Primary Keys

Understanding Primary Keys

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)
);
```

We have defined productId as our primary key

Understanding Primary Keys

What is a primary key?

is a column or a set of columns which can uniquely identify rows in a columnfamily

eg for our product columnfamily, productid is enough to distinguish between different products

(No 2 products will ever have the same productId)

Understanding Primary Keys

What is a primary key?

But in Cassandra there is more to
a primary key...

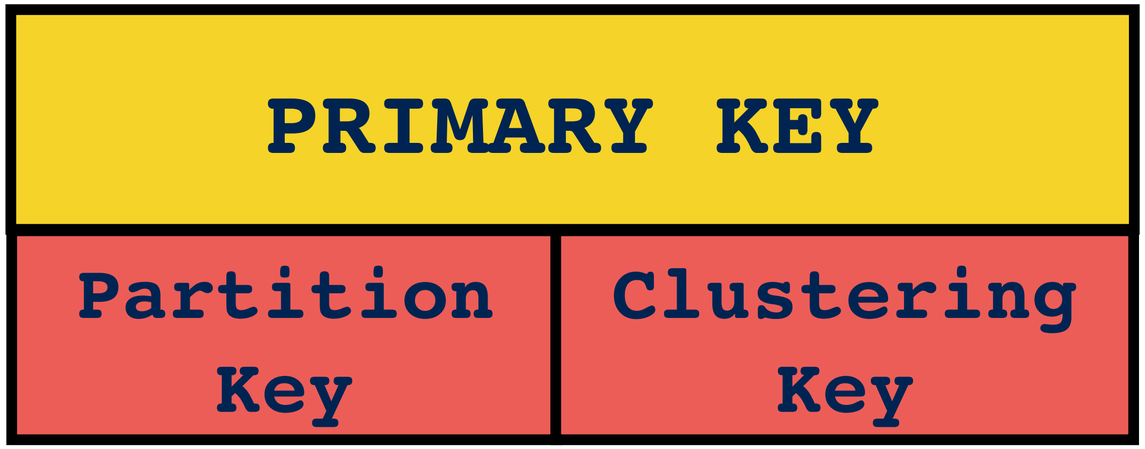
Primary key also tells cassandra
how to store and distribute our data

Let's understand this in detail..

Understanding Primary Keys

What is a primary key?

A Primary key consists of 2 parts



Partition key decides how the data is distributed

Clustering key decides how the data is stored on the disk

Understanding Primary Keys

What is a primary key?

If there are multiple columns in the primary key

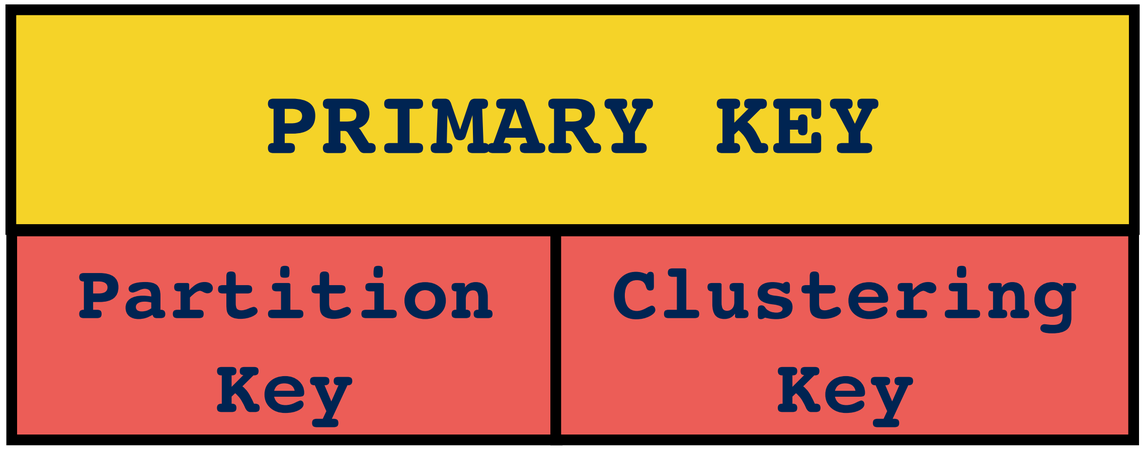
The first column becomes the partition key

The partition key can have multiple columns

The remaining columns form the clustering key

Understanding Primary Keys

What is a primary key?



