Chapter – 4
CRUD operations in Apache Cassandra

Bal Krishna Nyaupane

Assistant Professor

Department of Electronics and Computer Engineering

Paschimanchal Campus, IOE

bkn@wrc.edu.np

Apache Cassandra

- ***History of Cassandra***
  - Avinash Lakshman, one of the authors of Amazon's Dynamo, and Prashant Malik initially developed Cassandra at Facebook to power the Facebook inbox search feature.
  - Facebook released Cassandra as an open-source project on Google code in July 2008
  - Cassandra was accepted into Apache Incubator in March 2009.
  - It was made an Apache top-level project since February 2010.

Apache Cassandra

- It is ***scalable, fault-tolerant, and consistent.***
- It is a ***column-oriented database.***
- Its distribution design is ***based on Amazon's Dynamo*** and its data model on ***Google's Bigtable.***
- Cassandra implements a ***Dynamo-style replication model with no single point of failure***, but adds a more powerful "***column family" data model.***
- Cassandra is being used by some of the biggest companies such as ***Facebook, Twitter, Cisco, Rackspace, ebay, Twitter, Netflix, and more.***
- ***From Wikipedia:*** Apache Cassandra is an ***open source distributed database*** management system designed to ***handle large amounts of data*** across many commodity servers, providing ***high availability with no single point of failure***. Cassandra offers ***robust support for clusters*** spanning multiple datacenters, with ***asynchronous masterless replication*** allowing ***low latency*** operations for all clients.

Features of Cassandra

- ***Distributed Databases***
  - Cassandra is a global distributed database. ***Cassandra supports features like replication and partitioning.***
  - Replication is a process where system maintains n number of replicas on various data sites.
  - Data Partitioning is a scheme, where data may be distributed across multiple nodes.
  - Partitioning is usually for managing high availability/performance on data.
- ***Peer-to-Peer Design***
  - Cassandra storage architecture is peer-to-peer. Each node in a cluster is assigned the same role, making it a decentralized database. Each node is independent of the other but interconnected.
  - Nodes in a network are capable of serving read/write database requests, so at a given point even if a node goes down, subsequent read/write requests will be served from other nodes in the network, ***hence there is no SPOF (Single Point Of Failure).***

Features of Cassandra

- ***Configurable Data Consistency***
  - Data consistency is synchronization of data across multiple replica nodes.
  - Eventually the consistency-based data model returns the last updated record. Such a data model is widely supported by many distributed databases. Cassandra also offers configurable eventual consistency.
- ***Cassandra Query Language (CQL)***
  - CQL was introduced with Cassandra 0.8 release with the intention of having a RDBMS style SQL. CQL adds a flavor of DDL and DML statements.
  - CQL treats the ***database (Keyspace)*** as a container of tables.
- ***Replication Factor***
  - The replication factor determines the number of copies of data (replica) that will be stored across nodes in a cluster. If one wishes to store only one copy of each row on one node, they should set the replication factor to one.
  - The replication factor should ideally be more than one and not more than the number of nodes in the cluster.

Features of Cassandra

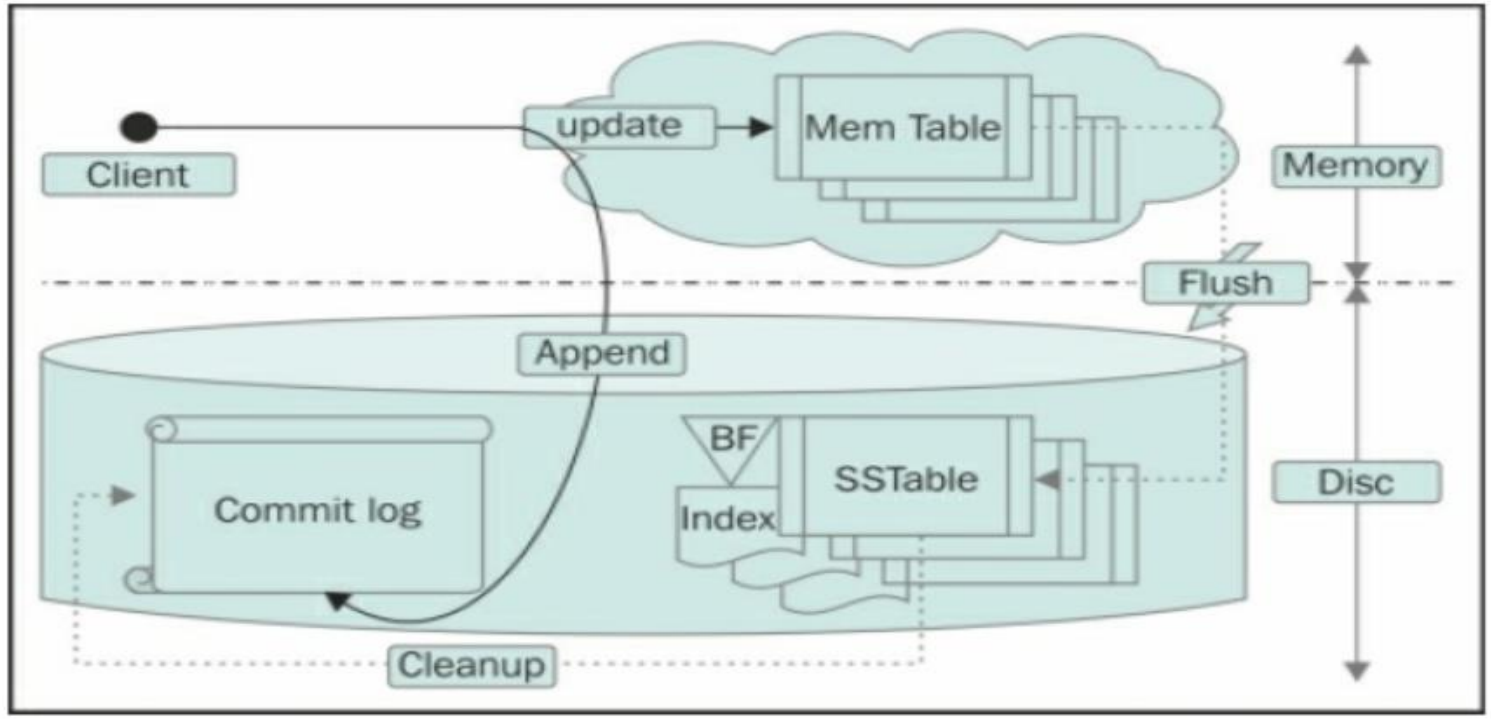
- ***When dealing with Cassandra, keep the following things in mind:***
  - **Denormalize, denormalize, and denormalize**: Denormalize wherever you can for quicker retrieval and let the application logic handle the responsibility of reliably updating all the redundancies.
  - **Rows are gigantic and sorted**: The giga-sized rows (a row can accommodate 2 billion cells per partition) can be used to store sortable and sliceable columns.
  - **One row, one machine**: Each row stays on one machine. *Rows are not sharded across nodes*. A high-demand row may create a hotspot.
  - **From query to model**: You may need to denormalize your model in such a way that all your queries stay limited to a bunch of simple commands *such as get, slice, count, multi_get,* and some simple indexed searches.

