Advanced Databases

Cassandra CA

Student Number: C19366191

Student Name: Lina Mir

Programme Code: TU856

1. **Setting up the cluster and keyspace**

   The cluster contains two nodes, one called C19366191-1 and the other called C19366191-2. They are both situated on rack1. They run on port 9042 and have pulled from the latest Cassandra container.

```
C:\Users\Windows>docker exec -it C19366191-1 nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns (effective)  Host ID                               Rack
UN  172.17.0.3  191.97 KiB  16      100.0%            10cd4463-51f7-492f-bbec-6c8cc4a14c5a  rack1
UN  172.17.0.2  208.73 KiB  16      100.0%            9f43cb43-67de-4498-aabd-6b88fdc0623d  rack1
```

   The UN shows the nodes are up and normal. The addresses on the nodes are two different ones and their size is different. They both own 100% of the nodes and share the data.

```
CREATE KEYSPACE advanceddb WITH replication = {'class': 'SimpleStrategy',
 'replication_factor': '2'}  AND durable_writes = true;
```

   This is the keyspace that was created. It has a replication factor of 2. This means that there will be two copies of data. The class is SimpleStrategy. This is a SimpleStrategy because a single data centre is used and the single rack, this strategy will be used.

2. **Porting the data to Cassandra**

   a. **The query used on the fact table that will be used in the python file is below.**

```
select fr.player_sk, dp.p_sname as p_lastname, dt.t_descriprion as t_desc, fr.prize, rank, year
from factresults fr
join dimplayer dp on fr.player_sk = dp.player_sk
join dimdate dd on fr.date_sk = dd.date_sk
join dimtournament dt on fr.tournament_sk = dt.tournament_sk;
```

      This will get the name of the player, their player sk, the tournament name, year, the rank, and prize they won then.

   b. **The python file submitted is what was used to port the data and create and populate the table in Cassandra. This was run in the terminal and the output was as follows:**

```
C:\Users\Windows\Desktop\CS\Year4\Semester1\AdvancedDatabases\CA\CA2>python c19366191.py
<PreparedStatement query="insert into factresults (player_sk,p_name,t_name,prize,rank,year) VALUES (?,?,?,?,?,?)", consistency=Not Set>
Uploading to Cassandra - row 1
Woods
Uploading to Cassandra - row 2
Smith
Uploading to Cassandra - row 3
Smith
Uploading to Cassandra - row 4
Smith
Uploading to Cassandra - row 5
Woods
Uploading to Cassandra - row 6
Baggio
Uploading to Cassandra - row 7
Smith
Uploading to Cassandra - row 8
Woods
Uploading to Cassandra - row 9
McDonald
Uploading to Cassandra - row 10
```

The uploading to Cassandra goes as far as row 28.
The JSON file is also submitted, where the results of the query are stored.

```
C:\Users\Windows\Desktop\CS\Year4\Semester1\AdvancedDatabases\CA\CA2>docker exec -it C19366191-1 nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns (effective)  Host ID                               Rack
UN  172.17.0.3  227.38 KiB  16      100.0%            10cd4463-51f7-492f-bbec-6c8cc4a14c5a  rack1
UN  172.17.0.2  224.03 KiB  16      100.0%            9f43cb43-67de-4498-aabd-6b88fdc0623d  rack1
```

The size of the load here is around 227kb and 224kb. This has now increased since the first time, where it was from 190kb to 200kb. Since more data has been added, this makes sense.

The command used to investigate the table was docker exec -it C19366191-1 nodetool tablestats advanceddb.factresults
The screenshot is below. Some key features to note from this:

