

Experiment No.14

Title: Implementing the concept of Database Programming

Aim: To study JDBC and ODBC connectivity in Java.

Theory:

JDBC Introduction

The JDBC API is a Java API that can access any kind of tabular data, especially data stored in a Relational Database.

JDBC helps you to write java applications that manage 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

JDBC includes four components:

1. The JDBC API provides programmatic access to relational data from the Java™ programming language. Using the JDBC API, applications can execute SQL statements, retrieve results, and propagate changes back to an underlying data source. The JDBC API can also interact with multiple data sources in a distributed, heterogeneous environment.

The JDBC API is part of the Java platform, which includes the *Java Standard Edition* (Java SE) and the *Java Enterprise Edition* (Java EE). The JDBC 4.0 API is divided into two packages: `java.sql` and `javax.sql`. Both packages are included in the Java SE and Java EE platforms.

2 JDBC Driver Manager —

The JDBC DriverManager class defines objects which can connect Java applications to a JDBC driver. DriverManager has traditionally been the backbone of the JDBC architecture. It is quite small and simple.

JDBC Test Suite —

The JDBC driver test suite helps you to determine that JDBC drivers will run your program. These tests are not comprehensive or exhaustive, but they do exercise many of the important features in the JDBC API.

3 JDBC-ODBC Bridge —

The Java Software bridge provides JDBC access via ODBC drivers. Note that you need to load ODBC binary code onto each client machine that uses this driver. As a result, the ODBC driver is most appropriate on a corporate network where client installations are not a major problem, or for application server code written in Java in a three-tier architecture.

SELECT Statements

SQL is a language designed to be used with relational databases. There is a set of basic SQL commands that is considered standard and is used by all RDBMSs. For example, all RDBMSs use the SELECT statement.

A SELECT statement, also called a query, is used to get information from a table. It specifies one or more column headings, one or more tables from which to select, and some criteria for selection. The

RDBMS returns rows of the column entries that satisfy the stated requirements. A SELECT statement such as the following will fetch the first and last names of employees who have company cars:

```
SELECT First_Name, Last_Name
FROM Employees
WHERE Car_Number IS NOT NULL
```

The result set (the set of rows that satisfy the requirement of not having null in the Car_Number column) follows. The first name and last name are printed for each row that satisfies the requirement because the SELECT statement (the first line) specifies the columns First_Name and Last_Name. The FROM clause (the second line) gives the table from which the columns will be selected.

FIRST_NAME	LAST_NAME
-----	-----
Axel	Washington
Florence	Wojokowski

The following code produces a result set that includes the whole table because it asks for all of the columns in the table Employees with no restrictions (no WHERE clause). Note that SELECT * means "SELECT all columns."

```
SELECT *
FROM Employees
```

WHERE Clauses

The WHERE clause in a SELECT statement provides the criteria for selecting values. For example, in the following code fragment, values will be selected only if they occur in a row in which the column Last_Name begins with the string 'Washington'.

```
SELECT First_Name, Last_Name
FROM Employees
WHERE Last_Name LIKE 'Washington%'
```

The keyword LIKE is used to compare strings, and it offers the feature that patterns containing wildcards can be used. For example, in the code fragment above, there is a percent sign (%) at the end of 'Washington', which signifies that any value containing the string 'Washington' plus zero or more additional characters will satisfy this selection criterion. So 'Washington' or 'Washingtonian' would be matches, but 'Washing' would not be. The other wildcard used in LIKE clauses is an underbar (_), which stands for any one character. For example,

```
WHERE Last_Name LIKE 'Ba_man'
```

would match 'Batman', 'Barman', 'Badman', 'Balman', 'Bagman', 'Bamman', and so on.

The code fragment below has a WHERE clause that uses the equal sign (=) to compare numbers. It selects the first and last name of the employee who is assigned car 12.

```
SELECT First_Name, Last_Name
```

```
FROM      Employees
```

```
WHERE Car_Number = 12
```

The next code fragment selects the first and last names of employees whose employee number is greater than 10005:

```
SELECT First_Name, Last_Name
```

```
FROM Employees
```

```
WHERE Employee_Number > 10005
```

WHERE clauses can get rather elaborate, with multiple conditions and, in some DBMSs, nested conditions. This overview will not cover complicated WHERE clauses, but the following code fragment has a WHERE clause with two conditions; this query selects the first and last names of employees whose employee number is less than 10100 and who do not have a company car.

```
SELECT First_Name, Last_Name
```

```
FROM Employees
```

```
WHERE Employee_Number < 10100 and Car_Number IS NULL
```

Common SQL Commands

SQL commands are divided into categories, the two main ones being Data Manipulation Language (DML) commands and Data Definition Language (DDL) commands. DML commands deal with data, either retrieving it or modifying it to keep it up-to-date. DDL commands create or change tables and other database objects such as views and indexes.

