Cassandra NoSQL

Fast, Distributed, High availability, linear scalability

By :
Mala Das (d.mala@iitg.ernet.in)
Hemanta Baruah (hemanta.b@iitg.ernet.in)

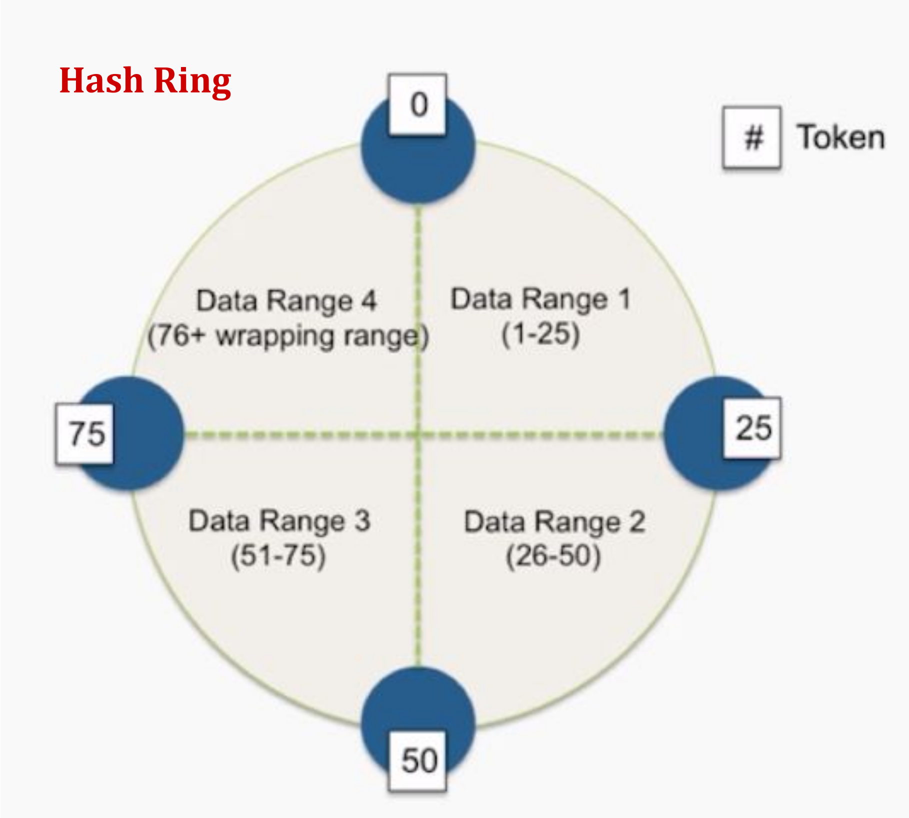
Objective

Cassandra

- Introduction to Cassandra
- Internal Architecture
- Data Model
- Data Model Principles
- Dataset - **Twitter**
- CQL - syntax
- References

Introduction to Cassandra

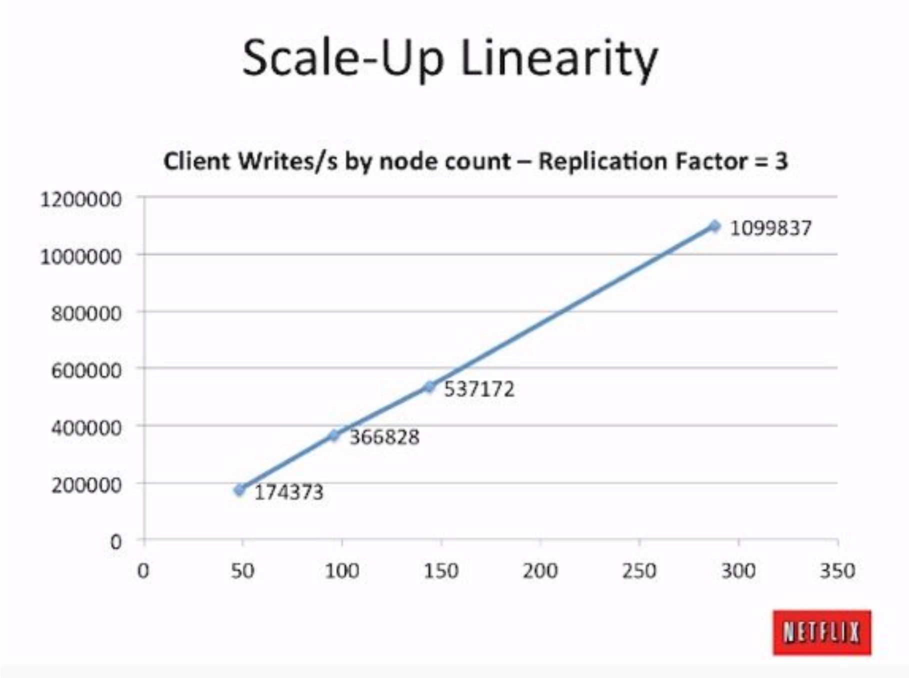
Cassandra Introduction



Hash Ring

- No - master / slave (peer -to -peer)
- Each node contain ranges of hashes(or token)
- Data is partitioned(distributed) around the ring
- Data is replicated to RF=N servers
- Location of data on ring is determined by Partition Key

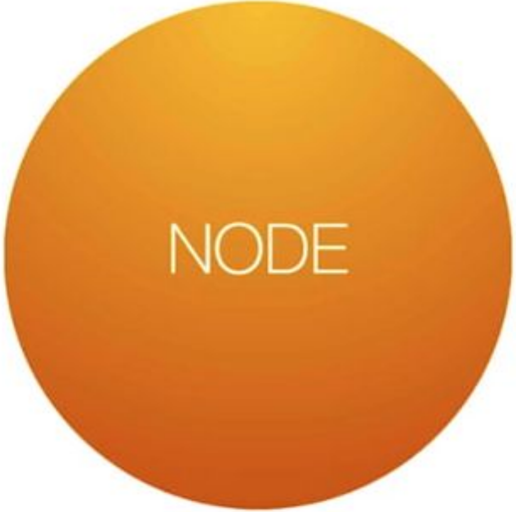
Cassandra Introduction continue.....



- Fast distributed Database
- High Availability
- Linear Scalability
- Predictable Performance
- Multi Data Center (DC)
- Commodity Hardware
- Easy to manage large cluster
- Not a drop in replacement for RDBMS

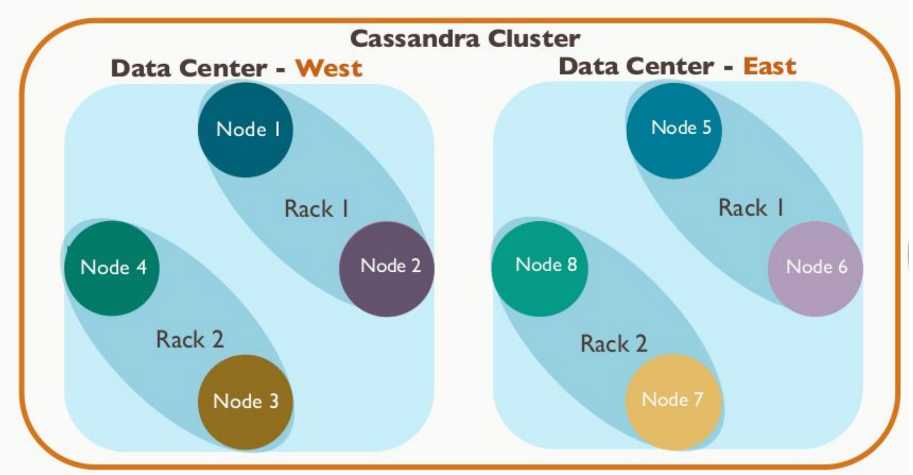
Internal Architecture

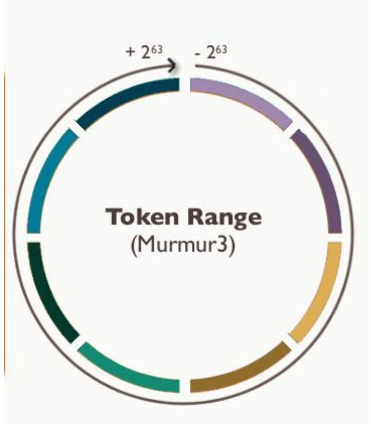
Standalone - Custer

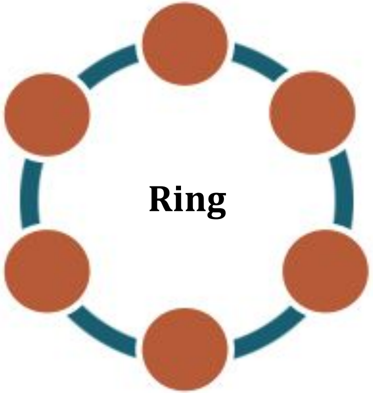


- **Cluster having only one node**
  - **Node** - one Cluster instance

Architecture



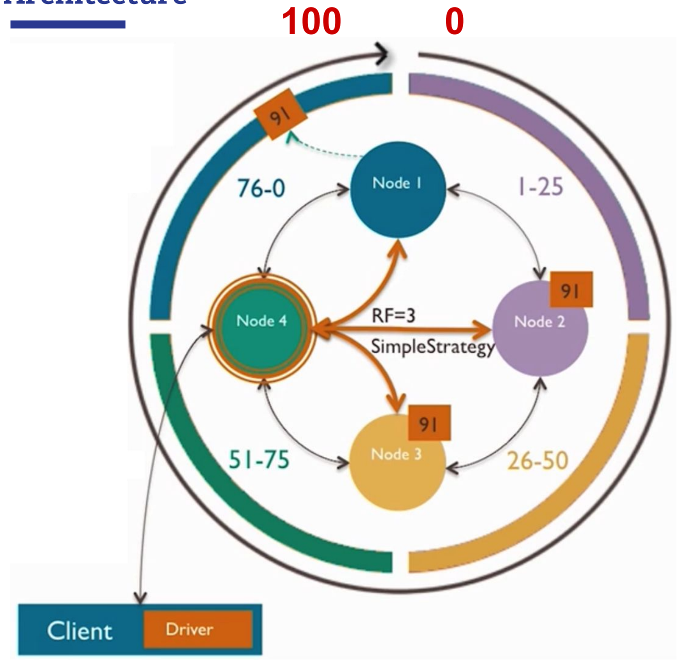


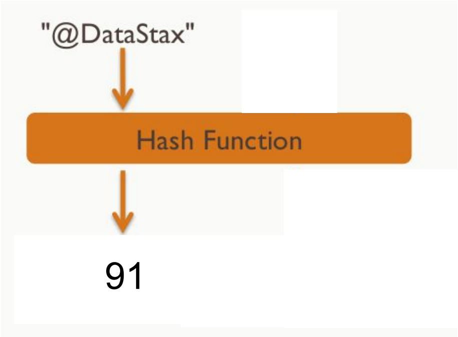


Distributed Cluster

- **A peer to peer set of nodes**
  - **Node** - one Cluster instance
  - **Rack** - a logical set of nodes
  - **Data Center** - a logical set of racks
  - **Cluster** - the full set of nodes (a single complete token ring)