- The write latency is 0.16010714285714286 ms which means it will take this amount of time to complete the most recent write request.
- The table name is here too, factresults.
- The memtable cell count is 28 and this means that 28 rows of data have been put in.
- The memtable size is 841, which is showing how much data is stored in the memtable.
- The bloom filter space is 0 which is the space used to store the bloom filter data.
- The bloom filter false positives ratio shows the fraction of all bloom filter checks resulting in false positives and here the number is 0.

```
C:\Users\Windows\Desktop\CS\Year4\Semester1\AdvancedDatabases\CA\CA2>do
Total number of tables: 43
----------------
Keyspace : advanceddb
        Read Count: 0
        Read Latency: NaN ms
        Write Count: 28
        Write Latency: 0.16010714285714286 ms
        Pending Flushes: 0
                Table: factresults
                SSTable count: 0
                Old SSTable count: 0
                Space used (live): 0
                Space used (total): 0
                Space used by snapshots (total): 0
                Off heap memory used (total): 0
                SSTable Compression Ratio: -1.0
                Number of partitions (estimate): 6
                Memtable cell count: 28
                Memtable data size: 841
                Memtable off heap memory used: 0
                Memtable switch count: 0
                Local read count: 0
                Local read latency: NaN ms
                Local write count: 28
                Local write latency: NaN ms
                Pending flushes: 0
                Percent repaired: 100.0
                Bytes repaired: 0.000KiB
                Bytes unrepaired: 0.000KiB
                Bytes pending repair: 0.000KiB
                Bloom filter false positives: 0
                Bloom filter false ratio: 0.00000
                Bloom filter space used: 0
                Bloom filter off heap memory used: 0
                Index summary off heap memory used: 0
                Compression metadata off heap memory used: 0
                Compacted partition minimum bytes: 0
                Compacted partition maximum bytes: 0
                Compacted partition mean bytes: 0
                Average live cells per slice (last five minutes): NaN
                Maximum live cells per slice (last five minutes): 0
                Average tombstones per slice (last five minutes): NaN
                Maximum tombstones per slice (last five minutes): 0
                Dropped Mutations: 0
                Droppable tombstone ratio: 0.00000
```

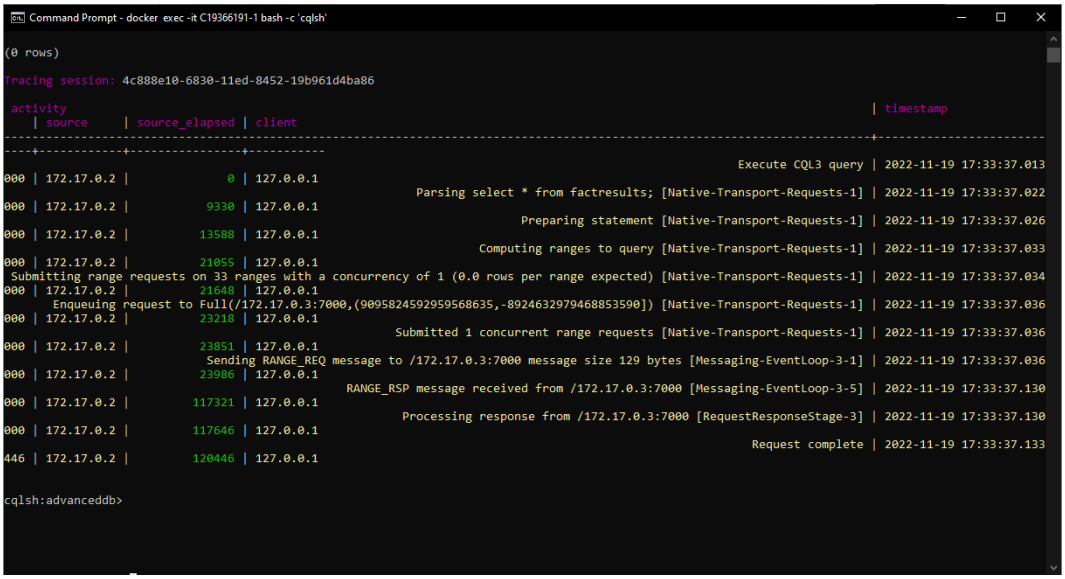
3. Golf data

a. Basic query on golf data

Tracing was set to be on.

The basic query done is select * from factresults;

The output for this query is here:



This shows the stages it takes for the query. The first step is to execute the query and then to parse it. After that the statement is prepared and the ranges to the query are computed.

