**NoSǪL-CFDB-CQL-PART-I**

**3.2 Use your keyspace**

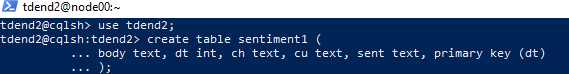


* 1. **Create five tables**

>>>create table **sentiment1** (

body text, dt int, ch text, cu text, sent text, primary key (dt)

);



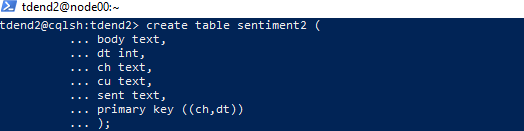
>>>create table **sentiment2** ( body text,

dt int, ch text, cu text,

sent text,

primary key ((ch,dt))

);



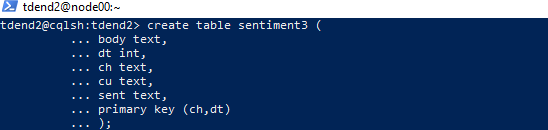
>>>create table **sentiment3** ( body text,

dt int, ch text, cu text,

sent text,

primary key (ch,dt)

);



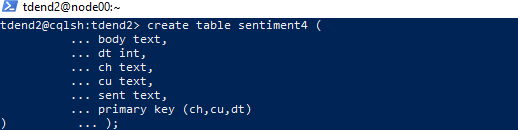
>>>create table **sentiment4** ( body text,

dt int, ch text, cu text,

sent text,

primary key (ch,cu,dt)

);



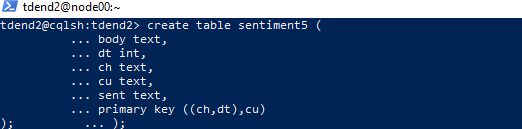
>>>create table **sentiment5** ( body text,

dt int, ch text, cu text,

sent text,

primary key ((ch,dt),cu)

);



* 1. **Populate the tables with data**

begin batch

insert into sentiment1 (body,dt,ch,cu,sent) values ('I am feeling sick.',20160102,'twitter','red bull','negative');

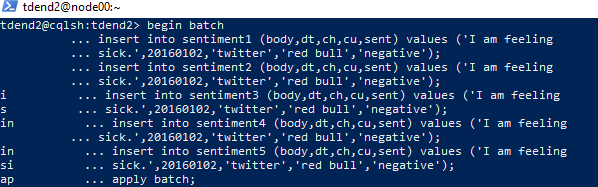
insert into sentiment2 (body,dt,ch,cu,sent) values ('I am feeling sick.',20160102,'twitter','red bull','negative');

insert into sentiment3 (body,dt,ch,cu,sent) values ('I am feeling sick.',20160102,'twitter','red bull','negative');

insert into sentiment4 (body,dt,ch,cu,sent) values ('I am feeling sick.',20160102,'twitter','red bull','negative');

insert into sentiment5 (body,dt,ch,cu,sent) values ('I am feeling sick.',20160102,'twitter','red bull','negative');

apply batch;



**Batch 2: Inserted row2 in all the above five tables created as stated earlier.**

begin batch

insert into sentiment1 (body,dt,ch,cu,sent) values ('That was sick!',20160101,'facebook','red bull','positive');

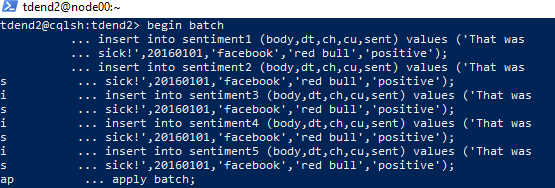
insert into sentiment2 (body,dt,ch,cu,sent) values ('That was sick!',20160101,'facebook','red bull','positive');

insert into sentiment3 (body,dt,ch,cu,sent) values ('That was sick!',20160101,'facebook','red bull','positive');

insert into sentiment4 (body,dt,ch,cu,sent) values ('That was sick!',20160101,'facebook','red bull','positive');

insert into sentiment5 (body,dt,ch,cu,sent) values ('That was sick!',20160101,'facebook','red bull','positive');

apply batch;



# Batch3: insert third row in the above tables.

begin batch

insert into sentiment1 (body,dt,ch,cu,sent) values ('I feel sick, too.',20160102,'facebook','dew tour','negative');

