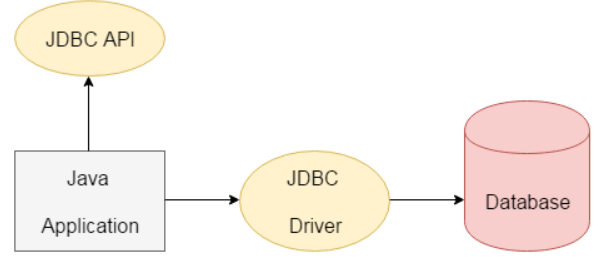
**JDBC intro**

1. JDBC is a Java API to connect and execute the query with the database.
2. JDBC is an API (Application programming interface) used to communicate Java application to database in database independent and platform independent manner.
3. It provides classes and interfaces to connect or communicate Java application with database.
4. It is a specification from Sun microsystems that provides a standard abstraction(API or Protocol) for java applications to communicate with various databases.
5. It enables Java programs to execute SQL statements. JDBC works with Java on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.



### ****Components of JDBC****

There are generally four main components of JDBC through which it can interact with a database. They are as mentioned below:

1. **JDBC API:**

It provides various methods and interfaces for easy communication with the database. It provides two packages as follows

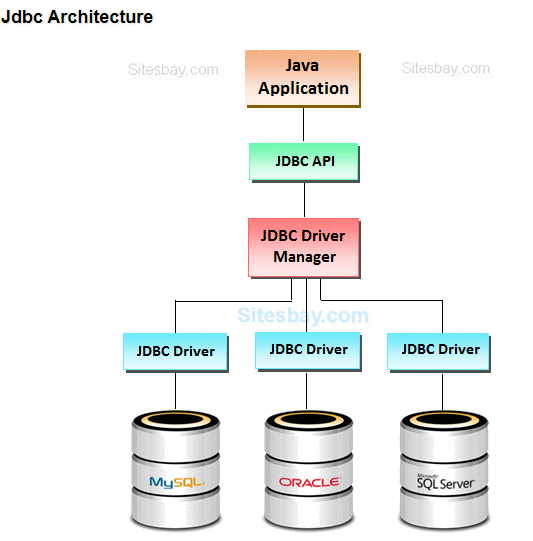
Java.sql.\*;

Javax.sql.\*;

It also provides a standard to connect a database to a client application.

1. **[JDBC Driver manager](https://www.geeksforgeeks.org/jdbc-drivers/):**It loads a database-specific driver in an application to establish a connection with a database. It is used to make a database-specific call to the database to process the user request.
2. **JDBC-ODBC Bridge Drivers**: It connects database drivers to the database. This bridge translates the JDBC method call to the ODBC function call. It makes use of the **sun.jdbc.odbc**package
3. **JDBC Test suite:** It is used to test the operation(such as insertion, deletion, updation) being performed by JDBC Drivers.

### Architecture of JDBC



**Description:**

**1.Application:** It is a java applet or a servlet that communicates with a data source.

**2.The JDBC API:** The JDBC API allows Java programs to execute SQL statements and retrieve results. Some of the important classes and interfaces defined in JDBC API are as follows:

**3.DriverManager:**It loads a database-specific driver in an application to establish a connection with a database. It is used to make a database-specific call to the database to process the user request.

**4.JDBC drivers:** To communicate with a data source through JDBC, you need a JDBC driver that intelligently communicates with the respective data source.

