DATASTAX ACADEMY

Exercise 4.3 – UDT Lists

This exercise is a continuation of the earlier exercise and is designed to familiarize users with UDT List queries.

In this exercise, you will:

- Understand how to perform writes using Lists
- Learn to execute updates to existing Lists
- Be able to run reads on Lists
- Review both CQL and Java driver statements for managing Lists

Part 1: The writeList() Function in IDE

1. Navigate to the following file within the IDE: *~/session1/src/test/java/com/datastax/training/killrvideo/dao/cassandra/UserDAOAddressTest.java*

2. Continue where you left off in the previous exercise. Locate the `writeList()` function within the file.

3. There are five (5) steps requiring completion in order for the function to work. Follow the instructions provided below; copy and paste each step to its appropriate location in the `writeList()` function.

Step 1: Run the following CQL script in the killrvideo_test keyspace. This is the table to which we will be writing data.

```
CREATE TABLE userlist (userid int PRIMARY KEY, things list<text>);
```

Step 2: Initialize the session and four list variables:

```
DseSession session = CassandraSession.getSession();
ArrayList<String> list0 = new ArrayList<String>();
ArrayList<String> list1 = new ArrayList<String>();
ArrayList<String> list2 = new ArrayList<String>();
ArrayList<String> list3 = new ArrayList<String>();
```

Step 3: Add values to each list variable:

```
list0.add("end table");
list1.add("table");
list2.add("table"); list2.add("chair");
list3.add("table"); list3.add("chair"); list3.add("lid");
```

Step 4: Create a new Batch Statement, a Prepared Statement that will insert the items into a table, and four Bound Statements for each of the pending four items:

```
BatchStatement batch = new BatchStatement();
PreparedStatement ps = session.prepare("INSERT INTO userlist (userid,
   things) VALUES (:userid, :things)");
```

Step 5: Now add a BoundStatement. Each of these will create a new row in the userlist table. A userid is included along with one of the list variables.

```
BoundStatement bs0 = ps.bind(0, list0);
BoundStatement bs1 = ps.bind(1, list1);
BoundStatement bs2 = ps.bind(2, list2);
BoundStatement bs3 = ps.bind(3, list3);
```

Step 6: Add the bound statements to the batch statement along with a statement that will execute the batch:

```
batch.add(bs0);
batch.add(bs1);
batch.add(bs2);
batch.add(bs3);
ResultSet result = session.execute(batch);
```

Step 5: Execute the `writeList()` function and query the table to verify the data was inserted.

```
cqlsh:killrvideo_test> SELECT * FROM userlist;
```

Part 2: The updateList() Function

Now, update the values inserted into the table.

1. Locate the `updateList()` function section in the file *UserDAOAddressTest.java.*

Step 1: Create a new session and create list variables.

```
DseSession session = CassandraSession.getSession();

ArrayList<String> list1 = new ArrayList<String>();
ArrayList<String> list2 = new ArrayList<String>();
ArrayList<String> list3 = new ArrayList<String>();
ArrayList<String> list4 = new ArrayList<String>();
```

Step 2: Now add items to each list.

```
list1.add("chair");
list2.add("table");
list3.add("box");
list4.add("sofa");
list4.add("lamp");
```

Step 3: Create a new `BatchStatement`. This is a single line statement.

```
BatchStatement batch = new BatchStatement();
```

Step 4: Now we'll create a simple statement for each update. Note the comments above each line as it describes what kind of update is being performed. Then note the syntax for that particular update.

```
// add an item to the list
SimpleStatement ss1 = new SimpleStatement("UPDATE userlist SET things =
   things + :things WHERE userid = :userid", list1, 1);

// remove an item from the list
SimpleStatement ss2 = new SimpleStatement("UPDATE userlist SET things =
   things - :things WHERE userid = :userid", list2, 2);

// replace an item in the list
SimpleStatement ss3 = new SimpleStatement("UPDATE userlist SET things =
   :things WHERE userid = :userid", list3, 3);

// add a new row in the table
SimpleStatement ss4 = new SimpleStatement("INSERT INTO userlist
   (userid, things) VALUES (:userid, :things)", 4, list4);

// update an item in a list
```

```
SimpleStatement ss5 = new SimpleStatement("UPDATE userlist SET
   things[0]='lava lamp' WHERE userid = :userid", 1);

// delete a row from the table
SimpleStatement ss6 = new SimpleStatement("DELETE FROM userlist WHERE
   userid = :userid", 0);
```