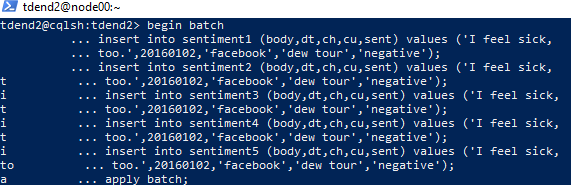
insert into sentiment2 (body,dt,ch,cu,sent) values ('I feel sick, too.',20160102,'facebook','dew tour','negative');

insert into sentiment3 (body,dt,ch,cu,sent) values ('I feel sick, too.',20160102,'facebook','dew tour','negative');

insert into sentiment4 (body,dt,ch,cu,sent) values ('I feel sick, too.',20160102,'facebook','dew tour','negative');

insert into sentiment5 (body,dt,ch,cu,sent) values ('I feel sick, too.',20160102,'facebook','dew tour','negative');

apply batch;



**Batch4:**

begin batch

insert into sentiment1 (body,dt,ch,cu,sent) values ('Dude, you are sick.',20160103,'facebook','red bull','positive');

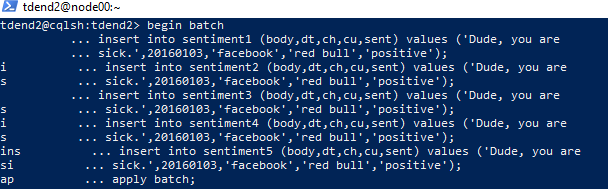
insert into sentiment2 (body,dt,ch,cu,sent) values ('Dude, you are sick.',20160103,'facebook','red bull','positive');

insert into sentiment3 (body,dt,ch,cu,sent) values ('Dude, you are sick.',20160103,'facebook','red bull','positive');

insert into sentiment4 (body,dt,ch,cu,sent) values ('Dude, you are sick.',20160103,'facebook','red bull','positive');

insert into sentiment5 (body,dt,ch,cu,sent) values ('Dude, you are sick.',20160103,'facebook','red bull','positive');

apply batch;



# Batch5

begin batch

insert into sentiment1 (body,dt,ch,cu,sent) values ('How sick was that?',20160103,'facebook','monster energy','positive');

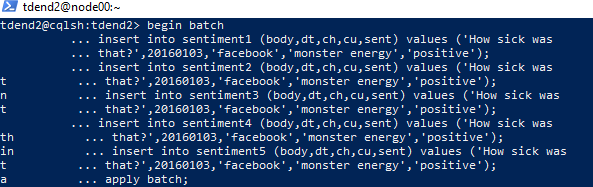
insert into sentiment2 (body,dt,ch,cu,sent) values ('How sick was that?',20160103,'facebook','monster energy','positive');

insert into sentiment3 (body,dt,ch,cu,sent) values ('How sick was that?',20160103,'facebook','monster energy','positive');

insert into sentiment4 (body,dt,ch,cu,sent) values ('How sick was that?',20160103,'facebook','monster energy','positive');

insert into sentiment5 (body,dt,ch,cu,sent) values ('How sick was

that?',20160103,'facebook','monster energy','positive'); apply batch;



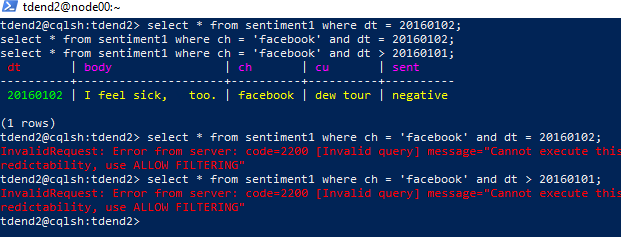
-

# Run queries

**// primary key (dt) in sentiment1 table**

select \* from sentiment1 where dt = 20160102;

select \* from sentiment1 where ch = 'facebook' and dt = 20160102; select \* from sentiment1 where ch = 'facebook' and dt > 20160101;



*>>>select \* from sentiment1 where ch = 'facebook' and dt = 20160102;*

InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thus may have unpredictable performance. If you want to execute this query despite the performance unpredictability, use ALLOW FILTERING"

# REASON:

***In situations where filtering on non-indexed columns is necessary, Cassandra requires explicit permission (ALLOW FILTERING) from the user to execute such queries****. Alternatively, denormalization and pre-aggregation can be used to optimize queries and avoid the need for filtering on non-indexed columns.*

**>>>**tdend2@cqlsh:tdend2> select \* from sentiment1 where ch = 'facebook' and dt > 20160101;

InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thus may have unpredictable performance. If you want to execute this query despite the performance unpredictability, use ALLOW FILTERING"

# REASON:

## Attempting to filter data based on a non-indexed column (dt in this case) without using

***a primary or secondary index caused this error as here only ‘dt’ is the primary key.***

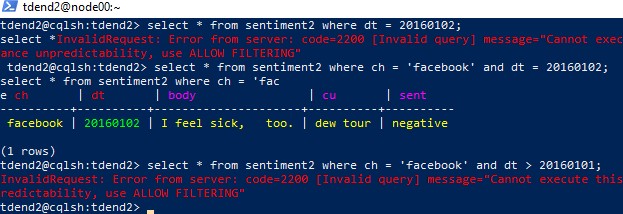
*We can create a secondary index on the* ***dt*** *column to efficiently locate the relevant rows based on the filter condition*.

