



Apache Cassandra

CSCE 5300 - Introduction to Big Data and Data Science

Overview

- Cassandra Introduction
- Strengths
- Architecture
- Data Types
- Commands

What is Cassandra?

- Cassandra is the choice when developers need a **scalability** and **high availability** for the end user(s)
- **Features:**
 - Best performance
 - Fault tolerant
 - No Single Point of Failure
 - Data replication and can be clustered into multiple data centers.

Why Cassandra?

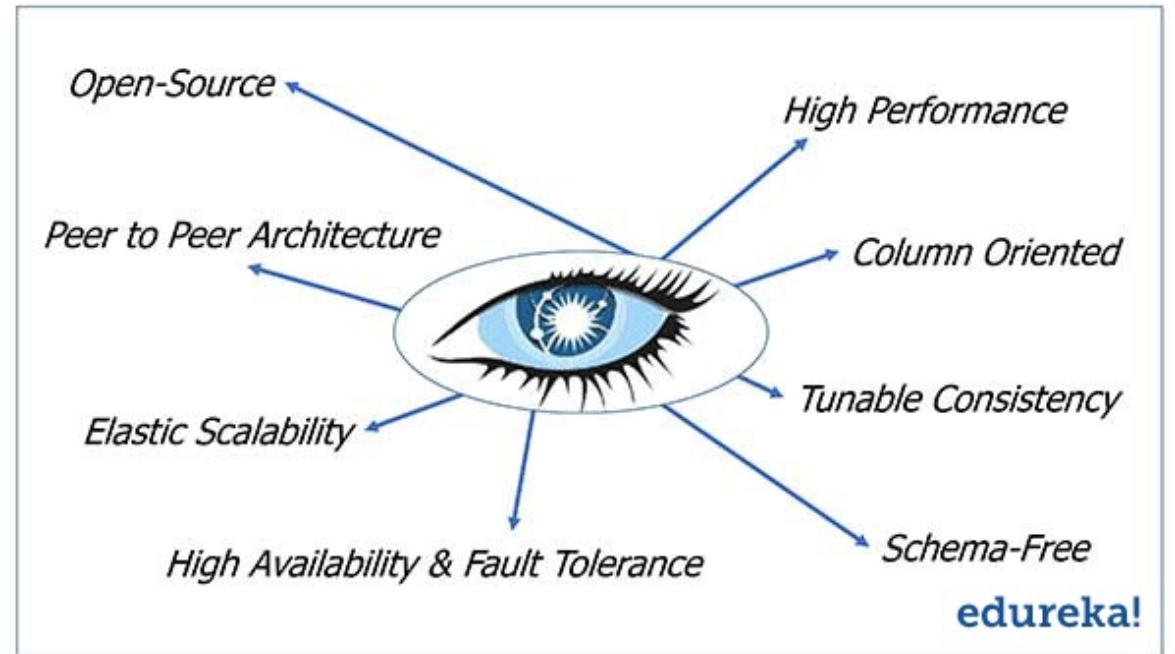
- It is an **open source and developed Database management system** that has the ability to handle very vast data
- The data can be **distributed over servers**; the main importance of the Cassandra is to provide service which has no **failure point**
- Cassandra can operate strongly on multiple data centres which are distributed over the World, along with replication which gives the **Cassandra the ability to operate on low latency servers**
- Cassandra is used in “**NetFlix, eBay, Twitter, Reddit and Many more**”
- While if we talk about the storage and nodes, then Cassandra documentation shows the highest setup of Cassandra is done, that consists of **400 TB of data distributed over 400 nodes**

Cassandra Usage

- One the most common use of Cassandra is in **Facebook inbox Searching**
- Facebook stores approximately **150 Terabytes** of its data on **150 nodes**
- Cassandra is also used in Twitter, Rackspace, CISCO and Many more

Strengths of Cassandra: *Elastic and Scalable*

- The Cassandra is elastic and scalable because of the read and writes operations per second which increases linearly when a new node is added to the channel of Cassandra database's seed machine
- With no downtime or restarting needed for the configuration of the system, whenever node is added to the channel it starts operating straight away without the system to be restarted




