Example 10:
cqlsh
Restrictions on partition key

Let's revisit the column definition of product

```
cassandra@cqlsh:catalog> CREATE COLUMNFAMILY product
  productId varchar,
  title text,
  brand varchar,
  publisher varchar,
  length int,
  breadth int,
  height int,
  PRIMARY KEY(productId)
);
```

Here productId is our partition key and we have not defined any clustering columns

We want to list the books published by Riverhead Books

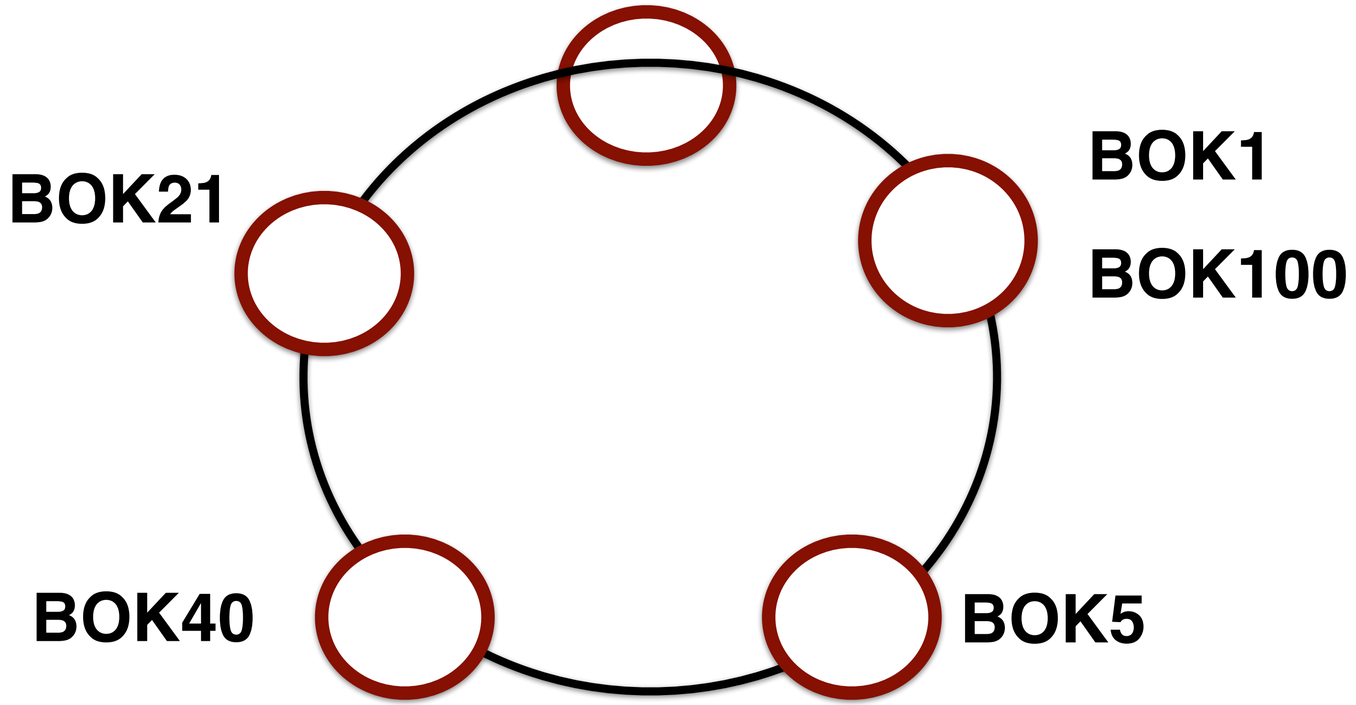
```
cassandra@cqlsh:catalog> select productid from product where publisher='Riverhead';
```

```
InvalidRequest: code=2200 [Invalid query] message="Cannot execute this query as it
might involve data filtering and thus may have unpredictable performance. If you wa
nt to execute this query despite the performance unpredictability, use ALLOW FILTER
ING"
```

What went wrong?