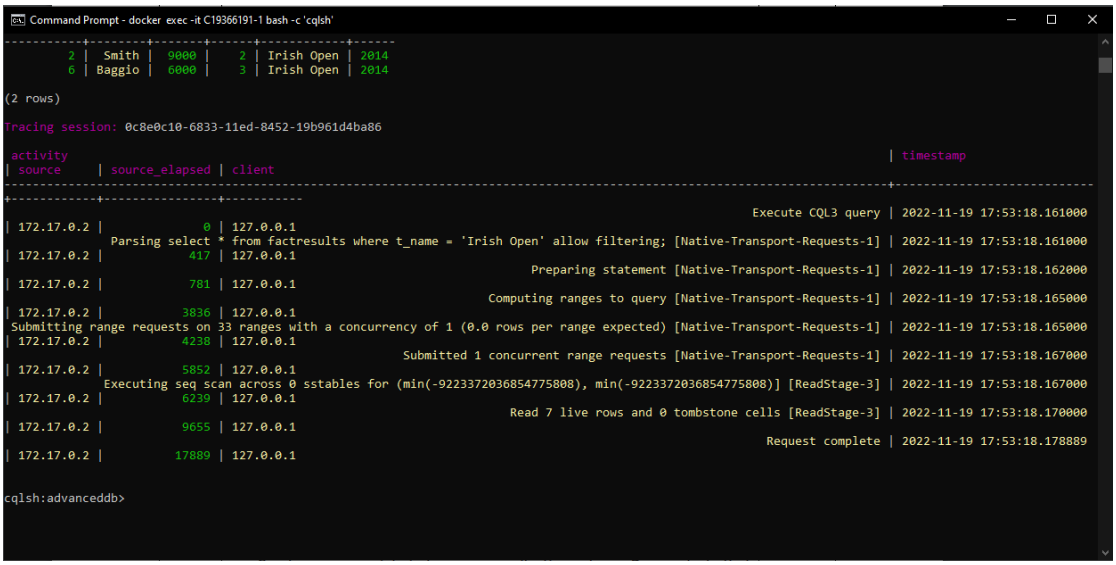
Then the range request with concurrency of 1 is submitted and the rest of show in the screenshot.

b. Query of non-primary index (without index)

Tracing was set to be on.

The query of a non-primary index was done is select * from factresults where t_name = ‘Irish Open’ allow filtering;

The output for this query is here:



This screenshot shows the process the query follows to get the output. The query is executed, and the statement is parsed and prepared. The ranges are computed and the range requests on 33 ranges with a concurrency of 1 are submitted. The concurrent range requests are submitted and then the seq scan is executed. 7 live rows are read and then the request is complete.

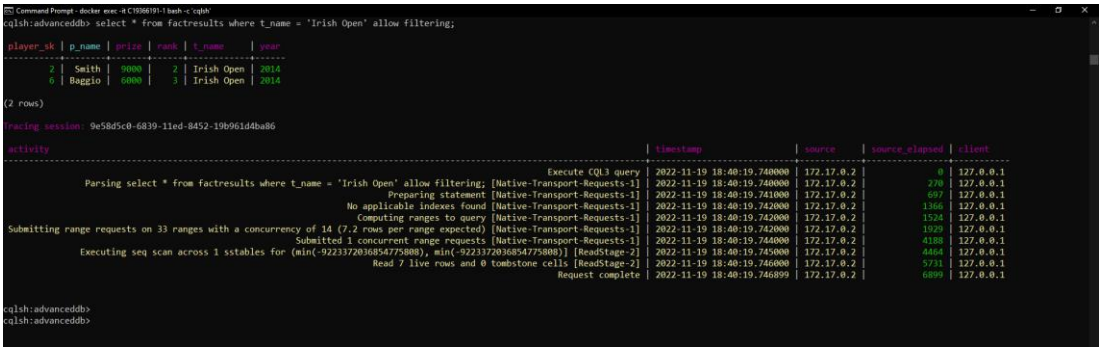
c. Adding a secondary index to golf data

Tracing was set to be on.

The index that was created was on the rank, create index player_rank on factresults(rank);

The query that was performed was select * from factresults where t_name = ‘Irish Open’ allow filtering;

This is the output that was given on this query.



The process that the query followed was it parsed the query, then the statement was prepared and the indexes that could be applicable were not found. This was because the query did not use the column that was indexed. Then the ranges to query were computed and the range requests on 33 ranges with a concurrency of 14 were submitted. The rows per range were 7.2. One concurrent range was submitted and the seq scan was executed. 7 live rows were read, and the request was completed.