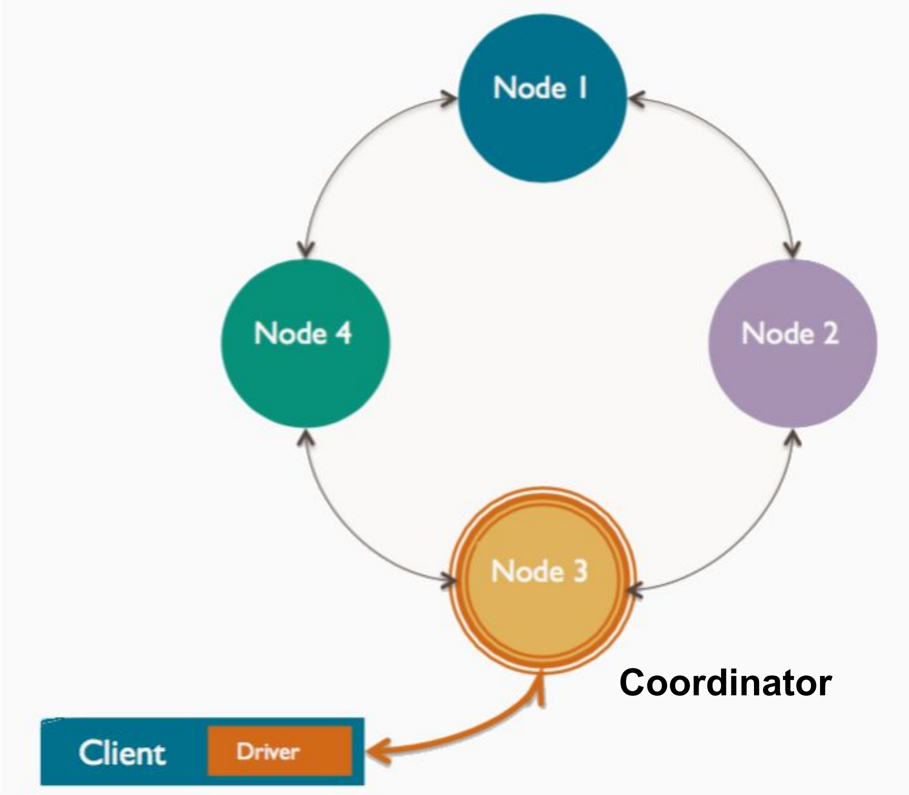
Architecture





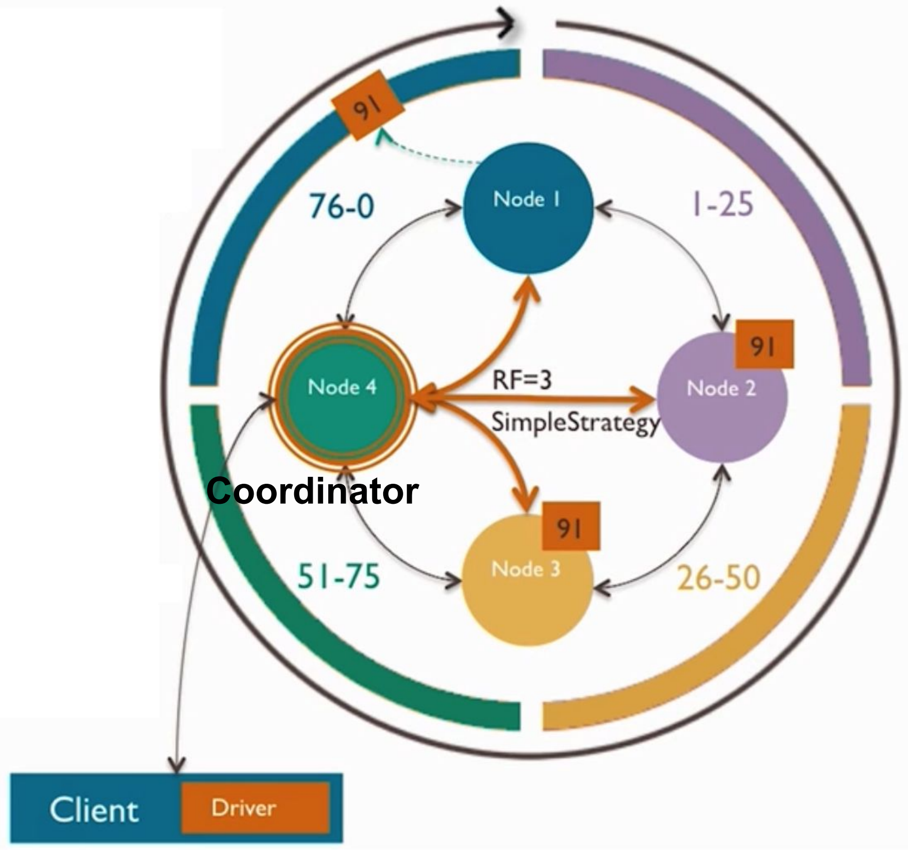
Partition Key, Hashing

- **Token** - integer value generated by a hashing algorithm
- A cluster have -> **2^64** hash keys



Coordinator

- The node chosen by the client to receive a particular read or write request to its cluster
- No single point of failure



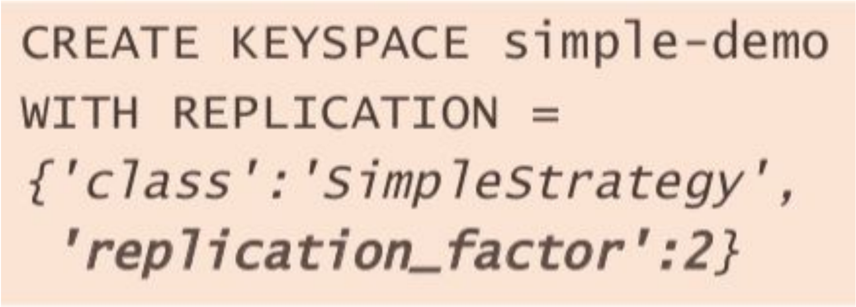
Replication

- **Replication factor (RF) -**
  - onto how many nodes should a write be copied?
  - RF is set for an entire **keyspace**, or for each **data center**, if multiple

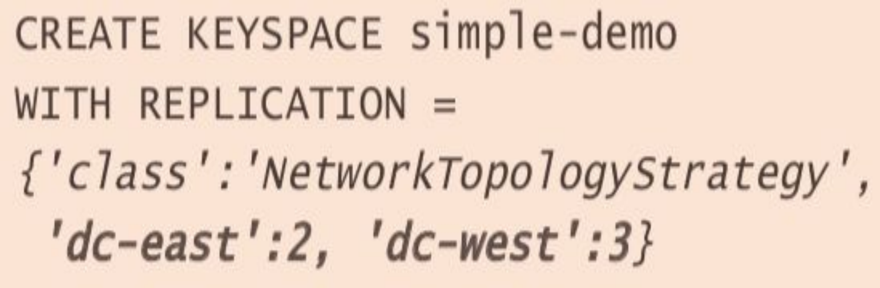
Architecture (Replication)

“**Keyspace**” impact on Replication

- **SimpleStrategy** (learning use only)
- **NetworkTopologyStrategy**

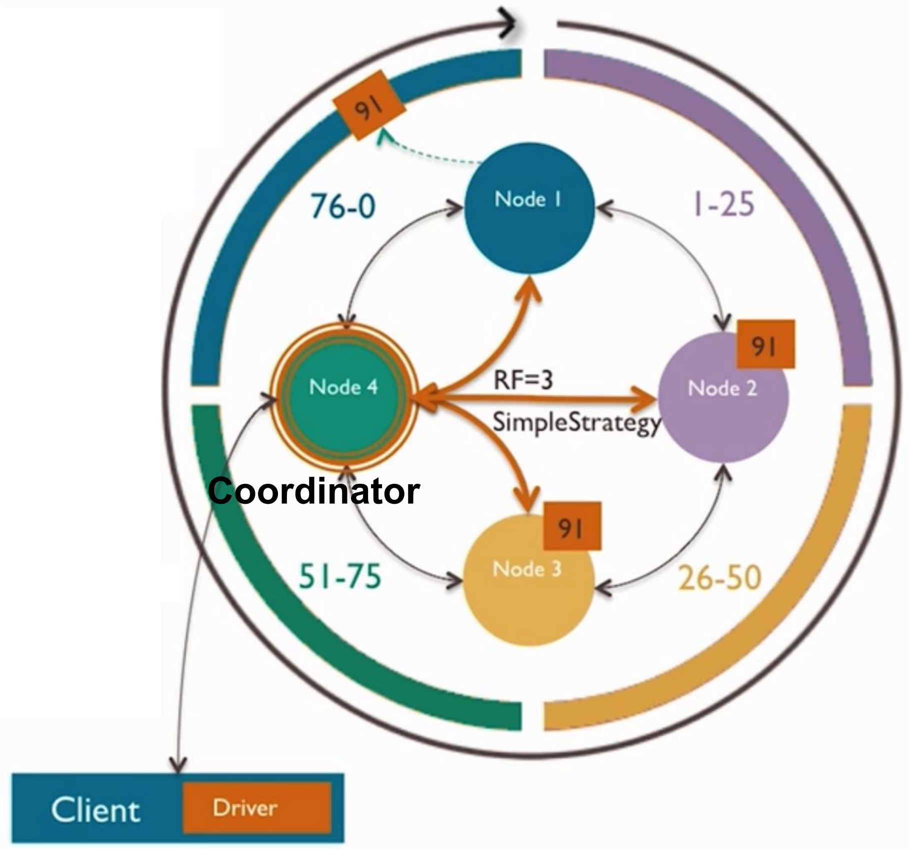


```
CREATE KEYSPACE simple-demo
WITH REPLICATION =
{'class':'SimpleStrategy',
  'replication_factor':2}
```



```
CREATE KEYSPACE simple-demo
WITH REPLICATION =
{'class':'NetworkTopologyStrategy',
  'dc-east':2, 'dc-west':3}
```