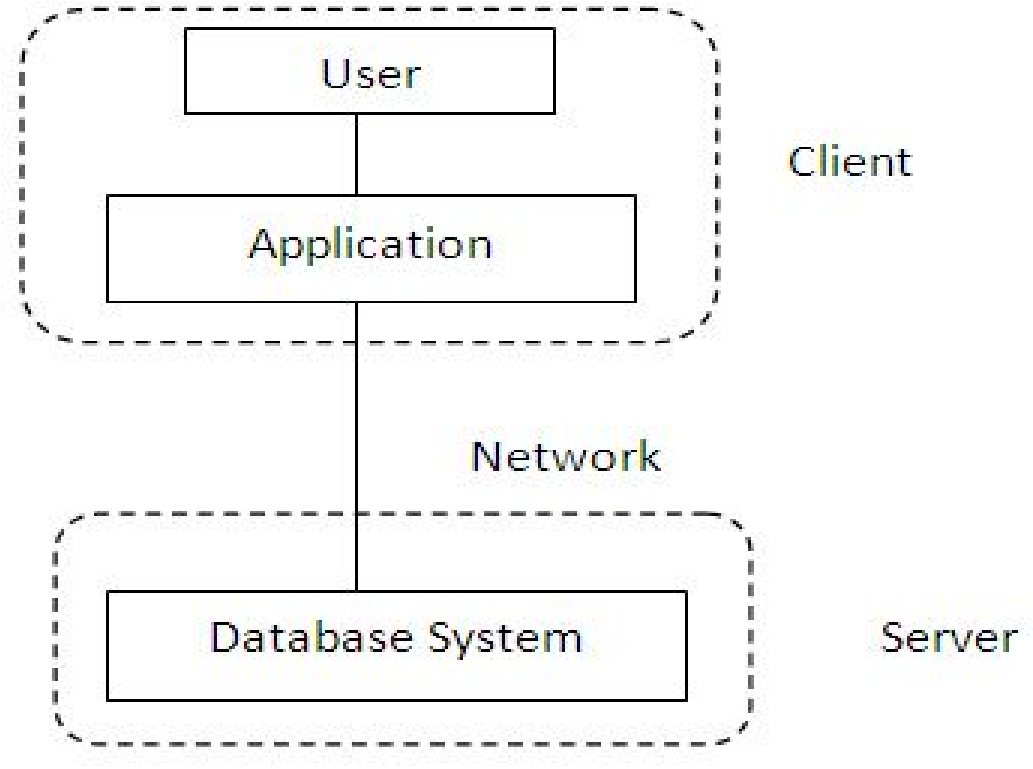
**Why Should We Use JDBC**

1. Before JDBC, ODBC API was the database API to connect and execute the query with the database.
2. But, ODBC API uses ODBC driver which is written in C language (i.e. platform dependent and unsecured).
3. That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).
4. JDBC API to handle database using Java program and can perform the following activities:

* Connect to the database
* Execute queries and update statements to the database
* Retrieve the result received from the database.

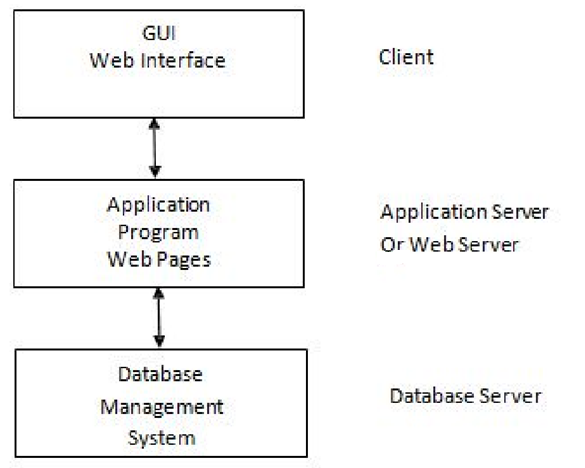
|  |  |
| --- | --- |
| **ODBC** | |
| Open Database connectivity | |
| Introduced by Microsoft in 1992 | |
| For any Language like C,C++,Java etc. | |
| Many ODBC drivers  implemented in native language like C/C++ | |
| Only for Windows | |
| Odbc is platform dependent and database independent | |
| For JAVA application it is not recommended to use ODBC because performance will be down due to internal conversation and application will be platform dependent. | |
| **JDBC** |
| Java Database Connectivity |
| Introduced by Sun Microsystem in 1997 |
| For JAVA Language only |
| Mostly JDBC drivers implemented in JAVA Language |
| For any platform |
| Jdbc is both platform and database independent. |
| For JAVA application it is highly recommended to Jdbc because there is no performance and platform dependent problem |

**Two tier architecture**



A java application communicates directly to the data source. The JDBC driver enables the communication between the application and the data source. When a user sends a query to the data source, the answers for those queries are sent back to the user in the form of results.   
The data source can be located on a different machine on a network to which a user is connected. This is known as a **client/server configuration**, where the user’s machine acts as a client, and the machine has the data source running acts as the server.

**Three tier architecture**



 In this, the user’s queries are sent to middle-tier services, from which the commands are again sent to the data source. The results are sent back to the middle tier, and from there to the user.   
This type of model is found very useful by management information system directors. advantage is that it simplifies the deployment of applications. Finally, in many cases, the three-tier architecture can provide performance advantages.

**What is API**

* **API (Application programming interface) is a document that contains a description of all the features of a product or software.**
* **It represents classes and interfaces that software programs can follow to communicate with each other.**
* **An API can be created for applications, libraries, operating systems, etc.**

**JDBC Driver**  
JDBC Driver is a software component that enables java application to interact with the database.

There are 4 types of JDBC drivers:

1. JDBC-ODBC bridge driver
2. Native-API driver (partially java driver)
3. Network Protocol driver (fully java driver)
4. Thin driver (fully java driver)

