Exercise 3.4 – Batch Statements

This exercise is designed to familiarize you with the basic code required to execute a `BatchStatement` statement in order to read data from and write data to DSE. Batch Statements are used to bundle CQL queries together and guarantee atomicity. If no errors are generated, they will all succeed together. But, if at least one of the CQL queries fail, they will all fail together. In Step 3 you will intentionally generate an error in the last step so you can observe the Batch Statement fail as a whole.

In this exercise, you will add additional code to the `addVideo()` function of the `CassandraVideoDAO` class. You will be executing the same inserts as performed in the previous exercise. The only difference is that you will be using a Batch Statement to do the inserts.

This exercise uses the following files in the *Session 2* exercise project:

- *CassandraVideoDAO.java:* The file that defines the class responsible for reading and writing data to the video table. Additionally, by the end of this exercise it will feature a `BatchStatement` statement that will write data to the *latest_videos* table.
- *BatchStatementVideoDAOTest.java:* This file implements a class for inserting a video. In the final step of this exercise you will modify the file to drop the *latest_video* table first in order to generate an error when the Batch Statement executes.

All steps will have most of the required code already written out. Your task is to place the additional code where it is needed and understand how to create and test a Batch Statement. Here is a summary of steps you should follow:

STEPS:

Step 1: Modify the `addVideo()` method of the file *CassandraVideoDAO.java* to use as a Batch Statement
Step 2: Run the file *BatchStatementVideoDAOTest.java* and verify it succeeds
Step 3: Modify and run *BatchStatementVideoDAOTest.java* to observe error as the Batch Statement fails

Step 1: Modify the addVideo() method of CassandraVideoDAO.java to use a Batch Statement

The purpose of this step is to modify the `addVideo()` method to provide atomicity to the statements previously executed. In other words, they will all succeed together or fail together.

1. Launch your IDE if necessary and open the *session2 killrvideo* project.
2. In the IDE, navigate to the following directory: *~/session2/src/main/java/com/datastax/training/killrvideo/model/dao/cassandra/CassandraVideoDAO.java*
3. Find the `addVideo()` method.

NOTE: There are two insert statements: one for *videos* and one for *latest_videos*. If either of these inserts fail, the tables will be out of sync. We will submit them via a `BatchStatement` instead.

1. Comment out both `session.execute()` statements:

```
// Execute the statement
//ResultSet result = session.execute(insertToVideos);

Insert insertToLatestVideos = QueryBuilder.insertInto("latest_videos")
    .value("video_bucket", currentDate)
    .value("video_id", newVideo.getVideoId())
    .value("title", newVideo.getTitle())
    .value("type", newVideo.getType())
    .value("tags", newVideo.getTags())
    .value("preview_thumbnail", newVideo.getPreviewThumbnail());

//session.execute(insertToLatestVideos);
```

4. Just below the last line you commented out, combine both of the `INSERT` statements (which we just commented out) into a single `BatchStatement` object, as shown below:

```
BatchStatement insertVideoBatch = new BatchStatement();
insertVideoBatch.add(insertToVideos);
insertVideoBatch.add(insertToLatestVideos);
```

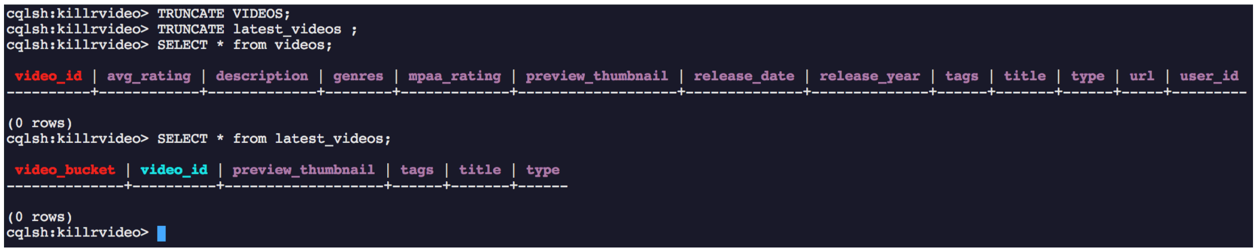
5. Pass the batch statement to `execute()` instead of calling `execute()` on each insert individually.

```
ResultSet result = session.execute(insertVideoBatch);
```

Step 2: Run BatchStatementVideoDAOTest.java and verify

The purpose of this step is to test that we modified the code properly in the first step. This step should succeed.