Partition key decides how the data is distributed

Clustering key decides how the data is stored on the disk

Understanding Primary Keys

How is data distributed?

Each node in cassandra is assigned a unique token

This token determines WHICH ROWS this node is going to store

How is this token generated?

Understanding Primary Keys

Token Generation

Tokens are hash values generated by a hashing function called Partitioner

It generates hash values in the range
[-2^63 to +2^63 - 1]

These tokens are distributed among the nodes

Understanding Primary Keys

Token Generation

Single data center

Calculate tokens by dividing the hash range by the number of nodes in the cluster

Multiple data center

First calculate the tokens for each data center

And then divide the tokens among the nodes in the datacenter

Understanding Primary Keys

Token Generation

Let's say the range for tokens is 1-100

Number of nodes in the cluster is 5

Node 1: Tokens 1-20

Node 2: Tokens 21-40

Node 3: Tokens 41-60

Node 4: Tokens 61-80

Node 4: Tokens 81-100

Understanding Primary Keys

What is a primary key?

Token Generation

You might recall the output of `ccm node1 show` command

```
node1: UP
       cluster=easybuy
       auto_bootstrap=False
       thrift=('127.0.0.1', 9160)
       binary=('127.0.0.1', 9042)
       storage=('127.0.0.1', 7000)
       jmx_port=7100
       remote_debug_port=0
       byteman_port=0
       initial_token=-9223372036854775808
       pid=3028
```

The first token in the token range that node1 owns is the initial_token

node1: [-9223372036854775808, -5534023222112865484)

Understanding Primary Keys

What is a primary key?

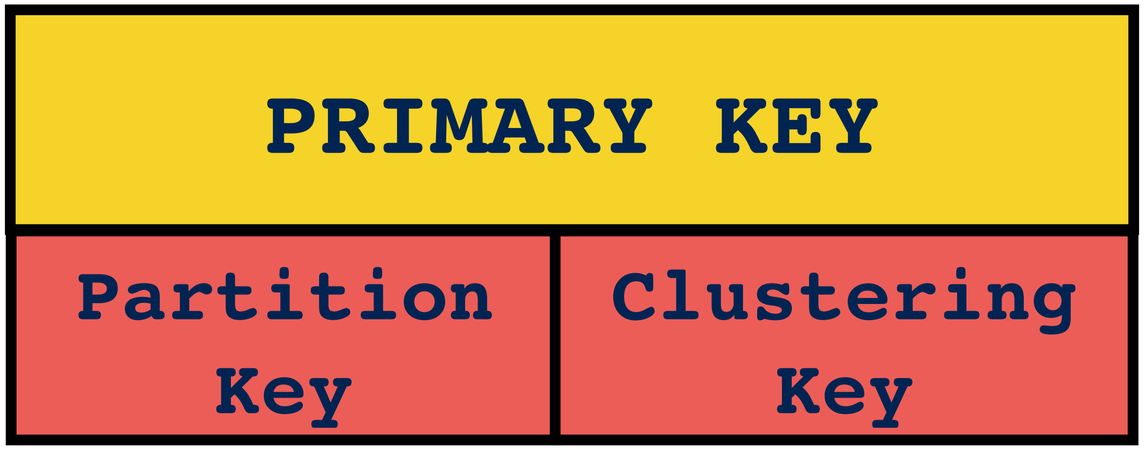
Token Generation

But how does this token determine which rows it stores?