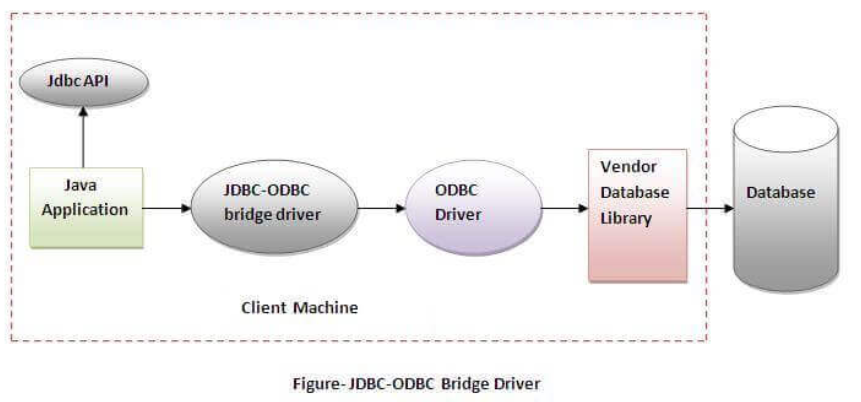
* **JDBC-ODBC bridge driver**  
  The JDBC-ODBC bridge driver uses ODBC driver to connect to the database.
* The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls. This is now discouraged because of thin driver.

**Advantages:**

* easy to use.
* can be easily connected to any database.

**Disadvantages:**

* Performance degraded because JDBC method call is converted into the ODBC function calls.
* The ODBC driver needs to be installed on the client machine.



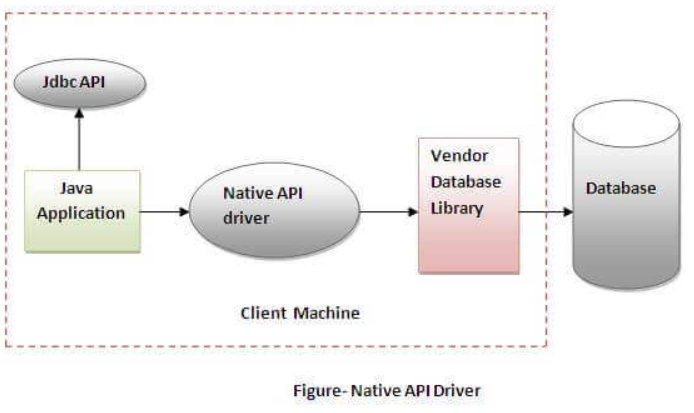
* **Native-API driver**  
  The Native API driver uses the client-side libraries of the database.
* The driver converts JDBC method calls into native calls of the database API. It is not written entirely in java.

**Advantage:**

performance upgraded than JDBC-ODBC bridge driver.

**Disadvantage:**

* The Native driver needs to be installed on the each client machine.
* The Vendor client library needs to be installed on client machine.



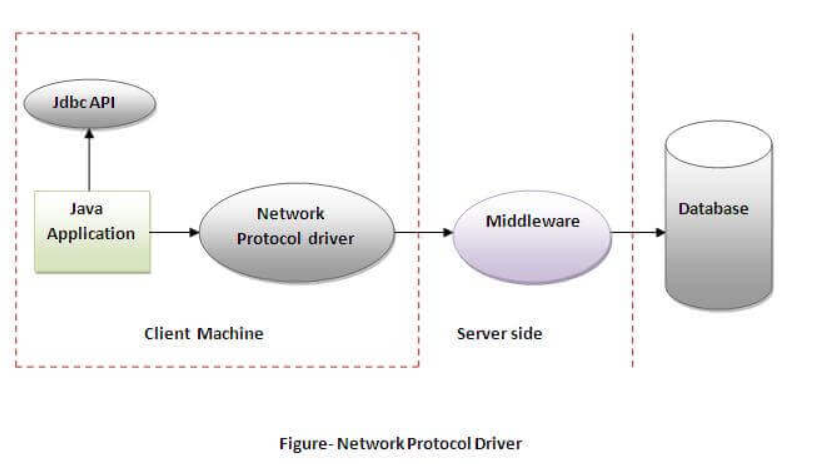
* **Network Protocol driver**  
  The Network Protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol.
* It is fully written in java.

**Advantage:**

* No client side library is required because of application server that can perform many tasks like auditing, load balancing, logging etc.

**Disadvantages:**

* Network support is required on client machine.
* Requires database-specific coding to be done in the middle tier.
* Maintenance of Network Protocol driver becomes costly because it requires database-specific coding to be done in the middle tier.



* **Thin driver**  
  The thin driver converts JDBC calls directly into the vendor-specific database protocol. That is why it is known as thin driver.
* It is fully written in Java language.