Strengths of Cassandra : *Reliable*

- When the developer was designing the Cassandra, in the development stage they expected that hardware may fail due to the unreliability
- Cassandra now has the ability to abide more than one node failure. Whenever a node fails, it does not affect the main Cassandra application and no down time is observed at all.
- These nodes can be replaced once the error has been removed or hardware has been replaced, and bring them back online in functioning state.
- The termination of data that originates across the data centres across terrestrial regions caused by the data replication is supported by the Cassandra database.
- All nodes in the Cassandra channel have same abilities with no node as a top priority node. Cassandra uses its feature known as accrual failure detector which is used to monitor the operation power of the nodes within a cluster.

Strengths of Cassandra : Durable & Without ETL

Durable

- Writing is also known as durability, and once the writing process is completed that written data will remain on disk forever even if the hardware failure occurs
- Same is the case with Cassandra, Cassandra writes records, and it appends those records to commit log initially and then for the flushing of data permanently to a disk it uses “fsync system call”



**ETL (Extract,
Transform and
Load)**

Analytics without ETL

- There is one to one access to the execution of jobs of Hadoop that is feed in Cassandra and with no impact on the working of the applications that runs of the real time

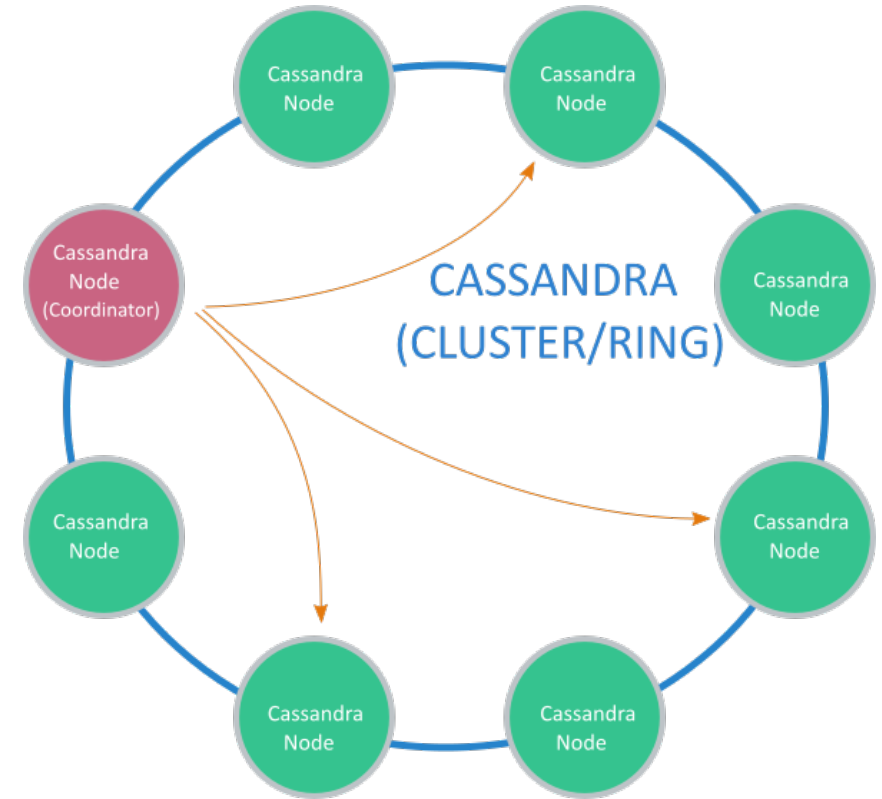
ETL is a process in data warehousing responsible for pulling data out of the source systems and placing it into a data warehouse.

