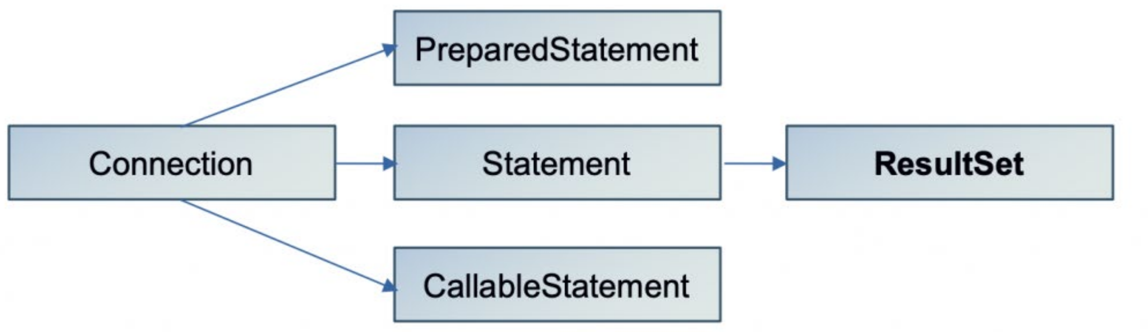
Java **ResultSet** interface is a part of the java.sql package. It is one of the core components of the JDBC Framework. ResultSet Object is used to access query results retrieved from the relational databases.

ResultSet maintains cursor/pointer which points to a single row of the query results. Using navigational and getter methods provided by ResultSet, we can iterate and access database records one by one. ResultSet can also be used to update data.

Java ResultSet Hierarchy



Java ResultSet Class Hierarchy
The above diagram shows the place of ResultSet in the JDBC Framework. **ResultSet** can be obtained by executing SQL Query using **Statement**, **PreparedStatement** or **CallableStatement**. **AutoCloseable**, **Wrapper** are super interfaces of ResultSet. Now we will see how to work with ResultSet in our Java programs.

ResultSet Example

We will be using MySQL for our example purpose. Use below DB script to create a database and table along with some records.

```
create database empdb;

use empdb;

create table tblemployee (empid integer primary key, firstname
varchar(32), lastname varchar(32), dob date);

insert into tblemployee values  (1, 'Mike', 'Davis',' 1998-11-11');
insert into tblemployee values  (2, 'Josh', 'Martin', '1988-10-22');
insert into tblemployee values  (3, 'Ricky', 'Smith', '1999-05-11');
```
Copy

Let’s have look at the below example program to fetch the records from the table and print them on the console. Please make sure you have the MySQL JDBC driver in the project classpath.

```
package com.journaldev.examples;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Date;


public class ResultSetDemo {

        public static void main(String[] args) {
                String query = "select empid, firstname, lastname, dob
from tblemployee";
                Connection conn = null;
                Statement stmt = null;
                try {
                        Class.forName("com.mysql.jdbc.Driver");
                        conn =
DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/empdb",
"root", "root");
                        stmt = conn.createStatement();
                        ResultSet rs = stmt.executeQuery(query);
                        while (rs.next()) {
                                Integer empId = rs.getInt(1);
                                String firstName = rs.getString(2);
                                String lastName = rs.getString(3);
                                Date dob = rs.getDate(4);
                                System.out.println("empId:" + empId);
                                System.out.println("firstName:" +
firstName);
                                System.out.println("lastName:" +
lastName);
                                System.out.println("dob:" + dob);
                                System.out.println("");
                        }
                        rs.close();
                } catch (Exception e) {
                        e.printStackTrace();
                } finally {
                        try {
                                stmt.close();
                                conn.close();
                        } catch (Exception e) {}
                }
        }
}
```
Output:
```
empId:1
firstName:Mike
lastName:Davis
dob:1998-11-11

empId:2
firstName:Josh
lastName:Martin
dob:1988-10-22

empId:3
firstName:Ricky
lastName:Smith
dob:1999-05-11
```
Explanation:

- ResultSet is obtained by calling the executeQuery method on Statement instance. Initially, the cursor of ResultSet points to the position before the first row.
- The method next of ResultSet moves the cursor to the next row. It returns true if there is further row otherwise it returns false.
- We can obtain data from **ResultSet** using getter methods provided by it. e.g. getInt(), getString(), getDate()
- All the getter methods have two variants. 1st variant takes column index as Parameter and 2nd variant accepts column name as Parameter.
- Finally, we need to call **close** method on **ResultSet** instance so that all resources are cleaned up properly.