1. Open a terminal window and truncate the *videos* and *latest_videos* table. Confirm both tables are now empty.



```
cqlsh:killrvideo> TRUNCATE VIDEOS;
cqlsh:killrvideo> TRUNCATE latest_videos ;
cqlsh:killrvideo> SELECT * from videos;

 video_id | avg_rating | description | genres | mpaa_rating | preview_thumbnail | release_date | release_year | tags | title | type | url | user_id
----------+------------+-------------+--------+-------------+-------------------+--------------+--------------+------+-------+------+-----+---------

(0 rows)
cqlsh:killrvideo> SELECT * from latest_videos;

 video_bucket | video_id | preview_thumbnail | tags | title | type
--------------+----------+-------------------+------+-------+------

(0 rows)
cqlsh:killrvideo>
```

2. In the IDE, navigate to the following directory:

~/session2/src/test/java/com/datastax/training/killrvideo/dao/cassandra/BatchStatementVideoDAOTest.java

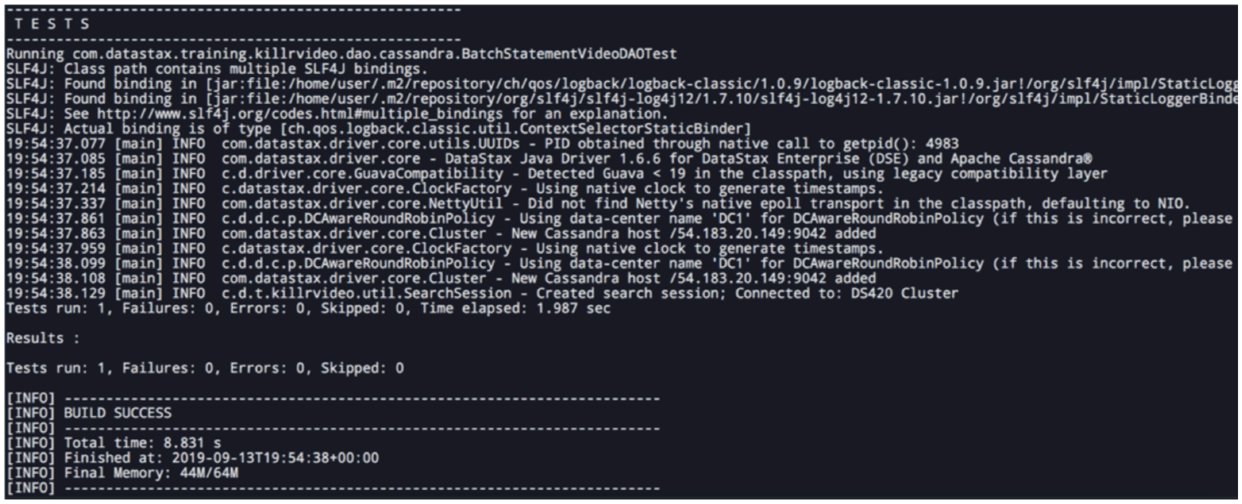
3. Find `testBatchInsert()` and the `TODO:` block:

```
//TODO: Add code here

//TODO: Your code ends here
```

4. Add the following line:

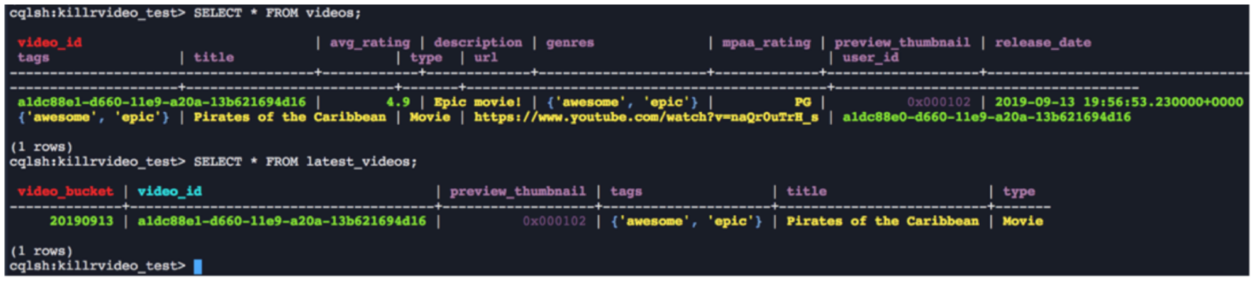
```
//TODO: Add code here

videoDAO.addVideo(originalVideo);

//TODO: Your code ends here
```

