JDBC: A Focus on the ResultSet Class

CMPUT 291

File and Database Management Systems

Objectives:

- ✔ ResultSet Class
- ✔ Creating a scrollable ResultSet
- ✔ Scroll in a result set
- ✔ Updating rows with a ResultSet
- ✔ Inserting and Deleting rows
- ✔ Using bind variables in Python
- ✔ Working through the JDBC Tutorial #2

ResultSet Class:

- A result set contains the result of a query to the database and is obtained as a ***ResultSet*** object.
- A result set can be read sequentially from the ***ResultSet*** object using the method ***next()***, for example,

  rset.next()

  where ***rset*** is an object of class ***ResultSet***.

Creating a ResultSet object with desired properties:

- Use the method ***createStatement*** from class ***Connection*** to specify properties of the ***ResultSet*** object, for example:

  Creating a scrollable ResultSet:

```
Statement stmt = conn.createStatement(
                                    ResultSet.TYPE_SCROLL_SENSITIVE,
                                    ResultSet.CONCUR_READ_ONLY);
ResultSet rset=stmt.executeQuery("select …");
```

ResultSet properties specification:

- **Types:**
  - TYPE_FORWARD_ONLY (default)
  - TYPE_SCROLL_INSENSITIVE (allows scroll and doesn’t reflect changes)
  - TYPE_SCROLL_SENSITIVE (reflects changes)
- **Concurrency:**
  - CONCUR_READ_ONLY (default, no updates)
  - CONCUR_UPDATABLE (updates allowed)

ResultSet properties specification:

Result Set Type	Can See Internal Delete	Can See Internal Update	Can See Internal Insert	Can See External Delete	Can See External Update	Can See External Insert
Forward-only	No	Yes	No	No	No	No
Scroll-insensitive	Yes	Yes	No	No	No	No
Scroll-sensitive	Yes	Yes	No	No	Yes	No

Internal changes – user changes in the ResultSet

External changes – changes made from elsewhere (user changes outside the ResultSet or other committed transactions)

ResultSet methods for moving the cursor:

- rset.next() – move forward one row;
- rset.previous() – move backward one row;
- rset.first() – move to the first row;
- rset.last() – move to the last row;
- rset.getRow() – obtain current position;
- rset.absolute(int n) – move to the n^{th} row;
- rset.relative(int n) – move *n* rows from current;
- …

Updating data using ResultSet:

- Statement must be **CONCUR_UPDATABLE**
- Move **ResultSet rset** cursor to the row to be changed
- Use method **rset.updateX(column,value)** to change, e.g., rset.updateInt(1,25)

  or equivalently:

  rset.updateInt(“age”,25)

  NOTE: Cannot be used if query is “SELECT * FROM... ”
- Use method **rset.updateRow()** to make changes permanent

Inserting data using ResultSet:

- Statement must be **sensitive** and **updatable**
- Move cursor to **special insert row** using method call **rset.moveToInsertRow()**
- Set every column value using method call **rset.updateX(…)**
- Insert new row in ResultSet and table using method call **rset.insertRow()**

Deleting data using ResultSet:

- Statement must be **sensitive** and **updatable**
- Move cursor to the desired row
- Delete row from **ResultSet and table** using method call **rset.deleteRow()**

Bind variables for insert statements in Python

- Create a data as an array of tuples that needs to be insert.
- Set the **bindarray** property of cursor object to number of rows that have to be inserted at once.
- Call **setinputsizes** method of cursor to indicate the types of rows to be inserted
- Call **executemany** method of cursor to perform the insert query

Example of the insert query with bind variables

- Say that we have a table *mytab* with two columns: *id* of type integer and *data* of type char(20).
- Lets assume that we have an opened connection **con.**

Example of the insert query with bind variables (continued)

```
rows=[(1, “First”), (2, “Second”), (3, “Third”)]
cur = con.cursor()
cur.bindarraysize = 3
cur.setinputsizes(int,20)
cur.executemany(“insert into mytab(id, data) values(:1,:2)”, rows)
con.commit
```

Using MetaData to display the ResultSet:

- **ResultSetMetaData** class provides information about types and properties in a ResultSet
- Use method getMetaData() on a ResultSet object to get the result set's metadata information
- Use methods from ResultSetMetaData class to get the available information:
  - ***getColumnCount()*** – returns number of columns in the ResultSet
  - ***getColumnLabel(int column)*** – returns column title (String)
  - ***getColumnType(int column)*** – returns column's SQL type

Using MetaData to display the ResultSet (cont'd):

An example from JDBC 2 tutorial, test5.java:

```
/* get metadata for result set */
ResultSetMetaData rsetMetaData = rset.getMetaData();

/* get number of columns in ResultSet */
int columnCount = rsetMetaData.getColumnCount();

/* display column names */
for ( int column = 1; column <= columnCount; column++) {
		value = rsetMetaData.getColumnLabel(column);
		…
}
```

Using MetaData to display the ResultSet (cont'd):

...more from JDBC 2 tutorial, test5.java:

```
/* display answers, one tuple at a time */
while ( rset.next() ) {
        for ( int index = 1; index <= columnCount; index++) {
                o = rset.getObject(index);        /* get value as an object */
                if (o != null )
                        value = o.toString();       /* convert String for display */
                else
                        value = "null";
                        ...
        }
        ...
}
```

MetaData analogue in cx_Oracle

- **Description** read-only attribute of class cursor provides information about types and properties of columns returned by query
- It is a sequence of 7-items sequences.
- Each of these sequences contains information describing one result column: (name, type, display_size, internal_size, precision, scale, null_ok).

MetaData analogue in cx_Oracle

Printing headers of columns:

```
#perform query
curs.execute("SELECT * from TOFFEES")

#getting metadata
rows = curs.description

#getting number of columns
int columnCount = len(rows);

#display column names
for row in rows:
		print(row[0])
```

What's Next?

- Work through the JDBC Tutorial #2 at course page on eClass at eclass.srv.ualberta.ca
- Solve the exercises at the end of tutorial.
- For more information on ResultSet follow the link(s) from the *Related Materials* of the JDBC 2 lab section on the course web page.