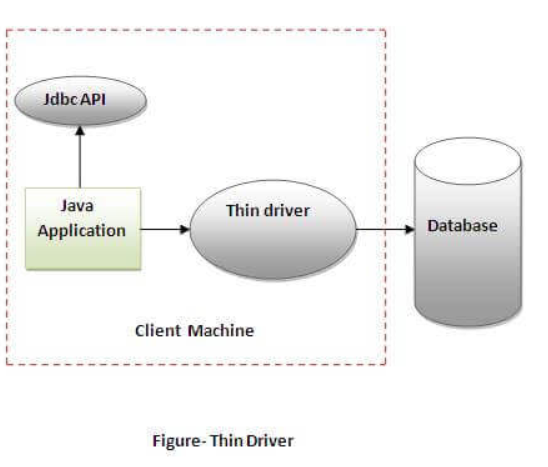
**Advantage:**

Better performance than all other drivers.

No software is required at client side or server side.

**Disadvantage:**

Drivers depend on the Database.



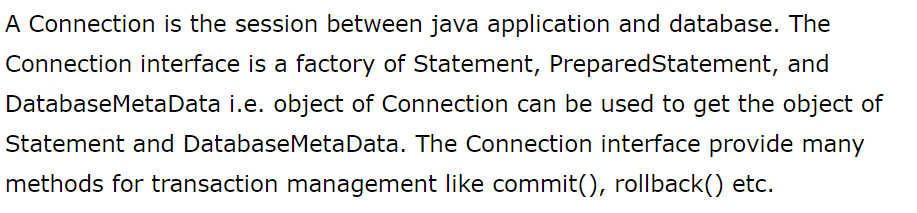
* **java.sql package contains classes and interfaces for JDBC API.**
* **A list of popular interfaces of JDBC API are given below:**

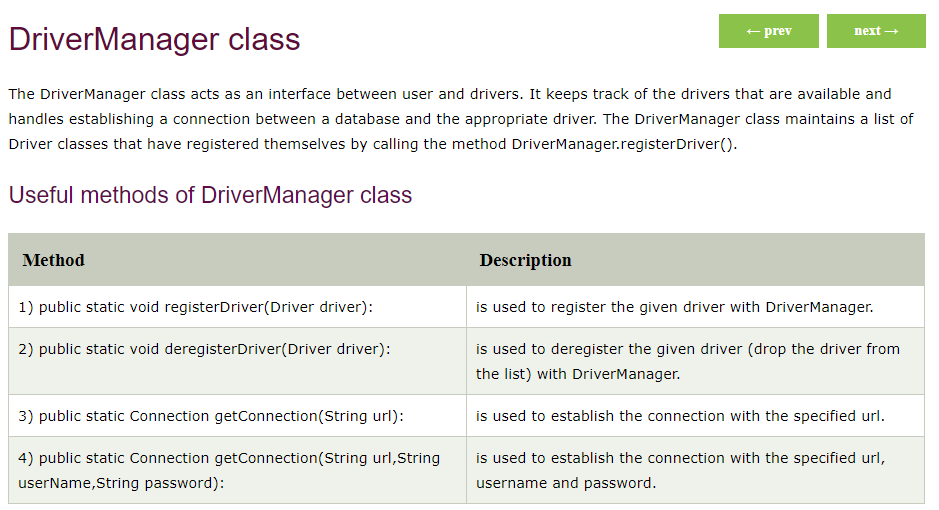
1. **Driver interface**
2. **Connection interface**
3. **Statement interface**
4. **PreparedStatement interface**
5. **CallableStatement interface**
6. **ResultSet interface**
7. **ResultSetMetaData interface**
8. **DatabaseMetaData interface**
9. **RowSet interface**

* **A list of popular classes of JDBC API are given below:**

1. **DriverManager class**
2. **Blob class**
3. **Clob class**
4. **Types class**

**Connection interface**





**JDBC - Statements, PreparedStatement and CallableStatement**

* Once a connection is obtained we can interact with the database.
* The JDBC Statement,CallableStatement, and PreparedStatement interfaces define the methods and properties that enable you to send SQL or PL/SQL commands and receive data from your database.
* They also define methods that help bridge data type differences between Java and SQL data types used in a database.

**Statement interface:**

The Statement interface provides methods to execute queries with the database. The statement interface is a factory of ResultSet i.e. it provides factory method to get the object of ResultSet.

The important methods of Statement interface are as follows:

1. public ResultSet executeQuery(String sql):

is used to execute SELECT query. It returns the object of ResultSet.

1. public int executeUpdate(String sql):

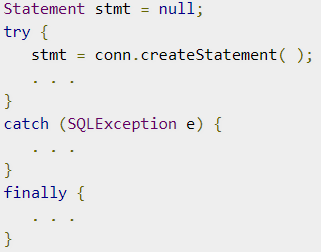
is used to execute specified query, it may be create, drop, insert, update, delete etc.

