SavePoint(I)

Within a transaction, if we want to rollback a particular group of operations based on some condition then we need to use savepoint

a. getting savepoint using the following method

Savepoint sp = connection.setSavePoint();

b. To perform rollback operation for a particular group of operations w.r.t savepoint then we need to use rollback.

connection.rollback(sp);

c. we can release the savepoint or delete the savepoint as shown below

connection.releaseSavePoint(sp);

Case study

====================================

connection.setAutocommit(false);

operation-1

operation-2

operation-3

SavePoint sp = connection.setSavePoint();

operation-4

operation-5

if(balance<1000)

connection.rollback(sp);

else

connection.releaseSavepoint(sp);

connection.commit();

if balance< 1000 then operation 4,5 will be rollbacked, otherwise all the operations will be committed.

eg:

connection.setAutocommit(false);

st.executeUpdate("insert into politicians values('BJP', 'Modi'));

st.executeUpdate("insert into politicians values('TRS','KCR')");

SavePoint sp = connection.setSavePoint();

st.executeUpdate("insert into politicians values('BJP','siddu')");

connection.rollback(sp);

..

...

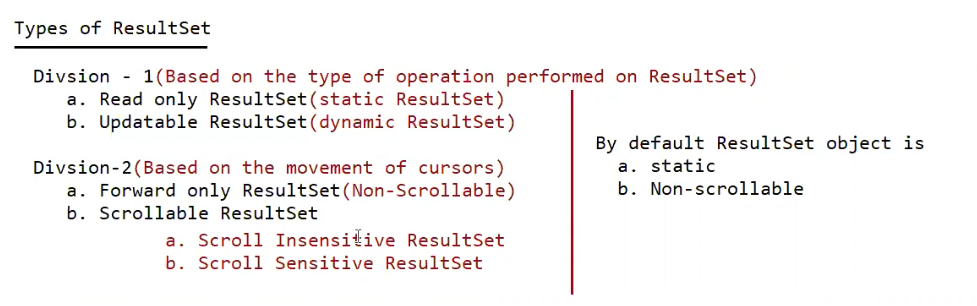
connection.releaseSavePoint(sp);

..

connection.commit();

Eg: Jdbc\_Transaction\_Savepoint\_Rollback

Types of ResultSet:



create table scrollableapp ( id int , name nvarchar2(15) , age int , address nvarchar2(15));

insert into scrollableapp values ( 1 , 'pavan' , 24 , 'vijayawada');

insert into scrollableapp values ( 2 , 'anand' , 23 , 'vijayawada');

insert into scrollableapp values ( 3 , 'pavankumar' , 24 , 'vijayawada');

insert into scrollableapp values ( 4 , 'charan' , 23 , 'ongole');

insert into scrollableapp values ( 5 , 'tarun' , 24 , 'vijayawada');

insert into scrollableapp values ( 6 , 'mpavan' , 24 , 'vijayawada');

insert into scrollableapp values ( 7 , 'ppavan' , 24 , 'vijayawada');

insert into scrollableapp values ( 8 , 'teja' , 24 , 'vijayawada');

absolute() -> it works from the BFR or from ALR.

relative() -> it works w.r.t current position.

In both the methods positive means move in forward direction, negative means move in backward direction.

Note:

rs.last() and rs.absolute(-1) both are equal

rs.first() and rs.absolute(1) both are equal

Eg: Scrollable\_App

//go through the code

Scroll InSensitive ResultSet

After getting resultSet if we are performing any operations in the database, and if those changes are not reflecting to the resultSet, such type of ResultSet are called as "Scroll Insensitive ResutlSet".

public static final int TYPE\_SCROLL\_INSENSITIVE

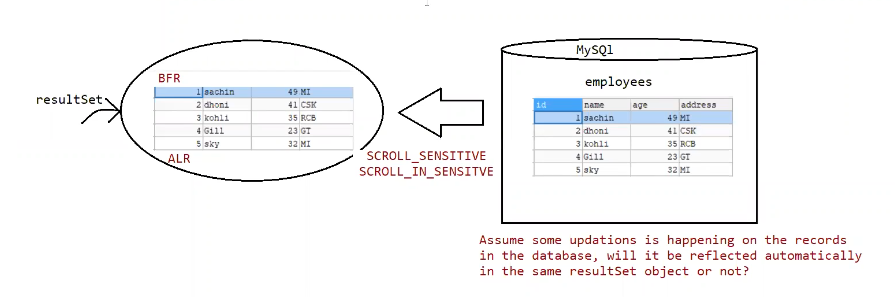
Scroll Sensitive ResultSet

After getting resultSet if we are performing any operations in the database, and if those changes are getting reflecting to the resultSet, such type of ResultSet are called as "Scroll Sensitive ResutlSet”.

public static final int TYPE\_SCROLL\_SENSITIVE

Note:

When we make the ResultSet as ScrollSensitive, then we need to use resultSet.refreshRow() to get the updated records from the database.