This will be determined by the Partition Key

Understanding Primary Keys

Partition Key

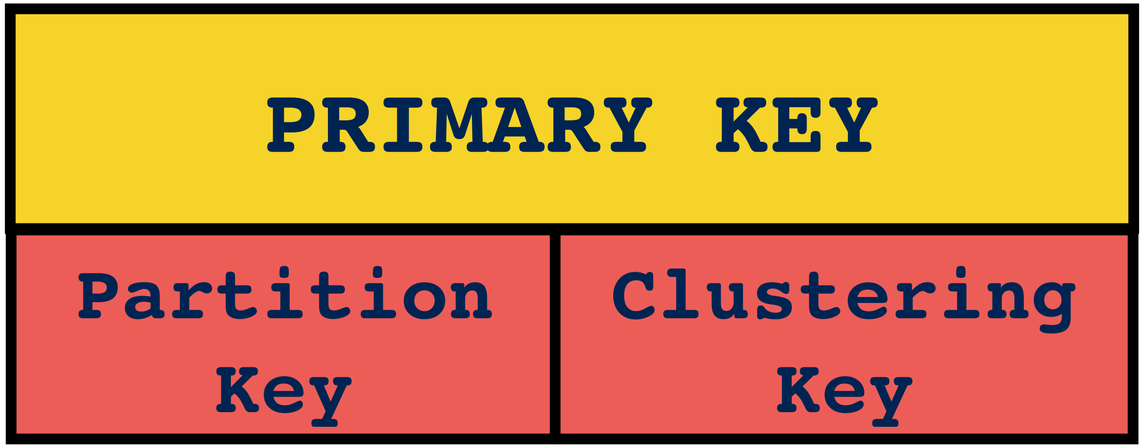


The Partitioner generates a token
from the partition key of a row

Understanding Primary Keys

Partition Key

The Partitioner generates a token from the partition key of a row



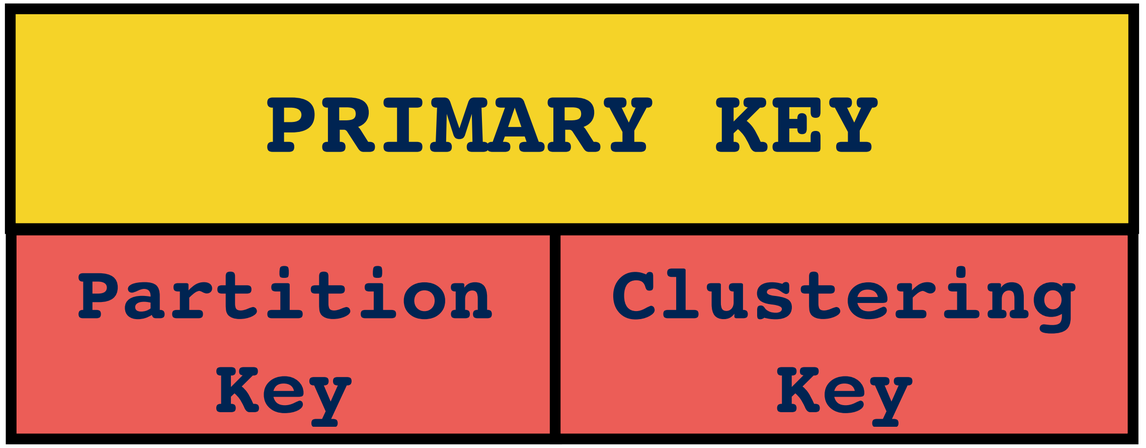
Every node has a range of tokens

The row is assigned to the node which contains the token generated from its partition key!

Understanding Primary Keys

Partition Key

The Partitioner generates a token from the partition key of a row



Every node has a range of tokens

The row is assigned to the node which contains the token generated from its partition key!

From our example a row with token 7 will be assigned to node1

Understanding Primary Keys

Partition Key

Then according to the replication_factor, a number of copies of a row are made

ReplicaPlacementStrategy decides on which nodes the replicas should be stored

Understanding Primary Keys

Partition Key

Points to be noted

Row data is not segmented across nodes

If a partition key exists on the node,
the entire row will exist on that node

all the rows sharing the same partition key (even across columnfamilys) are stored on the same physical node

Understanding Primary Keys

Partitioner

Let's have an overview of the different partitioners in Cassandra

RandomPartitioner

distributes data uniformly across the cluster using MD5 Hash

Understanding Primary Keys

Partitioner

Let's have an overview of the different partitioners in Cassandra

RandomPartitioner

Murmur3Partitioner

This is the default partitioner for cassandra

distributes data uniformly across the cluster using MurmurHash

Understanding Primary Keys

Partitioner

Let's have an overview of the different partitioners in Cassandra

RandomPartitioner

Murmur3Partitioner

Hashing speed and performance of Murmur3 is better than the RandomPartioner

Understanding Primary Keys

Partitioner

Let's have an overview of the different partitioners in Cassandra

RandomPartitioner

Murmur3Partitioner

Both of these partitioners do not allow range or aggregate queries

Understanding Primary Keys

Partitioner

Let's have an overview of the different partitioners in Cassandra

RandomPartitioner

Murmur3Partitioner

ByteOrderedPartitioner

This partitioner is never used. It is still supported only for backward compatibility

Understanding Primary Keys

Partitioner

ByteOrderedPartitioner

partitions using hexadecimal representation of the leading character(s) in the data of the partition key

keys are stored in a sorted way

Understanding Primary Keys

Partitioner

ByteOrderedPartitioner

e.g in product, book products will be stored together ordered by productId

This enables us to perform range queries

Then why don't we use it?