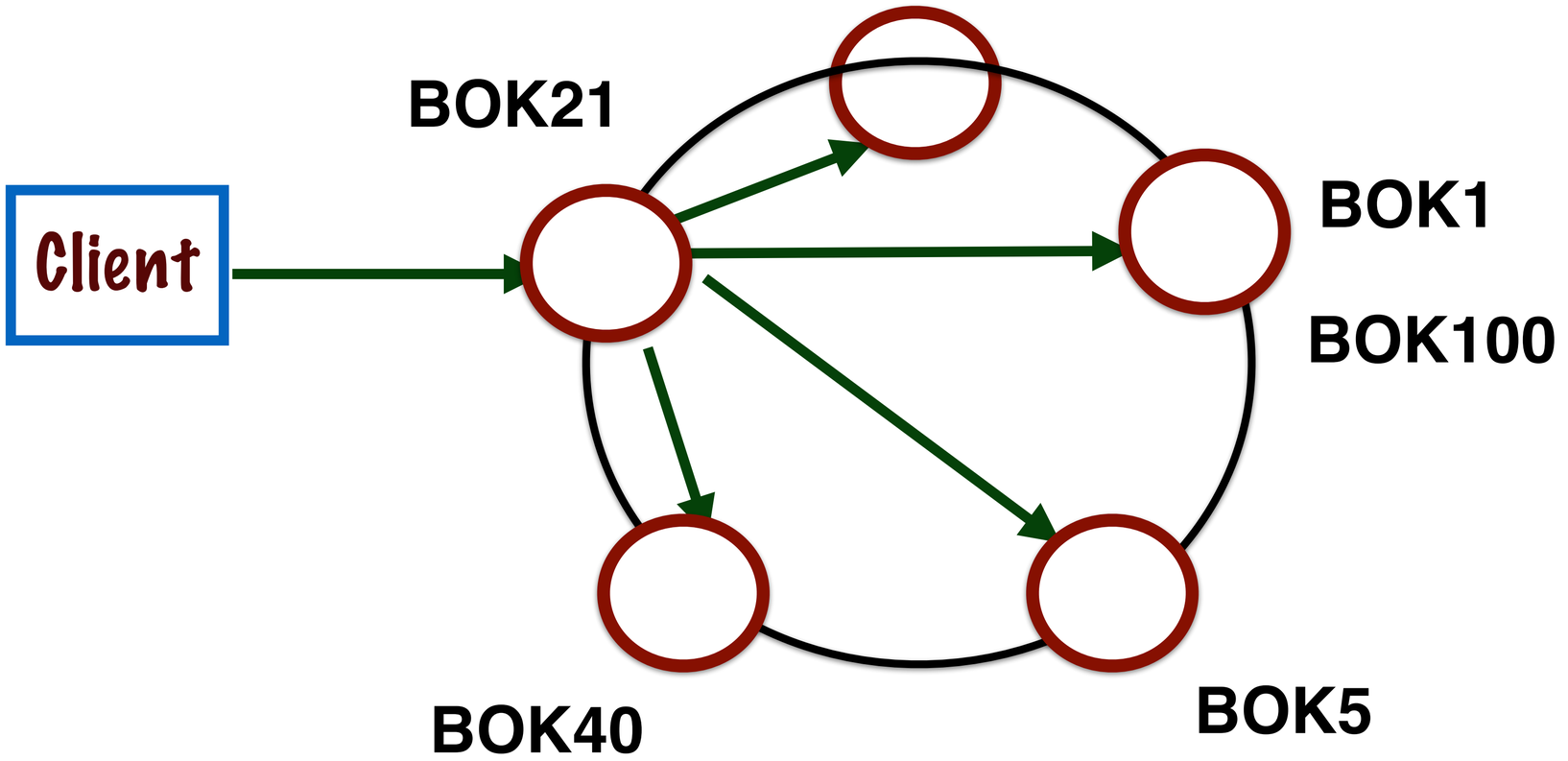
The Murmur3Partitioner distributes data uniformly across the cluster

So products of the book category are scattered across the cluster



```
cassandra@cqlsh:catalog> select productid from product where publisher='Riverhead';
```

For a query like this, the oordinator node will send requests to ALL the nodes in the cluster to get the result



And here if even 1 intermediate request fails, then the client request results in an error

```
cassandra@cqlsh:catalog> select productid from product where publisher='Riverhead';
```

if this query was run, cassandra would have to scan the entire column family across all nodes

Read will timeout while executing this query

To prevent such inefficient queries

cassandra has restrictions on

- the columns which can be queried
- the operations that are supported by the columns

Let's go through the restrictions on the partition key

Partition key restrictions

Let's create a columnfamily "skus" which stores metadata about the products

```
cassandra@cqlsh:catalog> CREATE COLUMNFAMILY skus (
        sellerId varchar,
        productId varchar,
        skuId varchar,
        title text,
        listingId varchar,
        isListingCreated boolean,
        timeofskucreation text,
        PRIMARY KEY((sellerId,skuId), timeofskucreation, productId));
```

Partition key restrictions

```
cassandra@cqlsh:catalog> CREATE COLUMNFAMILY skus (
        sellerId varchar,
        productId varchar,
        skuId varchar,
        title text,
        listingId varchar,
        isListingCreated boolean,
        timeofskucreation text,
        PRIMARY KEY (sellerId,skuId), timeofskucreation, productId));
```