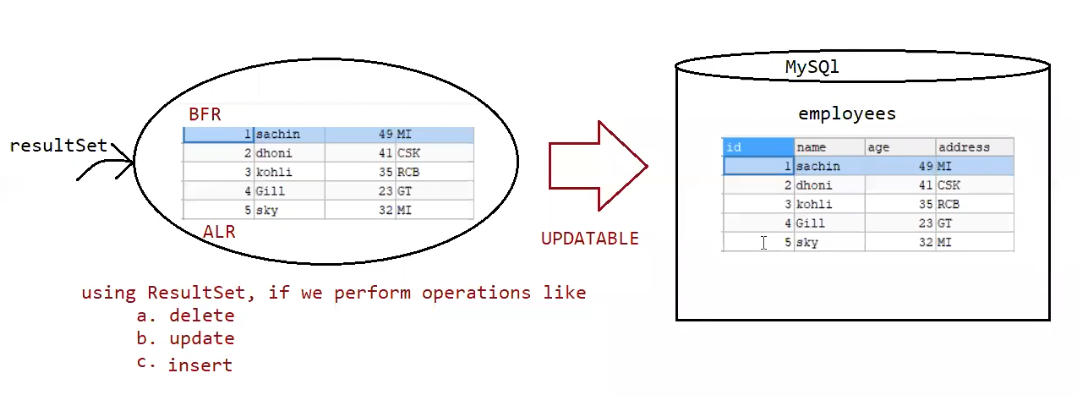
Eg: Scrollable\_Sensitive\_App

// go through the code

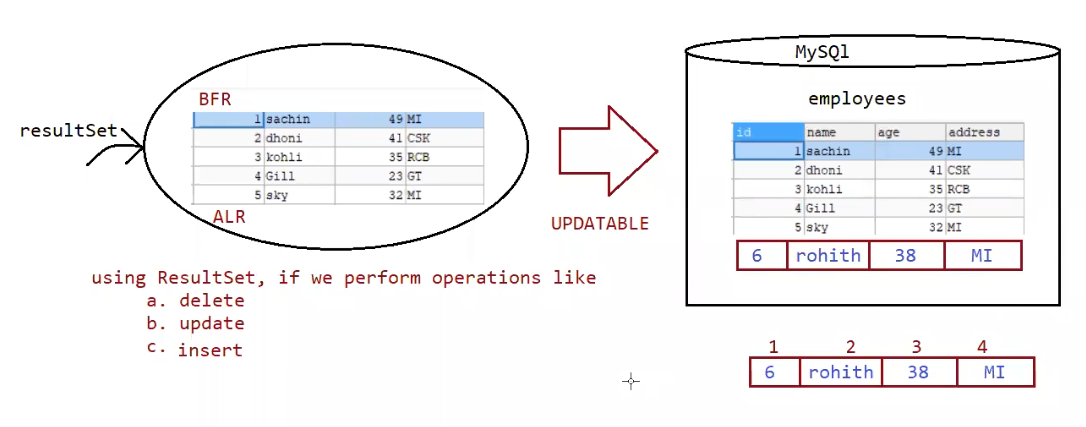
Updatable ResultSet:

It is possible to perform delete operation using resultSet without writing delete query

resultSet.deleteRow() -> This method would delete the record based on the cursor position of ResultSet.



Eg: Scrollable\_Sensitive\_App\_Row\_Deletion



It is possible to perform insert operation using resultSet without writing insert query

a. resultSet.moveTolnsertRow() //creates a empty record

b. resultSet.updateXXXX(int pos, Object value); // insert the value based on the column data

c. resultSet.insertRow(); // record will be inserted to the table with the updated values.

Eg: Scrollable\_Sensitive\_App\_Row\_Insertion

It is possible to peform update operation using resultSet without writing update query

a. resultSet.getXXXX(int pos) // getting the old value from resultSet

b. resultSet.updateXXXX(int pos, Object newValue); // update the newValue w.r.t resultSet

c. resultSet.updateRow();// record will be updated to the table as per the resultSet information.