Step 5: Now add the simple statements to the batch statement as well as the code that will execute the batch statement.

```
batch.add(ss1);
batch.add(ss2);
batch.add(ss3);
batch.add(ss4);
batch.add(ss5);
batch.add(ss6);
ResultSet result = session.execute(batch);
```

Step 6: Now, run `updateList()` statement.

Step 7: Within the CQLSH terminal, perform a select on the *userlist* table and confirm all items were added and modified correctly.

Part 3: The readList() Function

This final exercise will retrieve the contents of the *userlist* table and print them to the IDE console.

1. Locate the `readList()` function in the file *UserDAOAddressTest.java.*

2. Find the `TODO` code block.

3. Enter the code shown below into the `TODO` block:

```
        //TODO: Your code starts here
        DseSession session = CassandraSession.getSession();
        List<String> userstuff;
        Integer userid;

        //TODO: Your code ends here
```

4. This code will create a new session, a list variable called *userstuff*, and a variable called *userid*. Enter the following line of code below what you just entered:

```
        ResultSet result = session.execute("SELECT * FROM userlist");
```

5. This line of code fetches the contents of the *userlist* table. Enter the following code just below what you entered previously:

```
        for (Row row : result) {
            userid = row.get("userid", Integer.class);
            userstuff = row.getList("things", String.class);

        }
```

6. This is a for-next loop that will iterate through each row in the result object and assign the *userid* and the contents of the "things" field to a list variable called *userstuff*. Complete your code so it appears as shown below:

```
        //TODO: Your code starts here
        DseSession session = CassandraSession.getSession();
        List<String> userstuff;
        Integer userid;

        ResultSet result = session.execute("SELECT * FROM userlist");

        for (Row row : result) {
            userid = row.get("userid", Integer.class);
            userstuff = row.getList("things", String.class);

            for (String item: userstuff)
            {
                System.out.println(userid + " : " + item);
            }
        }
        //TODO: Your code ends here
```

7. The nested for-next loop you entered iterates through each item in the list and prints it along with the *userid* to the IDE console.

8. Run the code. You should see the following output in the console:

```
1 : lava lamp
1 : chair
2 : chair
4 : sofa
4 : lamp
3 : box

Process finished with exit code 0
```

Optional Exercise: CQL

The statement below are the CQL equivalent of the writes and updates that you just performed. You may truncate the table and then run these to see that they are the same. It can also be helpful to compare the CQL to the java statements to see how the java statements were written to accomplish what the CQL below does.

```
INSERT INTO userlist (userid, things) VALUES (1, ['table']);
INSERT INTO userlist (userid, things) VALUES (0, ['end table']);
INSERT INTO userlist (userid, things) VALUES (2, ['table','chair']);
INSERT INTO userlist (userid, things) VALUES (3,
   ['table','chair','lid']);
```

```
 userid | things
--------+----------------------------
      1 |                  ['table']
      0 |             ['end table']
      2 |       ['table', 'chair']
      3 | ['table', 'chair', 'lid']
```

```
UPDATE userlist SET things = things + ['chair'] WHERE userid = 1;
UPDATE userlist SET things = things - ['table'] WHERE userid = 2;
UPDATE userlist SET things = ['box'] WHERE userid = 3;
INSERT INTO userlist (userid, things) VALUES (4, ['sofa','lamp']);
UPDATE userlist SET things[0]='lava lamp' WHERE userid = 1;
DELETE FROM userlist WHERE userid = 0;
```

```
 userid | things
--------+-------------------------
      1 | ['lava lamp', 'chair']
      2 |             ['chair']
      4 |      ['sofa', 'lamp']
      3 |               ['box']
```