Components of Cassandra

- **Node:** It is the place where data is stored.
- **Data center:** It is a collection of related nodes.
- **Cluster:** A cluster is a component that contains one or more data centers.
- **Commit log:** The purpose of commit log is to ensure there is no data loss. The commit log is a crash-recovery mechanism in Cassandra. Every write operation is written to the commit log.
- **Mem-table:** A mem-table is a memory-resident data structure. After commit log, the data will be written to the mem-table. Sometimes, for a single-column family, there will be multiple mem-tables.
- **SSTable:** It is a disk file to which the data is flushed from the mem-table when its contents reach a threshold value.

Components of Cassandra

- ***Commit log, MemTable, and SSTable*** in a node are tightly coupled. Any write operation gets ***written to the commit log first and then the MemTable gets updated***.
- MemTable, based on certain criteria, ***gets flushed to a disk in immutable files called SSTable.*** The data in commit logs gets purged after its corresponding data in MemTable gets flushed to SSTable.



Components of Cassandra

- **The messaging service**
  - The messaging service is the mechanism that manages inter-node socket communication in a ring. Communications, for example gossip, read, read digest, write, and so on, processed via a messaging service, can be assumed as a gateway messaging server running at each node.
  - To communicate, each node creates two socket connections per node. This implies that if you have101 nodes, there will be 200 open sockets on each node to handle communication with other nodes.
- **Gossip**
  - Cassandra uses the gossip protocol for inter-node communication.
  - As the name suggests, the protocol spreads information in the same way an office rumor does. It can also be compared to a virus spread.
  - There is no central broadcaster, but the information (virus) gets transferred to the whole population.

Components of Cassandra

- It's a way for nodes to build the global map of the system with a small number of local interactions Cassandra uses gossip to find out the state and location of other nodes in the ring (cluster).
- The gossip process runs every second and exchanges information with, at the most, three other nodes in the cluster.
- Nodes exchange information about themselves and other nodes that they come to know about via some other gossip session.

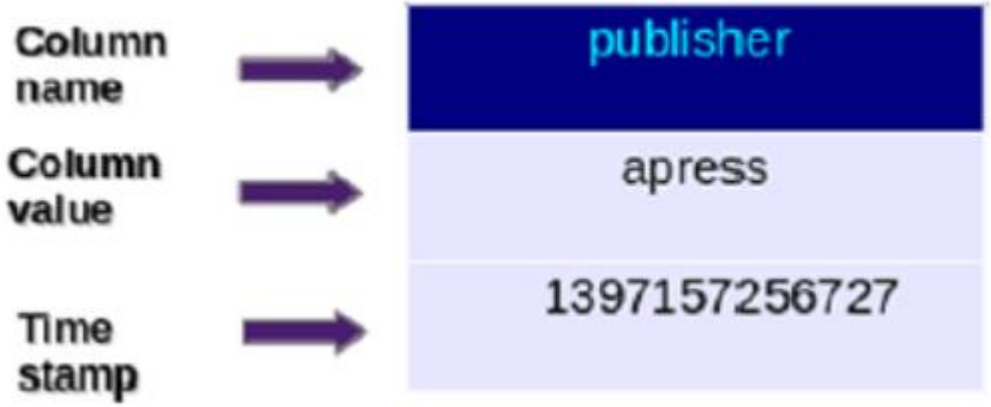
Cassandra Data Model

- ***The Cassandra data model:***

- At the heart of Cassandra lies ***two structures: column family and cell***. There is a container entity for these entities ***called keyspace.*** Keyspace is the outermost container for data in Cassandra.
- "Column family" is the old name for a table. There is still a slight difference between the column family and table, but for the most part, we can use them interchangeably.

