In this way, we can store some rows into emptab. To see all the rows of the table, we ... lect query as:

```
mysql> select * from emptab;
```

eno	ename	sal
1001	Nagesh	7800
1002	Ganesh	10000
1003	Vinaya	5000.55
1004	Rahul	7800

4 rows in set

Suppose we want to retrieve employees names whose salary is Rs. 7800.

```
mysql> select ename from emptab where sal = 7800;
```

ename
Nagesh
Rahul

2 rows in set

Suppose we want to delete the row of an employee whose employee number is 1001.

```
mysql> delete from emptab where eno = 1001;
```

Query OK, 1 row affected

To update the salary to Rs. 15000 of an employee whose name is "Rahul",

```
mysql> update emptab set sal= 15000 where name= "Rahul";
```

Query OK, 1 row affected

Rows matched: 1 changed: 1 warnings: 0

Finally, to close MySQL, we can give 'quit' command to see the system prompt as:

```
mysql> quit;
```

Bye

In the similar manner, we can work with Sybase, SQL Server or MSAccess databases. We can give commands to create tables, store data in the tables as well as retrieve data from the tables and process data.

Without the help of database clients, it is possible to connect to a database server by using a Java program as its client. This is where JDBC API plays significant role.

Stages in a JDBC Program

The following stages are used by Java programmers while using JDBC in their programs:

- **Registering the driver:** A database driver is a software containing classes and interfaces written according to JDBC API. Since there are several drivers available in the market, we

should first declare a driver which is going to be used for communication with the database server in a Java program.

- **Connecting to a database:** In this stage, we establish a connection with a specific database through the driver which is already registered in the previous step.
- **Preparing SQL statements in Java:** We should create SQL statements in our Java program using any of the interfaces like Statement, PreparedStatement, CallableStatement, etc., which are available in `java.sql` package.
- **Executing the SQL statements on the database:** For this purpose, we can use `execute()`, `executeQuery()` and `executeUpdate()` methods of Statement interface.
- **Retrieving the results:** The results obtained by executing the SQL statements can be stored in an object with the help of interfaces like ResultSet, ResultSetMetaData and DatabaseMetaData.
- **Closing the connection:** We should close the connection between the Java program and the database by using `close()` method of Connection interface.

Registering the Driver

This is the first step to connect to a database. A programmer should specify which database driver he is going to use to connect to the database. There are 4 ways to register a driver.

- By creating an object to driver class of the driver software, we can register the driver. For example, to register JdbcOdbcDriver of the Sun Microsystems, we can create an object to the driver class: JdbcOdbcDriver, as shown below:

```
sun.jdbc.odbc.JdbcOdbcDriver obj = new sun.jdbc.odbc.JdbcOdbcDriver();
```

- The second way to register a driver is by sending the driver class object to `registerDriver()` method of DriverManager class. For example, the object of JdbcOdbcDriver class can be passed to `registerDriver()` method as:

```
DriverManager.registerDriver( new sun.jdbc.odbc.JdbcOdbcDriver());
```

- The third way to register the driver is to send the driver class name directly to `forName()` method as:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

- In case, the user should specify the driver name at the time of running the program, we can use `getProperty()` method of System class to receive the driver name as:

```
String dname = System.getProperty("driver");
Class.forName(dname);
```

Here, `getProperty()` method accepts the driver name from the user and stores it in dname. Then `forName()` method creates an object to the class whose name is in dname. The question is how to provide the driver name while running the program? The following syntax will make it clear:

```
C:\> java -Ddriver = driverclassname Programname
```

For example, to send the driver class name: JdbcOdbcDriver to Myprog, we can type at system prompt as:

```
C:\> java -Ddriver = sun.jdbc.odbc.JdbcOdbcDriver  Myprog
```

In this case, the name: `sun.jdbc.odbc.JdbcOdbcDriver` is sent to `System.getProperty()` method and then `Class.forName()` method will create an object to it.

Important Interview Question

What is a database driver?

A database driver is a set of classes and interfaces, written according to JDBC API to communicate with a database.

How can you register a driver?

To register a database driver, we can follow one of the 4 options:

- *By creating an object to driver class*
- *By sending driver class object to DriverManager.registerDriver() method*
- *By sending the driver class name to Class.forName() method*
- *By using System class getProperty() method*

Connecting to a Database

To connect to a database, we should know 3 things:

- **URL of the database:** Here, URL(Uniform Resource Locator) represents a protocol to connect to the database. Simply speaking, it locates the database on the network.
- **Username:** To connect to a database, every user will be given a username which is generally allotted by the database administrator.
- **Password:** This is the password allotted to the user by the database administrator to connect to the database.

Let us take an example. To connect to Oracle database using Sun Microsystems' jdbc-odbc driver, we can write the following statement:

```
DriverManager.getConnection("jdbc:odbc:oradsn", "scott", "tiger");
```

Here, "`jdbc:odbc:oradsn`" is the URL. oradsn represents the DSN (data source name), a name given to the database for the reference in the Java program. "scott" is the username and "tiger" is password.

Let us take another example. To connect to Oracle database using the thin driver provided by Oracle corp, we can write the following statement:

```
DriverManager.getConnection("jdbc:  :thin:@localhost:1521:oracle", "scott",
"tiger");
```

We advise you to go through the user manuals supplied by the database vendors where they explain clearly the ways to connect to the database with examples.

Important Interview Question

What is DSN?

Data Source Name (DSN) is a name given to the database to identify it in the Java program. The DSN is linked with the actual location of the database.

Important Interview Question

What is parsing?

Parsing represents checking the syntax and grammar of a statement as a whole and also word by word.

What is the difference between Statement and PreparedStatement?