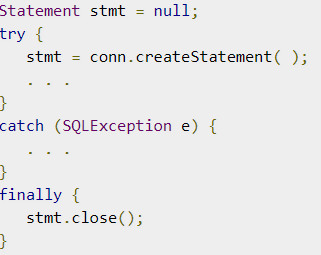
1. public boolean execute(String sql):

is used to execute queries that may return multiple results.

1. public int[] executeBatch():

is used to execute batch of commands.





**PreparedStatement interface**

The PreparedStatement interface is a sub interface of Statement.

It is used to execute parameterized query.

Let's see the example of parameterized query:

String sql="insert into emp values(?,?,?)";

As you can see, we are passing parameter (?) for the values.

Its value will be set by calling the setter methods of PreparedStatement.

Improves performance:

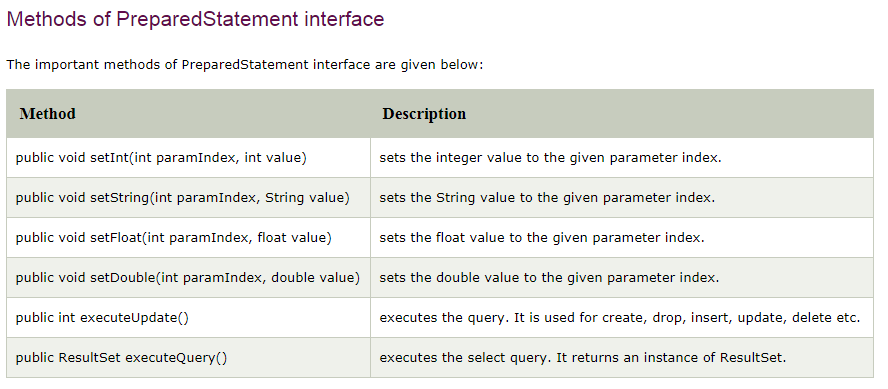
The performance of the application will be faster if you use PreparedStatement interface because query is compiled only once.

How to get the instance of PreparedStatement?

The prepareStatement() method of Connection interface is used to return the object of PreparedStatement.

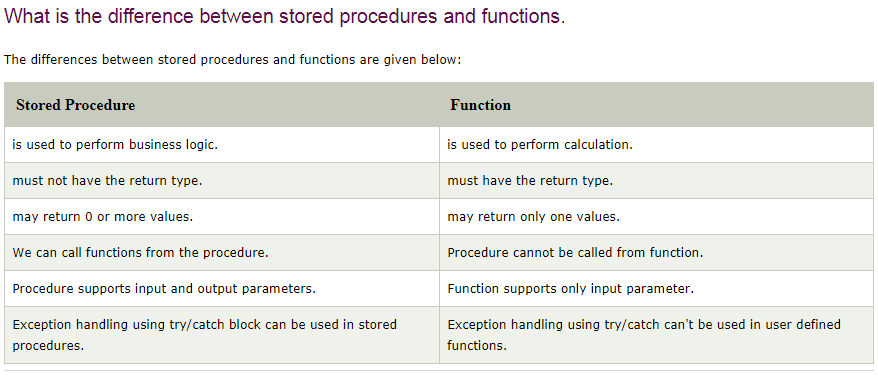
Syntax:

**public PreparedStatement prepareStatement(String query)throws SQLException{}**



**CallableStatement**

* CallableStatement interface is used to call the stored procedures and functions.
* We can have business logic on the database by the use of stored procedures and functions that will make the performance better because these are precompiled.
* Suppose you need the get the age of the employee based on the date of birth, you may create a function that receives date as the input and returns age of the employee as the output.



**1.Make sure stored procedure available in the database**

**2.Create a callable Statements with the procedure call.**

**Syntax: CallableStatement   cst = con.prepareCall ("{ call procedurename(?,?...?)}");**

**Example:  CallableStatement  cst = con .prepareCall ("{ call sum4 (?,?...?)}");**

**3.Provide values for every IN Parameter by using corresponding setter method**

**cst.setInt (1, 100);**

**cst.setInt (2, 200);**

**↓ ↓**

**Index value**

**4. If stored procedure has OUT parameter then to hold that output we should register every OUT parameter by using the following methods.**