The node status was performed on the table now. This is the output:

```
C:\Users\Windows>docker exec -it C19366191-1 nodetool tablestats advance
Total number of tables: 43
----------------
Keyspace : advanceddb
        Read Count: 7
        Read Latency: 1.766142857142857 ms
        Write Count: 35
        Write Latency: 0.15088571428571426 ms
        Pending Flushes: 0
                Table: factresults
                SSTable count: 1
                Old SSTable count: 0
                Space used (live): 5483
                Space used (total): 5483
                Space used by snapshots (total): 0
                Off heap memory used (total): 40
                SSTable Compression Ratio: 0.6231884057971014
                Number of partitions (estimate): 7
                Memtable cell count: 0
                Memtable data size: 0
                Memtable off heap memory used: 0
                Memtable switch count: 1
                Local read count: 7
                Local read latency: NaN ms
                Local write count: 28
                Local write latency: NaN ms
                Pending flushes: 0
                Percent repaired: 0.0
                Bytes repaired: 0.000KiB
                Bytes unrepaired: 0.404KiB
                Bytes pending repair: 0.000KiB
                Bloom filter false positives: 0
                Bloom filter false ratio: 0.00000
                Bloom filter space used: 24
                Bloom filter off heap memory used: 16
                Index summary off heap memory used: 16
                Compression metadata off heap memory used: 8
                Compacted partition minimum bytes: 51
                Compacted partition maximum bytes: 72
                Compacted partition mean bytes: 64
                Average live cells per slice (last five minutes): NaN
                Maximum live cells per slice (last five minutes): 0
                Average tombstones per slice (last five minutes): NaN
                Maximum tombstones per slice (last five minutes): 0
                Dropped Mutations: 0
                Droppable tombstone ratio: 0.00000
```

The read latency on this table now is 1.766142857142857 ms. The write latency on the table is 0.15088571428571426 ms. The write count is 35 now. The Index summary off heap memory used is 16 and the bloom filter space used is 24 while the bloom filter false positive ratio is 0.

d. Adding an SASI index to golf data to facilitate pattern matching

The SASI index that was put on the table is the following:

```
cqlsh:advanceddb> CREATE CUSTOM INDEX certain_tname ON factresults(t_name) USING 'org.apache.cassandra.index.sasi.SASIIndex';
```

This index was put on the tournament name as a prefix SASI index.

This is the query and the results that was run using pattern matching to see it in action:

```
cqlsh:advanceddb> SELECT * FROM factresults WHERE t_name LIKE 'I%';

 player_sk | p_name | prize | rank | t_name     | year
-----------+--------+-------+------+------------+------
         2 |  Smith |  9000 |    2 | Irish Open | 2014
         6 | Baggio |  6000 |    3 | Irish Open | 2014
```

It shows the details of the players matching the tournament name with the letter I, showing the Irish tournaments.