Architecture of Cassandra

- **Main Feature** : Master and slave working of the whole setup of a system of a point to point cluster
 - The system doesn't have any single peer failure
 - The system designed has no master node
 - If there is any failure in the node or if there are a lot of request sent by other peers this would not affect other nodes inside the cluster
 - The setup will continue its working normally as it was working before any point failure
- The Cassandra has *the ability to group* many servers regardless of their figure
- Cassandra has a very usable architecture which has *the ability to perform thousands or millions of operations in a second*
- It has also the capability to handle the *data that is in petabytes*, which is distributed over a huge number of data centers; it can perform operations that are very small

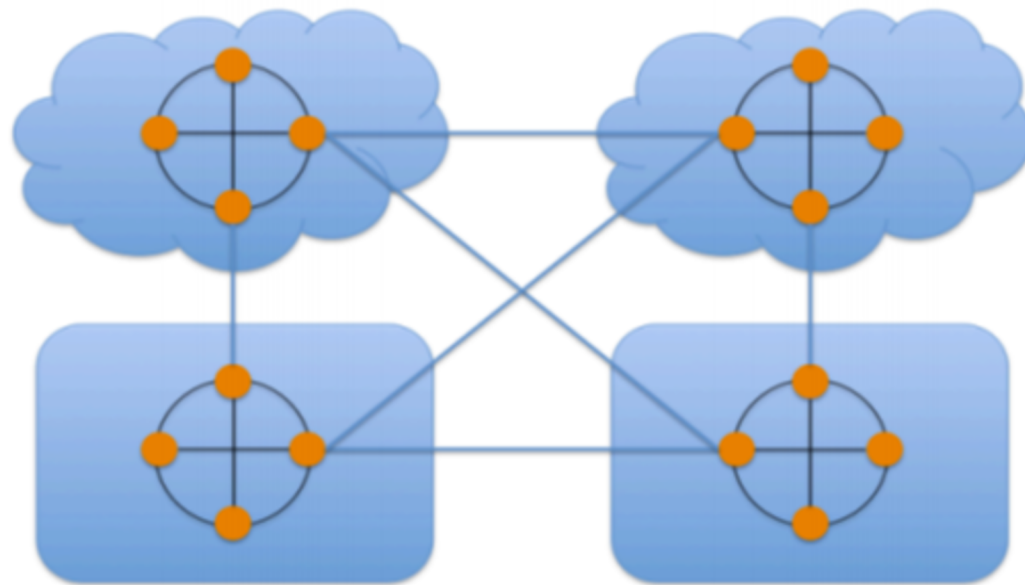
Distribution and Replication of Data

- ***Workable Database Management System:*** The ability to provide the distribution of the data to the number of machines connected to the cluster or we can say in other words a ring in automatic format.
- Distribution of Data is Automatic.
- Distribution is in an ordered or random manner, but initially when Cassandra is installed and no settings are done, the data is distributed randomly over the machines connected to the cluster.



Multi Data Centre and Cloud Support

- Cassandra is known as one of the most prominent NoSQL database, when someone talks about the duplication of the data across different data centres or rack spaces.
- For administrators or developers, it is very easy to handle and make only one Cassandra cluster, add as many nodes as they want. It can be configured from only one place rather than going to each and every node.
- When data is provided, it automatically distributes the data across several data centres or hybrid cloud fashion.



- When a developer intends to create a new Cassandra database in the cluster, he only has to use a simple command in the command line of Cassandra which is also known as creating a key space
- During the creation of a key space in command line, the developer can tell how many and which nodes to be considered and how much should be the replication
- After when he enters the command, it's the Cassandra responsibility to control and distribute the data functionality and replication over the nodes that are in the cluster in automatic format

Reading and Writing Data

- Cassandra provides architecture to the system if we talk about reading and writing of the information to the Cassandra cluster's nodes, "which is independent on location"
- It means that no one is aware of where the data has been written. This tells us that if there is any machine in the Cassandra ring, it will have the ability of reading and writing no matter if it's in single cluster setup or multi cluster setup
- This ability of Cassandra shows a true "read/write-anywhere design"

Data Consistency

- Data consistency is the main feature of Cassandra which provides a good command over the flexibility of Cassandra. As discussed that Cassandra provides “tunable” data consistency for the Cassandra cluster, so the main flexibility is that a developer is a commander to decide what type of data consistency is required
- This type of consistency is supported in the Cassandra’s clusters if they are in single or multiples data centres. The developer is now free to choose among the options of the data consistency which one is appropriate and which operation he wants to perform
- Consistency can also be handled on the basis of a single operation, which means that a developer decides which operation should be performed on all nodes at one time. The operations may be INSERT, UPDATE, DELETE and SELECT