- -

# // primary key ((ch,dt)) - sentiment2 table

select \* from sentiment2 where dt = 20160102;

select \* from sentiment2 where ch = 'facebook' and dt = 20160102; select \* from sentiment2 where ch = 'facebook' and dt > 20160101;



**>>>**select \* from sentiment2 where dt = 20160102;

select \*InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thus may have unpredictable performance. If you want to execute this query despite the performance unpredictability, use ALLOW FILTERING"

# REASON:

## When querying data without specifying the partition key or based on a non-indexed column (dt) without using a primary or secondary index causes this error as Cassandra must scan all partitions to find the relevant rows and so it works if specified both ch, dt which are part of primary key together as illustrated below:

**>>>**tdend2@cqlsh:tdend2> select \* from sentiment2 where ch = 'facebook' and dt = 20160102;

select \* from sentiment2 where ch = 'fac e ch | dt | body | cu | sent

+- +- + + - facebook | 20160102 | I feel sick, \ntoo. | dew tour | negative

(1 rows)

**>>>**tdend2@cqlsh:tdend2> select \* from sentiment2 where ch = 'facebook' and dt > 20160101;

InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thus may have unpredictable performance. If you want to execute this query despite the performance unpredictability, use ALLOW FILTERING"

# REASON:

***Filtering on non-indexed columns within a partition is efficient but filtering on non- indexed columns across multiple partitions (especially when using inequalities like > or <) can lead to performance issues. Here*** filtering on the partition key **ch** and the clustering column **dt** requires scanning multiple partitions.

- - -

# // primary key (ch,dt) - sentiment3 table

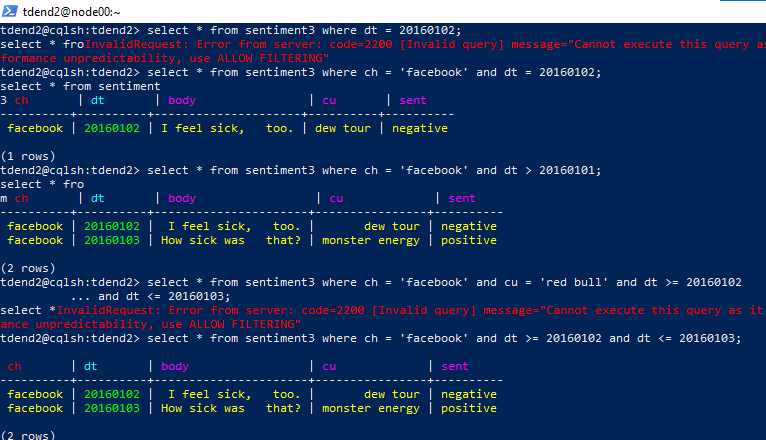
select \* from sentiment3 where dt = 20160102;

select \* from sentiment3 where ch = 'facebook' and dt = 20160102; select \* from sentiment3 where ch = 'facebook' and dt > 20160101;

select \* from sentiment3 where ch = 'facebook' and cu = 'red bull' and dt >= 20160102

and dt <= 20160103;

select \* from sentiment3 where ch = 'facebook' and dt >= 20160102 and dt <= 20160103;



**>>>**select \* from sentiment3 where dt = 20160102;

select \* froInvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thus may have unpredictable performance. If you want to execute this query despite the performance unpredictability, use ALLOW FILTERING"

# REASON:

(ch, dt) forms the primary key, where ch is the partition key and dt is the clustering column. This means that data is partitioned based on the values of ch, and within each partition, rows are ordered by dt. When querying data in Cassandra, it's essential to provide the partition key to ensure efficient data retrieva**l. *Attempting to filter data on a non-primary key column (dt) without providing the partition key (ch) causes this error.***

## -----------------------------------------------------------------------------------------------------------

**>>>**tdend2@cqlsh:tdend2> select \* from sentiment3 where ch = 'facebook' and cu = 'red bull' and dt >= 20160102

... and dt <= 20160103;

select \*InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thus may have unpredictable performance. If you want to execute this query despite the performance unpredictability, use ALLOW FILTERING"

