DATASTAX

Apache Cassandra:
Core Concepts, Skills, and Tools

Working with the Cassandra write path
Exercise Workbook

Joseph Chu
October, 2014

Exercise 1: Insert data into Cassandra and observe file system impact

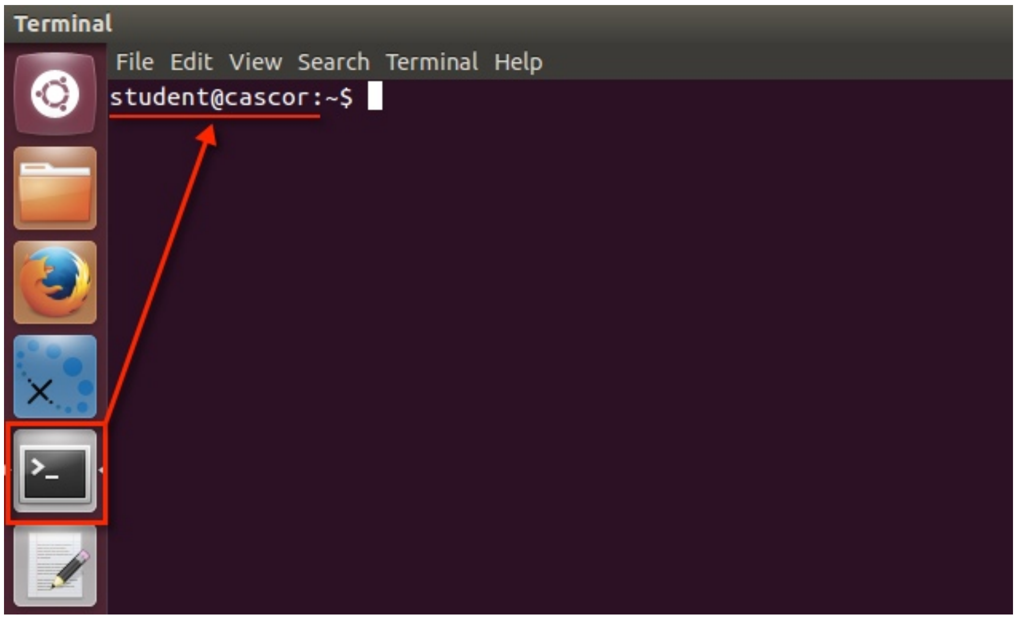
In this exercise, you will:

- Watch the changes to the MemTable and Commit Log during writes
- Observe the effects of flushing the MemTable and Commit Log to disk
- View how the Commit Log is read when a node starts

Steps

Watch the changes to the MemTable and Commit Log during writes

1. From the virtual machine, open a Terminal window or switch to an existing Terminal window running a Linux shell.



```
student@cascor:~$
```

2. In the Linux shell, navigate to the commit log directory for *node1* of the cascor cluster.

```
cd ~/node1/commitlogs
```

The commit log directory may be different depending on the commitlog_directory setting in the cassandra.yaml file.

3. In the *commitlogs* directory, list the files currently in the directory.

```
ls -lh
```

The files listed in this directory are the commit log segments. The size of each of these segments is dependent of the commitlog_segment_size_in_mb setting in the cassandra.yaml.

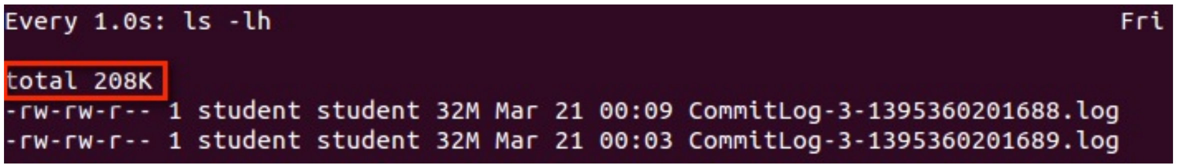
4. Run *nodetool cfstats* and view the current memtable size for the *performer* table.

```
ccm node1 nodetool cfstats musicdb.performer
```

5. Set a watch to view how the files in the commit log directory changes.

```
watch -n 1 -d "ls -lh"
```

Although each segment is allocated 32MB, that does not mean that there is 32MB of data written. Take note of the total near the top of the screen, which is the true size of the files.