ResultSet Types & Concurrency

We can specify type and concurrency of ResultSet while creating an instance of Statement, PreparedStatement or CallableStatement.
statement.createStatement(int resultSetType, int resultSetConcurrency)

ResultSet Types

1) Forward Only (ResultSet.*TYPE_FORWARD_ONLY*)

This type of ResultSet instance can move only in the forward direction from the first row to the last row. ResultSet can be moved forward one row by calling the next() method. We can obtain this type of ResultSet while creating Instance of Statement, PreparedStatement or CallableStatement.

```
Statement stmt =
connection.createStatement(ResultSet.TYPE_FORWARD_ONLY,
ResultSet.CONCUR_READ_ONLY);
ResultSet rs = stmt.executeQuery("select * from tbluser");
```
Copy

2) Scroll Insensitive (ResultSet.*TYPE_SCROLL_INSENSITIVE*)

Scroll Insensitive ResultSet can scroll in both forward and backward directions. It can also be scrolled to an absolute position by calling the absolute() method. But it is not sensitive to data changes. It will only have data when the query was executed and ResultSet was obtained. It will not reflect the changes made to data after it was obtained.

```
Statement stmt =
connection.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
       ResultSet.CONCUR_READ_ONLY);
ResultSet rs = stmt.executeQuery("select * from tbluser");
```
Copy

3) Scroll Sensitive (ResultSet.*TYPE_SCROLL_SENSITIVE)*

Scroll Sensitive ResultSet can scroll in both forward and backward directions. It can also be scrolled to an absolute position by calling the absolute() method. But it is sensitive to data changes. It will reflect the changes made to data while it is open.

```
Statement stmt =
connection.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
       ResultSet.CONCUR_READ_ONLY);
ResultSet rs = stmt.executeQuery("select * from tbluser");
```
Copy

ResultSet Concurrency

1) Read Only (ResultSet.*CONCUR_READ_ONLY)*

It is the default concurrency model. We can only perform Read-Only operations on ResultSet Instance. No update Operations are allowed.

2) Updatable (ResultSet.*CONCUR_UPDATABLE)*

In this case, we can perform update operations on ResultSet instance.

ResultSet Methods

We can divide ResultSet methods into the following categories.

- **Navigational Methods**
- **Getter/Reader Methods**
- **Setter/Updater Methods**
- **Miscellaneous Methods** - close() and getMetaData()

1. ResultSet Navigational Methods

- **boolean** absolute(**int** row) **throws** SQLException**:** This method moves ResultSet cursor to the specified row and returns true if the operation is successful.
- **void** afterLast() **throws** SQLException**:** This method moves ResultSet cursor to the position after the last row.
- **void** beforeFirst() **throws** SQLException**:** This method moves ResultSet cursor to the position before the first row.
- **boolean** first() **throws** SQLException: This method moves ResultSet cursor to the first row.
- **boolean** last() **throws** SQLException: This method moves ResultSet cursor to the last row.
- **boolean** next() **throws** SQLException: This method moves ResultSet cursor to the next row.
- **boolean** previous() **throws** SQLException: This method moves ResultSet cursor to the previous row.

```
package com.journaldev.examples;
import java.sql.*;

public class ResultSetDemo {

       public static void main(String[] args) {
              String query = "select empid, firstname, lastname, dob
from tblemployee";
              Connection conn = null;
              Statement stmt = null;
              try {
                     Class.forName("com.mysql.jdbc.Driver");
                     conn =
DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/empdb",
"root", "root");
                     stmt =
conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);
                     ResultSet rs = stmt.executeQuery(query);
```

```
                        System.out.println("All the rows of table=>");
                        while (rs.next()) {
                                // Go to next row by calling next() method
                                displayData(rs);
                        }
                        System.out.println("Now go directly to 2nd 
row=>");
                        rs.absolute(2); // Go directly to 2nd row
                        displayData(rs);
                        System.out.println("Now go to Previous row=>");
                        rs.previous();
                        // Go to 1st row which is previous of 2nd row
                        displayData(rs);
                        rs.close();
                } catch (Exception e) {
                        e.printStackTrace();
                } finally {
                        try {
                                stmt.close();
                                conn.close();
                        } catch (Exception e) {
                        }
                }
        }

        public static void displayData(ResultSet rs) throws SQLException
{
                System.out.println("empId:" + rs.getInt(1));
                System.out.println("firstName:" + rs.getString(2));
                System.out.println("lastName:" + rs.getString(3));
                System.out.println("dob:" + rs.getDate(4));
                System.out.println("");
        }
}
```
Copy

Output:

```
All the rows of table=>
empId:1
firstName:Mike
lastName:Davis
dob:1998-11-11
```

```
empId:2
firstName:Josh
lastName:Martin
dob:1988-10-22

empId:3
firstName:Ricky
lastName:Smith
dob:1999-05-11

Now go directly to 2nd row=>
empId:2
firstName:Josh
lastName:Martin
dob:1988-10-22

Now go to Previous row=>
empId:1
firstName:Mike
lastName:Davis
dob:1998-11-11
```
Copy