Working with Excel sheet

To work with Excel sheet, we need to use HXTT company supplied driver.

eg: EXCEL\_JDBC40.jar

Working with CSV files

To work with CSV files, we need to use HXTT company supplied driver.

eg: Text\_JDBC42.jar

Eg: Jdbc\_ExcelFile\_Select\_Operation

//go through the code

Eg: Jdbc\_ExcelFile\_Insert\_Operation

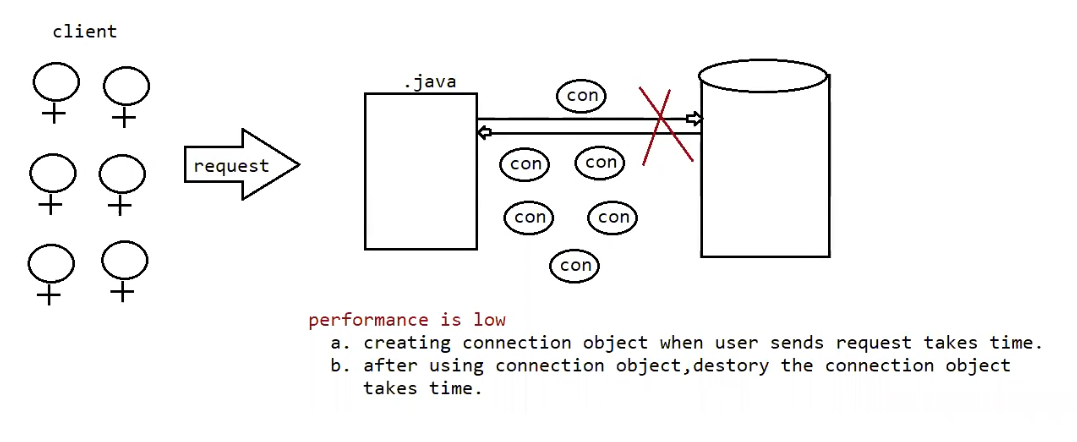
//go through the code

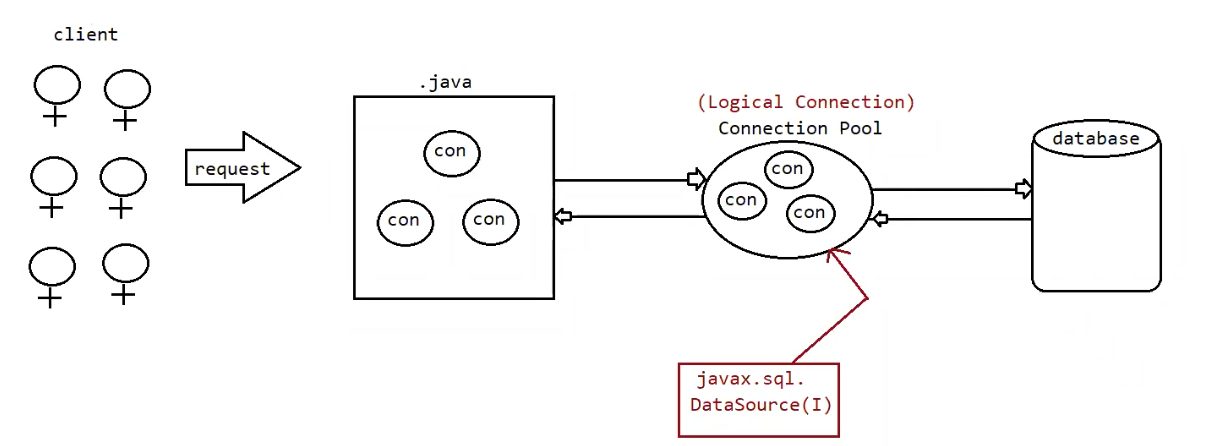
Eg: Jdbc\_CsvFile\_Select\_Operation

//go through the code

Eg: Jdbc\_CsvFile\_Insert\_Operation

//go through the code





Advantages

1. We need not want to make clients wait to get the connection object, if connection object is

available pick from connection pool and give the connection object for service.

2. After using the connection object, connection object will be returned back to connection pool so it supports reusability.

The no of objects created in the connection pool depends of the dbvendor and since the count of objects is less(10),it is not suitable for production environment.

DBVendors also supplied "Conneciton poooling" mechanism in the jars.

The best suited vendor who supplies connection pooling mechanism is "hikaricp".

For MySQL database to implement Connection pooling we need to use the class called

"com.mysql.cj.jdbc.MysqlConnectionPoolDataSource"

Oracle Connection Pooling Code Sample

This example first creates an OracleConnectionPoolDataSource instance, next initializes its connection properties, then gets a pooled connection instance from the connection pool data source instance, and finally gets a connection instance from the pooled connection instance. (The getPooledConnection() method actually returns an OraclePooledConnection instance, but in this case only generic PooledConnection functionality is required.)

...