TYPE	CONSTANTS SUPPORTED	DESCRIPTION
Int	integer	32-bit signed int
Smallint	integer	16-bit signed int
Text	string	UTF8 encoded string
Time	integer , string	A time (with no corresponding date value) with nanosecond precision. See Working with times below for details
Timestamp	integer , string	A timestamp (date and time) with millisecond precision. See Working with timestamps below for details
Timeuuid	uuid	Version 1 UUID , generally used as a “conflict-free” timestamp. Also see Timeuuid functions
Tinyint	integer	8-bit signed int
Uuid	uuid	A UUID (of any version)
Varchar	string	UTF8 encoded string
Varint	integer	Arbitrary-precision integer

TYPE	CONSTANTS SUPPORTED	DESCRIPTION
Ascii	string	ASCII character string
Bigint	integer	64-bit signed long
Blob	blob	Arbitrary bytes (no validation)
Boolean	boolean	Either true or false
Counter	integer	Counter column (64-bit signed value). See Counters for details
Date	integer , string	A date (with no corresponding time value). See Working with dates below for details
Decimal	integer , float	Variable-precision decimal
Double	integer float	64-bit IEEE-754 floating point
Duration	duration,	A duration with nanosecond precision. See Working with durations below for details
Float	integer , float	32-bit IEEE-754 floating point
Inet	string	An IP address, either IPv4 (4 bytes long) or IPv6 (16 bytes long). Note that there is no inet constant, IP address should be input as strings

SimpleStrategy vs NetworkTopologyStrategy

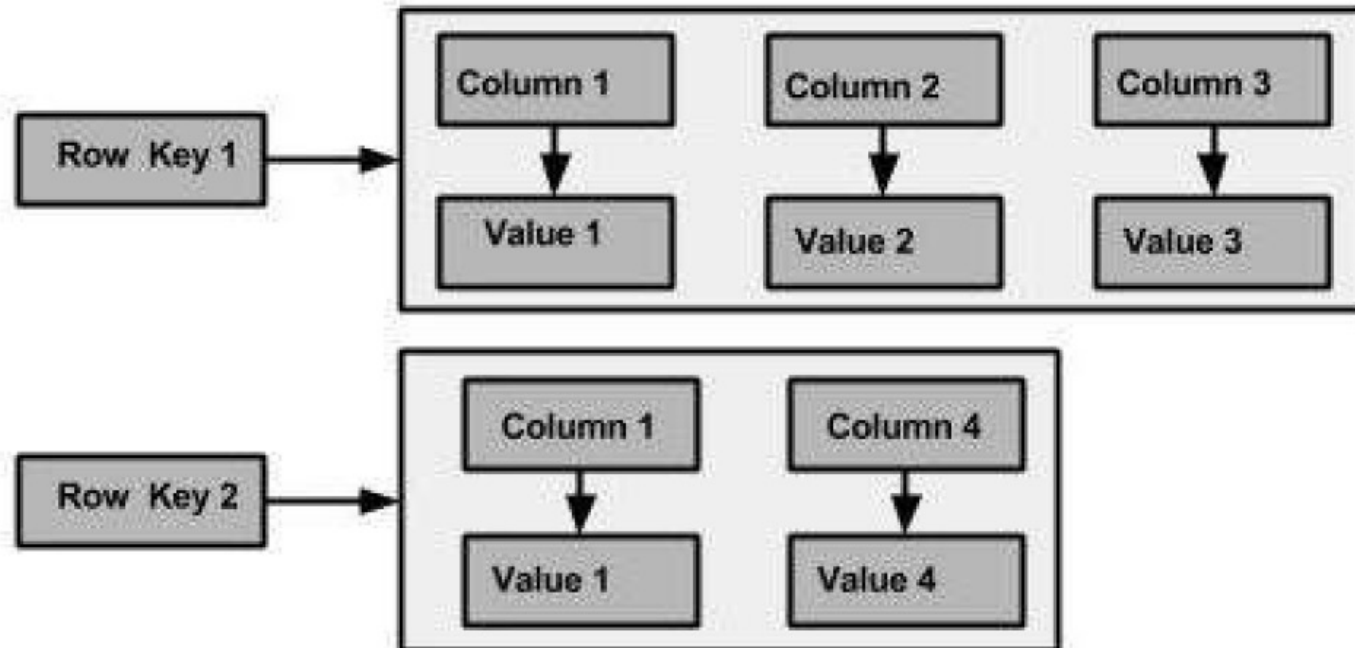
- SimpleStrategy is for data, that can be manageable within some limits such as the data would be text only or limited types of images.
- NetworkTopologyStrategy is for data such as High quality images and videos which had to travel through the network and data will be distributed among different clusters.