2. ResultSet Getter/Reader Methods

- **int** getInt(**int** columnIndex) **throws** SQLException: This method returns value of specified columnIndex as **int**.
- **long** getLong(**int** columnIndex) **throws** SQLException: This method returns value of specified columnIndex as **long**
- String getString(**int** columnIndex) **throws** SQLException: This method returns value of specified columnIndex as **String**
- java.sql.Date getDate(**int** columnIndex) **throws** SQLException: This method returns value of specified columnIndex as **java.sql.Date**
- **int** getInt(String columnLabel) **throws** SQLException: This method returns value of specified column name as **int**.
- **long** getLong(String columnLabel) **throws** SQLException: This method returns value of specified column name as **long**.
- String getString(String columnLabel) **throws** SQLException: This method returns the value of the specified column name as String.
- java.sql.Date getDate(String columnLabel) throws SQLException: This method returns the value of the specified column name as **java.sql.Date**.
- ResultSet contains getter methods that return other primitive datatypes like boolean, float and double. It also has methods to obtain array and binary data from the database.

3. ResultSet Setter/Updater Methods

- **void** updateInt(**int** columnIndex, **int** x) **throws** SQLException: This method updates the value of specified column of current row with **int** value.
- **void** updateLong(**int** columnIndex, **long** x) **throws** SQLException: This method updates the value of the specified column of the current row with long value.
- void updateString(int columnIndex, String x) throws SQLException: This method updates the value of the specified column of the current row with a String value.
- **void** updateDate(**int** columnIndex, java.sql.Date x) **throws** SQLException: This method updates the value of specified column of current row with java.sql.Date value.
- void updateInt(String columnLabel, int x) throws SQLException: This method updates the value of the specified column label of the current row with int value.
- void updateLong(String columnLabel, long x) throws SQLException: This method updates the value of the specified column label of the current row with long value.
- void updateString(String columnLabel, String x) throws SQLException: This method updates the value of the specified column label of the current row with a String value.
- **void** updateDate(String columnLabel, java.sql.Date x) **throws** SQLException: This method updates the value of specified columnLabel of current row with java.sql.Date value.

Note: Setter/Updater Methods doesn't directly update database values. **Database values will be inserted/updated after calling the insertRow or updateRow method**.

```
package com.journaldev.examples;
import java.sql.*;



public class ResultSetUpdateDemo {

       public static void main(String[] args) {
              String query = "select empid, firstname, lastname, dob
from tblemployee";
              Connection conn = null;
              Statement stmt = null;
              try {
                     Class.forName("com.mysql.jdbc.Driver");
                     conn =
DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/empdb",
"root", "root");
```

```
                        stmt =
conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_UPDATABLE);
                        ResultSet rs = stmt.executeQuery(query);
                        System.out.println("Now go directly to 2nd row
for Update");
                        if (rs.absolute(2)) {
                                // Go directly to 2nd row
                                System.out.println("Existing Name:" +
rs.getString("firstName"));
                                rs.updateString("firstname", "Tyson");
                                rs.updateRow();
                        }
                        rs.beforeFirst(); // go to start
                        System.out.println("All the rows of table=>");
                        while (rs.next()) {
                        // Go to next row by calling next() method
                                displayData(rs);
                        }
                        rs.close();
                } catch (Exception e) {
                        e.printStackTrace();
                } finally {
                        try {
                                stmt.close();
                                conn.close();
                        } catch (Exception e) {
                        }
                }
        }

        public static void displayData(ResultSet rs) throws SQLException
{
                System.out.println("empId:" + rs.getInt(1));
                System.out.println("firstName:" + rs.getString(2));
                System.out.println("lastName:" + rs.getString(3));
                System.out.println("dob:" + rs.getDate(4));
                System.out.println("");
        }
}
```

Output:

```
Now go directly to 2nd row for Update
```

```
Existing Name:Josh
All the rows of table=>
empId:1
firstName:Mike
lastName:Davis
dob:1998-11-11

empId:2
firstName:Tyson
lastName:Martin
dob:1988-10-22

empId:3
firstName:Ricky
lastName:Smith
dob:1999-05-11
```

4. ResultSet Miscellaneous Methods

- **void** close() **throws** SQLException**:** This method frees up resources associated with ResultSet Instance. It must be called otherwise it will result in resource leakage.
- **ResultSetMetaData** getMetaData() **throws** SQLException: This method returns ResultSetMetaData instance. It gives information about the type and property of columns of the query output.