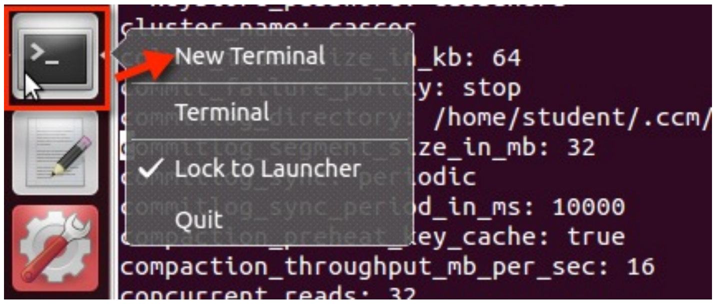


```
Every 1.0s: ls -lh                                                    Fri

total 208K
-rw-rw-r-- 1 student student 32M Mar 21 00:09 CommitLog-3-1395360201688.log
-rw-rw-r-- 1 student student 32M Mar 21 00:03 CommitLog-3-1395360201689.log
```

If you need to stop watching the directory contents, you can do so by pressing ctrl-c.

6. On the Ubuntu desktop, right-click on the Terminal launcher and open a new Terminal window.



```
cluster_name: cascor
...ize_in_kb: 64
...y: stop
... /home/student/.ccm/
...ze_in_mb: 32
...odic
...d_in_ms: 10000
...ey_cache: true
compaction_throughput_mb_per_sec: 16
concurrent_reads: 32
```

There are also keyboard shortcuts as well. To open a new Terminal window, press Ctrl-Shift-N. To open a new tab, you can use the shortcut Ctrl-Shift-T.

7. In the new Terminal window, navigate to the *exercise-1* directory for the *write-path* module.

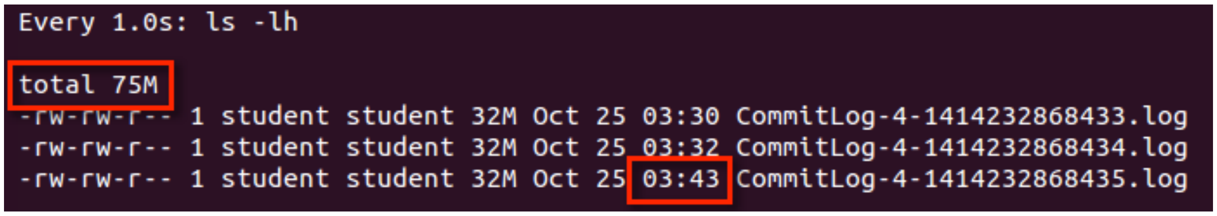
```
cd ~/cascor/write-path/exercise-1
```

8. In the *exercise-1* directory, run *cassandra-stress* using the cqlstress.xml profile to insert 250,000 partitions using 1 client thread with no warmup.

```
cassandra-stress user profile=cqlstress.yaml
ops\(insert=1\) no-warmup n=250000 -rate threads=1
```

9. While *cassandra-stress* is running, switch back to the other Terminal window running *watch*.

10. Watch the changes to the commit log as keys are inserted.



```
Every 1.0s: ls -lh

total 75M
-rw-rw-r-- 1 student student 32M Oct 25 03:30 CommitLog-4-1414232868433.log
-rw-rw-r-- 1 student student 32M Oct 25 03:32 CommitLog-4-1414232868434.log
-rw-rw-r-- 1 student student 32M Oct 25 03:43 CommitLog-4-1414232868435.log
```

There are two things to watch out for while keys are being inserted:
- The total size will continue to increase
- The timestamp will change for the current segment being written.

11. From the window that has completed *cassandra-stress*, run *nodetool cfstats again* and view the current memtable size for the *performer* table.

```
ccm node1 nodetool cfstats musicdb.performer
```

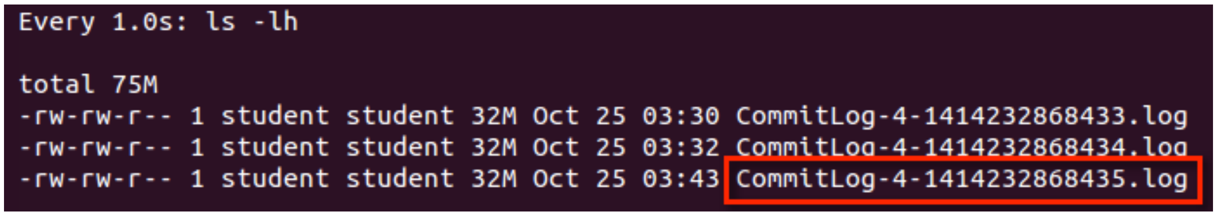
The memtable data size should be different than it was before. It should be larger, but depends on if and when any previous memtables were flushed.