Replication Factor

- Replication factor is the strategy in Cassandra which makes replicas in nodes around the cluster to ensure reliability and scalability.
- The total number of replicas around the clusters is referred to as Replication Factor.

Cassandra - Data Model

The following figure shows an example of a Cassandra **column family**.



Cassandra - Cqlsh

By default, Cassandra provides a prompt Cassandra query language shell (cqlsh) that allows users to communicate with it. Using this shell, you can execute Cassandra Query Language (CQL).

Using cqlsh, you can

- 1)insert data, and**
- 2)define a schema,**
- 3)execute a query.**

Documented Shell Commands:-

DESCRIBE	Describes the current cluster of Cassandra and its objects.
EXPAND	Expands the output of a query vertically.
EXIT	Using this command, you can terminate cqlsh.
PAGING	Enables or disables query paging
SHOW	Displays the details of current cqlsh session such as Cassandra version, host, or data type assumptions.
SOURCE	Executes a file that contains CQL statements.
TRACING	Enables or disables request tracing.

CQL Data Definition Commands

command	use
CREATE KEYSPACE	Creates a KeySpace in Cassandra.
USE	Connects to a created KeySpace.
ALTER KEYSPACE	Changes the properties of a KeySpace.
DROP KEYSPACE	Removes a KeySpace
CREATE TABLE	Creates a table in a KeySpace
ALTER TABLE	Modifies the column properties of a table.
DROP TABLE	Removes a table.
TRUNCATE	Removes all the data from a table.
CREATE INDEX	Defines a new index on a single column of a table.
DROP INDEX	Deletes a named index.

CQL Data Manipulation Commands

COMMAND	USE
INSERT	Adds columns for a row in a table.
UPDATE	Updates a column of a row
DELETE	Deletes data from a table.
BATCH	Executes multiple DML statements at once.

CQL Clauses

command	use
SELECT	This clause reads data from a table
WHERE	The where clause is used along with select to read a specific data
ORDERBY	The orderby clause is used along with select to read a specific data in a specific order.

ALTER

ALTER KEYSPACE

```
ALTER ( KEYSPACE | SCHEMA ) keyspace_name
WITH REPLICATION = map
| ( WITH DURABLE_WRITES = ( true | false ))
AND ( DURABLE_WRITES = ( true | false ))
```

`map` is a map collection, a JSON-style array of literals, such as 'class': 'SimpleStrategy' or 'NetworkTopologyStrategy' in this format:

```
{ literal : literal, literal : literal ... }
```

ALTER TABLE

```
ALTER TABLE keyspace_name.table_name instruction
```

`instruction` is:

```
ALTER column_name TYPE cql_type
| ( ADD column_name cql_type )
| ( DROP column_name )
| ( RENAME column_name TO column_name )
| ( WITH property AND property ... )
```

`cql_type` is compatible with the original type and is a CQL type other than a collection or counter. Exceptions: `ADD` supports a collection type and also, if the table is a counter, a counter type.

`property` is a CQL table property (p. 4) and value, such as `read_repair_chance = .5`.

ALTER TYPE

```
ALTER TYPE name instruction
```

`name` is an identifier of a user-defined type.