Tracing was set to on for this query.

```
Tracing session: 145df030-6c44-11ed-b84a-7fa5bd83bdad

 activity                                                                                                                       | timestamp                  | source     | source_elapsed | client
--------------------------------------------------------------------------------------------------------------------------------+----------------------------+------------+----------------+-----------
                                                                                                             Execute CQL3 query | 2022-11-24 22:05:17.363000 | 172.17.0.2 |              0 | 127.0.0.1
                                      Parsing SELECT * FROM factresults WHERE t_name LIKE 'I%'; [Native-Transport-Requests-1] | 2022-11-24 22:05:17.364000 | 172.17.0.2 |            527 | 127.0.0.1
                                                                               Preparing statement [Native-Transport-Requests-1] | 2022-11-24 22:05:17.364000 | 172.17.0.2 |            832 | 127.0.0.1
           Index mean cardinalities are certain_tname:-9223372036854775808. Scanning with certain_tname. [Native-Transport-Requests-1] | 2022-11-24 22:05:17.365000 | 172.17.0.2 |           1680 | 127.0.0.1
                                                                          Computing ranges to query [Native-Transport-Requests-1] | 2022-11-24 22:05:17.365000 | 172.17.0.2 |           2207 | 127.0.0.1
 Submitting range requests on 33 ranges with a concurrency of 1 (-2.59407332E17 rows per range expected) [Native-Transport-Requests-1] | 2022-11-24 22:05:17.366000 | 172.17.0.2 |           2500 | 127.0.0.1
                 Enqueuing request to Full(/172.17.0.3:7000,(9095824592959568635,-8924632979468853590]) [Native-Transport-Requests-1] | 2022-11-24 22:05:17.367000 | 172.17.0.2 |           3675 | 127.0.0.1
                                                                Submitted 1 concurrent range requests [Native-Transport-Requests-1] | 2022-11-24 22:05:17.367000 | 172.17.0.2 |           4088 | 127.0.0.1
                                       Sending RANGE_REQ message to /172.17.0.3:7000 message size 160 bytes [Messaging-EventLoop-3-2] | 2022-11-24 22:05:17.367000 | 172.17.0.2 |           4190 | 127.0.0.1
                                                         RANGE_REQ message received from /172.17.0.2:7000 [Messaging-EventLoop-3-3] | 2022-11-24 22:05:17.368000 | 172.17.0.3 |            106 | 127.0.0.1
                                                        Executing read on advanceddb.factresults using index certain_tname [ReadStage-1] | 2022-11-24 22:05:17.369000 | 172.17.0.3 |            600 | 127.0.0.1
                                                                       Executing single-partition query on factresults [ReadStage-1] | 2022-11-24 22:05:17.370000 | 172.17.0.3 |           2262 | 127.0.0.1
                                                                                    Acquiring sstable references [ReadStage-1] | 2022-11-24 22:05:17.371000 | 172.17.0.3 |           2471 | 127.0.0.1
                                          Skipped 0/1 non-slice-intersecting sstables, included 0 due to tombstones [ReadStage-1] | 2022-11-24 22:05:17.371000 | 172.17.0.3 |           2782 | 127.0.0.1
                                                                                     Key cache hit for sstable 1 [ReadStage-1] | 2022-11-24 22:05:17.371000 | 172.17.0.3 |           3035 | 127.0.0.1
                                                                          Merged data from memtables and 1 sstables [ReadStage-1] | 2022-11-24 22:05:17.372000 | 172.17.0.3 |           3383 | 127.0.0.1
                                                                       Executing single-partition query on factresults [ReadStage-1] | 2022-11-24 22:05:17.372000 | 172.17.0.3 |           3761 | 127.0.0.1
                                                                                    Acquiring sstable references [ReadStage-1] | 2022-11-24 22:05:17.372000 | 172.17.0.3 |           3962 | 127.0.0.1
                                          Skipped 0/1 non-slice-intersecting sstables, included 0 due to tombstones [ReadStage-1] | 2022-11-24 22:05:17.372000 | 172.17.0.3 |           4111 | 127.0.0.1
                                                                                     Key cache hit for sstable 1 [ReadStage-1] | 2022-11-24 22:05:17.372000 | 172.17.0.3 |           4208 | 127.0.0.1
                                                                          Merged data from memtables and 1 sstables [ReadStage-1] | 2022-11-24 22:05:17.373000 | 172.17.0.3 |           4447 | 127.0.0.1
                                                                             Read 2 live rows and 0 tombstone cells [ReadStage-1] | 2022-11-24 22:05:17.373000 | 172.17.0.3 |           4705 | 127.0.0.1
                                                                            Enqueuing response to /172.17.0.2:7000 [ReadStage-1] | 2022-11-24 22:05:17.373000 | 172.17.0.3 |           4830 | 127.0.0.1
                           Sending RANGE_RSP message to cassandra/172.17.0.2:7000 message size 150 bytes [Messaging-EventLoop-3-1] | 2022-11-24 22:05:17.373000 | 172.17.0.3 |           5060 | 127.0.0.1
                                                         RANGE_RSP message received from /172.17.0.3:7000 [Messaging-EventLoop-3-8] | 2022-11-24 22:05:17.374000 | 172.17.0.2 |          10678 | 127.0.0.1
                                                            Processing response from /172.17.0.3:7000 [RequestResponseStage-3] | 2022-11-24 22:05:17.375000 | 172.17.0.2 |          11774 | 127.0.0.1
                                                                                                           Request complete | 2022-11-24 22:05:17.375620 | 172.17.0.2 |          12620 | 127.0.0.1
```

The tracing output shows that it went through these stages to execute the query. It parsed the query first, then prepared the statement and scanned using the index created above. Then it computed ranges to the query and submitted range requests. It went through many other stages as seen in the screenshot.

This is what nodetool tablestats shows:

```
Total number of tables: 44
----------------
Keyspace : advanceddb
        Read Count: 0
        Read Latency: NaN ms
        Write Count: 35
        Write Latency: 0.05631428571428572 ms
        Pending Flushes: 0
                Table: factresults
                SSTable count: 1
                Old SSTable count: 0
                Space used (live): 5484
                Space used (total): 5484
                Space used by snapshots (total): 0
                Off heap memory used (total): 40
                SSTable Compression Ratio: 0.6280193236714976
                Number of partitions (estimate): 7
                Memtable cell count: 0
                Memtable data size: 0
                Memtable off heap memory used: 0
                Memtable switch count: 1
                Local read count: 0
                Local read latency: NaN ms
                Local write count: 28
                Local write latency: NaN ms
                Pending flushes: 0
                Percent repaired: 0.0
                Bytes repaired: 0.000KiB
                Bytes unrepaired: 0.404KiB
                Bytes pending repair: 0.000KiB
                Bloom filter false positives: 0
                Bloom filter false ratio: 0.00000
                Bloom filter space used: 24
                Bloom filter off heap memory used: 16
                Index summary off heap memory used: 16
                Compression metadata off heap memory used: 8
                Compacted partition minimum bytes: 51
                Compacted partition maximum bytes: 72
                Compacted partition mean bytes: 64
                Average live cells per slice (last five minutes): NaN
                Maximum live cells per slice (last five minutes): 0
                Average tombstones per slice (last five minutes): NaN
                Maximum tombstones per slice (last five minutes): 0
                Dropped Mutations: 0
                Droppable tombstone ratio: 0.00000
```