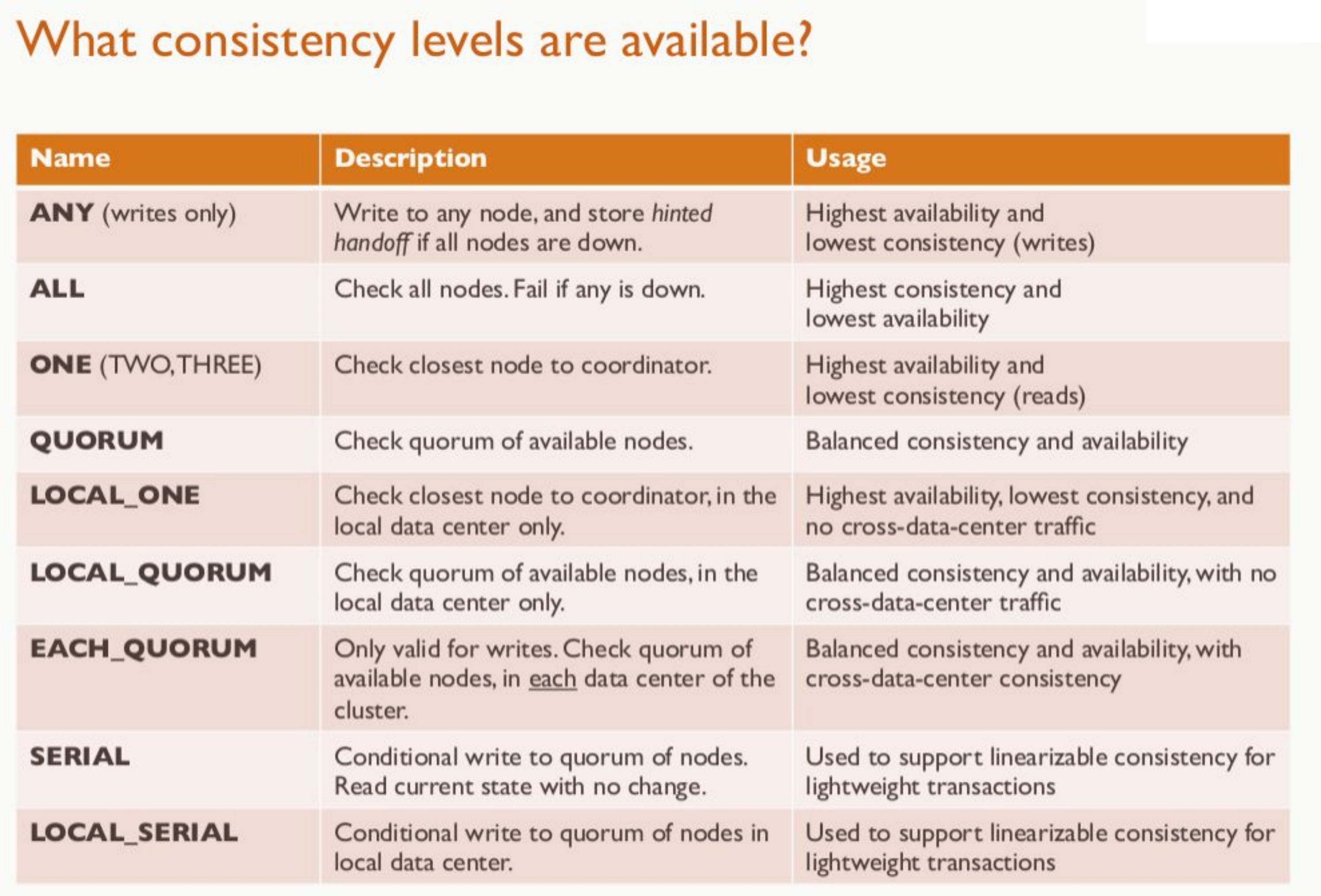
Architecture



Consistency Level

- Consistency level **(CL)**- **no. of nodes must acknowledge to the coordinator**
- CL may vary for each request
- Write / Read request
- If Consistency is top priority-
  - (Nodes written + nodes read) > RF
- **Immediate** Consistency
- **Eventual** Consistency

Architecture (CL)

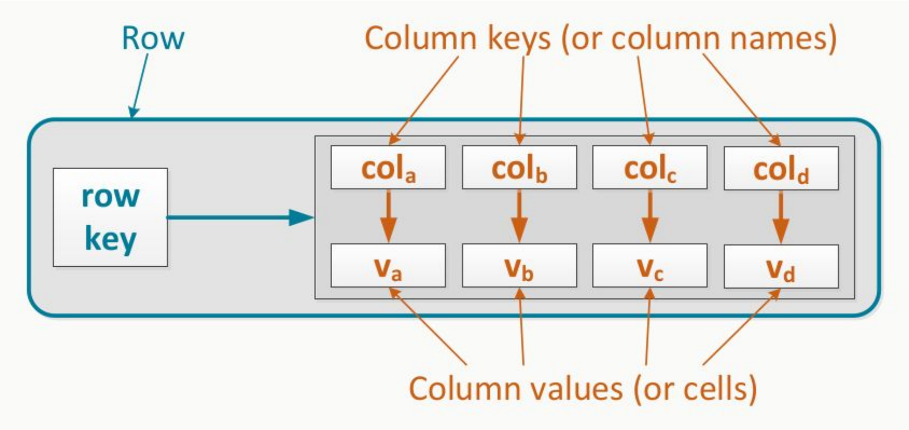


What consistency levels are available?

Name	Description	Usage
ANY (writes only)	Write to any node, and store *hinted handoff* if all nodes are down.	Highest availability and lowest consistency (writes)
ALL	Check all nodes. Fail if any is down.	Highest consistency and lowest availability
ONE (TWO,THREE)	Check closest node to coordinator.	Highest availability and lowest consistency (reads)
QUORUM	Check quorum of available nodes.	Balanced consistency and availability
LOCAL_ONE	Check closest node to coordinator, in the local data center only.	Highest availability, lowest consistency, and no cross-data-center traffic
LOCAL_QUORUM	Check quorum of available nodes, in the local data center only.	Balanced consistency and availability, with no cross-data-center traffic
EACH_QUORUM	Only valid for writes. Check quorum of available nodes, in each data center of the cluster.	Balanced consistency and availability, with cross-data-center consistency
SERIAL	Conditional write to quorum of nodes. Read current state with no change.	Used to support linearizable consistency for lightweight transactions
LOCAL_SERIAL	Conditional write to quorum of nodes in local data center.	Used to support linearizable consistency for lightweight transactions

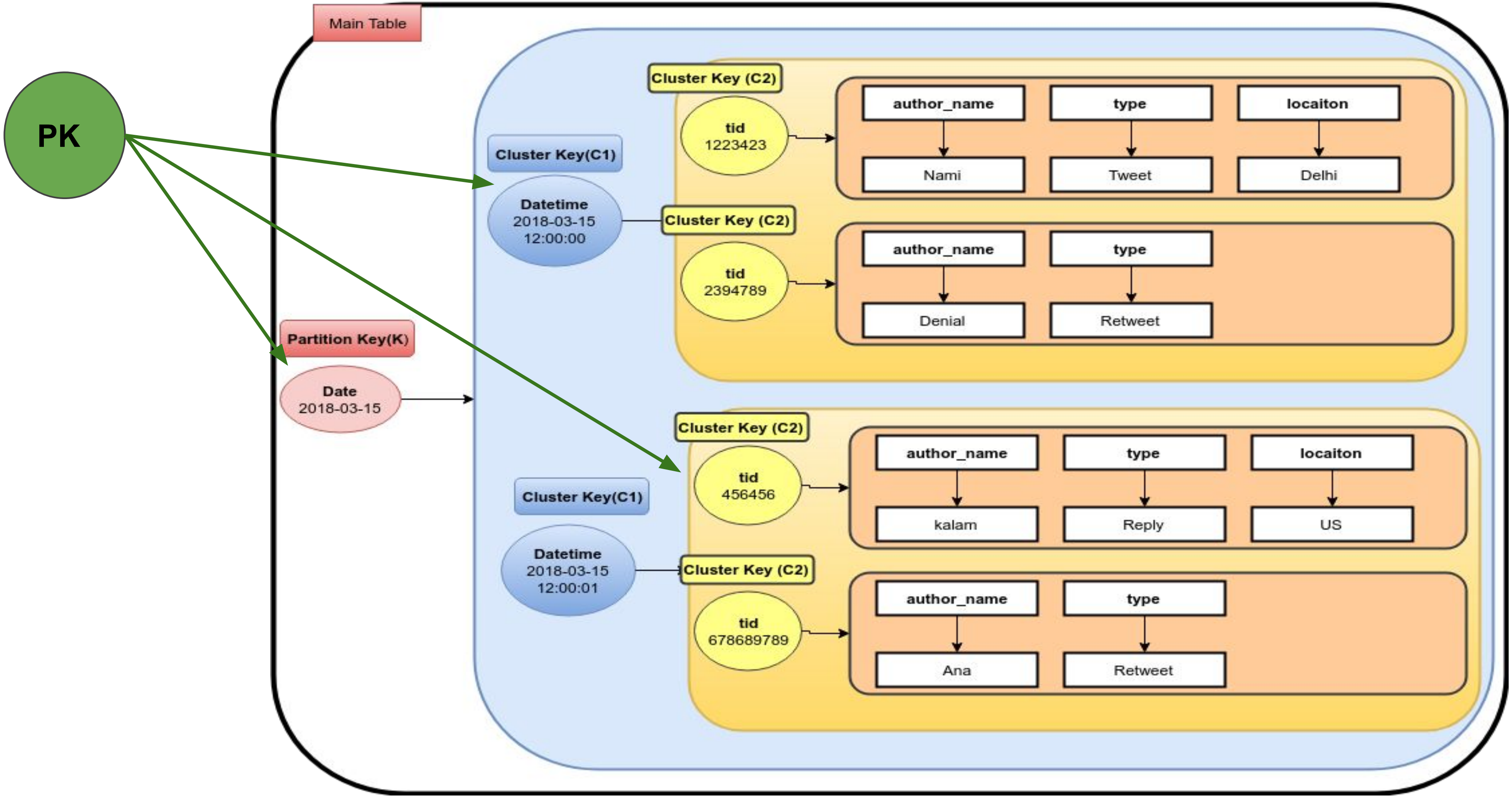
Data Model

Data Model



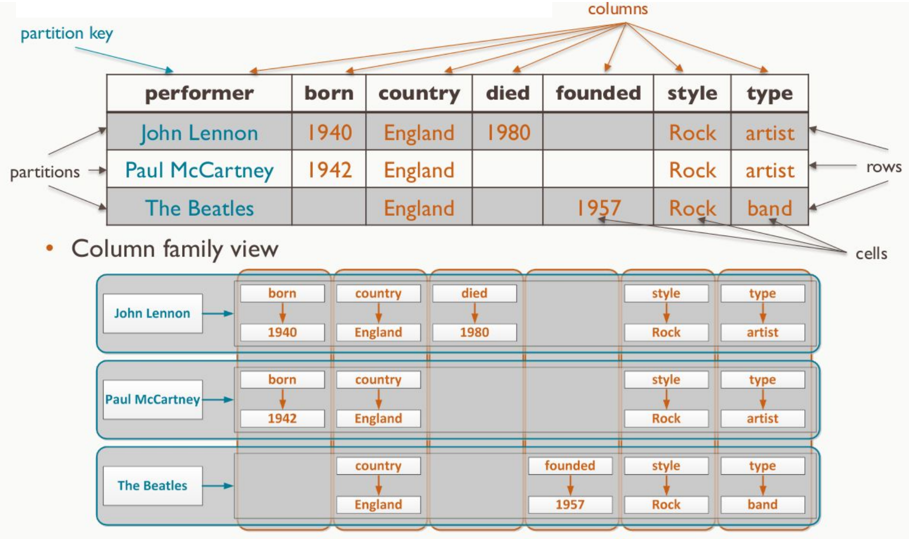
- Row
- Row Key
- Rows
- Column Key
- Column Value