`field_name` is an arbitrary identifier for the field.

`new_type` is an identifier other than the reserved type names.

Creation of Index and Keyspace

CREATE INDEX

```
CREATE CUSTOM INDEX IF NOT EXISTS index_name  
ON keyspace_name.table_name ( KEYS (column_name) )  
( USING class_name ) ( WITH OPTIONS = map )
```

Restrictions:

`USING class_name` is allowed only if `CUSTOM` is used and `class_name` is a string literal containing a java class name.

`index_name` is an identifier, enclosed or not enclosed in double quotation marks, excluding reserved words.

`map` is described in ALTER KEYSPACE.

CREATE KEYSPACE

```
CREATE ( KEYSPACE | SCHEMA ) IF NOT EXISTS keyspace_name  
WITH REPLICATION = map  
AND DURABLE_WRITES = ( true | false )
```

`map` is described in ALTER KEYSPACE.

Creation of Table

CREATE TABLE

```
CREATE TABLE IF NOT EXISTS keyspace_name.table_name  
( column_definition, column_definition, ... )  
WITH property AND property ...
```

column_definition is:

```
column_name cql_type STATIC PRIMARY KEY  
| column_name frozen<tuple<tuple_type>  
    tuple_type ... > PRIMARY KEY  
| column_name frozen<user-defined_type> PRIMARY KEY  
| ( PRIMARY KEY ( partition_key ) )
```

Restrictions:

- There should always be exactly one primary key definition.
- `cql_type` of the primary key must be a CQL type or user-defined type.
- `cql_type` of a collection uses this syntax:

```
LIST<cql_type>  
| ( SET<cql_type> ) | ( MAP<cql_type, cql_type> )
```

PRIMARY KEY is:

```
column_name  
| ( column_name1, column_name2, column_name3 ... )  
| ((column_name4,column_name5), column_name6, column_name7 ...)
```

`column_name1` is the partition key.

`column_name2, column_name3 ...` are clustering columns.

`column_name4, column_name5` are partitioning keys.

`column_name6, column_name7 ...` are clustering columns.

Creation of Table, Trigger, Type and User

CREATE TABLE (continued)

`property` is a CQL table storage property or one of these directives:

```
COMPACT STORAGE
| ( CLUSTERING ORDER BY (clustering_column
  ( ASC | DESC ), ... ) )
```

CREATE TRIGGER

```
CREATE TRIGGER trigger_name ON table_name
USING 'java_class'
```

CREATE TYPE

```
CREATE TYPE IF NOT EXISTS keyspace.type_name
(field, field, . . . )
```

`type_name` is a type identifier other than reserved type names.
`field` is: `field_name type`
`field_name` is an arbitrary identifier for the field.
`type` is a CQL collection or non-collection type other than a counter type.

CREATE USER

```
CREATE USER IF NOT EXISTS user_name
WITH PASSWORD 'password'
NOSUPERUSER | SUPERUSER
```

DELETE

```
DELETE column_name, ... | ( column_name term )  
FROM keyspace_name.table_name  
USING TIMESTAMP integer  
WHERE row_specification  
( IF ( EXISTS | ( condition ( AND condition ) ... ) ) )
```

term is:

```
[ list_position ] | key_value
```

row_specification is one of:

- primary_key_name = key_value
- primary_key_name IN (key_value, key_value, ...)

condition is:

```
| column_name [ list_position ] = key_value  
column_name = key_value
```

Delete / Drop

Permission Specification



DROP TRIGGER

```
DROP TRIGGER trigger_name ON table_name
```

DROP TYPE

```
DROP TYPE IF EXISTS type_name
```

type_name is the name of a user-defined type.