This shows now that the space used is more. The write count has increased from 7 to 35 since the last change. The number of partitions is 5 whereas the previous ones is 7. The compacted partition minimum bytes are now 73. The compacted maximum is not 103 and the mean bytes are 93.

4. Data including collection data type

a. Creating and inserting into table

<<use tracing, capture the output and include it here, comment on what is happening>>

The table that was created was a movie table. This contained collection data type, which is a map with two text fields, one for the movie name and the main lead.

```
create table movies (movieid int, movie_info MAP<text, text>, m_year int, PRIMARY KEY(movieid));
```

Data was inserted into the table.

```
insert into movies(movieid, movie_info, m_year) values (1, {'Movie Name':'Batman', 'Main Lead': 'Robert Pattinson'}, 2022);
insert into movies(movieid, movie_info, m_year) values (2, {'Movie Name':'Dune', 'Main Lead': 'Timothee Chalamet'}, 2021);
insert into movies(movieid, movie_info, m_year) values (3, {'Movie Name':'Glass Onion', 'Main Lead': 'Daniel Craig'}, 2022);
insert into movies(movieid, movie_info, m_year) values (4, {'Movie Name':'Tenet', 'Main Lead': 'John David Washington'}, 2020);
insert into movies(movieid, movie_info, m_year) values (5, {'Movie Name':'Interstellar', 'Main Lead': 'Matthew McConaughey'}, 2014);
```

b. Query of non-primary index (without index)

This is the query without the index. It gets the movies where the year was 2022.

```
 movieid | m_year | movie_info
---------+--------+-------------------------------------------------------------
       1 |   2022 |  {'Main Lead': 'Robert Pattinson', 'Movie Name': 'Batman'}
       3 |   2022 | {'Main Lead': 'Daniel Craig', 'Movie Name': 'Glass Onion'}
```

This is the query with tracing on:

```
Tracing session: f455ceb0-6c49-11ed-b84a-7fa5bd83bdad

 activity                                                                                                                   | timestamp                  | source     | source_elapsed | client
----------------------------------------------------------------------------------------------------------------------------+----------------------------+------------+----------------+-----------
                                                                                                         Execute CQL3 query | 2022-11-24 22:47:20.603000 | 172.17.0.2 |              0 | 127.0.0.1
                              Parsing select * from movies where m_year = 2022 allow filtering; [Native-Transport-Requests-1] | 2022-11-24 22:47:20.603000 | 172.17.0.2 |            309 | 127.0.0.1
                                                                             Preparing statement [Native-Transport-Requests-1] | 2022-11-24 22:47:20.603000 | 172.17.0.2 |            667 | 127.0.0.1
                                                                        Computing ranges to query [Native-Transport-Requests-1] | 2022-11-24 22:47:20.604000 | 172.17.0.2 |           1106 | 127.0.0.1
 Submitting range requests on 33 ranges with a concurrency of 1 (0.0 rows per range expected) [Native-Transport-Requests-1] | 2022-11-24 22:47:20.604000 | 172.17.0.2 |           1244 | 127.0.0.1
                                                           Submitted 1 concurrent range requests [Native-Transport-Requests-1] | 2022-11-24 22:47:20.605000 | 172.17.0.2 |           2524 | 127.0.0.1
            Executing seq scan across 0 sstables for (min(-9223372036854775808), min(-9223372036854775808)] [ReadStage-2] | 2022-11-24 22:47:20.605000 | 172.17.0.2 |           2595 | 127.0.0.1
                                                                  Read 5 live rows and 0 tombstone cells [ReadStage-2] | 2022-11-24 22:47:20.606000 | 172.17.0.2 |           3308 | 127.0.0.1
                                                                                                           Request complete | 2022-11-24 22:47:20.607457 | 172.17.0.2 |           4457 | 127.0.0.1
```