Partition Key(K) & Cluster Key(C)



Data Model

Single Row Partition, No Cluster Key



performer	born	country	died	founded	style	type
John Lennon	1940	England	1980		Rock	artist
Paul McCartney	1942	England			Rock	artist
The Beatles		England		1957	Rock	band

- Column family view

- Primary Key (PK) = (Partition Key(K) + Cluster Key(C))
- Single K , no C
- Single K, single C
- Composite K, no C
- Composite K , single C

Data Model Principles

Data Model Principles

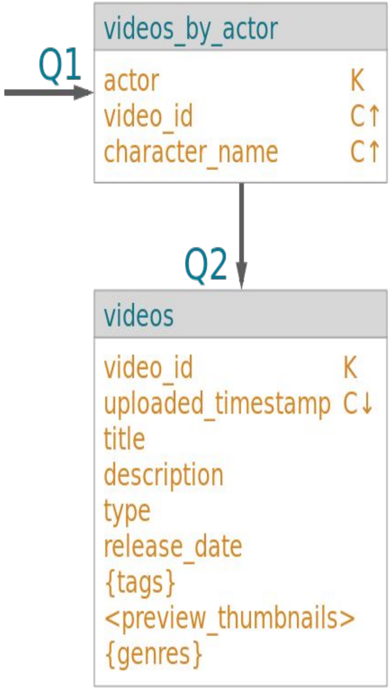
The goal of Cassandra is **lightning fast Queries**.

Cassandra Data Modeling Principles

- Know your Data
- Know your queries
- Nested Data
- Duplicate Data

Data Model Principles

Query : Give me video details of a particular Actor?



Q1

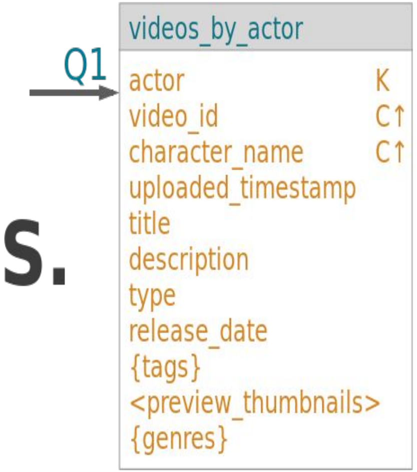
videos_by_actor	
actor	K
video_id	C↑
character_name	C↑

Q2

videos	
video_id	K
uploaded_timestamp	C↓
title	
description	
type	
release_date	
{tags}	
<preview_thumbnails>	
{genres}	

Data Model 1 - Acceptable

VS.



Q1

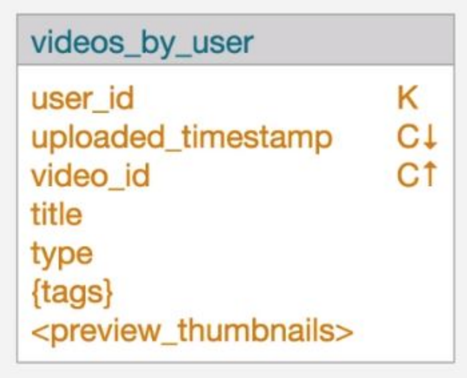
videos_by_actor	
actor	K
video_id	C↑
character_name	C↑
uploaded_timestamp	
title	
description	
type	
release_date	
{tags}	
<preview_thumbnails>	
{genres}	

Data Model 2 - Ideal

- Know your queries
  - Partition per query - ideal
  - Partitions per query - acceptable
  - Table scan - unacceptable
  - Multi-table - unacceptable

Data Model Principles

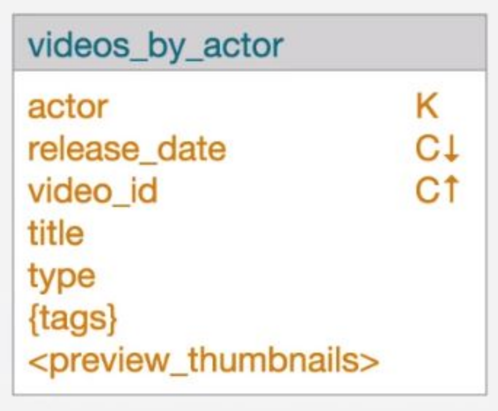
Query：Find videos uploaded by user with a known id (show most recently uploaded videos first)?



videos_by_user	
user_id	K
uploaded_timestamp	C↓
video_id	C↑
title	
type	
{tags}	
<preview_thumbnails>	

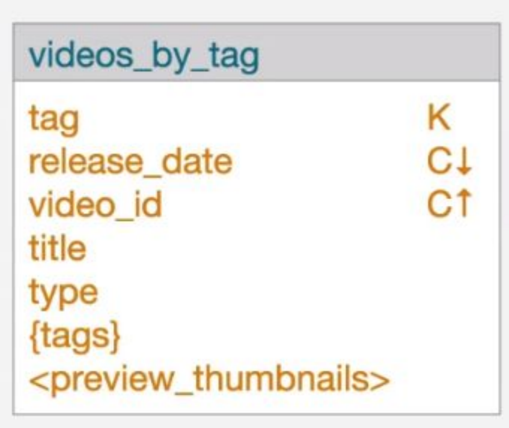
- Nested Data
  - Clustering columns - multiple rows

Data Model Principles



videos_by_actor	
actor	K
release_date	C↓
video_id	C↑
title	
type	
{tags}	
<preview_thumbnails>	

Query：Find videos by actor name (show most recent videos first)?



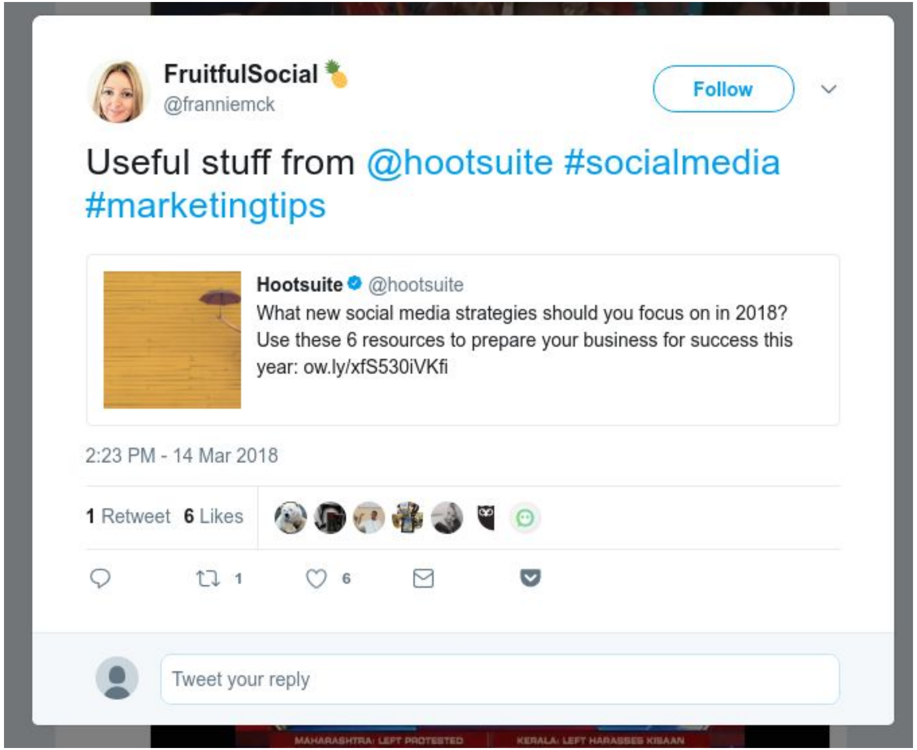
videos_by_tag	
tag	K
release_date	C↓
video_id	C↑
title	
type	
{tags}	
<preview_thumbnails>	

Query：Find videos by tag (show most recent videos first)?

- Duplicate Data
  - (denormalize)
  - (no joining)
  - (super fast response)

Dataset - Twitter

Twitter Dataset



FruitfulSocial
@franniemck

Useful stuff from @hootsuite #socialmedia #marketingtips

Hootsuite @hootsuite
What new social media strategies should you focus on in 2018? Use these 6 resources to prepare your business for success this year: ow.ly/xfS530iVKfi