5. Run the file *BatchStatementVideoDAOTest.java*. The execution should succeed:



```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.datastax.training.killrvideo.dao.cassandra.BatchStatementVideoDAOTest
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/user/.m2/repository/ch/qos/logback/logback-classic/1.0.9/logback-classic-1.0.9.jar!/org/slf4j/impl/StaticLogg
SLF4J: Found binding in [jar:file:/home/user/.m2/repository/org/slf4j/slf4j-log4j12/1.7.10/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinde
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [ch.qos.logback.classic.util.ContextSelectorStaticBinder]
19:54:37.077 [main] INFO  com.datastax.driver.core.utils.UUIDs - PID obtained through native call to getpid(): 4983
19:54:37.085 [main] INFO  com.datastax.driver.core - DataStax Java Driver 1.6.6 for DataStax Enterprise (DSE) and Apache Cassandra®
19:54:37.185 [main] INFO  c.d.driver.core.GuavaCompatibility - Detected Guava < 19 in the classpath, using legacy compatibility layer
19:54:37.214 [main] INFO  c.datastax.driver.core.ClockFactory - Using native clock to generate timestamps.
19:54:37.337 [main] INFO  com.datastax.driver.core.NettyUtil - Did not find Netty's native epoll transport in the classpath, defaulting to NIO.
19:54:37.861 [main] INFO  c.d.d.c.p.DCAwareRoundRobinPolicy - Using data-center name 'DC1' for DCAwareRoundRobinPolicy (if this is incorrect, please
19:54:37.863 [main] INFO  com.datastax.driver.core.Cluster - New Cassandra host /54.183.20.149:9042 added
19:54:37.959 [main] INFO  c.datastax.driver.core.ClockFactory - Using native clock to generate timestamps.
19:54:38.099 [main] INFO  c.d.d.c.p.DCAwareRoundRobinPolicy - Using data-center name 'DC1' for DCAwareRoundRobinPolicy (if this is incorrect, please
19:54:38.108 [main] INFO  com.datastax.driver.core.Cluster - New Cassandra host /54.183.20.149:9042 added
19:54:38.129 [main] INFO  c.d.t.killrvideo.util.SearchSession - Created search session; Connected to: DS420 Cluster
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.987 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 8.831 s
[INFO] Finished at: 2019-09-13T19:54:38+00:00
[INFO] Final Memory: 44M/64M
[INFO] ------------------------------------------------------------------------
```

6. Return to the terminal window and verify the writes occurred.

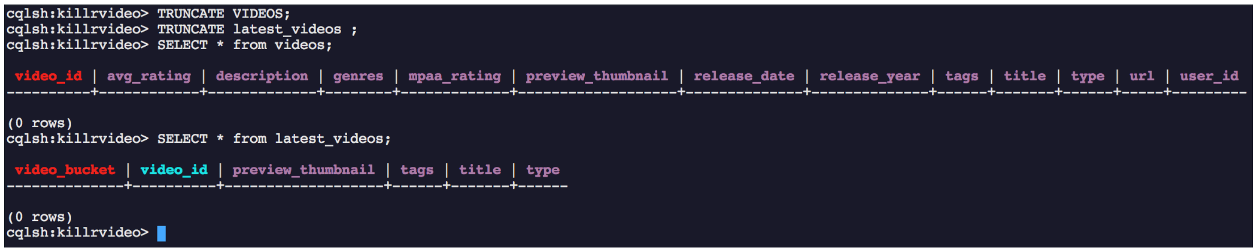


```
cqlsh:killrvideo_test> SELECT * FROM videos;

 video_id                             | avg_rating | description | genres              | mpaa_rating | preview_thumbnail | release_date
 tags                 | title                    | type  | url                                            | user_id
--------------------------------------+------------+-------------+---------------------+-------------+-------------------+-------------------------------------
----------------------+--------------------------+-------+------------------------------------------------+--------------------------------------
 a1dc88e1-d660-11e9-a20a-13b621694d16 |        4.9 | Epic movie! | {'awesome', 'epic'} |          PG |          0x000102 | 2019-09-13 19:56:53.230000+0000
 {'awesome', 'epic'} | Pirates of the Caribbean | Movie | https://www.youtube.com/watch?v=naQr0uTrH_s | a1dc88e0-d660-11e9-a20a-13b621694d16

(1 rows)
cqlsh:killrvideo_test> SELECT * FROM latest_videos;

 video_bucket | video_id                             | preview_thumbnail | tags                | title                    | type
--------------+--------------------------------------+-------------------+---------------------+--------------------------+-------
     20190913 | a1dc88e1-d660-11e9-a20a-13b621694d16 |          0x000102 | {'awesome', 'epic'} | Pirates of the Caribbean | Movie

(1 rows)
cqlsh:killrvideo_test>
```

Step 3: Modify and run BatchStatementVideoDAOTest.java and observe error as the Batch Statement fails

The purpose of this step is to modify the `testBatchInsert()` method such that the `BatchStatement` will fail. At the end we will confirm that both queries in the Batch Statement failed, thereby achieving atomicity.