Observe the effects of flushing the MemTable and Commit Log to disk

12. Switch back to the terminal window running *watch*, and interrupt the program by pressing *ctrl-c* to go back to the Linux shell.

13. In the *commitlogs* directory, list the contents of the *commitlogs* directory and take note of the file name for each of the segments.

```
ls -lh
```



```
Every 1.0s: ls -lh

total 75M
-rw-rw-r-- 1 student student 32M Oct 25 03:30 CommitLog-4-1414232868433.log
-rw-rw-r-- 1 student student 32M Oct 25 03:32 CommitLog-4-1414232868434.log
-rw-rw-r-- 1 student student 32M Oct 25 03:43 CommitLog-4-1414232868435.log
```

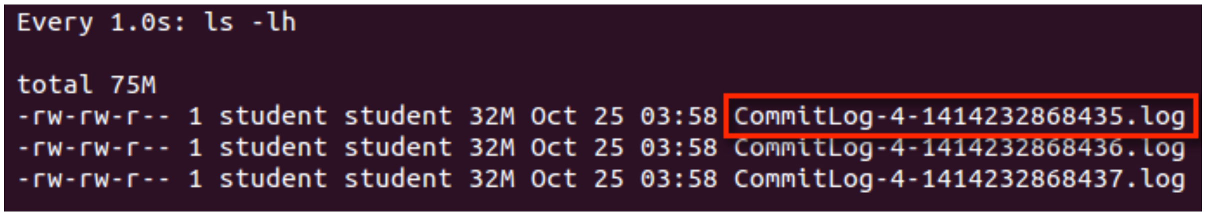
14. Run the *nodetool flush* command.

```
ccm flush
```

The flush operation will flush any data remaining in memory to disk.

15. List the contents of the *commitlogs* directory again.

```
ls -lh
```



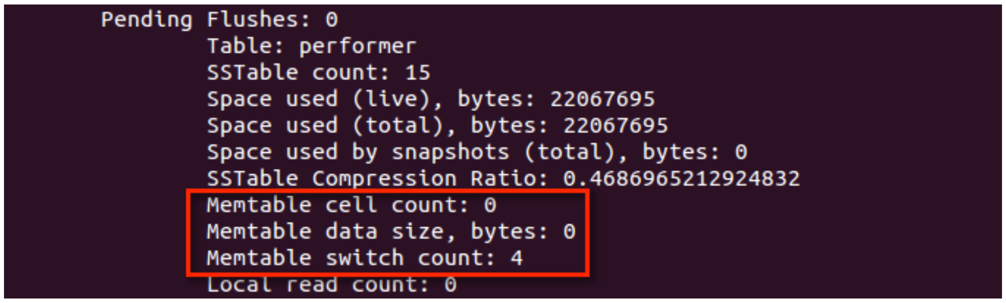
```
Every 1.0s: ls -lh

total 75M
-rw-rw-r-- 1 student student 32M Oct 25 03:58 CommitLog-4-1414232868435.log
-rw-rw-r-- 1 student student 32M Oct 25 03:58 CommitLog-4-1414232868436.log
-rw-rw-r-- 1 student student 32M Oct 25 03:58 CommitLog-4-1414232868437.log
```

Since there is no longer any data in memory, the commit log segments can be recycled. The current segment being used should be at the top and the earlier segments are set to be reused. The disk space used remains the same.

16. Run *nodetool cfstats again* and view the current memtable size for the *performer* table.

```
ccm node1 nodetool cfstats musicdb.performer
```



```
        Pending Flushes: 0
                Table: performer
                SSTable count: 15
                Space used (live), bytes: 22067695
                Space used (total), bytes: 22067695
                Space used by snapshots (total), bytes: 0
                SSTable Compression Ratio: 0.4686965212924832
                Memtable cell count: 0
                Memtable data size, bytes: 0
                Memtable switch count: 4
                Local read count: 0
```

The memtable data size should now be 0 after flushing.

