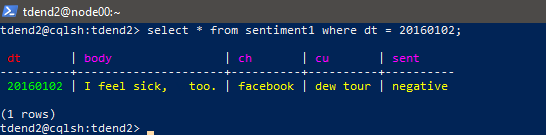
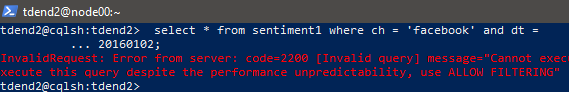
**NoSQL-CFDB-CQL-II**

select \* from sentiment1 where dt = 20160102;



tdend2@cqlsh:tdend2> 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

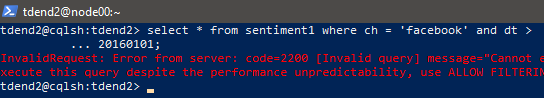
performance. If you want to execute this query despite the performance

unpredictability, use ALLOW FILTERING"

**REASON#** ch is not a part of the primary key

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

20160101;



InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot

execute ...

**REASON#** è ch is not a part of the primary key and ‘>’ is allowed for a

clustering(ordering) key only. Here dt is theprimary key and partition key but not the clustering key and so the error occurred.

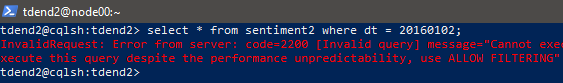
**// Sentiment 2**

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

tdend2@cqlsh:tdend2> select \* from sentiment2 where dt = 20160102;

InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot

execute …

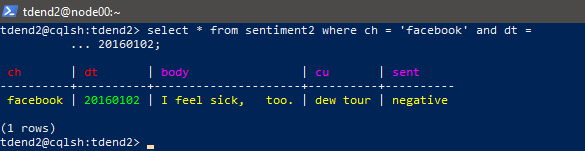


**REASON#** è (ch,dt) is a primary key (composite key) and partition key, so we need to use

both like below.

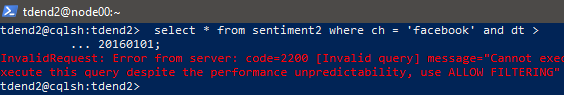
tdend2@cqlsh:tdend2> select \* from sentiment2 where ch = 'facebook' and dt =

20160102;



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

20160101;



InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot

execute ...

**REASON#** è (ch,dt) is a primary key (composite key), but ‘>’ is allowed for a

clustering(ordering) key only.

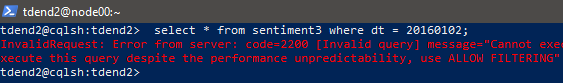
**// Sentiment 3**

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

tdend2@cqlsh:tdend2> select \* from sentiment3 where dt = 20160102;

InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot

execute …

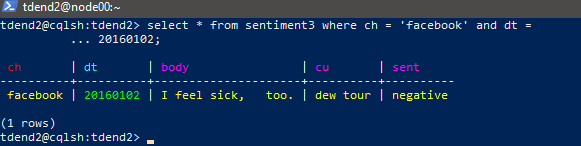


**REASON#** è ch is a partition key, and dt is a clustering key. We should specify the

partition key first, like below.

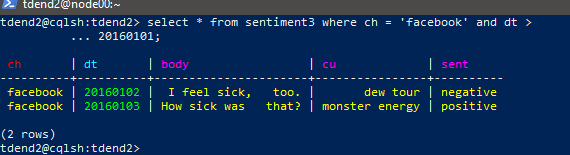
tdend2@cqlsh:tdend2> select \* from sentiment3 where ch = 'facebook' and dt =

20160102;



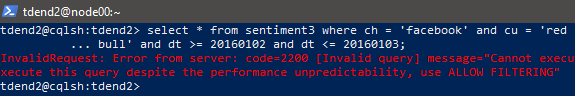
tdend2@cqlsh:tdend2> select \* from sentiment3 where ch = 'facebook' and dt >

20160101;



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

bull' and dt >= 20160102 and dt <= 20160103;



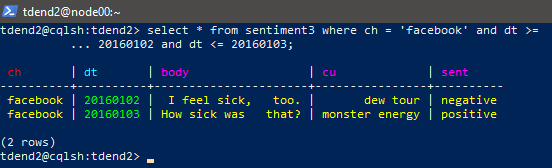
InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot

execute ...