1. Open a terminal window and truncate the *videos* and *latest_videos* table.

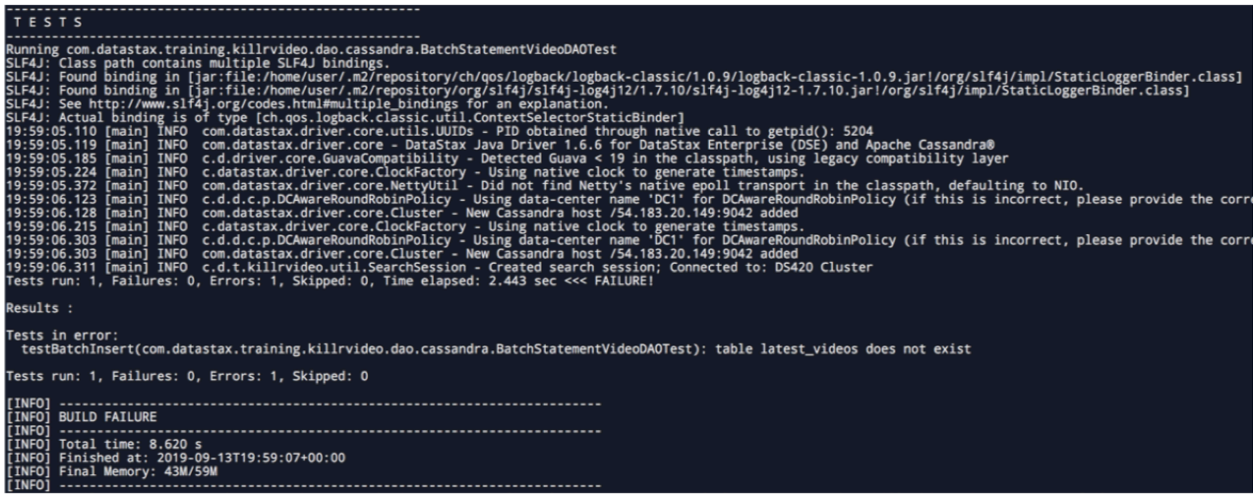


```
cqlsh:killrvideo> TRUNCATE VIDEOS;
cqlsh:killrvideo> TRUNCATE latest_videos ;
cqlsh:killrvideo> SELECT * from videos;

 video_id | avg_rating | description | genres | mpaa_rating | preview_thumbnail | release_date | release_year | tags | title | type | url | user_id
----------+------------+-------------+--------+-------------+-------------------+--------------+--------------+------+-------+------+-----+---------

(0 rows)
cqlsh:killrvideo> SELECT * from latest_videos;

 video_bucket | video_id | preview_thumbnail | tags | title | type
--------------+----------+-------------------+------+-------+------

(0 rows)
cqlsh:killrvideo>
```

2. In the IDE, navigate to the following directory: *~/session2/src/test/java/com/datastax/training/killrvideo/dao/cassandra/BatchStatementVideoDAOTest.java*
3. Remove any code in the `TODO:` block and add the following lines:

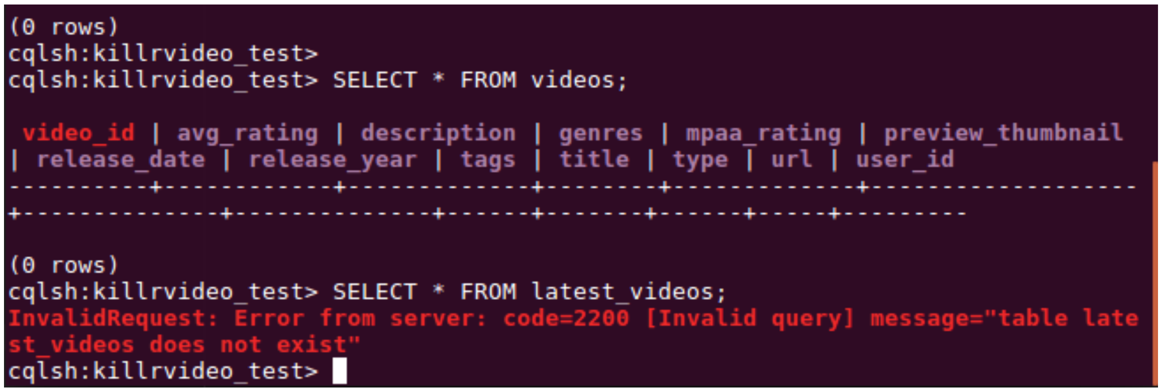
```
//TODO: Add code here

session.execute("DROP TABLE latest_videos;");
try {
    videoDAO.addVideo(originalVideo);
}
catch(InvalidQueryException e) {
throw e;
}

//TODO: Your code ends here
```