Cassandra Data Model

- A cell is the smallest unit of the Cassandra data model. Cells are contained within a column family. A cell is essentially a key-value pair.
- The key of a cell is called **cell name** and value is called **cell value**. A cell can be represented as a triplet of the cell name, value, and timestamp.
- The timestamp is used to resolve conflicts during read repair or to reconcile two writes that happen to the same cell at the same time; the one written later wins.



Cassandra Vs. RDMS

Cassandra	RDBMS
• *Cassandra is used to deal with unstructured data.*	• *RDBMS is used to deal with structured data.*
• *Cassandra has flexible schema.*	• *RDBMS has fixed schema.*
• *In Cassandra, a table is a list of "nested key-value pairs". (Row x Column Key x Column value)*	• *In RDBMS, a table is an array of arrays. (Row x Column)*
• *In Cassandra, keyspace is the outermost container which contains data corresponding to an application.*	• *In RDBMS, database is the outermost container which contains data corresponding to an application.*
• *In Cassandra, tables or column families are the entity of a keyspace.*	• *In RDBMS, tables are the entities of a database.*
• *In Cassandra, row is a unit of replication.*	• *In RDBMS, row is an individual record.*
• *In Cassandra, column is a unit of storage.*	• *In RDBMS, column represents the attributes of a relation.*
• *In Cassandra, relationships are represented using collections.*	• *In RDBMS, there are concept of foreign keys, joins etc.*

Cassandra Installation on Windows

- Download: http://cassandra.apache.org/download.
- Extract Cassandra and put on C drive.
- Installed JDK 1.8 and Python 2.7
- Set Environment variables for Cassandra, Java and Python

System variables

Variable	Value
ComSpec	C:\WINDOWS\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
JAVA_HOME	C:\Program Files\Java\jdk1.8.0_101
NUMBER_OF_PROCESSORS	8

Edit environment variable

C:\Python27
C:\apache-cassandra-3.11.4\bin

New

Cassandra Installation on Windows

- Go to C:\apache-cassandra-3.11.4\bin and ***edit the Cassandra bath file.*** Set the Java home: Set JAVA_HOME=C:\Program Files\Java\jdk1.8.0_101
- Go to C:\apache-cassandra-3.11.4\bin and ***edit the nodetool bath file.*** Set the Java home: Set JAVA_HOME=C:\Program Files\Java\jdk1.8.0_101
- **Open command Prompt: type** ***cassandra –f*** ***(Don't close it)***
- ***Open another command prompt:*** ***type nodetool status press enter.***

Again type cqlsh

nodetool - Notepad

File Edit Format View Help

```
@REM (the "License"); you may not use this file except in compliance with
@REM the License. You may obtain a copy of the License at
@REM
@REM http://www.apache.org/licenses/LICENSE-2.0
@REM
@REM Unless required by applicable law or agreed to in writing, software
@REM distributed under the License is distributed on an "AS IS" BASIS,
@REM WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
@REM See the License for the specific language governing permissions and
@REM limitations under the License.
set JAVA_HOME=C:\Program Files\Java\jdk1.8.0_101
@echo off
if "%OS%" == "Windows_NT" setlocal
```

Cassandra Data Types

Data Type	Constants	Description
bigint	bigint	Represents 64-bit signed long
blob	blobs	Represents arbitrary bytes
counter	integers	Represents counter column
decimal	integers, floats	Represents variable-precision decimal
double	integers	Represents 64-bit IEEE-754 floating point
float	integers, floats	Represents 32-bit IEEE-754 floating point
inet	strings	Represents an IP address, IPv4 or IPv6
text	strings	Represents UTF8 encoded string
timestamp	integers, strings	Represents a timestamp
timeuuid	uuids	Represents type 1 UUID
varchar	strings	Represents uTF8 encoded string
varint	integers	Represents arbitrary-precision integer

Cqlsh basic commands

- **HELP** – Displays help topics for all cqlsh commands.
- **CAPTURE** – Captures the output of a command and adds it to a file.
- **CONSISTENCY** – Shows the current consistency level, or sets a new consistency level.
- **COPY** – Copies data to and from Cassandra.
- **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.