View how the Commit Log is read when a node starts

17. In the Linux shell, restart the Cassandra cluster.

```
ccm stop
ccm start
```

18. From the Terminal, run the *CCM* command to show the system log for *node1*.

```
ccm node1 showlog
```

19. In the *system.log* file, search for occurrences of the text *CommitLog.java*.

To go to the end of the log, press shift-g
To search forward, press the / key and type the pattern to search for.
To search backwards press the ? key and type the pattern to search for.
To exit the log display, press the q key.

```
<shift>-<g>
?CommitLog.java
?
```

If there were no commit log segments found during startup, no replay needs to be done. If Cassandra does find commit log files, the mutations in those files will be replayed into MemTables, and then flushed to disk.

END OF EXERCISE

Demo 2: Show data directory configuration in cassandra.yaml

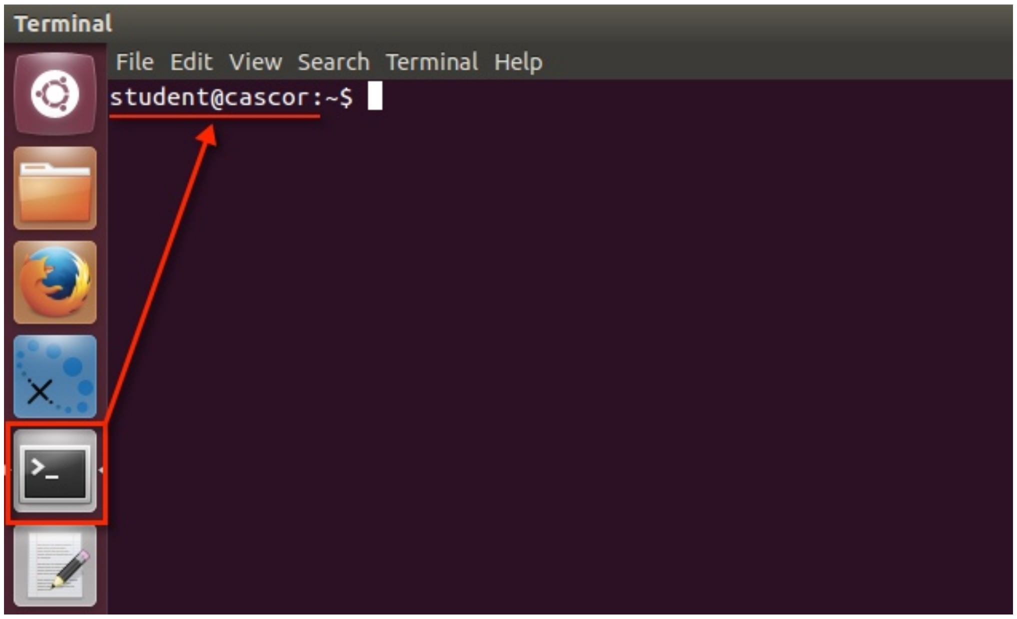
In this exercise, you will:

- Add multiple disk devices as data directories

Steps

Add multiple disk devices as data directories

1. From the virtual machine, open a Terminal window or switch to an existing Terminal window running a Linux shell.



```
student@cascor:~$
```

2. In the Terminal window, navigate to the *mnt* directory.

```
cd /mnt
```

3. In the *mnt* directory, list the files and directories.

```
ls -l
```

There are several directories here: disk1, disk2, and disk3. These directories are different disk devices that have been mounted for use in Linux.

4. Stop the 3-node cascor cluster, if it is online.

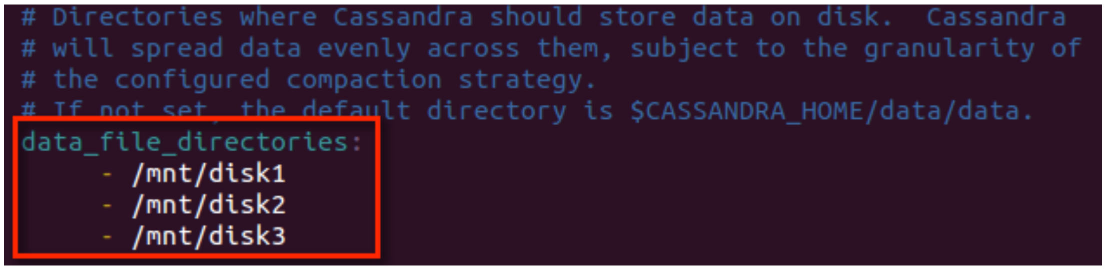
```
ccm stop
```

5. Navigate to the *conf* directory of your Cassandra installation.

```
cd /home/student/cassandra/conf
```

6. Open the *cassandra.yaml* file with a text editor.

7. For the setting *data_file_directories*, update the value to use /mnt/disk1, */mnt/disk2*, and */mnt/disk3*.



```
# Directories where Cassandra should store data on disk.  Cassandra
# will spread data evenly across them, subject to the granularity of
# the configured compaction strategy.
# If not set, the default directory is $CASSANDRA_HOME/data/data.
data_file_directories:
     - /mnt/disk1
     - /mnt/disk2
     - /mnt/disk3
```