(sellerId,skuId) make the partition key

Partition key restrictions

```
cassandra@cqlsh:catalog> CREATE COLUMNFAMILY skus (
        sellerId varchar,
        productId varchar,
        skuId varchar,
        title text,
        listingId varchar,
        isListingCreated boolean,
        timeofskucreation text,
        PRIMARY KEY((sellerId,skuId), timeofskucreation, productId );
```

timeofskucreation and productid are the clustering columns

Partition key restrictions

We have data for some SKUs in a csv file

```
Next,COM1,SKU1,Acer One,LISTINGNextCOM1,true,2015-08-12 12:21
Next,COM2,SKU2,Acer One,LISTINGNextCOM2,true,2015-03-01 11:11
Chroma,COM1,CHROMASKU1,Acer One,,false,2016-01-01 00:01
Fab,SOFA1,FABSKU,Urban Living Derby,LISTINGFabSOFA1,true,2015-12-11 12:21
Decor,SOFA1,DECORSKU,Urban Living Derby,,false,2016-07-01 22:30
SOFA3,Urban Living 4 seater,Fab,FABSKU,2015-12-11 12:21,False
SOFA2,Urban Living 2 seater,Fab,FABSKU1,2015-12-11 12:21,False
```

Partition key restrictions

We can import this data in the columnfamily(CF) by using the COPY command

```
cassandra@cqlsh:catalog> COPY skus (sellerid, productid, skuid, title, listingid,
islistingcreated,timeofskucreation) FROM 'skus.csv' ;
```

We provide the columns for which the file has data

Partition key restrictions

```
cassandra@cqlsh:catalog> select * from skus;

 sellerid | skuid       | timeofskucreation | productid | islistingcreated | listing
id        | title
----------+-------------+-------------------+-----------+------------------+--------
---------+----------------------
    Chroma | CHROMASKU1 |  2016-01-01 00:01 |      COM1 |            False |
     null |                Acer One
     Decor |    DECORSKU |  2016-07-01 22:30 |     SOFA1 |            False |
     null |     Urban Living Derby
      Next |        SKU2 |  2015-03-01 11:11 |      COM2 |             True | LISTING
NextCOM2 |                Acer One
       Fab |     FABSKU1 |  2015-12-11 12:21 |     SOFA2 |            False |
     null | Urban Living 2 seater
      Next |        SKU1 |  2015-08-12 12:21 |      COM1 |             True | LISTING
NextCOM1 |                Acer One
       Fab |      FABSKU |  2015-12-11 12:21 |     SOFA1 |             True | LISTING
FabSOFA1 |     Urban Living Derby
       Fab |      FABSKU |  2015-12-11 12:21 |     SOFA3 |            False |
     null | Urban Living 4 seater

(7 rows)
```

data is imported in the skus CF

Partition key restrictions

Let's query with only 1 column of the partition key - sellerid

```
cassandra@cqlsh:catalog> select * from skus where sellerid = ('Decor');
```

```
InvalidRequest: code=2200 [Invalid query] message="Partition key parts: skuid must
be restricted as other parts are"
```

What went wrong?

Partition key restrictions

Let's query with only 1 column of the partition key - sellerid

```
cassandra@cqlsh:catalog> select * from skus where sellerid = ('Decor');
```

```
InvalidRequest: code=2200 [Invalid query] message="Partition key parts: skuid must
be restricted as other parts are"
```

token for the row is generated by hashing the data of the partition key

For the skus column family, data of the partition key is <sellerid data, skuid data>

Partition key restrictions

Let's query with only 1 column of the partition key - sellerid

```
cassandra@cqlsh:catalog> select * from skus where sellerid = ('Decor');
```

output

```
InvalidRequest: code=2200 [Invalid query] message="Partition key parts: skuid must
be restricted as other parts are"
```

Without skuid data, the token cannot be generated

And without a token, coordinator node will not be able to know which node owns the partition

Partition key restrictions

Let's query with only 1 column of the partition key - sellerid

```
cassandra@cqlsh:catalog> select * from skus where sellerid = ('Decor');
```

output

```
InvalidRequest: code=2200 [Invalid query] message="Partition key parts: skuid must
be restricted as other parts are"
```

Let's understand the error message

Partition key restrictions

Let's query with only 1 column of the partition key - sellerid

```
cassandra@cqlsh:catalog> select * from skus where sellerid = ('Decor');
```

output

```
InvalidRequest: code=2200 [Invalid query] message="Partition key parts: skuid must
be restricted as other parts are"
```

restricted here means that definite data should be provided for the column in the query

Query condition is required on the skuid column as well

i.e the skuld column should be restricted

Partition key restrictions

Let's query with only the sellerid

```
cassandra@cqlsh:catalog> select * from skus where sellerid = ('Decor');
```

output

```
InvalidRequest: code=2200 [Invalid query] message="Partition key parts: skuid must
be restricted as other parts are"
```