**REASON#** è cu is not a part of the primary key.

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

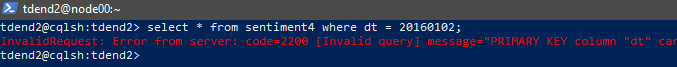
20160102 and dt <= 20160103;



**/ Sentiment 4**

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

tdend2@cqlsh:tdend2> select \* from sentiment4 where dt = 20160102;



InvalidRequest: Error from server: code=2200 [Invalid query] message="PRIMARY KEY

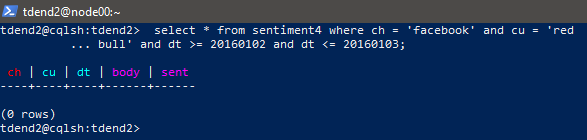
column "dt" cannot be restricted as preceding column "cu" is not restricted"

**REASON#** è ch is a partition key; cu and dt are clustering keys. Need to specify the

partition key first. cu also needs to be specified before dt like below.

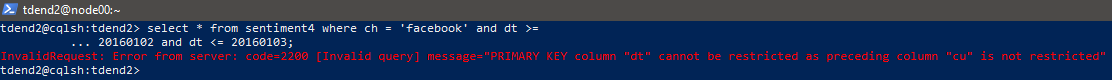
tdend2@cqlsh:tdend2> select \* from sentiment4 where ch = 'facebook' and cu = 'red

bull' and dt >= 20160102 and dt <= 20160103;



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

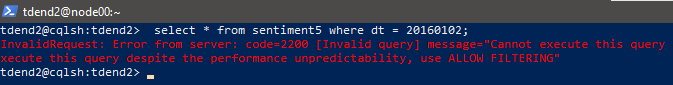
column "dt" cannot be restricted as preceding column "cu" is not restricted"

**REASON#** è cu needs to be specified before dt.

/**/ Sentiment 5**

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

tdend2@cqlsh:tdend2> select \* from sentiment5 where 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

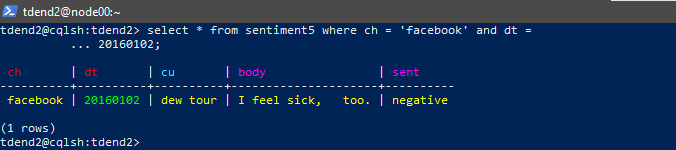
performance. If you want to execute this query despite the performance

unpredictability, use ALLOW FILTERING"

**REASON#** è (ch,dt) is a partition key, and cu is a clustering key. Need ch like below.

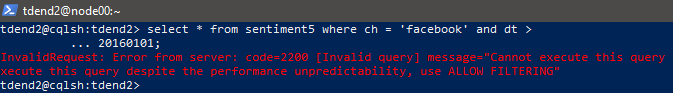
tdend2@cqlsh:tdend2> select \* from sentiment5 where ch = 'facebook' and dt =

20160102;



tdend2@cqlsh:tdend2> select \* from sentiment5 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#** è ‘>’ is not allowed in a partition key whereas it is allowed with a clustering key as rows are ordered using it.

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

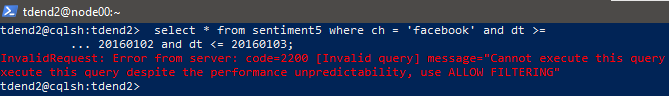
20160102 and dt <= 20160103;

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#** è ‘>=’ or ‘<=’ is not allowed in a partition key but can be used with a clustering key.

—----------------------------------------------------------------------------------------------------------------------------

Answer three questions below (Q1-Q3)

***Question 1 (20 points)***

***Did the query "select \* from sentimentX where ch = 'facebook' and dt >= 20160102 and dt <=***

***20160103;" work for any of the tables? Why or why not? Also, explain the primary key, partition***

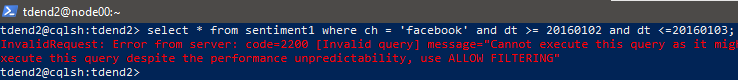
***key, or clustering key. Like Section 3.5, explain those for one table at a time (5 tables)***

***Note: Don’t mess up names, primary key, partition key, and clustering key when you explain the***

***reasons or results***

SENTIMENT1:

select \* from sentiment1 where ch = 'facebook' and dt >= 20160102 and dt <=20160103;



primary key: dt

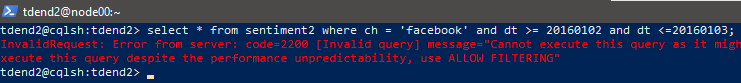
partition key:dt

and clustering key: not provided in schema

So, dt cannot be used as a clustering column as it is the partition key and also ch is a non-primary key which can also not be used to filter the query on non-equality operations.