C:\WINDOWS\system32\cmd.exe - cqlsh

```
CREATE_AGGREGATE          DROP_KEYSPACE              PERMISSIONS
CREATE_COLUMNFAMILY       DROP_MATERIALIZED_VIEW     REVOKE
CREATE_FUNCTION           DROP_ROLE                  SELECT
CREATE_INDEX              DROP_TABLE                 SELECT_JSON

cqlsh> capture
Currently not capturing query output.
cqlsh> show
Improper show command.
cqlsh> source
Improper source command.
cqlsh> help

Documented shell commands:
==========================
CAPTURE  CLS          COPY  DESCRIBE  EXPAND  LOGIN   SERIAL  SOURCE   UNICODE
CLEAR    CONSISTENCY  DESC  EXIT      HELP    PAGING  SHOW    TRACING

CQL help topics:
================
AGGREGATES                CREATE_KEYSPACE            DROP_TRIGGER       TEXT
ALTER_KEYSPACE            CREATE_MATERIALIZED_VIEW   DROP_TYPE          TIME
ALTER_MATERIALIZED_VIEW   CREATE_ROLE                DROP_USER          TIMESTAMP
ALTER_TABLE               CREATE_TABLE               FUNCTIONS          TRUNCATE
ALTER_TYPE                CREATE_TRIGGER             GRANT              TYPES
ALTER_USER                CREATE_TYPE                INSERT             UPDATE
APPLY                     CREATE_USER                INSERT_JSON        USE
ASCII                     DATE                       INT                UUID
BATCH                     DELETE                     JSON
BEGIN                     DROP_AGGREGATE             KEYWORDS
BLOB                      DROP_COLUMNFAMILY          LIST_PERMISSIONS
BOOLEAN                   DROP_FUNCTION              LIST_ROLES
COUNTER                   DROP_INDEX                 LIST_USERS
CREATE_AGGREGATE          DROP_KEYSPACE              PERMISSIONS
CREATE_COLUMNFAMILY       DROP_MATERIALIZED_VIEW     REVOKE
CREATE_FUNCTION           DROP_ROLE                  SELECT
CREATE_INDEX              DROP_TABLE                 SELECT_JSON

cqlsh> _
```

Create, Alter and Drop Keyspaces

- ***Create a keyspace Syntax:***

```
Create keyspace KeyspaceName with replicaton={'class':strategy name,
'replication_factor': No of replications on different nodes}
```

- **Strategy:** There are ***two types of strategy*** declaration in Cassandra syntax:
  - **Simple Strategy:** Simple strategy is used in the case of one data center. In this strategy, the first replica is placed on the selected node and the remaining nodes are placed in clockwise direction in the ring without considering rack or node location.
  - **Network Topology Strategy:** This strategy is used in the case of more than one data centers. *In this strategy, you have to provide replication factor for each data center separately.*

- ***Drop a keyspace:***

```
DROP keyspace KeyspaceName ;
```

```
{ 'class' : 'SimpleStrategy', 'replication_factor' : <positive_integer> }
```

```
{ 'class' : 'NetworkTopologyStrategy'[, '<datacenter_name>' :
<positive_integer>, '<datacenter_name>' : <positive_integer>, ...] }
```

Here are a couple of examples:

```
# Single data center, replication factor: 3
{ 'class' : 'NetworkTopologyStrategy', 'DC1' : 3}

# Three data centers, with RF as 3, 2, and 1
{ 'class' : 'NetworkTopologyStrategy', 'DC_NY' : 3, 'TokyoDC' : 2, 'DC_Hadoop':
1}

# Two data centers with three replica each
{ 'class' : 'NetworkTopologyStrategy', 'DC1' : 3, 'DC2' : 3}
```

Create, Alter and Drop Keyspaces

- *Create a keyspace by the name "Student".*
- *List all the existing keyspace:* **describe keyspaces.**

```
cqlsh> CREATE KEYSPACE Student
   ... WITH replication = {'class':'SimpleStrategy', 'replication_factor' : 3};
cqlsh> describe keyspaces

system_schema  system_auth  system  student  system_distributed  system_traces
```

- *Alter the keyspace "Student" strategy from 'SimpleStrategy' to 'NetworkTopologyStrategy' and replication factor from 3 to 1 for DataCenter1.*

```
cqlsh:student> ALTER KEYSPACE Student
          ... WITH replication = {'class':'NetworkTopologyStrategy', 'replication_factor' : 3};
ConfigurationException: replication_factor is an option for SimpleStrategy, not NetworkTopologyStrategy
cqlsh:student> ALTER KEYSPACE Student
          ... WITH replication = {'class':'NetworkTopologyStrategy', 'DC1' : 1};
cqlsh:student> describe keyspaces
```

Cassandra Create Table

- In Cassandra, CREATE TABLE command is used to create a table.

```
CREATE (TABLE | COLUMNFAMILY) <tablename>
('<column-definition>' , '<column-definition>')
(WITH <option> AND <option>)
```

```
CREATE TABLE tablename(
  column1 name datatype PRIMARYKEY,
  column2 name data type,
  column3 name data type.
  )
```

- Create a Table *by the name* ***"Student_Info"*** *with* following information: ***Name, Age, Mobile, Year, and Address.***

```
cqlsh:student> CREATE TABLE Student_Info(
           ...     ID  int PRIMARY KEY,
           ...     Name varchar,
           ...     Age int,
           ...     Mobile varint,
           ...     Year int,
           ...     Address varchar
           ...     );
cqlsh:student> select * from Student_Info;

 id | address | age | mobile | name | year
----+---------+-----+--------+------+------

(0 rows)
```

