIONS

1. JDBC perseverance logic supports SQL statements based programming; these are database software dependent statements. So JDBC perseverance logic is database software dependent perseverance logic. That's means if we want to change database software in the middle of project development it is not possible, we have to re develop JDBC perseverance logic for other database software's.

2. By default JDBC ResultSet object is not serializable, so we can not send that object over the network. To conquer this RowSet introduced but that is not sustaining for all JDBC drivers.

IV CONCLUSION

JDBC provides API to communicate with various databases. With the help of JDBC drivers we can interact with different databases. This knowledge defines how a user may access the database. It provides querying and modification of data in the database. A JDBC – ODBC bridge enables interaction to any ODBC accessible data source in the JVM host environment.

REFERENCES:

1. "The Complete Reference" by Herbert Schildt, 7th Edition, Mc Graw Hill Publication.
2. "Programming with Java" by E Balaguruswamy, 4th Edition, Mc Graw Hill Publication.
3. https://www.studytonight.com/java/features-of-java.php
4. https://www.studytonight.com/java/component-of-java.php
5. https://www.studytonight.com/java/introduction-to-jdbc.php
6. https://www.geeksforgeeks.org/jdbc-drivers/
7. https://www.tutorialspoint.com/jdbc/jdbc-db-connections.htm
8. https://www.geeksforgeeks.org/the-complete-history-of-java-programming-language/
9. https://javapapers.com/jdbc/jdbc-introduction/
10. https://www.roseindia.net/answers/viewqa/JDBC/517-JDBC.html