Statement parses a statement before its execution on the database. This parsing is done every time the statement is executed, and hence it may take more time when the same statement gets executed repeatedly. PreparedStatement conducts parsing only once when the same statement is executed repeatedly and hence it gives better performance.

Here, two more examples of using PreparedStatement is furnished:

```
1. PreparedStatement stmt = con.prepareStatement("update emptab set sal = ?
where eno= ? ");
stmt.setFloat(1, 10000.55f); //set 10000.55 to first ?
stmt.setInt(2,1002);  //set 1002 to second ?
int n = stmt.executeUpdate();  //n gives no. of rows affected
```

```
2. PreparedStatement stmt = con.prepareStatement("select * from emptab");
ResultSet rs = stmt.executeQuery();  //rs contains rows of emptab
```

Stored Procedures and CallableStatement

Stored procedure is a set of statements written using PL/SQL (Procedural language/Structured Query language). To perform some calculations on the tables of a database, we can use stored procedures. Stored procedures are written and stored at database server. When a client contacts the server, the stored procedure is executed, and the results are sent to the client. Let us see why stored procedures are written in Oracle at the server side.

When client/server software is created, we observe two main parts of the software:

- The first part represents the screens which accept the input from the user and also display results to the user. The program code that helps to create such screens is called 'Presentation logic'.
- The second part represents the logic that converts the input into the output. It contains some business procedures and calculations related to the activities of an organization. This is called 'business logic'.

It is advisable to create presentation logic and business logic separately without mixing them together. When they are developed separately, their maintenance becomes easy. For example, to modify presentation logic, the programmer need not think about business logic. He can modify any one without affecting the other. Similarly, it is possible to develop presentation logic and business logic using separate technologies. For example, the screens representing the presentation logic can be created using Java or VisualBasic; whereas, it is possible to create the business logic in Oracle in the form of 'stored procedures'.

Important Interview Question

What are stored procedures?

A stored procedure represents a set of statements that is stored and executed at database server, sending results to the client.

tput:

```
C:\> javac CallFun.java
C:\> java CallFun
Square value= 2500
```

Important Interview Question

What is the use of CallableStatement?

CallableStatement is used to call stored procedures and functions which run at a database server and send the results to the client.

pes of Result Sets

here are two types of result sets, namely, Forward ResultSet and Scrollable ResultSet. So far, in ur programs, we retrieved the results into a ResultSet object rs, as shown:

```
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("select * from emptab");
```

Since the results are available in 'rs', we can retrieve the rows one by one in forward direction using `s.next()` method. Suppose we want to retrieve the rows in reverse direction, is there any method ike `rs.previous()`? Fortunately this method is available, but it is not supported by the `ResultSet` object 'rs'. This means using `ResultSet`, it is possible to move in forward direction only. This is called 'forward result set'.

There is another type of result set, where we can move in both forward and backward directions. This is called 'scrollable result set'. This type of result set is created by passing two constants to `createStatement()` method as:

```
Statement stmt = con.createStatement(CONST1, CONST2);
```

Here, `CONST1` may take any one of the following:

```
ResultSet.TYPE_SCROLL_SENSITIVE
ResultSet.TYPE_SCROLL_INSENSITIVE
```

And `CONST2` may be one of the following:

```
ResultSet.CONCUR_READ_ONLY
ResultSet.CONCUR_UPDATABLE
```

The constant `ResultSet.TYPE_SCROLL_SENSITIVE` represents that any changes done to the `ResultSet` will also affect the database. The constant `ResultSet.TYPE_SCROLL_INSENSITIVE` indicates that any changes done to the ResultSet will not reflect in the database. Most of the driver vendors provide this type of result set only (`TYPE_SCROLL_INSENSITIVE`), since they feel that there will be more load on the driver if they provide `TYPE_SCROLL_SENSITIVE` nature to the driver.

`ResultSet.CONCUR_READ_ONLY` represents that we can read the data from the ResultSet object, it cannot be modified. When `ResultSet.CONCUR_UPDATABLE` is used, it allows insertion, updating and deletion activities on the rows of the result set.

```
        //go to first row
        rs.next();

        //read data from col1
        Clob c = rs.getClob(1);

        //display file length
        System.out.println("File size= "+ c.length());

        //read file data from c and store into Reader object
        Reader r = c.getCharacterStream();

        //read data from Reader and write into newfile.txt
        int ch;
        Filewriter fw = new FileWriter("newfile.txt");
        while((ch = r.read()) != -1)
        fw.write((char)ch);

        //close the file
        fw.close();

        //close the connection
        con.close();
        }
}
```

Output:

```
C:\> set classpath=C:\jars\ojdbc14.jar;.;
C:\> javac ClobDemo1.java
C:\> java ClobDemo1
File size= 1881
```

Important Interview Question

What is BLOB?

Binary Large Object (BLOB) is a SQL datatype that represents binary data to be stored into a database. BLOB helps us to store images into a database.

What is CLOB?

Character Large Object (CLOB) is a SQL datatype that represents larger volumes of text data. CLOB helps to store text files into a database.

ResultSetMetaData

`ResultSetMetaData` is an interface which contains methods to get information about the types and properties of the columns in a `ResultSet` object. The following program demonstrates how to use retrieve the information about the `ResultSet`.

ResultSet has a method `getMetaData()` which returns the information about the result set into `ResultSetMetaData` object. It can be called as:

```
ResultSetMetaData rsmd =rs.getMetaData();
```

Program 19: Write a program to find out the information of the columns of the table: emptab. This information is available in ResultSet object which can be again retrieved into ResultSetMetaData.

```
/* Demonstrates how to find out ResultSet information
   using ResultSetMetaData
```