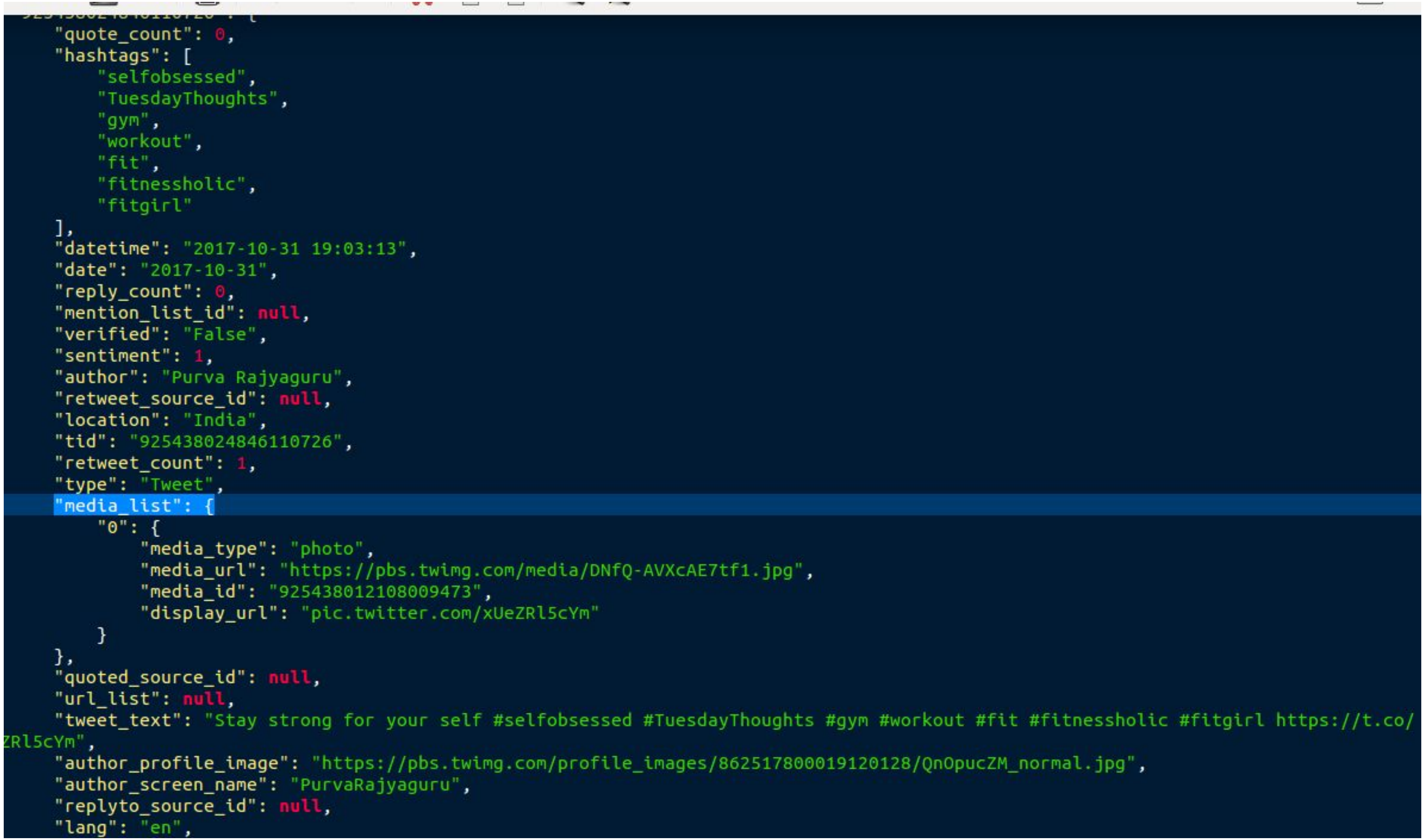
2:23 PM - 14 Mar 2018

1 Retweet 6 Likes

1 6

- Tweet
    - Tweet
    - Reply Tweet
    - Retweet Tweet
    - Quote Tweet
- User
    - Actual Name
    - ScreenName (@handle)
- @mention
- #hashtag

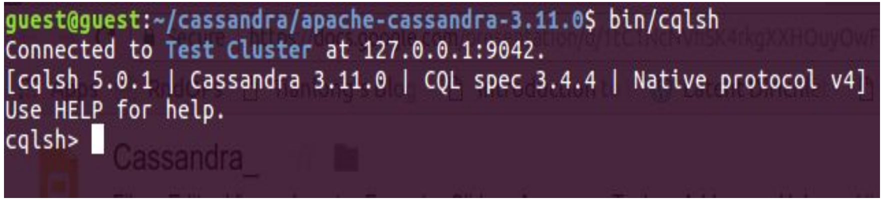
Twitter Dataset



```
    "quote_count": 0,
    "hashtags": [
        "selfobsessed",
        "TuesdayThoughts",
        "gym",
        "workout",
        "fit",
        "fitnessholic",
        "fitgirl"
    ],
    "datetime": "2017-10-31 19:03:13",
    "date": "2017-10-31",
    "reply_count": 0,
    "mention_list_id": null,
    "verified": "False",
    "sentiment": 1,
    "author": "Purva Rajyaguru",
    "retweet_source_id": null,
    "location": "India",
    "tid": "925438024846110726",
    "retweet_count": 1,
    "type": "Tweet",
    "media_list": {
        "0": {
            "media_type": "photo",
            "media_url": "https://pbs.twimg.com/media/DNfQ-AVXcAE7tf1.jpg",
            "media_id": "925438012108009473",
            "display_url": "pic.twitter.com/xUeZRl5cYm"
        }
    },
    "quoted_source_id": null,
    "url_list": null,
    "tweet_text": "Stay strong for your self #selfobsessed #TuesdayThoughts #gym #workout #fit #fitnessholic #fitgirl https://t.co/
ZRl5cYm",
    "author_profile_image": "https://pbs.twimg.com/profile_images/862517800019120128/QnOpucZM_normal.jpg",
    "author_screen_name": "PurvaRajyaguru",
    "replyto_source_id": null,
    "lang": "en",
```

CQL - Syntax

CQL Syntax

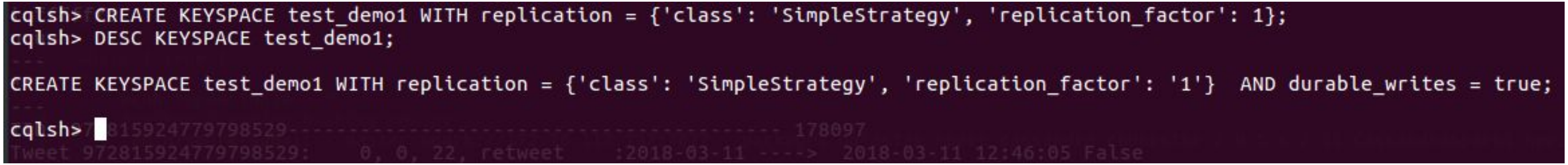


```
guest@guest:~/cassandra/apache-cassandra-3.11.0$ bin/cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.0 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> 
```

- Using “Cqlsh” - Connected to Cluster

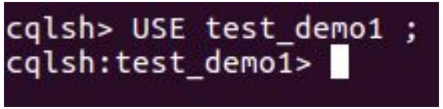
KEYSPACE

- ## To Create Keyspace



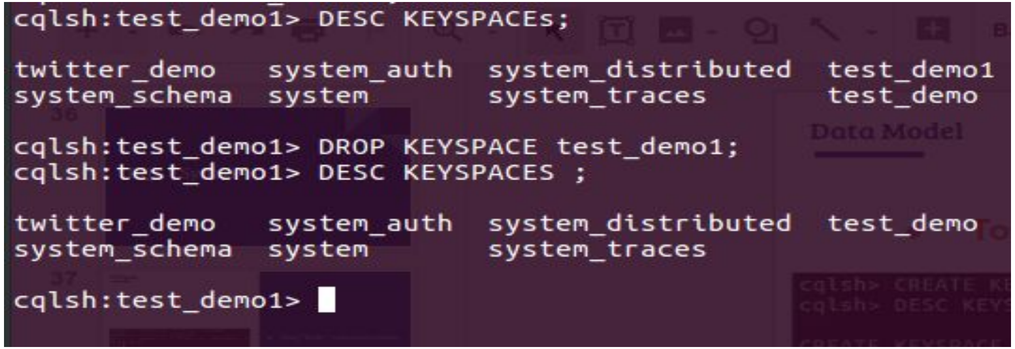
```
cqlsh> CREATE KEYSPACE test_demo1 WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
cqlsh> DESC KEYSPACE test_demo1;

CREATE KEYSPACE test_demo1 WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'}  AND durable_writes = true;

cqlsh> 
```

- ## To Use Keyspace



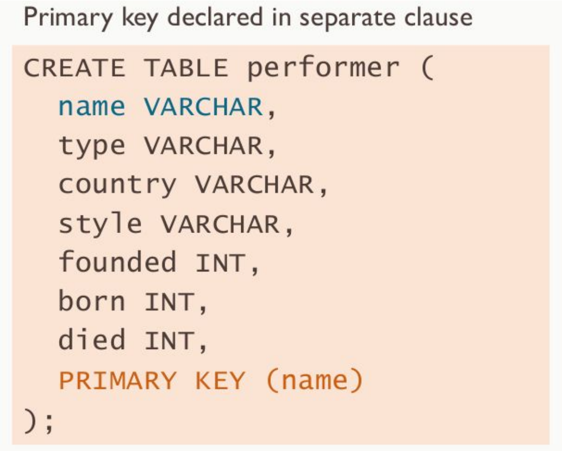
```
cqlsh> USE test_demo1 ;
cqlsh:test_demo1> 
```

- ## To Delete Keyspace



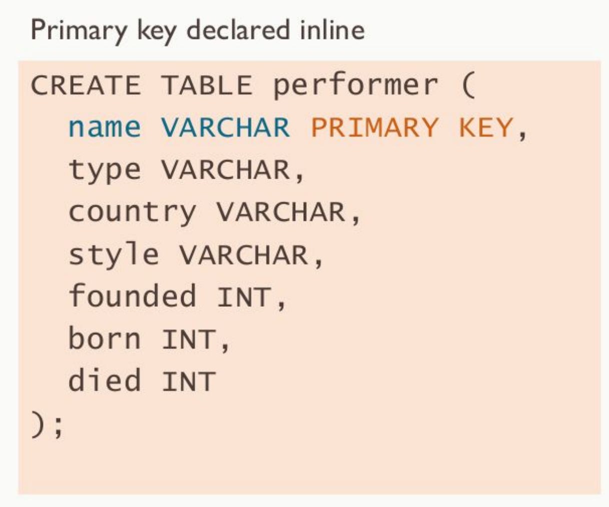
```
cqlsh:test_demo1> DESC KEYSPACEs;

twitter_demo   system_auth  system_distributed  test_demo1
system_schema  system       system_traces       test_demo

cqlsh:test_demo1> DROP KEYSPACE test_demo1;
cqlsh:test_demo1> DESC KEYSPACES ;

twitter_demo   system_auth  system_distributed  test_demo
system_schema  system       system_traces

cqlsh:test_demo1> 
```

CQL Syntax



Primary key declared in separate clause

```
CREATE TABLE performer (
  name VARCHAR,
  type VARCHAR,
  country VARCHAR,
  style VARCHAR,
  founded INT,
  born INT,
  died INT,
  PRIMARY KEY (name)
);
```