Since you may need to uncomment the setting, make sure you remove the hash and also remove any spaces in front of the setting name, as seen above.

8. Save the *cassandra.yaml* file and exit back to the Linux shell.

9. In the Linux shell, remove commitlog segments from your Cassandra installation.

```
rm -f ~/cassandra/data/commitlog/*
```

Since the data directory will essentially be re-initialized, having commitlog entries that gets replayed may cause unexpected behavior.

10. In the Linux shell, start your installed Cassandra cluster.

```
/home/student/cassandra/bin/cassandra
```

11. Navigate to */mnt/disk1* and list the contents of the directory.

```
cd /mnt/disk1
ls
```

The structure is the same as the original data directory, with a directory for each keyspace in Cassandra. Since the original data directory is not included, only the system and the system_traces directory should be found.

12. Navigate to the *cascor/write-path/demo-2* directory in the *student* home directory.

```
cd /home/student/cascor/write-path/demo-2
```

13. In the *cascor/write-path/demo-2* directory, run *cassandra-stress* using the cqlstress.xml profile and 1 client thread with no warmup.

```
cassandra-stress user profile=cqlstress.yaml
ops\(insert=1\) no-warmup n=20000 -rate threads=1
```

14. Run *nodetool flush* to create a SSTable for the remaining data in memory.

```
nodetool flush
```

15. In the *demo-2* directory, run the script *list_disks.db* to check if where the SSTables were created.

```
./list_disks.sh
```

16. Repeat steps 12-14 to create additional SSTables, and see them written among the three disks.

17. Stop the installed Cassandra instance and start up the 3-node cascor cluster again.

```
ps auwx | grep cassandra
kill [cassandra process id]
ccm start
```

END OF DEMO

Exercise 3: Insert Data and observe SSTables created

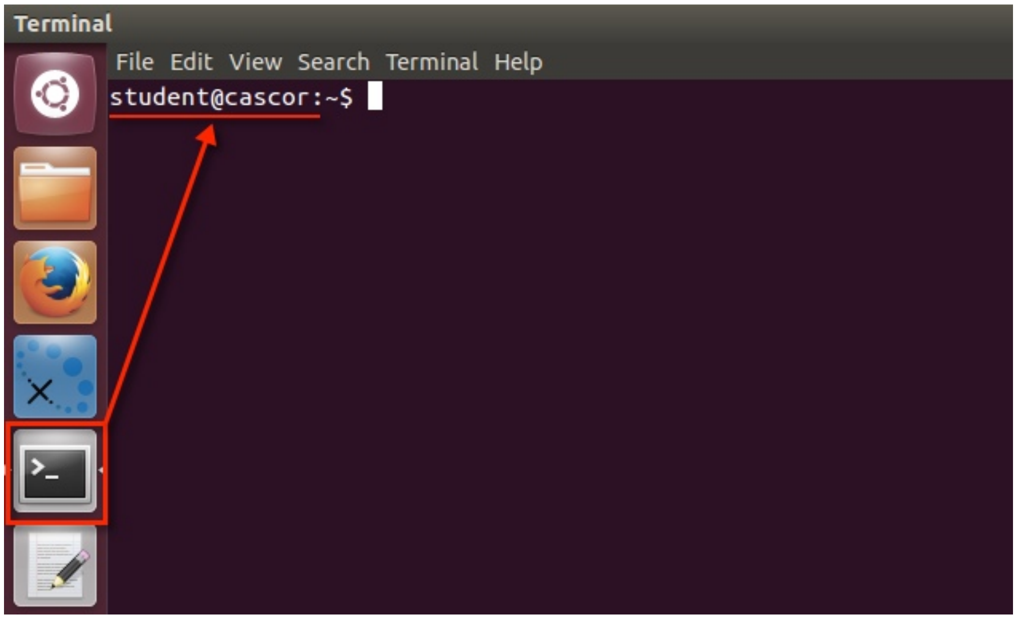
In this exercise, you will:

- Examine the directory structure for keyspaces and tables
- Explore the files in a table directory
- Convert a SSTable into a JSON file

Steps

Examine the directory structure for keyspaces and tables

1. From the virtual machine, open a Terminal window or switch to an existing Terminal window running a Linux shell.



```
student@cascor:~$
```

2. Navigate to the *data* directory for *node1* of the 3-node *cascor* cluster.

```
cd /home/student/node1/data
```

3. In the *data* directory, list the directories that reside there.

```
ls
```

At the top level of the data directory, you will find the directories for the keyspaces in Cassandra.

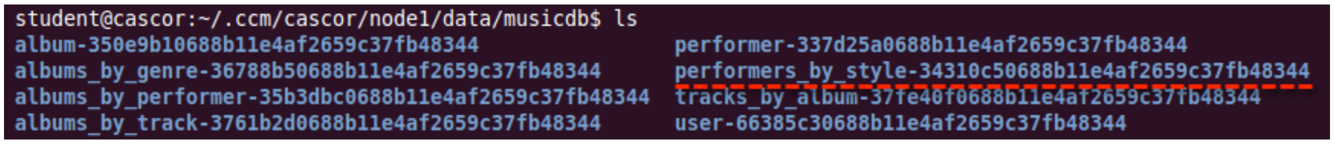
4. From the data directory, go to the *musicdb* directory and list the directories there.

```
cd musicdb
ls
```

Inside of a keyspace directory, the other directories found here are the tables that have been created within the keyspace.

Explore the files in a table directory

5. From the *musicdb* directory, go into the directory that corresponds with the *performers_by_style* table.

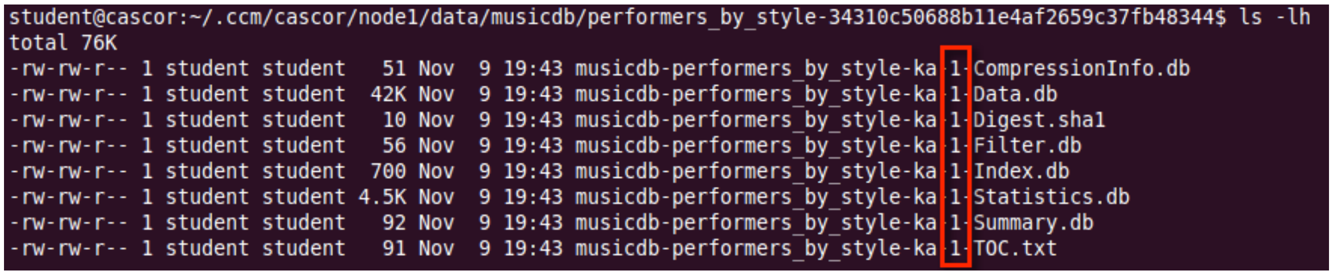


```
student@cascor:~/.ccm/cascor/node1/data/musicdb$ ls
album-350e9b10688b11e4af2659c37fb48344               performer-337d25a0688b11e4af2659c37fb48344
albums_by_genre-36788b50688b11e4af2659c37fb48344     performers_by_style-34310c50688b11e4af2659c37fb48344
albums_by_performer-35b3dbc0688b11e4af2659c37fb48344 tracks_by_album-37fe40f0688b11e4af2659c37fb48344
albums_by_track-3761b2d0688b11e4af2659c37fb48344     user-66385c30688b11e4af2659c37fb48344
```

```
cd performers_by_style-[table id]
```