Alter/Drop/Truncate Table Commands

- ***Adding Column Syntax:***

```
ALTER TABLE table name
ADD new column datatype;
```

```
ALTER TABLE [keyspace_name.] table_name
[ALTER column_name TYPE cql_type]
[ADD (column_definition_list)]
[DROP column_list | COMPACT STORAGE ]
[RENAME column_name TO column_name]
[WITH table_properties];
```

- ***Dropping a Column Syntax:***

```
ALTER table name
DROP column name;
```

- ***Truncate Table:*** Truncate command is used to truncate a table. If you truncate a table, all the rows of the table are deleted permanently.

```
TRUNCATE <tablename>
```

Alter/Drop/Truncate Table Commands

```
cqlsh:student> select * from Student_Info;

 id | address | age | name | year
----+---------+-----+------+------
  1 |   Bagar |  23 |  Ram | 2072

(1 rows)
cqlsh:student> ALTER Table Student_Info Drop Year;
cqlsh:student> select * from Student_Info;

 id | address | age | name
----+---------+-----+------
  1 |   Bagar |  23 |  Ram
```

```
cqlsh:student> ALTER Table Student_Info Add Year int;
cqlsh:student> select * from Student_Info;

 id | address | age | name | year
----+---------+-----+------+------
  1 |   Bagar |  23 |  Ram | null

(1 rows)
```

- ***Rename Column id to sid***

```
cqlsh:student> alter table Student_Info rename address to place;
InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot rename non PRIMARY KEY p
cqlsh:student> alter table Student_Info rename id to Sid;
cqlsh:student> select * from student_Info;

 sid | address     | age | name    | year
-----+-------------+-----+---------+------
   5 |     Simpani |  23 |   Shyam | 2073
   1 |       Bagar |  23 |     Ram | null
   8 | Madhyampath |  22 | Bhimesh | 2073
   2 | Madhyampath |  22 |   Shyam | 2072
   4 |       Bagar |  22 |  Dinesh | 2073
   7 | Madhyampath |  23 |  Biplav | 2074
   6 |    Lakeside |  24 |  Mahesh | 2071
   3 |    New Road |  23 |    Hari | 2073

(8 rows)
```

Create/Drop Index Commands

- ***Create Index Command***
  - *It* is used to create an index on the column specified by the user.
  - If the data already exists for the column which you choose to index, Cassandra creates indexes on the data during the 'create index' statement execution.
  - If the index name was not specified during index creation, ***then index name is TableName_ColumnName_idx.***
  - ***Rules for creating Index***
    - ✓The index cannot be created on primary key as a primary key is already indexed.
    - ✓In Cassandra, Indexes on collections are not supported.
    - ✓Without indexing on the column, Cassandra can't filter that column unless it is a primary key.
- ***Syntax:*** **CREATE INDEX** <identifier> **ON** <tablename>

```
cqlsh:student> CREATE INDEX Sname ON Student_Info (Name);
cqlsh:student> select * from Student_Info;

 id | address | age | name | year
----+---------+-----+------+------
  1 |   Bagar |  23 |  Ram | null

(1 rows)
cqlsh:student> CREATE INDEX Sname ON Student_Info (Name);
InvalidRequest: Error from server: code=2200 [Invalid query] message="Index sname already exists"
```

Create/Drop Index Commands

- ***Drop Index Command***
  - It is used to drop a specified index.
  - ***Rules for dropping a Index.***
    - ✓If the index does not exist, it will return an error unless you use IF EXISTS which returns no operation.
    - ✓During index creation, you have to specify keyspace name with the index name otherwise index will be dropped from the current keyspace.
- ***Syntax:*** **DROP INDEX** <identifier> ***or*** **Drop index** IF EXISTS KeyspaceName.IndexName

```
cqlsh:student> Drop index Sname
           ... ;
cqlsh:student> CREATE INDEX Sname ON Student_Info (Name);
cqlsh:student>
```

Create/Insert Data

- ***Insert Command Syntax:***

```
INSERT INTO <tablename>
(<column1 name>, <column2 name>....)
VALUES (<value1>, <value2>....)
USING <option>
```

```
cqlsh:student> INSERT INTO Student_Info (id, Name, Age,Year,Address ) VALUES ( 2, 'Shyam',22,2072,'Madhyampath');
cqlsh:student> INSERT INTO Student_Info (id, Name, Age,Year,Address ) VALUES ( 3, 'Hari',23,2073,'New Road');
cqlsh:student> INSERT INTO Student_Info (id, Name, Age,Year,Address ) VALUES ( 4, 'Dinesh',22,2073,'Bagar');
cqlsh:student>
cqlsh:student> INSERT INTO Student_Info (id, Name, Age,Year,Address ) VALUES ( 5, 'Shyam',23,2073,'Simpani');
cqlsh:student> INSERT INTO Student_Info (id, Name, Age,Year,Address ) VALUES ( 6, 'Mahesh',24,2071,'Lakeside');
cqlsh:student> INSERT INTO Student_Info (id, Name, Age,Year,Address ) VALUES ( 7, 'Biplav',23,2074,'Madhyampath');
cqlsh:student> INSERT INTO Student_Info (id, Name, Age,Year,Address ) VALUES ( 8, 'Bhimesh',22,2073,'Madhyampath');
cqlsh:student> select * from Student_info;

 id | address      | age | name    | year
----+--------------+-----+---------+------
  5 |      Simpani |  23 |   Shyam | 2073
  1 |        Bagar |  23 |     Ram | null
  8 | Madhyampath  |  22 | Bhimesh | 2073
  2 | Madhyampath  |  22 |   Shyam | 2072
  4 |        Bagar |  22 |  Dinesh | 2073
  7 | Madhyampath  |  23 |  Biplav | 2074
  6 |     Lakeside |  24 |  Mahesh | 2071
  3 |     New Road |  23 |    Hari | 2073
```