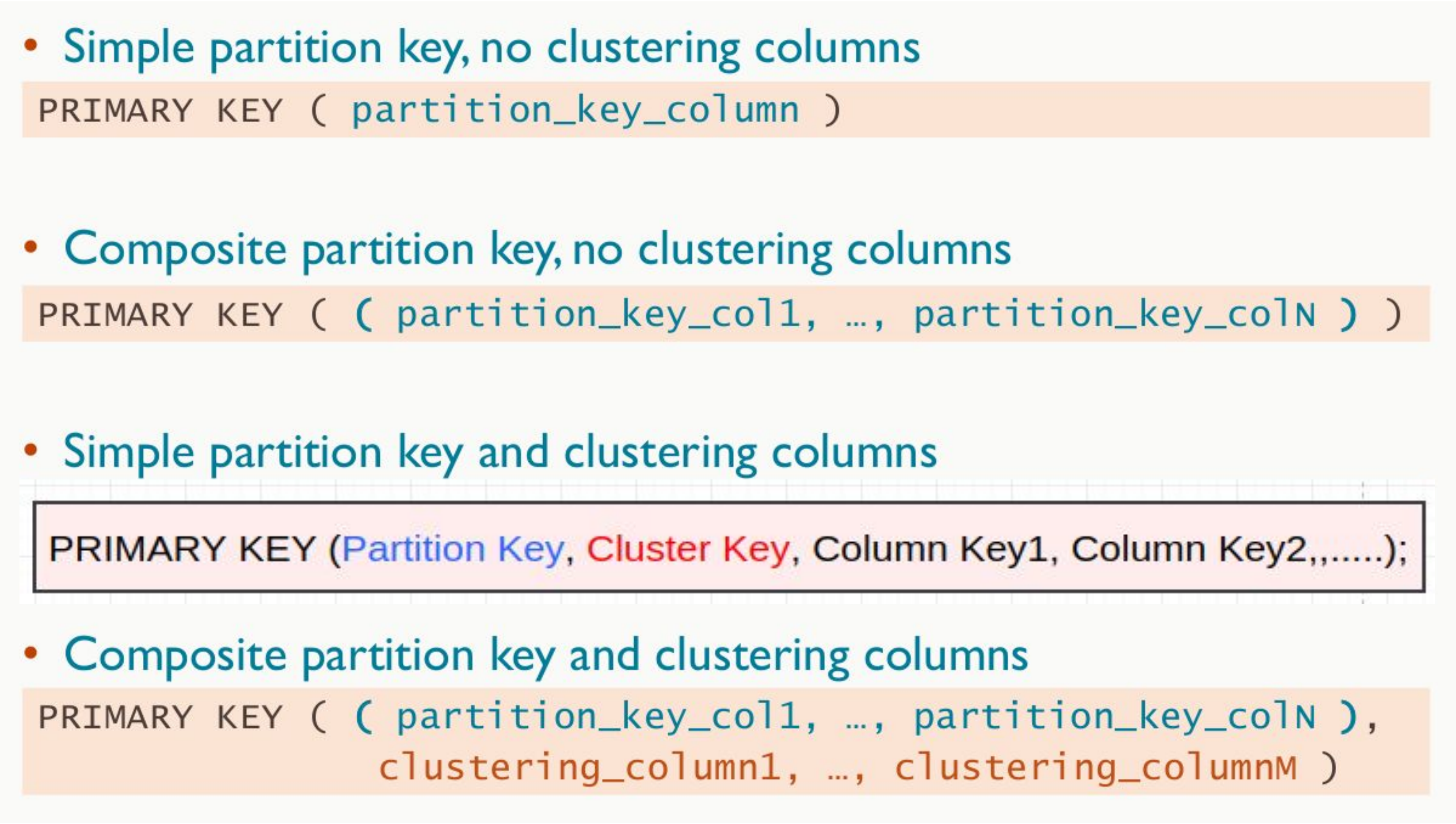


Primary key declared inline

```
CREATE TABLE performer (
  name VARCHAR PRIMARY KEY,
  type VARCHAR,
  country VARCHAR,
  style VARCHAR,
  founded INT,
  born INT,
  died INT
);
```

- CREATE TABLE

Partition Key , CLuster Key



- Simple partition key, no clustering columns

```
PRIMARY KEY ( partition_key_column )
```

- Composite partition key, no clustering columns

```
PRIMARY KEY ( ( partition_key_col1, …, partition_key_colN ) )
```

- Simple partition key and clustering columns

```
PRIMARY KEY (Partition Key, Cluster Key, Column Key1, Column Key2,,.....);
```

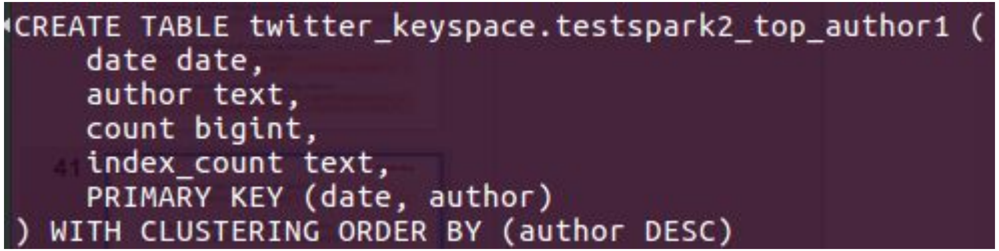
- Composite partition key and clustering columns

```
PRIMARY KEY ( ( partition_key_col1, …, partition_key_colN ),
               clustering_column1, …, clustering_columnM )
```

Data Model

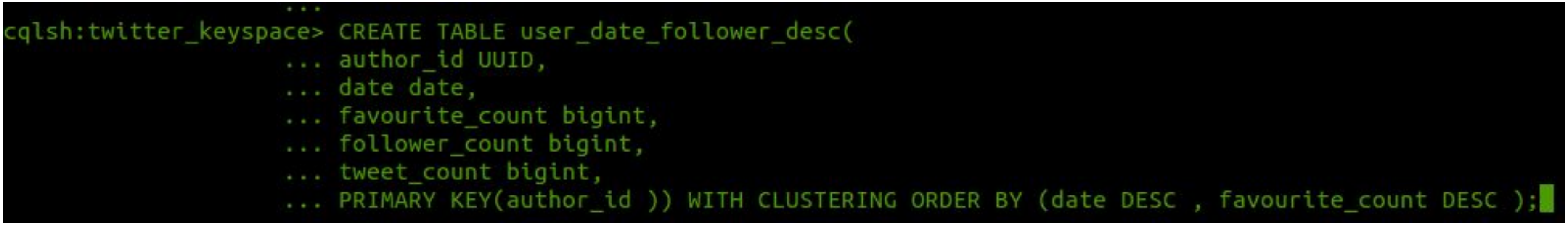
**Simple Partition Key ,
Simple Cluster Key**

Partition Key , CLuster Key



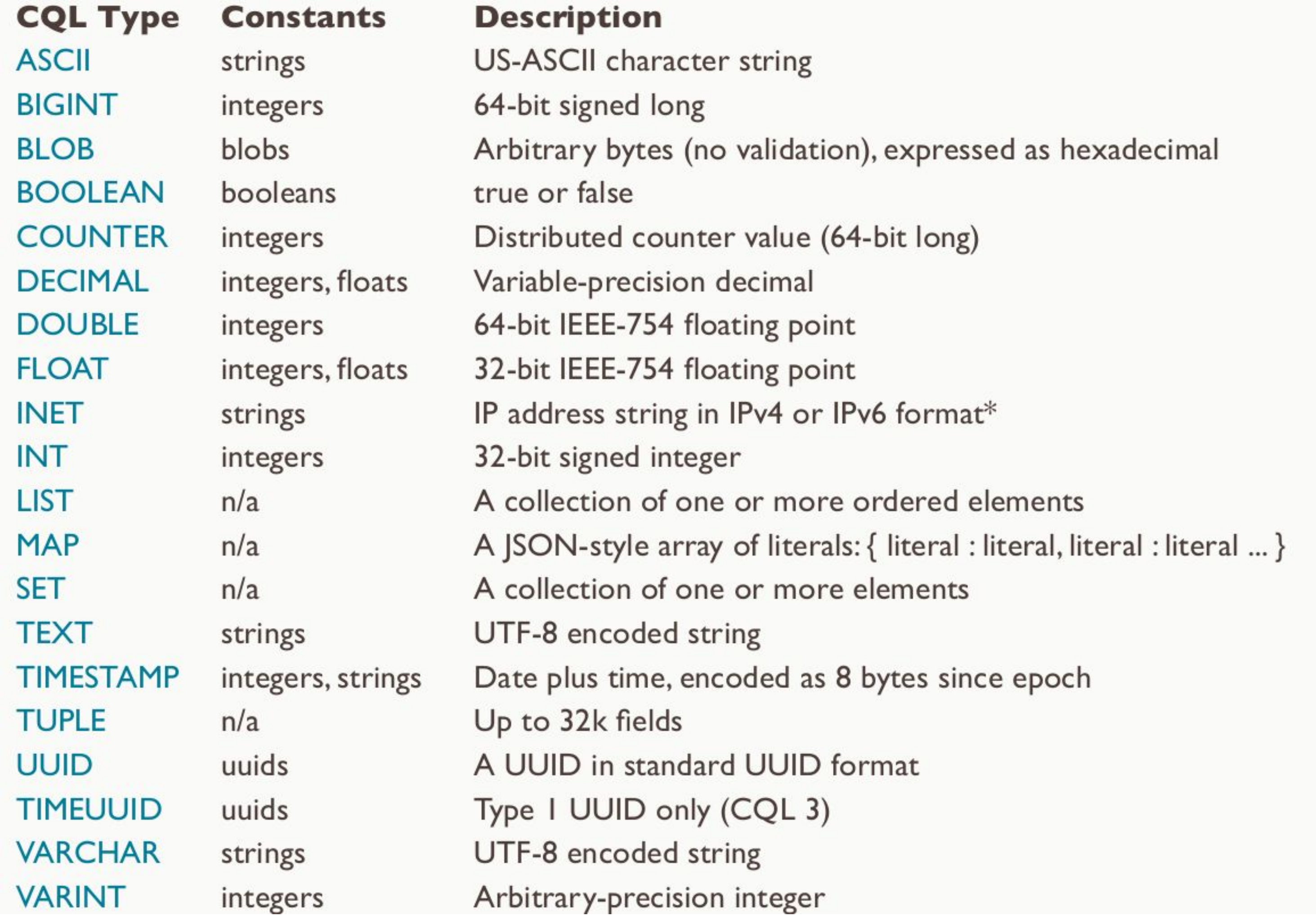
```
CREATE TABLE twitter_keyspace.testspark2_top_author1 (
    date date,
    author text,
    count bigint,
    index_count text,
    PRIMARY KEY (date, author)
) WITH CLUSTERING ORDER BY (author DESC)
```