A list of the more common DML commands follows:

- ☐ **SELECT** — used to query and display data from a database. The SELECT statement specifies which columns to include in the result set. The vast majority of the SQL commands used in applications are SELECT statements.
- ☐ **INSERT** — adds new rows to a table. INSERT is used to populate a newly created table or to add a new row (or rows) to an already-existing table.
- ☐ **DELETE** — removes a specified row or set of rows from a table
- ☐ **UPDATE** — changes an existing value in a column or group of columns in a table

The more common DDL commands follow:

- ☐ **CREATE TABLE** — creates a table with the column names the user provides. The user also needs to specify a type for the data in each column. Data types vary from one RDBMS to another, so a user might need to use metadata to establish the data types used by a particular database. CREATE TABLE is normally used less often than the data manipulation commands because a table is created only once, whereas adding or deleting rows or changing individual values generally occurs more frequently.
- ☐ **DROP TABLE** — deletes all rows and removes the table definition from the database. A JDBC API implementation is required to support the DROP TABLE command as specified

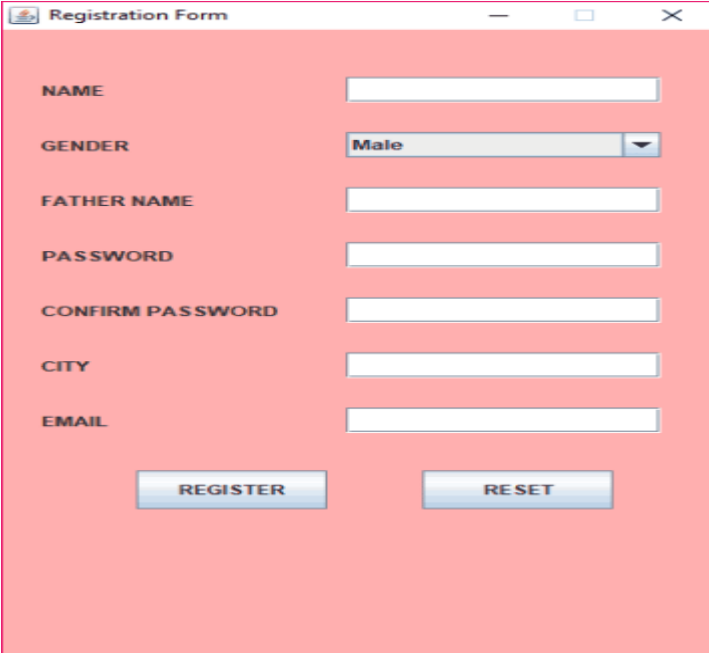
by SQL92, Transitional Level. However, support for the CASCADE and RESTRICT options

of DROP TABLE is optional. In addition, the behavior of DROP TABLE is implementation-defined when there are views or integrity constraints defined that reference the table being dropped.

- ❑ ALTER TABLE — adds or removes a column from a table. It also adds or drops table constraints and alters column attributes

Problem Statement:

Write a GUI based program to create a student registration and Login. Store Registration data in Database and take Login information from Database.



The image shows a graphical user interface for a student registration form. The window is titled "Registration Form" and has a standard Windows-style title bar with minimize, maximize, and close buttons. The form itself is light red and contains the following elements:

- NAME:** A text input field.
- GENDER:** A dropdown menu currently showing "Male".
- FATHER NAME:** A text input field.
- PASSWORD:** A text input field.
- CONFIRM PASSWORD:** A text input field.
- CITY:** A text input field.
- EMAIL:** A text input field.
- Buttons:** Two buttons at the bottom, "REGISTER" and "RESET", both with a blue gradient and white text.

Conclusion: Hence we have studied the program is used to deal with the database.