Read Commands

- ***Read Command Syntax:***

```
SELECT * | select_expression | DISTINCT partition
FROM [keyspace_name.] table_name
[WHERE partition_value
   [AND clustering_filters
   [AND static_filters]]]
[ORDER BY PK_column_name ASC|DESC]
[LIMIT N]
[ALLOW FILTERING]
```

- ***Display the records of students who lives in Madhyampath.***

```
cqlsh:student> select * from Student_info where Address='Madhyampath' allow filtering;

 id | address     | age | name    | year
----+-------------+-----+---------+-----
  8 | Madhyampath |  22 | Bhimesh | 2073
  2 | Madhyampath |  22 |   Shyam | 2072
  7 | Madhyampath |  23 |  Biplav | 2074
```

Read Commands

- ***Display the records of students whose id column has either 1 or 3 or 5.***

```
cqlsh:student> select * from Student_info where id in(3,5,7) allow filtering;

 id | address      | age | name    | year
----+--------------+-----+---------+------
  3 |     New Road |  23 |    Hari | 2073
  5 |      Simpani |  23 |   Shyam | 2073
  7 | Madhyampath  |  23 | Biplav  | 2074
```

- ***Display the only TWO records of students whose Batch year is 2073.***

```
cqlsh:student> select * from Student_info where year=2073 Limit 2 allow filtering;

 id | address      | age | name     | year
----+--------------+-----+----------+------
  5 |      Simpani |  23 |    Shyam | 2073
  8 | Madhyampath  |  22 | Bhimesh  | 2073
```

- ***Display the only TWO records of students whose Batch year is 2073.And display name as First_Name.***

```
cqlsh:student> select id,name as First_Name from Student_info where year=2073 Limit 2 allow filtering;

 id | first_name
----+-----------
  5 |      Shyam
  8 |    Bhimesh
```

Cassandra Update Commands

```
UPDATE <tablename>
SET <column name> = <new value>
<column name> = <value>....
WHERE <condition>
```

- ***Update Student name Ram to Rahul and Year to 2073 whose sid=1.***

```
cqlsh:student> update Student_info set name='Rahul', year=2073 where sid=1;
cqlsh:student> select * from student_Info;

 sid | address     | age | name    | year
-----+-------------+-----+---------+------
   5 |     Simpani |  23 |   Shyam | 2073
   1 |       Bagar |  23 |   Rahul | 2073
   8 | Madhyampath |  22 | Bhimesh | 2073
   2 | Madhyampath |  22 |   Shyam | 2072
   4 |       Bagar |  22 |  Dinesh | 2073
   7 | Madhyampath |  23 |  Biplav | 2074
   6 |    Lakeside |  24 |  Mahesh | 2071
   3 |    New Road |  23 |    Hari | 2073
```

Cassandra Delete Commands

```
DELETE [column_name (term)][, ...]
FROM [keyspace_name.] table_name
[USING TIMESTAMP timestamp_value]
WHERE PK_column_conditions
[IF EXISTS | IF static_column_conditions]
```

- ***Delete Student record whose sid=1:*** ***delete Student_Info where sid=1;***
- ***Delete Year column only whose sid=5;***

```
cqlsh:student> delete year from Student_Info where sid=5;
cqlsh:student> select * from student_Info;

 sid | address      | age | name     | year
-----+--------------+-----+----------+------
   5 |      Simpani |  23 |    Shyam | null
   8 | Madhyampath  |  22 | Bhimesh  | 2073
   2 | Madhyampath  |  22 |    Shyam | 2072
   4 |        Bagar |  22 |   Dinesh | 2073
   7 | Madhyampath  |  23 |   Biplav | 2074
   6 |     Lakeside |  24 |   Mahesh | 2071
   3 |     New Road |  23 |     Hari | 2073
```

Cassandra Batch Statement

- In ***Cassandra BATCH*** is used to execute multiple modification statements ***(insert, update, delete) simultaneously.***