**Simple Partition Key ,
Simple Cluster Key**



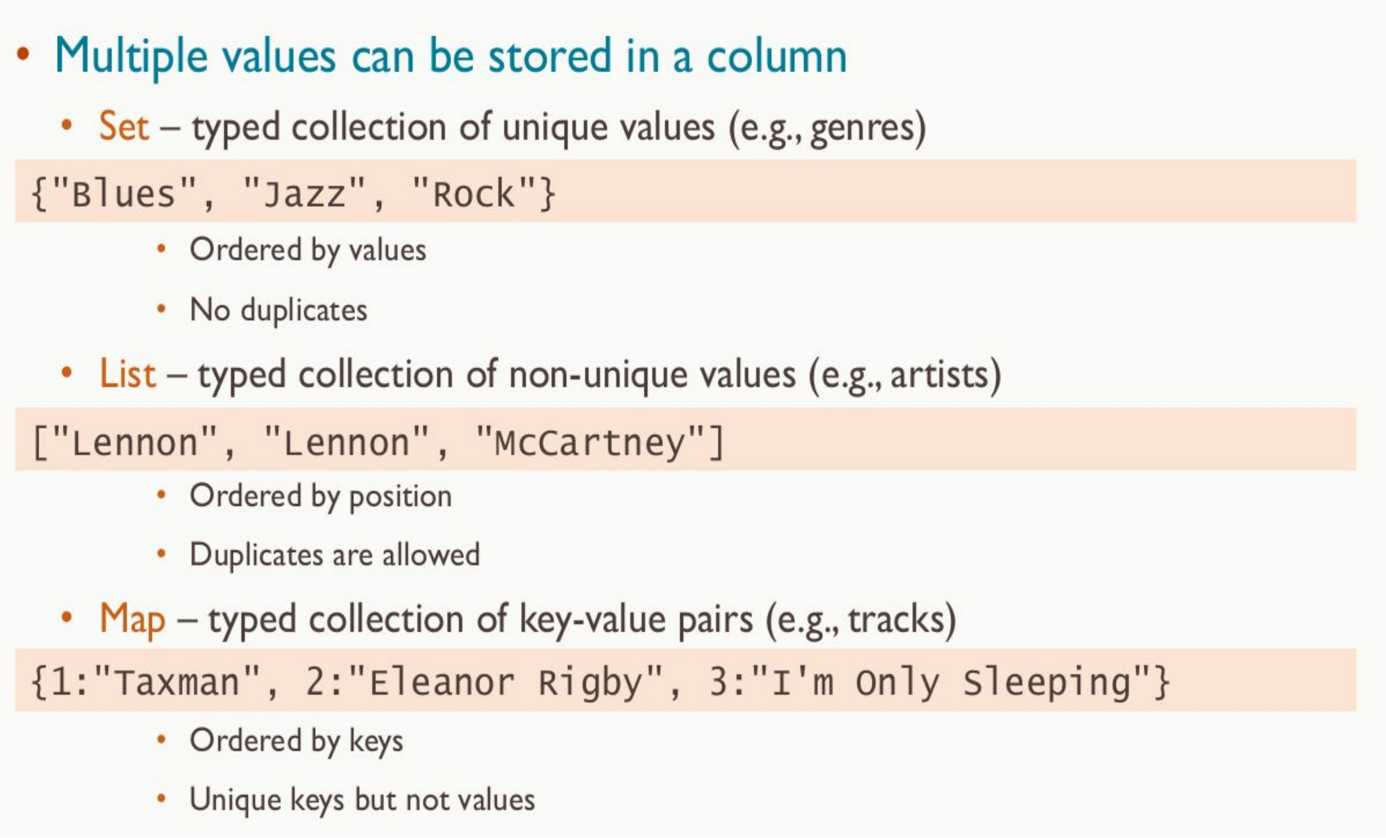
```
                    ...
cqlsh:twitter_keyspace> CREATE TABLE user_date_follower_desc(
                    ... author_id UUID,
                    ... date date,
                    ... favourite_count bigint,
                    ... follower_count bigint,
                    ... tweet_count bigint,
                    ... PRIMARY KEY(author_id )) WITH CLUSTERING ORDER BY (date DESC , favourite_count DESC );
```

Cassandra Data Type



CQL Type	Constants	Description
ASCII	strings	US-ASCII character string
BIGINT	integers	64-bit signed long
BLOB	blobs	Arbitrary bytes (no validation), expressed as hexadecimal
BOOLEAN	booleans	true or false
COUNTER	integers	Distributed counter value (64-bit long)
DECIMAL	integers, floats	Variable-precision decimal
DOUBLE	integers	64-bit IEEE-754 floating point
FLOAT	integers, floats	32-bit IEEE-754 floating point
INET	strings	IP address string in IPv4 or IPv6 format*
INT	integers	32-bit signed integer
LIST	n/a	A collection of one or more ordered elements
MAP	n/a	A JSON-style array of literals: { literal : literal, literal : literal ... }
SET	n/a	A collection of one or more elements
TEXT	strings	UTF-8 encoded string
TIMESTAMP	integers, strings	Date plus time, encoded as 8 bytes since epoch
TUPLE	n/a	Up to 32k fields
UUID	uuids	A UUID in standard UUID format
TIMEUUID	uuids	Type 1 UUID only (CQL 3)
VARCHAR	strings	UTF-8 encoded string
VARINT	integers	Arbitrary-precision integer

Collection Data types



- Multiple values can be stored in a column
  - Set – typed collection of unique values (e.g., genres)

```
{"Blues", "Jazz", "Rock"}
```

    - Ordered by values
    - No duplicates
  - List – typed collection of non-unique values (e.g., artists)

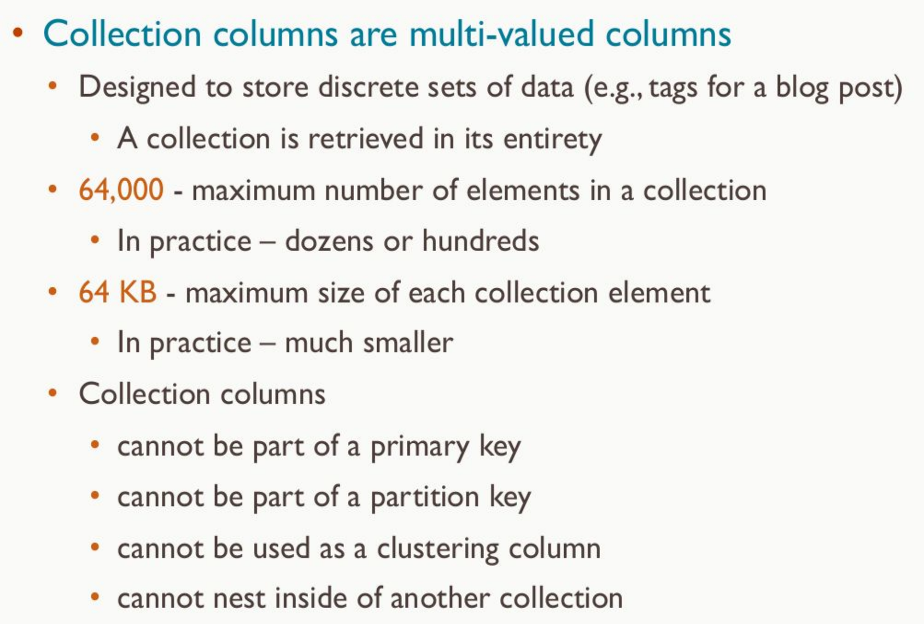
```
["Lennon", "Lennon", "McCartney"]
```

    - Ordered by position
    - Duplicates are allowed
  - Map – typed collection of key-value pairs (e.g., tracks)

```
{1:"Taxman", 2:"Eleanor Rigby", 3:"I'm Only Sleeping"}
```

    - Ordered by keys
    - Unique keys but not values

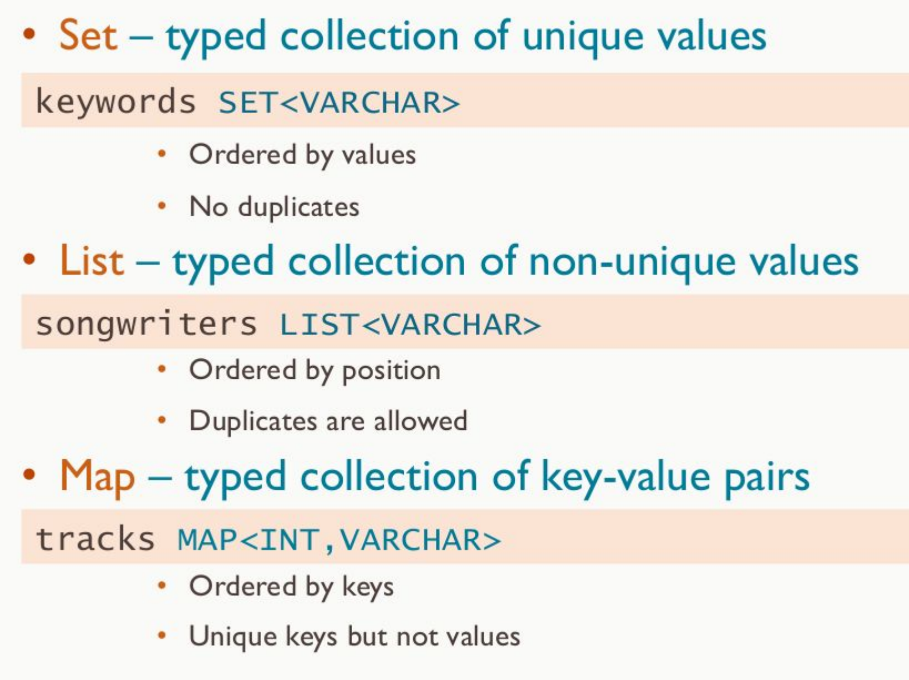
CQL Syntax



- Collection columns are multi-valued columns
  - Designed to store discrete sets of data (e.g., tags for a blog post)
    - A collection is retrieved in its entirety
  - 64,000 - maximum number of elements in a collection
    - In practice – dozens or hundreds
  - 64 KB - maximum size of each collection element
    - In practice – much smaller
  - Collection columns
    - cannot be part of a primary key
    - cannot be part of a partition key
    - cannot be used as a clustering column
    - cannot nest inside of another collection

- Collection Type Column

CQL Syntax



- Set – typed collection of unique values

```
keywords SET<VARCHAR>
```

  - Ordered by values
  - No duplicates

- List – typed collection of non-unique values

```
songwriters LIST<VARCHAR>
```

