DATASTAX ACADEMY

Exercise 4.2 – UDT Writes and Updates

This exercise is designed to familiarize users with writing and updating data to UDT fields.

In this exercise, you will:

- Learn the concepts of User Defined Types (UDTs)
- Understand how to use UDTs in production environments using both the IDE and CQLSH
- Familiarize yourself with the process of adding, reading, writing, and updating UDTs

Step 1: The writeUDT() Function

1. Create the TYPE and TABLE within the *killrvideo_test* keyspace. Login using CQL and then USE the keyspace *killrvideo_test* keyspace. Run the following commands:

```
cqlsh
USE killrvideo_test;
CREATE TYPE username (firstname text, lastname text);
CREATE TABLE userUDT (userid int PRIMARY KEY, username
        frozen<username>, email text);
```

2. Next, navigate to the following file within the IDE: *~/session3/src/test/java/com/datastax/training/killrvideo/dao/cassandra/UserDAOAddressTest.java*

3. Notice there are several stub functions currently in place throughout this file. Some will remain commented, others will need to be edited or uncommented and then run.

4. The first UDT we will work with is `writeUDT`. Locate the function within the IDE.

Step 1: Create a new session.

```
CassandraUserDAO userDAO = new CassandraUserDAO();
DseSession session = CassandraSession.getSession();
```

Step 2: Instantiate a new UDT object identical to UDT type in keyspace via `.getUserType()`

```
UserType udtUser = session.getCluster()
        .getMetadata()
        .getKeyspace("killrvideo_test")
        .getUserType("username");
```

Step 3: Instantiate an additional *UDTValue* object using the new UDT object

```
UDTValue udtUserName = udtUser.newValue();
udtUserName.setString("firstname", "Jamie");
udtUserName.setString("lastname", "King");
```

Step 4: Populate new UDT object

```
PreparedStatement ps = session.prepare("INSERT INTO userudt (userid,
        username) VALUES (:pk, :v)");
```

Step 5: Create a new *BoundStatement* and bind the UDT objects

```
BoundStatement bs = ps.bind()
            .setInt("pk", 2)
            .setUDTValue("v", udtUserName);
```

Step 6: Execute the *BoundStatement*

```
session.execute(bs);
```

5. Add the code above and then run the function.

6. Confirm the user was added via CQL. Return to the CQL terminal and run the following statement:

```
SELECT * FROM userUDT;
```

7. Confirm your output resembles the following:

```
userid | email | username
--------+-------+--------------------------------------------
      2 |  null | {firstname: 'Jamie', lastname: 'King'}
```

Step 2: The updateUDT() Function

Next, we will use code to update the row we entered in the previous step. Copy and paste the following code into the `updateUDT` function:

Step 1: Create a new session

```
CassandraUserDAO userDAO = new CassandraUserDAO();
DseSession session = CassandraSession.getSession();
```

Step 2: instantiate a new UDT object identical to UDT type in keyspace via `.getUserType()`

```
UserType udtUser = session.getCluster()
        .getMetadata()
        .getKeyspace("killrvideo_test")
        .getUserType("username");
```

Step 3: Step 2: Instantiate a new *UDTValue* object using the new UDT object

```
UDTValue udtUserName = udtUser.newValue();
udtUserName.setString("firstname", "Steve");
udtUserName.setString("lastname", "Halladay");
```

Step 4: Populate the new UDT object and execute

```
PreparedStatement ps = session.prepare("UPDATE userUDT SET username =
        :v WHERE userid = :pk");
```

Step 5: Create a new *BoundStatement* and bind the objects

```
BoundStatement bs = ps.bind()
        .setInt("pk", 2)
        .setUDTValue("v", udtUserName);
```

Step 6: Execute the session

```
session.execute(bs);
```

1. Add the code above and then run the function.

2. Run a CQL statement to check that the update indeed took place.

Step 3: The readUDT() function

This next exercise will check the IDE's ability to read information from the *userUDT* table within the *killrvideo_test* keyspace. This exercise builds on the previous steps and adds an extra user to the *killrvideo_test* keyspace and then display the result within the IDE console.

1. Locate the `readUDT()` function within the code. Place the following code within the TODO block of the function.

```
CassandraUserDAO userDAO = new CassandraUserDAO();
DseSession session = CassandraSession.getSession();
UserType udtUser = session.getCluster()
                .getMetadata()
                .getKeyspace("killrvideo_test")
                .getUserType("username");
```

2. The code you just pasted defines a variable of the same type as the UDT in the table. Now enter the code that will execute a select statement and fetch data from the database.

```
Row row = session.execute("SELECT username FROM userudt WHERE userid =
        2").one();
```

3. Enter the following code to extract values from the row.

```
        UDTValue udtUserName = udtUser.newValue();
        udtUserName = row.getUDTValue("username");
```

4. Now enter the code that will extract the individual values and print them to the console:

```
        String fn = udtUserName.getString("firstname");
        String ln = udtUserName.getString("lastname");
        System.out.println(fn + ' ' + ln);
```

5. Run readUDT(). The following should print to the console as the second to the last line:

```
Steve Halladay

Process finished with exit code 0
```

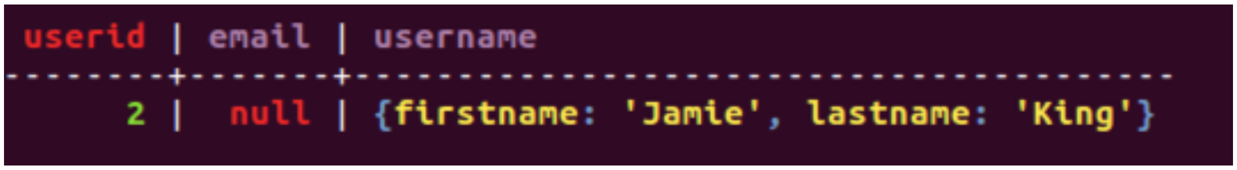
Optional Step: The updateUDT() Function in CQL

This optional step gives you a chance to work with CQL to insert and update into a UDT field. Truncate the user table and run the code shown below:

```
INSERT INTO userudt (userid, username) VALUES (2, {firstname: 'Jamie', lastname:
        'King'});
```

Note the CQL syntax is `{ fieldname1: 'value1', fieldname2: 'value2' }`. Curly braces are used to enclose the fieldname and value pairs. Fieldnames do not require the apostrophes but the values do.

Now, execute a SELECT statement to retrieve the following row:

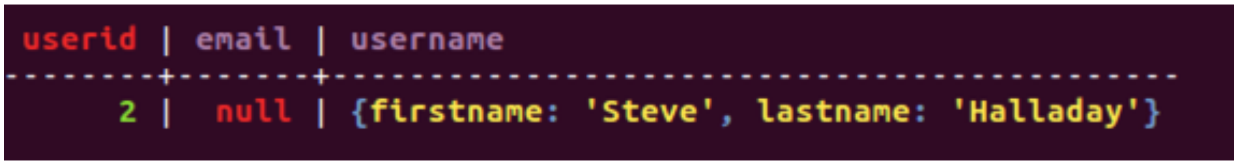


```
userid | email | username
--------+-------+----------------------------------------
     2 |  null | {firstname: 'Jamie', lastname: 'King'}
```

Run the `UPDATE` statement.

```
UPDATE userUDT SET username = {firstname: 'Steve', lastname: 'Halladay'} WHERE
        userid = 2;
```

Now execute a `SELECT *` statement. The row should appear as shown below.



```
userid | email | username
--------+-------+-----------------------------------------
     2 |  null | {firstname: 'Steve', lastname: 'Halladay'}
```