BEGIN BATCH

<**insert**-stmt>/ <**update**-stmt>/ <**delete**-stmt>

APPLY BATCH

```
cqlsh:student> Begin Batch
           ... INSERT INTO Student_Info (sid, Name, Age,Year,Address ) VALUES ( 9, 'Prakash',24,2072,'Bagar');
           ... update Student_Info set year=2073 where sid=5;
           ... INSERT INTO Student_Info (sid, Name, Age,Year,Address ) VALUES ( 10, 'Padam',24,2073,'New Road');
           ... Apply Batch;
cqlsh:student> select * from student_Info;

 sid | address     | age | name    | year
-----+-------------+-----+---------+-----
   5 |     Simpani |  23 |   Shyam | 2073
  10 |    New Road |  24 |   Padam | 2073
   8 | Madhyampath |  22 | Bhimesh | 2073
   2 | Madhyampath |  22 |   Shyam | 2072
   4 |       Bagar |  22 |  Dinesh | 2073
   7 | Madhyampath |  23 |  Biplav | 2074
   6 |    Lakeside |  24 |  Mahesh | 2071
   9 |       Bagar |  24 | Prakash | 2072
   3 |    New Road |  23 |    Hari | 2073
```

```
cqlsh:student> describe table student_info;

CREATE TABLE student.student_info (
    sid int PRIMARY KEY,
    address text,
    age int,
    name text,
    year int
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.S
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apa
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
CREATE INDEX sname ON student.student_info (name);
```

Cassandra CQL Collections

- CQL provides the facility of using Collection data types. Using these Collection types, you can store multiple values in a single variable.
- ***When to use Collection:*** to store or denormalize a small amount of data.
- There are ***limitations in Cassandra collections***.
  - Cassandra collection cannot store data more than 64KB.
  - Keep a collection small to prevent the overhead of querying collection because entire collection needs to be traversed.
  - If you store more than 64 KB data in the collection, only 64 KB will be able to query, it will result in loss of data.
- There are ***three types of collection*** supported by Cassandra:
  - ***Set***
  - ***List***
  - ***Map***

Collections - Set

- A column of type set consists of unordered unique values. However, when the column is queried, it returns the values in sorted order.
- ***Alter the table Student_Info to add a column Hobbies.***
- ***And Update the table to provide the values for Hobby for the sid=1***

```
cqlsh:student> alter table Student_info Add Hobby set<text>;
cqlsh:student> update Student_Info
           ... Set Hobby=Hobby +{'Football','Watching Movie','Cycling'}
           ... where sid=5;
cqlsh:student> select * from student_info;

 sid | address     | age | hobby                                          | name    | year
-----+-------------+-----+------------------------------------------------+---------+------
   5 |     Simpani |  23 | {'Cycling', 'Football', 'Watching Movie'}      |   Shyam | 2073
  10 |    New Road |  24 |                                           null |   Padam | 2073
   8 | Madhyampath |  22 |                                           null | Bhimesh | 2073
   2 | Madhyampath |  22 |                                           null |   Shyam | 2072
   4 |       Bagar |  22 |                                           null |  Dinesh | 2073
   7 | Madhyampath |  23 |                                           null |  Biplav | 2074
   6 |    Lakeside |  24 |                                           null |  Mahesh | 2071
   9 |       Bagar |  24 |                                           null | Prakash | 2072
   3 |    New Road |  23 |                                           null |    Hari | 2073

(9 rows)
```

Collections - List

- List is ***used in the cases where***
  - **the order** of the elements is **to be maintained**, and
  - a **value** is to be stored ***multiple times***.
- You can ***get the values*** of a list data type using the ***index of the elements*** in the list.
- ***Alter the table Student_Info to add a column Subjects.***
- ***And Update the table to provide the values for Subjects for the sid=10.***

```
cqlsh:student> alter table Student_info Add Subjects list<text>;
cqlsh:student> update Student_Info
           ... Set Subjects=Subjects + ['Ml','Big Data','Cloud']
           ... where sid=10;
cqlsh:student> select * from student_info;

 sid | address     | age | hobby                                        | name    | subjects                        | year
-----+-------------+-----+----------------------------------------------+---------+---------------------------------+------
   5 |     Simpani |  23 | {'Cycling', 'Football', 'Watching Movie'}    |   Shyam |                            null | 2073
  10 |    New Road |  24 |                                         null |   Padam | ['Ml', 'Big Data', 'Cloud'] | 2073
   8 | Madhyampath |  22 |                                         null | Bhimesh |                            null | 2073
   2 | Madhyampath |  22 |                                         null |   Shyam |                            null | 2072
```

Collections – List/Set

- ***Insert values for all columns into table Student_Info.***
  - *VALUES (7, 'Biplav',23,{'Cricket','Chess'},['CNN','IM'],'Madhyampath');*
  - *VALUES (8, 'Bhimesh',22,{'Cricket','Ludo'},['RNN','DL'],'Bagar');*

```
cqlsh:student> INSERT INTO Student_Info (sid, Name, Age,Hobby,Subjects,Address ) VALUES ( 7, 'Biplav',23,{'Cricket','Chess'},['CNN','IM'],'Madhyampath');
cqlsh:student> INSERT INTO Student_Info (sid, Name, Age,Hobby,Subjects,Address ) VALUES ( 8, 'Bhimesh',22,{'Cricket','Ludo'},['RNN','DL'],'Bagar');
cqlsh:student> select * from student_info;

 sid | address     | age | hobby                                | name    | subjects                  | year
-----+-------------+-----+--------------------------------------+---------+---------------------------+------
   5 |     Simpani |  23 | {'Cycling', 'Football', 'Watching Movie'} |   Shyam |                      null | 2073
  10 |    New Road |  24 |                                 null |   Padam | ['Ml', 'Big Data', 'Cloud'] | 2073
   8 |       Bagar |  22 |                  {'Cricket', 'Ludo'} | Bhimesh |              ['RNN', 'DL'] | null
   7 | Madhyampath |  23 |                 {'Chess', 'Cricket'} |  Biplav |              ['CNN', 'IM'] | null
```