**Syntax : Public void registerOutParameter (int index, int jdbctype)**

**Example : cst. registerOutParameter(3, Types.INTEGER);**

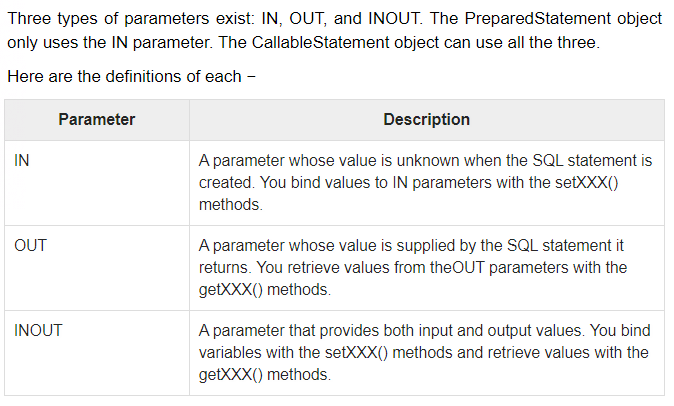
**5. EXECUTE Procedure call**

**cst.execute();**

**6.Get the result from OUT parameter by using that corresponding getXXX() method.**

**Example: int result =cst.getInt(3);**

**7. Close connection.**



**PROCEDURE:**

1. **import** java.sql.\*;
2. **public** **class** Proc {
3. **public** **static** **void** main(String[] args) **throws** Exception{
5. Class.forName("oracle.jdbc.driver.OracleDriver");
6. Connection con=DriverManager.getConnection(
7. "jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
9. CallableStatement stmt=con.prepareCall("{call insertR(?,?)}");
10. stmt.setInt(1,1011);
11. stmt.setString(2,"Amit");
12. stmt.execute();
14. System.out.println("success");
15. }
16. }

**FUNCTION:**

1. **import** java.sql.\*;
3. **public** **class** FuncSum {
4. **public** **static** **void** main(String[] args) **throws** Exception{
6. Class.forName("oracle.jdbc.driver.OracleDriver");
7. Connection con=DriverManager.getConnection(
8. "jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
9. CallableStatement stmt=con.prepareCall("{?= call sum4(?,?)}");
10. stmt.setInt(2,10);
11. stmt.setInt(3,43);
12. stmt.registerOutParameter(1,Types.INTEGER);
13. stmt.execute();
15. System.out.println(stmt.getInt(1));
17. }
18. }

**RESULTSET**

The SQL statements that read data from a database query, return the data in a result set. The SELECT statement is the standard way to select rows from a database and view them in a result set. The *java.sql.ResultSet* interface represents the result set of a database query.

A ResultSet object maintains a cursor that points to the current row in the result set. The term "result set" refers to the row and column data contained in a ResultSet object.

The methods of the ResultSet interface can be broken down into three categories −

**Navigational methods** − Used to move the cursor around.

**Get methods** − Used to view the data in the columns of the current row being pointed by the cursor.

**Update methods** − Used to update the data in the columns of the current row. The updates can then be updated in the underlying database as well.

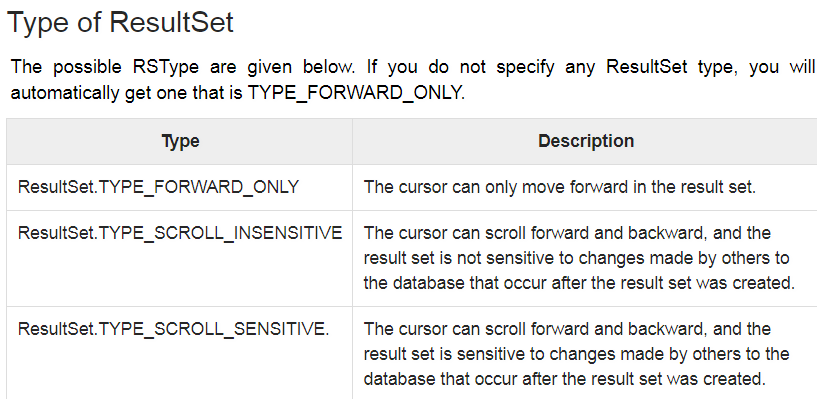
JDBC provides the following connection methods to create statements with desired ResultSet −

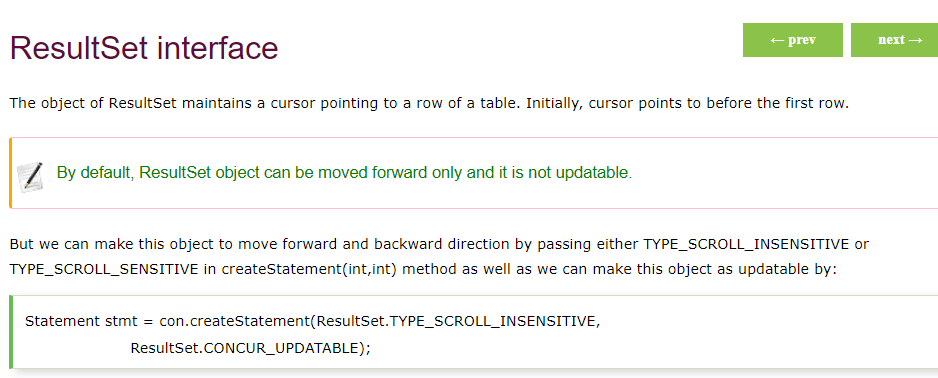
**createStatement(int RSType, int RSConcurrency);**

**prepareStatement(String SQL, int RSType, int RSConcurrency);**

**prepareCall(String sql, int RSType, int RSConcurrency);**

The first argument indicates the type of a ResultSet object and the second argument is one of two ResultSet constants for specifying whether a result set is read-only or updatable.