It starts with parsing and then prepares the statement. It computes the ranges to query and then submits the range requests on 33 ranges with a concurrency of 1. It submits the concurrent range requests. It executes the seq scan across 0 sstables. And it had read 5 lives rows and 0 tombstone cells.

c. Adding a secondary index

<<use tracing, capture the output and include it here, comment on what is happening>>

<<use nodetool tablestats to investigate your table, comment on key pieces of Information>>

The index was created on the map column as shown below.

```
create index movies_index on movies (movie_info);
```

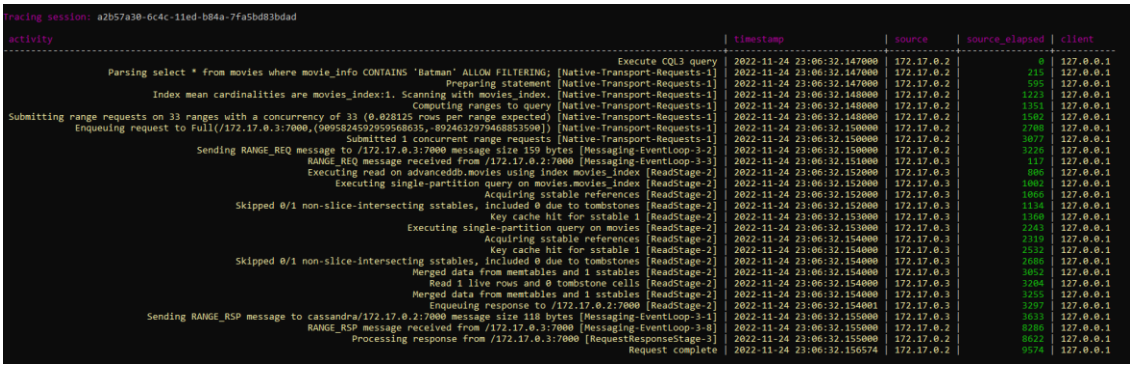
The query that was done this time was this:

```
cqlsh:advanceddb> select * from movies where movie_info CONTAINS 'Batman' ALLOW FILTERING;

 movieid | m_year | movie_info
---------+--------+-----------------------------------------------------------
       1 |   2022 | {'Main Lead': 'Robert Pattinson', 'Movie Name': 'Batman'}
```

This found the movie information where the movie name was batman.

Then with tracing on this was the output.