  - Ordered by position
  - Duplicates are allowed

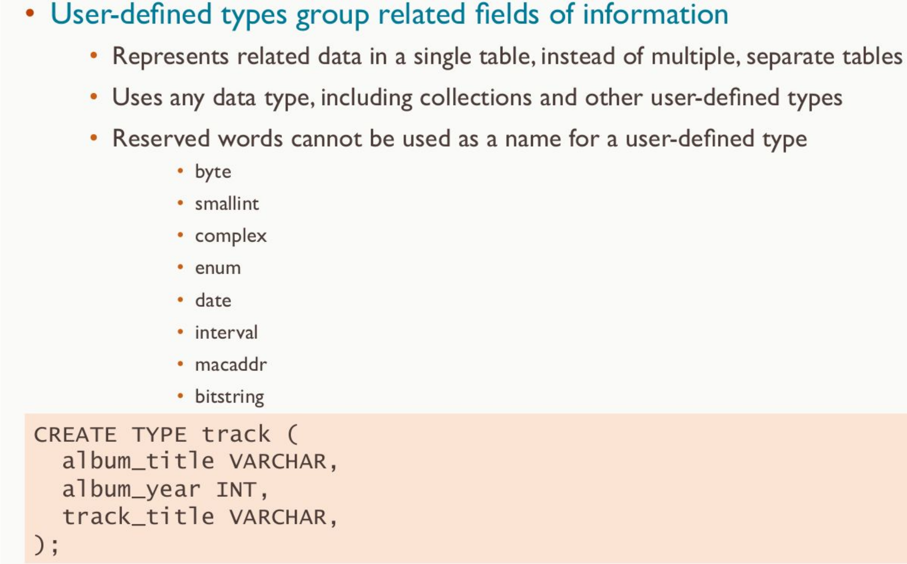
- Map – typed collection of key-value pairs

```
tracks MAP<INT,VARCHAR>
```

  - Ordered by keys
  - Unique keys but not values

- Collection Column (Syntax)

CQL Syntax

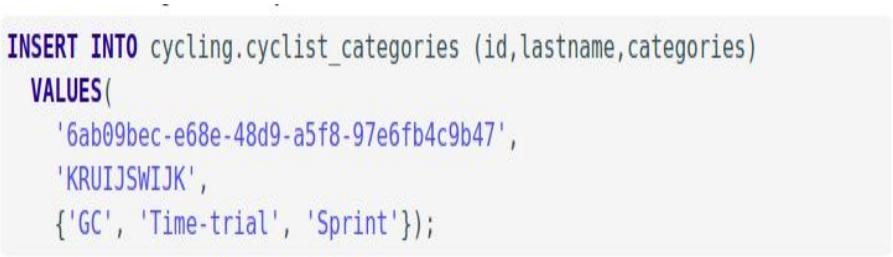


- User-defined types group related fields of information
  - Represents related data in a single table, instead of multiple, separate tables
  - Uses any data type, including collections and other user-defined types
  - Reserved words cannot be used as a name for a user-defined type
    - byte
    - smallint
    - complex
    - enum
    - date
    - interval
    - macaddr
    - bitstring

```
CREATE TYPE track (
  album_title VARCHAR,
  album_year INT,
  track_title VARCHAR,
);
```

- UDT

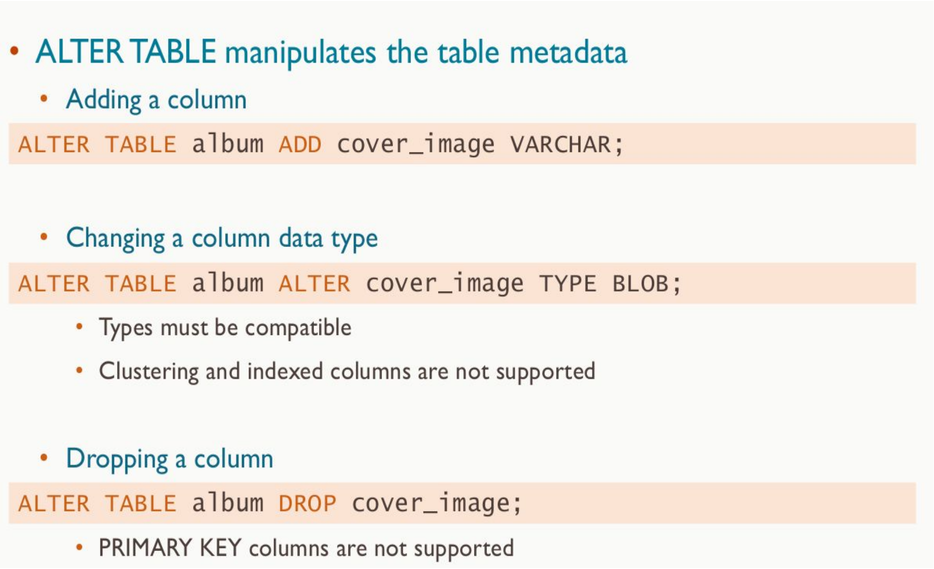
CQL Syntax



```
INSERT INTO cycling.cyclist_categories (id,lastname,categories)
  VALUES(
    '6ab09bec-e68e-48d9-a5f8-97e6fb4c9b47',
    'KRUIJSWIJK',
    {'GC', 'Time-trial', 'Sprint'});
```

- INSERT

CQL Syntax



- ALTER TABLE manipulates the table metadata
  - Adding a column

```
ALTER TABLE album ADD cover_image VARCHAR;
```

  - Changing a column data type

```
ALTER TABLE album ALTER cover_image TYPE BLOB;
```

    - Types must be compatible
    - Clustering and indexed columns are not supported

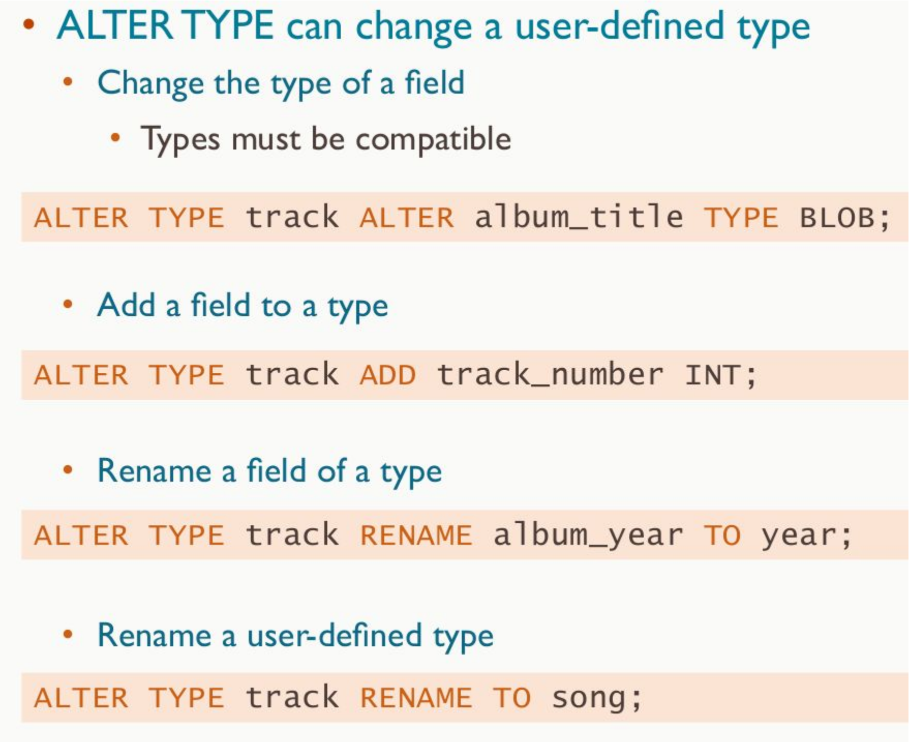
  - Dropping a column

```
ALTER TABLE album DROP cover_image;
```

    - PRIMARY KEY columns are not supported

- ALTER TABLE
  - Add a column
  - Column data type
  - Drop a column

CQL Syntax



- ALTER TYPE can change a user-defined type
  - Change the type of a field
    - Types must be compatible

```
ALTER TYPE track ALTER album_title TYPE BLOB;
```

  - Add a field to a type

```
ALTER TYPE track ADD track_number INT;
```

  - Rename a field of a type

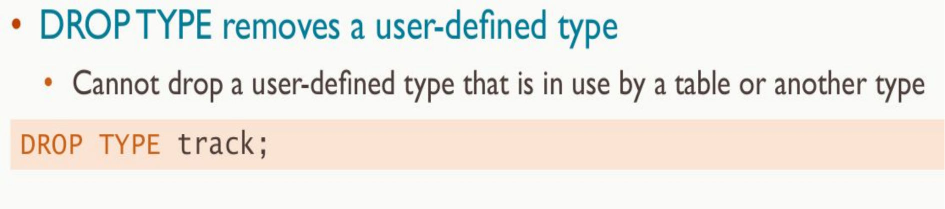
```
ALTER TYPE track RENAME album_year TO year;
```

  - Rename a user-defined type

```
ALTER TYPE track RENAME TO song;
```

- ALTER TYPE
  - To change the type of a field
  - Add a field to a type
  - Rename a field of a type
  - Rename a UDT

CQL Syntax



- DROP TYPE removes a user-defined type
  - Cannot drop a user-defined type that is in use by a table or another type

```
DROP TYPE track;
```

- DROP TYPE

CQL Syntax

To drop the Table including data

DROP “table name”

- DROP TABLE

CQL Syntax

Delete All Data of the Table

TRUNCATE “table name”

- TRUNCATE Table

CQL Syntax

```
SELECT * FROM cyclist_name LIMIT 50000;
```

```
SELECT lastname FROM cyclist_name LIMIT 50000;
```

- SELECT query

CQL Syntax

```
UPDATE cyclist_name
SET comments = ‘Rides hard, gets along with others, a
real winner' WHERE id = 123 IF EXISTS;
```

```
UPDATE cyclists SET firstname = 'Marianne,
 lastname = 'VOS'  WHERE id = 56;
```

- UPDATE query

CQL Syntax

Delete data from row

```
DELETE firstname, lastname FROM cyclist_name
WHERE id = 67;
```

Delete entire row

```
DELETE  FROM cyclist_name WHERE id = 68;
```

- DELETE query

References

References

- DataStax Cassandra Course
  - https://academy.datastax.com/courses
  - https://docs.datastax.com/en/cql/3.3/cql/cql_reference/cqlUpdate.html
  - https://docs.datastax.com/en/cql/3.3/cql/ddl/ddlCQLDataModelingTOC.html
  - https://docs.datastax.com/en/cql/3.3/cql/cqlIntro.html

- CAP theorem -
  - https://pandaforme.gitbooks.io/introduction-to-cassandra/content/cap_theorem.html

- PHP Driver -
  - https://docs.datastax.com/en/developer/php-driver/1.3/
- Python Driver -
  - https://docs.datastax.com/en/developer/python-driver/3.13/