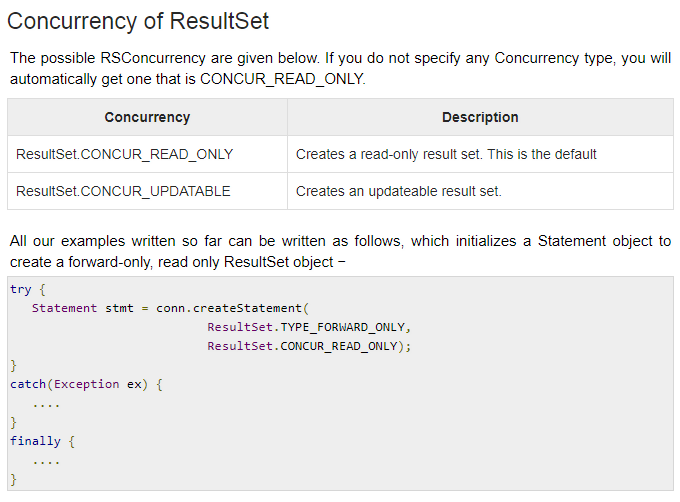


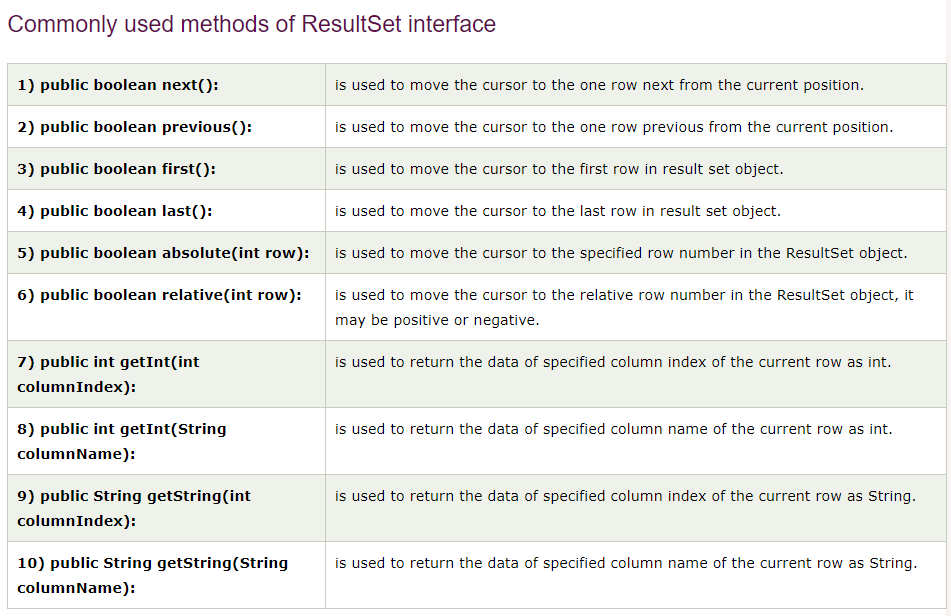


## Concurrency of ResultSet

The possible RSConcurrency are given below. If you do not specify any Concurrency type, you will automatically get one that is CONCUR\_READ\_ONLY.

|  |  |
| --- | --- |
| **Concurrency** | **Description** |
| ResultSet.CONCUR\_READ\_ONLY | Creates a read-only result set. This is the default |
| ResultSet.CONCUR\_UPDATABLE | Creates an updateable result set. |





## **Navigating a Result Set**

There are several methods in the ResultSet interface that involve moving the cursor, including −

|  |  |
| --- | --- |
| **S.N.** | **Methods & Description** |
| 1 | **public void beforeFirst() throws SQLException**  Moves the cursor just before the first row. |
| 2 | **public void afterLast() throws SQLException**  Moves the cursor just after the last row. |
| 3 | **public boolean first() throws SQLException**  Moves the cursor to the first row. |
| 4 | **public void last() throws SQLException**  Moves the cursor to the last row. |
| 5 | **public boolean absolute(int row) throws SQLException**  Moves the cursor to the specified row. |
| 6 | **public boolean relative(int row) throws SQLException**  Moves the cursor the given number of rows forward or backward, from where it is currently pointing. |
| 7 | **public boolean previous() throws SQLException**  Moves the cursor to the previous row. This method returns false if the previous row is off the result set. |
| 8 | **public boolean next() throws SQLException**  Moves the cursor to the next row. This method returns false if there are no more rows in the result set. |
| 9 | **public int getRow() throws SQLException**  Returns the row number that the cursor is pointing to. |
| 10 | **public void moveToInsertRow() throws SQLException**  Moves the cursor to a special row in the result set that can be used to insert a new row into the database. The current cursor location is remembered. |
| 11 | **public void moveToCurrentRow() throws SQLException**  Moves the cursor back to the current row if the cursor is currently at the insert row; otherwise, this method does nothing |