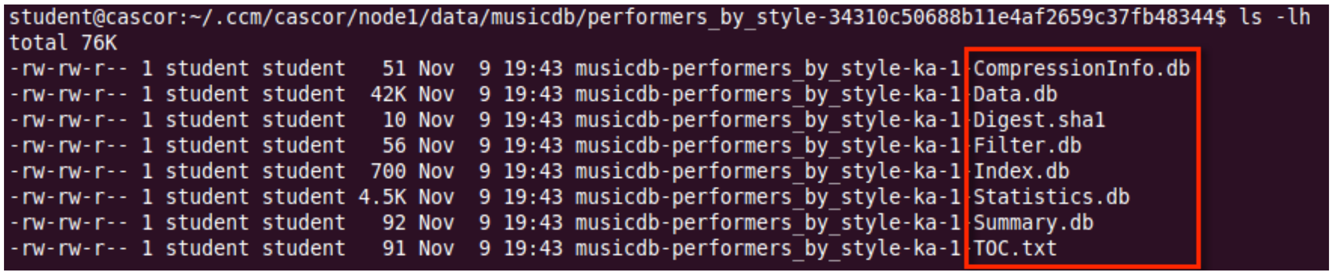
6. In the *performers_by_style* directory, list the contents of the directory.

```
ls -lh
```



```
student@cascor:~/.ccm/cascor/node1/data/musicdb/performers_by_style-34310c50688b11e4af2659c37fb48344$ ls -lh
total 76K
-rw-rw-r-- 1 student student   51 Nov  9 19:43 musicdb-performers_by_style-ka-1-CompressionInfo.db
-rw-rw-r-- 1 student student  42K Nov  9 19:43 musicdb-performers_by_style-ka-1-Data.db
-rw-rw-r-- 1 student student   10 Nov  9 19:43 musicdb-performers_by_style-ka-1-Digest.sha1
-rw-rw-r-- 1 student student   56 Nov  9 19:43 musicdb-performers_by_style-ka-1-Filter.db
-rw-rw-r-- 1 student student  700 Nov  9 19:43 musicdb-performers_by_style-ka-1-Index.db
-rw-rw-r-- 1 student student 4.5K Nov  9 19:43 musicdb-performers_by_style-ka-1-Statistics.db
-rw-rw-r-- 1 student student   92 Nov  9 19:43 musicdb-performers_by_style-ka-1-Summary.db
-rw-rw-r-- 1 student student   91 Nov  9 19:43 musicdb-performers_by_style-ka-1-TOC.txt
```

Here we see the SSTable files created for the album *table that resides on* node1. *The generation number identifies the different component files for one SSTable. A new SSTable that is created will then have the next generation number.*



```
student@cascor:~/.ccm/cascor/node1/data/musicdb/performers_by_style-34310c50688b11e4af2659c37fb48344$ ls -lh
total 76K
-rw-rw-r-- 1 student student   51 Nov  9 19:43 musicdb-performers_by_style-ka-1-CompressionInfo.db
-rw-rw-r-- 1 student student  42K Nov  9 19:43 musicdb-performers_by_style-ka-1-Data.db
-rw-rw-r-- 1 student student   10 Nov  9 19:43 musicdb-performers_by_style-ka-1-Digest.sha1
-rw-rw-r-- 1 student student   56 Nov  9 19:43 musicdb-performers_by_style-ka-1-Filter.db
-rw-rw-r-- 1 student student  700 Nov  9 19:43 musicdb-performers_by_style-ka-1-Index.db
-rw-rw-r-- 1 student student 4.5K Nov  9 19:43 musicdb-performers_by_style-ka-1-Statistics.db
-rw-rw-r-- 1 student student   92 Nov  9 19:43 musicdb-performers_by_style-ka-1-Summary.db
-rw-rw-r-- 1 student student   91 Nov  9 19:43 musicdb-performers_by_style-ka-1-TOC.txt
```

The files that have the same generation number represent a SSTable and its other component files.

Convert a SSTable into a JSON file

7. From the *performers_by_style* directory, run *sstable2json* to display the data stored within a specific SSTable file in the *performers_by_style* table.

Use the -e *option if you wish to only display the partition keys.*

```
sstable2json -e musicdb-performers_by_style-ka-1-Data.db
```

sstable2json *can only display the data in one SSTable file at a time.*

8. Run *sstable2json* to display the data in a specific SSTable file for a particular partition key.

```
sstable2json musicdb-performers_by_style-ka-1-Data.db -k
Rock
```

9. Start up *cqlsh.*

```
cqlsh
```

10. In *cqlsh*, insert a new band into the *performers_by_style* table using the style *Rock*.

```
INSERT INTO musicdb.performers_by_style (style,name)
VALUES ('Rock', 'OneRepublic');
```

11. Quit *cqlsh.*

```
QUIT
```

12. Flush and list a new SSTable for the *performers_by_style* table.

```
ccm flush
ls -lh *-Data.db
```

13. Run sstable2json to display the contents of the new SSTable.

```
sstable2json musicdb-performers_by_style-ka-2-Data.db
```

The partition for Rock *is now fragmented between the first SSTable and the new SSTable that was created with the newly inserted row.*

END OF EXERCISE