4. Run the file *BatchStatementVideoDAOTest.java*. You should get the following error.



```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.datastax.training.killrvideo.dao.cassandra.BatchStatementVideoDAOTest
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/user/.m2/repository/ch/qos/logback/logback-classic/1.0.9/logback-classic-1.0.9.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/user/.m2/repository/org/slf4j/slf4j-log4j12/1.7.10/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [ch.qos.logback.classic.util.ContextSelectorStaticBinder]
19:59:05.110 [main] INFO  com.datastax.driver.core.utils.UUIDs - PID obtained through native call to getpid(): 5204
19:59:05.119 [main] INFO  com.datastax.driver.core - DataStax Java Driver 1.6.6 for DataStax Enterprise (DSE) and Apache Cassandra®
19:59:05.185 [main] INFO  c.d.driver.core.GuavaCompatibility - Detected Guava < 19 in the classpath, using legacy compatibility layer
19:59:05.224 [main] INFO  c.datastax.driver.core.ClockFactory - Using native clock to generate timestamps.
19:59:05.372 [main] INFO  com.datastax.driver.core.NettyUtil - Did not find Netty's native epoll transport in the classpath, defaulting to NIO.
19:59:06.123 [main] INFO  c.d.d.c.p.DCAwareRoundRobinPolicy - Using data-center name 'DC1' for DCAwareRoundRobinPolicy (if this is incorrect, please provide the corr
19:59:06.128 [main] INFO  com.datastax.driver.core.Cluster - New Cassandra host /54.183.20.149:9042 added
19:59:06.215 [main] INFO  c.datastax.driver.core.ClockFactory - Using native clock to generate timestamps.
19:59:06.303 [main] INFO  c.d.d.c.p.DCAwareRoundRobinPolicy - Using data-center name 'DC1' for DCAwareRoundRobinPolicy (if this is incorrect, please provide the corr
19:59:06.303 [main] INFO  com.datastax.driver.core.Cluster - New Cassandra host /54.183.20.149:9042 added
19:59:06.311 [main] INFO  c.d.t.killrvideo.util.SearchSession - Created search session; Connected to: DS420 Cluster
Tests run: 1, Failures: 0, Errors: 1, Skipped: 0, Time elapsed: 2.443 sec <<< FAILURE!

Results :

Tests in error: 
  testBatchInsert(com.datastax.training.killrvideo.dao.cassandra.BatchStatementVideoDAOTest): table latest_videos does not exist

Tests run: 1, Failures: 0, Errors: 1, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 8.620 s
[INFO] Finished at: 2019-09-13T19:59:07+00:00
[INFO] Final Memory: 43M/59M
[INFO] ------------------------------------------------------------------------
```

5. In the terminal window, execute `select` statements on *latest_videos* and *videos*. The *latest_videos* table should be non-existent, and the videos table should be empty:



```
(0 rows)
cqlsh:killrvideo_test>
cqlsh:killrvideo_test> SELECT * FROM videos;

 video_id | avg_rating | description | genres | mpaa_rating | preview_thumbnail
| release_date | release_year | tags | title | type | url | user_id
----------+------------+-------------+--------+-------------+-------------------
+--------------+--------------+------+-------+------+-----+---------

(0 rows)
cqlsh:killrvideo_test> SELECT * FROM latest_videos;
InvalidRequest: Error from server: code=2200 [Invalid query] message="table late
st_videos does not exist"
cqlsh:killrvideo_test> 
```

6. Should you need to recreate the table in order to re-do the exercise, here is the code to do so:

```
CREATE TABLE killrvideo_test.latest_videos (
    video_bucket int,
    video_id timeuuid,
    preview_thumbnail blob,
    tags set<text>,
    title text,
    type text,
    PRIMARY KEY (video_bucket, video_id)
) WITH CLUSTERING ORDER BY (video_id DESC);
```