# REASON:

## Here ‘cu’ is non- primary key and so quering on it caused the error.

***----------------------------------------------------------------------------------------------------------***

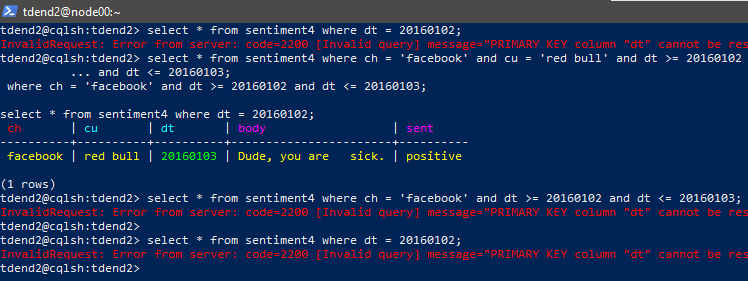
* // primary key (ch,cu,dt)

select \* from sentiment4 where dt = 20160102;

select \* from sentiment4 where ch = 'facebook' and cu = 'red bull' and dt >= 20160102

and dt <= 20160103;

select \* from sentiment4 where ch = 'facebook' and dt >= 20160102 and dt <= 20160103;



**>>>**select \* from sentiment4 where dt = 20160102;

select \* InvalidRequest: Error from server: code=2200 [Invalid query] message="PRIMARY KEY column "dt" cannot be restricted as preceding column "cu" is not restricted"

# REASON:

***When querying data with a compound primary key, all preceding primary key columns must be included in the filter condition before filtering on subsequent columns.***In the provided schema, the primary key is defined as **(ch, cu, dt)**, where **ch** is the partition key and **cu** is the clustering column preceding **dt**. This means that you must filter on **ch** and **cu** before filtering on **dt**.

--------------------------------------------------------------------------------------------

**>>>**tdend2@cqlsh:tdend2> select \* from sentiment4 where ch = 'facebook' and dt >= 20160102 and dt <= 20160103;

InvalidRequest: Error from server: code=2200 [Invalid query] message="PRIMARY KEY column "dt" cannot be restricted as preceding column "cu" is not restricted"

# REASON:

***By filtering on ch, cu, and dt in the correct order****, Cassandra can efficiently locate the requested data within the specified partition.*

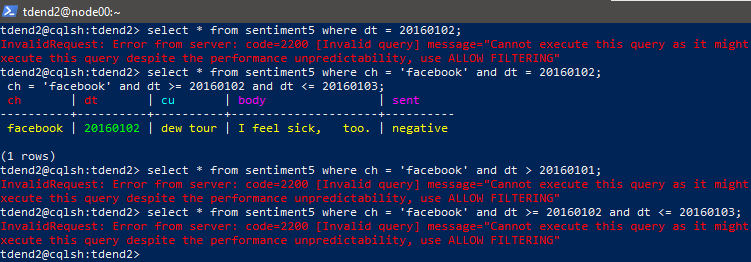
-

# // primary key ((ch,dt),cu) - sentiment5 table

select \* from sentiment5 where dt = 20160102;

select \* from sentiment5 where ch = 'facebook' and dt = 20160102; select \* from sentiment5 where ch = 'facebook' and dt > 20160101;

select \* from sentiment5 where ch = 'facebook' and dt >= 20160102 and dt <= 20160103;



**>>>**select \* from sentiment5 where dt = 20160102;

select InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thus may have unpredictable

performance. If you want to execute this query despite the performance unpredictability, use ALLOW FILTERING"

# REASON:

**((ch, dt), cu) forms the compound primary key, where (ch, dt) is the partition key and cu is the clustering column. This means that data is partitioned based on the combination of ch and dt, and within each partition, rows are sorted by cu.**Specifying both the partition key (**ch**, **dt**) and the clustering column (**cu**) resolves the error as illustrated below:

**>>>**tdend2@cqlsh:tdend2> select \* from sentiment5 where ch = 'facebook' and dt = 20160102;

select \* from sentiment5

w ch | dt | cu | body | sent

+- +- +- + - facebook | 20160102 | dew tour | I feel sick, \ntoo. | negative

(1 rows)

-

**>>>**tdend2@cqlsh:tdend2> select \* from sentiment5 where ch = 'facebook' and dt > 20160101;

selectInvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thus may have unpredictable performance. If you want to execute this query despite the performance unpredictability, use ALLOW FILTERING"

# REASON:

## data is partitioned based on the combination of ch and dt, and within each partition, rows are sorted by cu and here cu is not given in query and so the error occurred.

Specifying the complete primary key resolves the error.

--------------------------------------------------------------------------------------------

**>>>**tdend2@cqlsh:tdend2> select \* from sentiment5 where ch = 'facebook' and dt >= 20160102 and dt <= 20160103;

InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thus may have unpredictable performance. If you want to execute this query despite the performance unpredictability, use ALLOW FILTERING"

# REASON:

Here ‘cu’ is the clustering column using which rows are sorted and it is also part of primary key . So,specifying the complete primary key resolves the error.

============================THE END====================================