All the columns of the partition key should be restricted in the query

Partition key restrictions

All the columns of the partition key should be restricted in the query

Partition key restrictions

Similarly IN query will fail, if all the columns of partition key are not restricted

```
cassandra@cqlsh:catalog> select * from skus where sellerid IN ('Decor');
```

```
InvalidRequest: code=2200 [Invalid query] message="Partition key parts: skuid must
be restricted as other parts are"
```

Partition key restrictions

Let's do a range query using '>' on the partition key

```
cassandra@cqlsh:catalog> select * from skus where sellerId > 'Chroma' and skuid > '
ChromaSKU1';
```

```
InvalidRequest: code=2200 [Invalid query] message="Only EQ and IN relation are supp
orted on the partition key (unless you use the token() function)"
```

We can perform range operations efficiently only if the stored data is sorted

Partition key restrictions

Let's do a range query using '>' on the partitioner key

```
cassandra@cqlsh:catalog> select * from skus where sellerId > 'Chroma' and skuid > '
ChromaSKU1';
```

```
InvalidRequest: code=2200 [Invalid query] message="Only EQ and IN relation are supp
orted on the partition key (unless you use the token() function)"
```

Since, the partitions are distributed across the cluster, we cannot perform range operations on them

Partition key restrictions

Let's do a range query using '>' on the partitioner key

```
cassandra@cqlsh:catalog> select * from skus where sellerId > 'Chroma' and skuid > '
ChromaSKU1';

InvalidRequest: code=2200 [Invalid query] message="Only EQ and IN relation are supp
orted on the partition key (unless you use the token() function)
```

We cannot use the >, >=, <=, < operator directly on the partition key

Partition key restrictions

All the columns of the partition key should be restricted in the query

We cannot use > , >=, <=, < operator directly on the partition key

Only IN and = operators are allowed on the partition key

Partition key restrictions

Let's do a range query using '>' on the partitioner key

```
cassandra@cqlsh:catalog> select * from skus where sellerId > 'Chroma' and skuid > '
ChromaSKU1';
```

output

```
InvalidRequest: code=2200 [Invalid query] message="Only EQ and IN relation are supp
orted on the partition key (unless you use the token() function)"
```

The coordinator node knows the token range of all nodes

Partition key restrictions

Range query using '>' using token (partition key)

```
cassandra@cqlsh:catalog> select * from skus where token(sellerid,skuid) >= token('C
hroma', 'ChromaSKU1') ;
```

```
 sellerid | skuid    | timeofskucreation | productid | islistingcreated | listingid
      | title
----------+----------+-------------------+-----------+------------------+-----------
------+----------------------
      Fab | FABSKU1 |  2015-12-11 12:21 |     SOFA2 |            False |
 null | Urban Living 2 seater
     Next |    SKU1 |  2015-08-12 12:21 |      COM1 |             True | LISTINGNex
tCOM1 |               Acer One
      Fab |  FABSKU |  2015-12-11 12:21 |     SOFA1 |             True | LISTINGFab
SOFA1 |    Urban Living Derby
      Fab |  FABSKU |  2015-12-11 12:21 |     SOFA3 |            False |
 null | Urban Living 4 seater

(4 rows)
```

Partition key restrictions

All the columns of the partition key should be restricted in the query

We cannot use > , >=, <=, < operator directly on the partition key

Only IN and = operators are allowed on the partition key

We can use > , >=, <=, < operator on the token

Partition key restrictions

Let's order the data by one of the partition key

```
cassandra@cqlsh:catalog> select * from skus where sellerid = 'Fab' AND skuid in ('F
ABSKU') order by sellerid;

InvalidRequest: code=2200 [Invalid query] message="Order by is currently only
supported on the clustered columns of the PRIMARY KEY, got sellerid"
```

data cannot be ordered on partition key

Partition key restrictions

All the columns of the partition key should be restricted in the query

We cannot use > , >=, <=, < operator directly on the partition key

Only IN and = operators are allowed on the partition key

We can use > , >=, <=, < operator on the token

ORDER BY is not supported with partition key

Let's query the skus CF with a restricted partition key

```
cassandra@cqlsh:catalog> select * from skus where sellerid in ('Decor', 'Fab') and
skuid IN('FABSKU', 'DECORSKU');

 sellerid | skuid     | timeofskucreation | productid | islistingcreated | listingid
        | title
----------+-----------+-------------------+-----------+------------------+----------
-------+--------------------
    Decor | DECORSKU |  2016-07-01 22:30 |     SOFA1 |            False |
  null | Urban Living Derby
      Fab |    FABSKU |  2015-12-11 12:21 |     SOFA1 |             True | LISTINGFa
bSOFA1 | Urban Living Derby

(2 rows)
```

Partition key restrictions

All the columns of the partition key should be restricted in the query

We cannot use > , >=, <=, < operator directly on the partition key

Only IN and = operators are allowed on the partition key

We can use > , >=, <=, < operator on the token

ORDER BY is not supported with partition key