Understanding Primary Keys

Partitioner

ByteOrderedPartitioner

Limitations

Difficult to load balance

Some nodes will always get more requests than the others

we will not be able to achieve a balanced qps on all nodes

Understanding Primary Keys

PARTITIONER

ByteOrderedPartitioner

Limitations

Hotspots

Sequential writes can cause hot spots

e.g. Sale on mobiles will result in heavy load on nodes containing mobile keys

Understanding Primary Keys

PARTITIONER

ByteOrderedPartitioner

Limitations

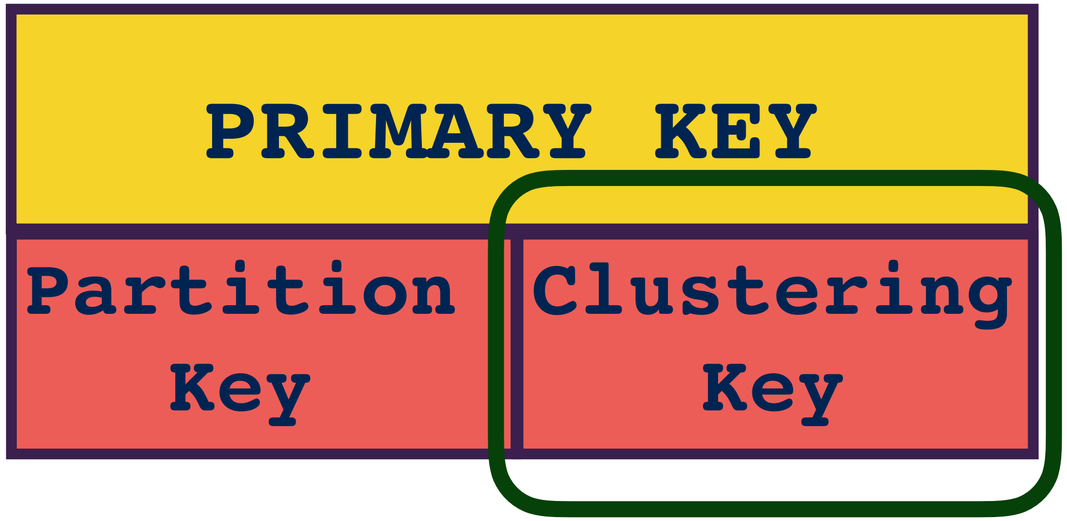
Hotspots

Difficult to load balance

MAKES THE SYSTEM UNSTABLE

Understanding Primary Keys

If there is more than 1 column in the primary key, it is called compound primary key



Understanding Primary Keys

Clustering Key

The columns in the clustering key determine how the data is stored on the disk within a single node

Understanding Primary Keys

Storage within a node

For a compound primary key which has 1 partition key (PK) and 2 clustering column keys (CK-a, CK-b)

PK1	CK-a	CK-b	rest of the columns

Understanding Primary Keys

Storage within a node

If multiple rows have the same partition key

PK1	CK-a1	CK-b1	rest of the columns
	CK-a2	CK-b2	rest of the columns
	CK-a3	CK-b3	rest of the columns

The rows are sorted by the values of the clustering keys

By one column and then the other