OracleConnectionPoolDataSource ocpds = new OracleConnectionPoolDataSource();

ocpds.setDriverType("oci");

ocpds.setServerName("dlsun999");

ocpds.setNetworkProtocol("tcp");

ocpds.setDatabaseName("816");

ocpds.setPortNumber(1521);

ocpds.setUser("scott");

ocpds.setPassword("tiger");

PooledConnection pc = ocpds.getPooledConnection();

Connection conn = pc.getConnection();

Eg: Connection\_Pool

// go through the code

HikariCp:

HikariCP is a "zero-overhead" production ready JDBC connection pool.

It is the 3rd party supplied jar which helps us to imporve connection pooling apporach.

To use hikaricp in our applications we need to use 2 jars

a. hikaricp jar

b. sl4j jar

Eg: Pooled\_Connection\_Hikaricp

//go through the code

Note:

Since hikaricp datasource is best for connection pooling, In Spring, SpringBoot by default hikaricp connection pooling is available.

javax.sql.RowSet :

====================

It is an alternative to ResultSet.

We can use RowSet to handle group of recordds in more effective way than ResultSet.

RowSet is a child interface of ResultSet.

By default RowSet is scrollable and updatable.

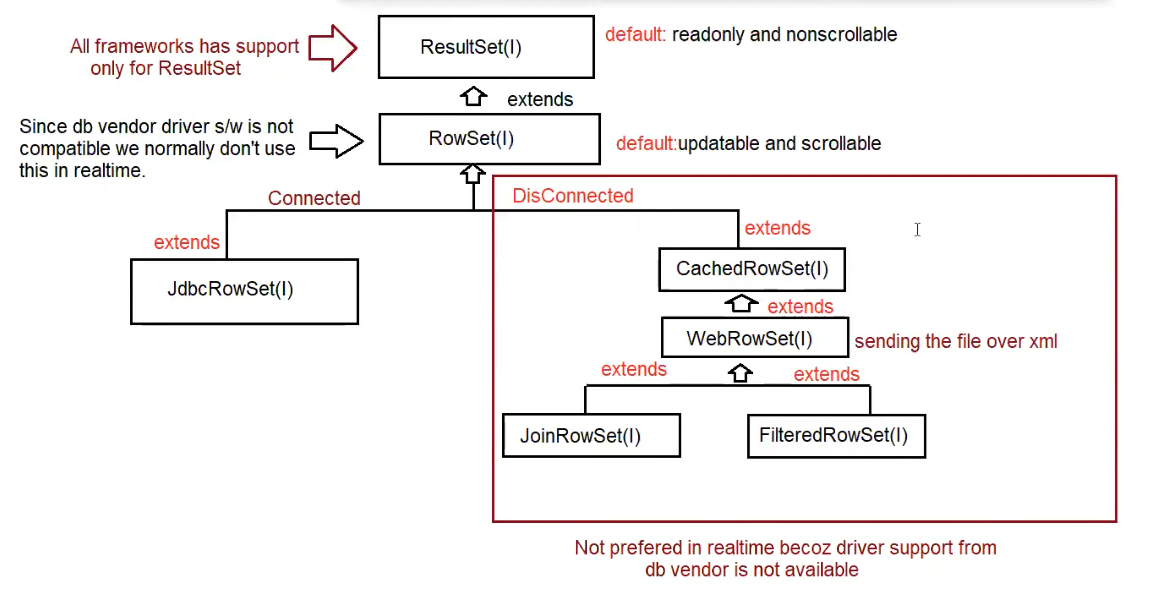
By default RowSet object implements Serializable, so RowSet object can be sent over the network.

ResultSet object is by default Connected object, where as if use RowSet we can work in both the modes like Connected/ DisConnected mode.

There are 2 types of RowSet

a. Connected RowSet.

b. DisConnected RowSet.



RowSet Implementation is given by the sun-micro systems, so there are many limitations

Eg: RowSet\_Eg1

// go through the code

Note: Implementation for RowSet is provided by java vendor only, not by db vendors.

Eg: Jdbc\_RowSet\_Select\_Operation

// go through the code

Eg: Jdbc\_RowSet\_Insert\_Operation

// go through the code

ID NAME AGE ADDRESS

---------- -------------------- ---------- ---------------

1 pavan 24 vijayawada

2 dpavan 24 vijayawada

3 anand 24 vijayawada

Eg: Jdbc\_RowSet\_Update\_Operation

// go through the code

ID NAME AGE ADDRESS SALARY

---------- -------------------- ---------- --------------- ----------

1 pavan 24 vijayawada 15000

2 dpavan 24 vijayawada 15000

3 anand 24 vijayawada 15000

Eg: Jdbc\_RowSet\_Delete\_Operation

// go through the code

Eg: Cached\_RowSet\_Select\_Operation

// go through the code

Eg: Cached\_RowSet\_Select\_Operation\_Scrollable

// go through the code

Eg: Cached\_RowSet\_Insert\_Operation

// go through the code