DROP USER

```
DROP USER IF EXISTS user_name
```

GRANT

```
GRANT permission_name PERMISSION  
| ( GRANT ALL PERMISSIONS ) ON resource TO user_name
```

permission_name is one of:

- ALTER
- AUTHORIZE
- CREATE
- DROP
- MODIFY
- SELECT

resource is one of:

- ALL KEYSPACES
- KEYSPACE keyspace_name
- TABLE keyspace_name.table_name

Insert Command

INSERT

```
INSERT INTO keyspace_name.table_name  
( column_name, column_name ... )  
VALUES ( value, value, ... ) IF NOT EXISTS  
USING option AND option
```

value is one of:

- a literal
- a set
{ literal, literal, ... }
- a list
[literal, literal, ...]
- a map collection, described in ALTER KEYSPACE

option is one of:

- **TIMESTAMP** microseconds
- **TTL** seconds

Select Command

SELECT

```
SELECT select_expression
FROM keyspace_name.table_name
WHERE relation AND relation ...
ORDER BY (clustering_column ( ASC | DESC ), ... )
LIMIT n
ALLOW FILTERING
```

select_expression is:
selection_list | (COUNT (* | 1))

selection_list is:
selector AS alias, selector AS alias, ... | *

alias is an alias for a column.

selector is:
column_name
| (WRITETIME (column_name))
| (TTL (column_name))
| (function (selector, selector, ...))

function is a timeuuid function, a token function, or a blob conversion function.

relation is:
column_name op term
| (column_name, column_name, ...) op term-tuple
| column_name IN (term, (term, ...))
| column_name, column_name, ...) IN (term-tuple, (term-tuple ...))
| TOKEN (column_name, ...) op (term)

op is:
= | < | > | <= | >= | = | CONTAINS | CONTAINS KEY

term-tuple is:
(term, term, ...)

term is a constant, such as a true or false, a bind marker (?), or a set, list, or map.

Update Command

UPDATE

```
UPDATE keyspace_name.table_name
USING option AND option
SET assignment, assignment ...
WHERE row_specification
IF column_name = literal
    AND column_name = literal ...
```

option is one of:

- `TIMESTAMP` microseconds
- `TTL` seconds

assignment is one of:

- `column_name = value`
- `set_or_list_item = set_or_list_item (+ | -) ...`
- `map_name = map_name (+ | -) ...`
- `column_name [term] = value`
- `counter_column_name = counter_column_name (+ | -) integer`

`set`, `list`, `map` are defined in `INSERT`.

term is:

```
[ list_position ] | key_value
```

row_specification is one of:

- `primary_key_name = key_value`
- `primary_key_name IN (key_value ,...)`

Creating Keyspace

- create keyspace bigdata with replication={'class':'SimpleStrategy', 'replication_factor':1};

File Import

Cassandra CQL Shell

```
(5 rows)
cqlsh:zeenat> copy employeee (employee_id, department, lastname, years_with_company) from 'C:\Users\Maham\Desktop\IOT\Big data\Lesson 7\employee_entries.csv' WITH DELIMITER='|' AND Header = TRUE;
Using 3 child processes

Starting copy of zeenat.employeee with columns [employee_id, department, lastname, years_with_company].
Processed: 5 rows; Rate:      4 rows/s; Avg. rate:      6 rows/s
5 rows imported from 1 files in 0.789 seconds (0 skipped).
cqlsh:zeenat> select * from employeee;
```

employee_id	department	lastname	years_with_company
5	Engineering	Gonzales	2
1	Engineering	Stevens	1
2	Engineering	Jones	2
4	Sales	Howard	1
3	Marketing	Smith	3

```
(5 rows)
cqlsh:zeenat>
```

References

- <https://downloads.datastax.com/#ddac>
- <https://www.datastax.com/blog/2015/06/datastax-community-217-and-2016-ready-download>
- <https://www.datastax.com/blog/2012/01/working-apache-cassandra-mac-os-x>
- <https://www.datastax.com/blog/2012/01/getting-started-apache-cassandra-windows-easy-way>
- <https://www.guru99.com/cassandra-architecture.html>
- <http://cassandra.apache.org/doc/latest/>

Issues References

- <https://issues.apache.org/jira/browse/CASSANDRA-8222>
- <https://stackoverflow.com/questions/27004773/cassandra-cqlsh-unable-to-connect-to-any-servers-127-0-0-19160-closed-is-a>
- <https://www.vitalsofttech.com/trying-to-connect-to-127-0-0-1-9042/>