https://docs.oracle.com/cd/B28359_01/java.111/b31224/resltset.htm

Scrollability refers to the ability to move backward as well as forward through a result set. Associated with scrollability is the ability to move to any particular position in the result set, through either relative positioning or absolute positioning.

A sensitive result set can see changes made to the database while the result set is open, providing a dynamic view of the underlying data. Changes made to the underlying columns values of rows in the result set are visible.

To summarize, the following result set types are available with JDBC 2.0:

- Forward-only

  This is a JDBC 1.0 functionality. This type of result set is not scrollable, not positionable, and not sensitive.

- Scroll-insensitive

  This type of result set is scrollable and positionable, but not sensitive to underlying database changes.

- Scroll-sensitive

  This type of result set is scrollable and positionable. It is also sensitive to underlying database changes.

You can specify one of the following `static` constant values for result set type:

- `ResultSet.TYPE_FORWARD_ONLY`
- `ResultSet.TYPE_SCROLL_INSENSITIVE`
- `ResultSet.TYPE_SCROLL_SENSITIVE`

Updatability

Updatability refers to the ability to update data in a result set and then copy the changes to the database. This includes inserting new rows into the result set or deleting existing rows.

Under JDBC 2.0, the following concurrency types are available:

- Read-only

  The result set cannot be modified in any way.

- Updatable

  In this case, updates, inserts, and deletes can be performed on the result set and copied to the database.

you can specify one of the following `static` constant values for concurrency type:

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

Summary of Result Set Categories

Because scrollability and sensitivity are independent of updatability, the three result set types and two concurrency types combine for a total of six result set categories, as follows:

- forward-only/read-only
- forward-only/updatable
- scroll-sensitive/read-only
- scroll-sensitive/updatable
- scroll-insensitive/read-only
- scroll-insensitive/updatable

Positioning in a Scrollable Result Set

```
void beforeFirst() throws SQLException
void afterLast() throws SQLException
boolean first() throws SQLException
boolean last() throws SQLException

boolean absolute(int row) throws SQLException

boolean relative(int row) throws SQLException
```

Methods for Checking the Current Position

```
boolean isBeforeFirst() throws SQLException

boolean isAfterLast() throws SQLException


boolean isFirst() throws SQLException


boolean isLast() throws SQLException


int getRow() throws SQLException
```

```
To process the entire result set going backward, call afterLast(), then use the
previous() method.
For example:

/* NOTE: The specified concurrency type, CONCUR_UPDATABLE, is not relevant to
this example. */


Statement stmt = conn.createStatement
         (ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);
```

```
ResultSet rs = stmt.executeQuery("SELECT empno, sal FROM emp");

rs.afterLast();
while (rs.previous())
{
   System.out.println(rs.getString("empno") + " " + rs.getFloat("sal"));
}
```

Example

Following is an example of a result set `UPDATE` operation that is also copied to the database. The tenth row is updated. The column number is used to specify column 1, and the column name, `sal`, is used to specify column 2.

```
Statement stmt = conn.createStatement
         (ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);
ResultSet rs = stmt.executeQuery("SELECT empno, sal FROM emp");
if (rs.absolute(10))        // (returns false if row does not exist)
{
   rs.updateString(1, "28959");
   rs.updateFloat("sal", 100000.0f);
   rs.updateRow();
}
// Changes are made permanent with the next COMMIT operation.
```

Example

The following example performs a result set `INSERT` operation, moving to the insert-row, writing the data, copying the data into the database, and then returning to what was the current row prior to going to the insert-row. The column number is used to specify column 1, and the column name, `sal`, is used to specify column 2.

```
...
Statement stmt = conn.createStatement
         (ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);

ResultSet rs = stmt.executeQuery("SELECT empno, sal FROM emp");

rs.moveToInsertRow();
rs.updateString(1, "28959");
rs.updateFloat("sal", 100000.0f);
rs.insertRow();
// Changes will be made permanent with the next COMMIT operation.
rs.moveToCurrentRow();  // Go back to where we came from...
...
```

Example

Delete row

```
...
rs.absolute(5);
rs.deleteRow();
...Oracle JDBC drivers use the ROWID to uniquely identify a row in a database
table. As long as the ROWID is valid when a driver tries to send an UPDATE or
DELETE operation to the database, the operation will be run.
```