Tracing session: a2b57a30-6c4c-11ed-b84a-7fa5bd83bdad

activity	timestamp	source	source_elapsed	client
Execute CQL3 query	2022-11-24 23:06:32.147000	172.17.0.2	0	127.0.0.1
Parsing select * from movies where movie_info CONTAINS 'Batman' ALLOW FILTERING; [Native-Transport-Requests-1]	2022-11-24 23:06:32.147000	172.17.0.2	215	127.0.0.1
Preparing statement [Native-Transport-Requests-1]	2022-11-24 23:06:32.147000	172.17.0.2	595	127.0.0.1
Index mean cardinalities are movies_index:1. Scanning with movies_index. [Native-Transport-Requests-1]	2022-11-24 23:06:32.148000	172.17.0.2	1223	127.0.0.1
Computing ranges to query [Native-Transport-Requests-1]	2022-11-24 23:06:32.148000	172.17.0.2	1351	127.0.0.1
Submitting range requests on 33 ranges with a concurrency of 33 (0.028125 rows per range expected) [Native-Transport-Requests-1]	2022-11-24 23:06:32.148000	172.17.0.2	1502	127.0.0.1
Enqueuing request to Full(/172.17.0.3:7000,(9095824592959568635,-8924632979468853590]) [Native-Transport-Requests-1]	2022-11-24 23:06:32.150000	172.17.0.2	2708	127.0.0.1
Submitted 1 concurrent range requests [Native-Transport-Requests-1]	2022-11-24 23:06:32.150000	172.17.0.2	3077	127.0.0.1
Sending RANGE_REQ message to /172.17.0.3:7000 message size 159 bytes [Messaging-EventLoop-3-2]	2022-11-24 23:06:32.150000	172.17.0.2	3226	127.0.0.1
RANGE_REQ message received from /172.17.0.2:7000 [Messaging-EventLoop-3-3]	2022-11-24 23:06:32.151000	172.17.0.3	117	127.0.0.1
Executing read on advanceddb.movies using index movies_index [ReadStage-2]	2022-11-24 23:06:32.152000	172.17.0.3	806	127.0.0.1
Executing single-partition query on movies.movies_index [ReadStage-2]	2022-11-24 23:06:32.152000	172.17.0.3	1002	127.0.0.1
Acquiring sstable references [ReadStage-2]	2022-11-24 23:06:32.152000	172.17.0.3	1066	127.0.0.1
Skipped 0/1 non-slice-intersecting sstables, included 0 due to tombstones [ReadStage-2]	2022-11-24 23:06:32.152000	172.17.0.3	1134	127.0.0.1
Key cache hit for sstable 1 [ReadStage-2]	2022-11-24 23:06:32.153000	172.17.0.3	1360	127.0.0.1
Executing single-partition query on movies [ReadStage-2]	2022-11-24 23:06:32.153000	172.17.0.3	2243	127.0.0.1
Acquiring sstable references [ReadStage-2]	2022-11-24 23:06:32.154000	172.17.0.3	2319	127.0.0.1
Key cache hit for sstable 1 [ReadStage-2]	2022-11-24 23:06:32.154000	172.17.0.3	2532	127.0.0.1
Skipped 0/1 non-slice-intersecting sstables, included 0 due to tombstones [ReadStage-2]	2022-11-24 23:06:32.154000	172.17.0.3	2686	127.0.0.1
Merged data from memtables and 1 sstables [ReadStage-2]	2022-11-24 23:06:32.154000	172.17.0.3	3052	127.0.0.1
Read 1 live rows and 0 tombstone cells [ReadStage-2]	2022-11-24 23:06:32.154000	172.17.0.3	3204	127.0.0.1
Merged data from memtables and 1 sstables [ReadStage-2]	2022-11-24 23:06:32.154000	172.17.0.3	3255	127.0.0.1
Enqueuing response to /172.17.0.2:7000 [ReadStage-2]	2022-11-24 23:06:32.154001	172.17.0.3	3297	127.0.0.1
Sending RANGE_RSP message to cassandra/172.17.0.2:7000 message size 118 bytes [Messaging-EventLoop-3-1]	2022-11-24 23:06:32.155000	172.17.0.3	3633	127.0.0.1
RANGE_RSP message received from /172.17.0.3:7000 [Messaging-EventLoop-3-8]	2022-11-24 23:06:32.155000	172.17.0.2	8286	127.0.0.1
Processing response from /172.17.0.3:7000 [RequestResponseStage-3]	2022-11-24 23:06:32.155000	172.17.0.2	8622	127.0.0.1
Request complete	2022-11-24 23:06:32.156574	172.17.0.2	9574	127.0.0.1

It went through these following stages. After parsing and preparing the statement, it finds the index mean cardinalities and scans with the index. It submits a range request on 33 ranges and goes through more stages.

The tablestats appear are shown.

```
Total number of tables: 45
----------------
Keyspace : advanceddb
        Read Count: 5
        Read Latency: 1.837 ms
        Write Count: 50
        Write Latency: 0.05634 ms
        Pending Flushes: 0
                Table: movies
                SSTable count: 1
                Old SSTable count: 0
                Space used (live): 5514
                Space used (total): 5514
                Space used by snapshots (total): 0
                Off heap memory used (total): 40
                SSTable Compression Ratio: 0.7219626168224299
                Number of partitions (estimate): 5
                Memtable cell count: 0
                Memtable data size: 0
                Memtable off heap memory used: 0
                Memtable switch count: 2
                Local read count: 5
                Local read latency: NaN ms
                Local write count: 5
                Local write latency: NaN ms
                Pending flushes: 0
                Percent repaired: 0.0
                Bytes repaired: 0.000KiB
                Bytes unrepaired: 0.418KiB
                Bytes pending repair: 0.000KiB
                Bloom filter false positives: 0
                Bloom filter false ratio: 0.00000
                Bloom filter space used: 24
                Bloom filter off heap memory used: 16
                Index summary off heap memory used: 16
                Compression metadata off heap memory used: 8
                Compacted partition minimum bytes: 73
                Compacted partition maximum bytes: 103
                Compacted partition mean bytes: 93
                Average live cells per slice (last five minutes): NaN
                Maximum live cells per slice (last five minutes): 0
                Average tombstones per slice (last five minutes): NaN
                Maximum tombstones per slice (last five minutes): 0
                Dropped Mutations: 0
                Droppable tombstone ratio: 0.00000
```

The write count has increased since the last tablestats and the