## **Viewing a Result Set**

The ResultSet interface contains dozens of methods for getting the data of the current row.

There is a get method for each of the possible data types, and each get method has two versions −

One that takes in a column name.

One that takes in a column index.

For example, if the column you are interested in viewing contains an int, you need to use one of the getInt() methods of ResultSet −

|  |  |
| --- | --- |
| **S.N.** | **Methods & Description** |
| 1 | **public int getInt(String columnName) throws SQLException**  Returns the int in the current row in the column named columnName. |
| 2 | **public int getInt(int columnIndex) throws SQLException**  Returns the int in the current row in the specified column index. The column index starts at 1, meaning the first column of a row is 1, the second column of a row is 2, and so on. |

Similarly, there are get methods in the ResultSet interface for each of the eight Java primitive types, as well as common types such as java.lang.String, java.lang.Object, and java.net.URL.

There are also methods for getting SQL data types java.sql.Date, java.sql.Time, java.sql.TimeStamp, java.sql.Clob, and java.sql.Blob. Check the documentation for more information about using these SQL data types.

For a better understanding, let us study [Viewing - Example Code](https://www.tutorialspoint.com/jdbc/viewing-result-sets.htm).

## Updating a Result Set

The ResultSet interface contains a collection of update methods for updating the data of a result set.

As with the get methods, there are two update methods for each data type −

One that takes in a column name.

One that takes in a column index.

For example, to update a String column of the current row of a result set, you would use one of the following updateString() methods −

|  |  |
| --- | --- |
| **S.N.** | **Methods & Description** |
| 1 | **public void updateString(int columnIndex, String s) throws SQLException**  Changes the String in the specified column to the value of s. |
| 2 | **public void updateString(String columnName, String s) throws SQLException**  Similar to the previous method, except that the column is specified by its name instead of its index. |

There are update methods for the eight primitive data types, as well as String, Object, URL, and the SQL data types in the java.sql package.

Updating a row in the result set changes the columns of the current row in the ResultSet object, but not in the underlying database. To update your changes to the row in the database, you need to invoke one of the following methods.

|  |  |
| --- | --- |
| **S.N.** | **Methods & Description** |
| 1 | **public void updateRow()**  Updates the current row by updating the corresponding row in the database. |
| 2 | **public void deleteRow()**  Deletes the current row from the database |
| 3 | **public void refreshRow()**  Refreshes the data in the result set to reflect any recent changes in the database. |
| 4 | **public void cancelRowUpdates()**  Cancels any updates made on the current row. |
| 5 | **public void insertRow()**  Inserts a row into the database. This method can only be invoked when the cursor is pointing to the insert row. |

Assignment No 4: JDBC

1. Explain Connectivity steps for JDBC
2. Explain types of JDBC Drivers in detail
3. Describe how to Insert, Delete, and Update record in the result set.
4. Give the use of connection and statement interface
   1. getConnection()
   2. CreateStatement()
   3. executerStatement()
   4. getResultSet()
5. List and explain JDBC API.
6. Explain two and three tier architecture in detail.
7. Explain following interfaces with its methods
   1. Connection Interface
   2. ResultSet
   3. Statement
   4. Callable statement
   5. Prepared statement
8. Explain Connectivity steps for JDBC
9. WAP to create employee table in database having two columns “emp\_id” and “emp\_name”.
10. Develop a program to display name and roll\_no of student from STUDENT table having percentage>70
11. Develop a program to change name of product from “TV” to “COMPUTER” in PRODUCTS table.
12. Create database and write java programs

Columns :- company ,number, cost , memory capacity

DSN :-Mobile

Records;- 05

Using above database determine total number of mobile belongs to ‘ABC’ company .Arrange them in ascending order.

1. WAP for to create database to insert a record into employee table and delete a record and update recorded for same table(Using Access, Mysql and Oracle database)
2. WAP for Prepared Statement (Insert ,Delete AND Update )
3. WAP for callable Statement to insert a record name, id ,address and mobile no into student table.
4. WAP for callable Statement to calculate factorial of number .