SENTIMENT2:

select \* from sentiment2 where ch = 'facebook' and dt >= 20160102 and dt <=20160103;



primary key: ((ch,dt)) here is a compound primary key

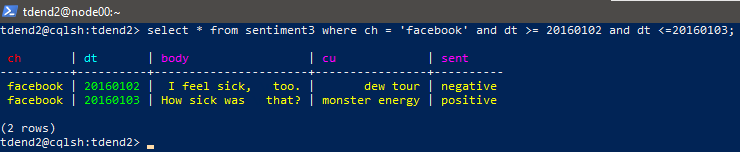
partition key:((ch,dt)) should be used together to locate a specified row

clustering key: not provided in schema

So, dt cannot be used as a clustering column as it is the partition key. Clustering key should be used to perform non-eqaulity operations on a query.

SENTIMENT3:

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



primary key: (ch,dt) is a compound primary key/ composite key

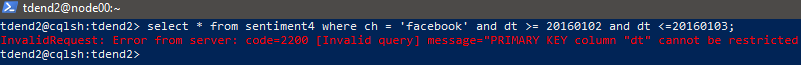
partition key:ch

and clustering key: dt

So, dt used as a clustering column is used to order rows and ch as the partition key specified in the order provided in the schema to filter the columns required in a query and so the result is obtained.

SENTIMENT4:

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



primary key: (ch,cu,dt) is a compound primary key/ composite key

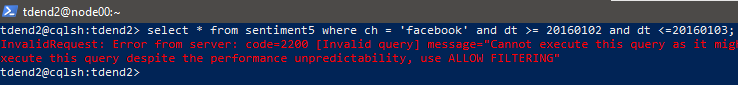
partition key: ch

and clustering key:(cu,dt)

Here, ch, cu, dt should be provided to filter just as provided in the schema following the same order but cu is missing here which has to be provided before dt..

SENTIMENT5:

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



primary key: ((ch,dt),cu) is a compound primary key/ composite key

partition key:(ch,dt) is a composite partition key

and clustering key: cu

So, dt here cannot be used for‘>=’ or ‘<=’ operations as it is not the clustering column and (ch,dt) as the partition key specified in the order provided in the schema to filter the columns required in a query but cu is not provided after dt and so the error occurred.

—----------------------------------------------------------------------------------------------------------------------------

***Question 2 What would you do if you needed to find all messages with a positive sentiment?***

select \* from sentimentX where sent=’positive’ ***does not give the required result as ‘sent’***

***Is neither a partition column nor a clustering column*** and so ***ALLOW FILTERING can be used to get the result.*** Here as non-equality operations are not used in the query ***if sent would have been a partition key could give the desired result.***

***Other method:***

If ‘sent column is a secondary indexed column in all the tables , then the desired result can be obtained without compromising on performance.

Sentiment1:

If ‘sent’ added either be a partition key or clustering key or secondary indexed:

select \* from sentiment1 where dt=’date where positive sentiment is available’ and sent=’positive’

Sentiment2:

If ‘sent’ would be a clustering key:

select \* from sentiment2 where ch=’facebook’ and dt=’date of positive sentiment’ and sent=’positive’

Sentiment3:

If ‘sent’ would be added as clustering key ora secondary indexed column:

select \* from sentiment3 where ch=’facebook’ and dt [non-equality operation] ’date of positive sentiment’ and sent=’positive’

Sentiment4:

If ‘sent’ would be added as clustering key or as a secondary indexed column to the existent table:

select \* from sentiment4 where ch=’facebook’ and cu=’[customer]’ and dt [non-equality operation] ’date of positive sentiment’ and sent=’positive’

Sentiment5:

If ‘sent’ was a clustering key as part of primary key of the existent table or secondary indexed :

select \* from sentiment5 where ch=’facebook’ and dt=’date of positive sentiment’ and cu=’[customer]’ and sent=’positive’

***—----------------------------------------------------------------------------------------------------------------------------***

***Question 3***

***In the real world, how many tweets are posted per day? As of this writing, Twitter generates***

***~500M tweets/day. (https://www.omnicoreagency.com/twitter-statistics/) Let's say we need to***

***run a query that captures all tweets over a specified time range. Given our data model scenario,***

***we simply data model a primary key value of (ch, dt) to capture all tweets in a single Cassandra***

***row sorted in order of time, right? Easy! But, alas, the Cassandra logical limit of single row size***

***(2 billion cells per partition in Cassandra) would fill up after about 4 days! Our primary key***

***won't work. What would you do to solve our query or this problem?***

This can be solved using the following ways:

1)See that each partition will contain a subset of tweets for a specific time period, ensuring that you don't exceed the row size limit.

2)Implement data archiving and retention policies to periodically move older tweets to separate storage or databases.

3)Distribute data across multiple Cassandra nodes or clusters to distribute the workload and mitigate the risk of hitting scalability limits.

4)Consider adding secondary indexes or denormalizing data if it improves query performance without significantly increasing storage requirements.

**===========================THE END============================**