Collections – List/Set

- *Add one more hobby "Football" to the record sid=7.*

```
cqlsh:student> update Student_Info
           ... Set Hobby=Hobby +{'Football'}
           ... where sid=7;
cqlsh:student> select * from student_info;

 sid | address     | age | hobby                                          |
-----+-------------+-----+------------------------------------------------+-
   5 |     Simpani |  23 | {'Cycling', 'Football', 'Watching Movie'} |
  10 |    New Road |  24 |                                          null |
   8 |       Bagar |  22 |                          {'Cricket', 'Ludo'} |
   7 | Madhyampath |  23 |             {'Chess', 'Cricket', 'Football'} |

(4 rows)
```

- *Remove "Watching Movie " hobby from the record with sid=5.*

```
cqlsh:student> update Student_Info Set Hobby=Hobby - {'Watching Movie'} where sid=5;
cqlsh:student> select * from student_info;

 sid | address     | age | hobby                                | name    | subjects                          | year
-----+-------------+-----+--------------------------------------+---------+-----------------------------------+------
   5 |     Simpani |  23 |             {'Cycling', 'Football'} |   Shyam |                              null | 2073
  10 |    New Road |  24 |                                 null |   Padam | ['Ml', 'Big Data', 'Cloud'] | 2073
   8 |       Bagar |  22 |                 {'Cricket', 'Ludo'} | Bhimesh |                     ['RNN', 'DL'] | null
   7 | Madhyampath |  23 | {'Chess', 'Cricket', 'Football'} |  Biplav |                     ['CNN', 'IM'] | null
```

Collections – List/Set

- ***Add Two Subjects "ML" and "IM" to the record sid=5.***

```
cqlsh:student> update Student_Info Set Subjects=Subjects + ['ML','IM'] where sid=5;
cqlsh:student> select * from student_info;

 sid | address     | age | hobby                           | name    | subjects                    | year
-----+-------------+-----+---------------------------------+---------+-----------------------------+------
   5 |     Simpani |  23 |          {'Cycling', 'Football'} |   Shyam |                 ['ML', 'IM'] | 2073
```

- ***Display the records of Student whose hobby is 'Cricket'.***

```
cqlsh:student> select Hobby, Name from Student_Info where Hobby contains 'Cricket' allow filtering;

 hobby                              | name
-----------------------------------+---------
              {'Cricket', 'Ludo'} | Bhimesh
 {'Chess', 'Cricket', 'Football'} |  Biplav
```

- ***Display the records of Student Who are studying 'IM'.***

```
cqlsh:student> select Subjects, Name from Student_Info where Subjects contains 'IM' allow filtering;

 subjects       | name
----------------+--------
  ['ML', 'IM'] |  Shyam
 ['CNN', 'IM'] | Biplav
```

Collections – List/Set

- ***Remove all elements** from a set of sid=5 by using **UPDATE or DELETE command**.*

  update Student_Info Set Hobby={} where sid=5; ***or***

  Delete Hobby from Student_Info where sid=5;

- ***Remove an element** from index 1 of Subjects list **whose sid=10.***

```
cqlsh:student> Delete Subjects['Big Data'] from Student_Info where sid=10;
InvalidRequest: Error from server: code=2200 [Invalid query] message="Invalid STRING constant (Big Data) for "idx(subjects)" of type int"
cqlsh:student> Delete Subjects[1] from Student_Info where sid=10;
cqlsh:student> select * from student_info;

 sid | address     | age | hobby                          | name    | subjects        | year
-----+-------------+-----+--------------------------------+---------+-----------------+------
   5 |     Simpani |  23 |                           null |   Shyam |    ['ML', 'IM'] | 2073
  10 |    New Road |  24 |                           null |   Padam | ['Ml', 'Cloud'] | 2073
   8 |       Bagar |  22 |              {'Cricket', 'Ludo'} | Bhimesh |   ['RNN', 'DL'] | null
   7 | Madhyampath |  23 | {'Chess', 'Cricket', 'Football'} |  Biplav |   ['CNN', 'IM'] | null
```

Collections – List/Set

- ***Remove an element "Cricket" from Hubby set whose sid=7.***

```
cqlsh:student> Delete Hobby['Chess'] from Student_Info where sid=7;
cqlsh:student> select * from student_info;

 sid | address     | age | hobby                    | name    | subjects        | year
-----+-------------+-----+--------------------------+---------+-----------------+------
   5 |     Simpani |  23 |                     null |   Shyam |    ['ML', 'IM'] | 2073
  10 |    New Road |  24 |                     null |   Padam | ['Ml', 'Cloud'] | 2073
   8 |       Bagar |  22 |     {'Cricket', 'Ludo'} | Bhimesh |   ['RNN', 'DL'] | null
   7 | Madhyampath |  23 | {'Cricket', 'Football'} |  Biplav |   ['CNN', 'IM'] | null

(4 rows)
cqlsh:student> Delete Hobby[1] from Student_Info where sid=7;
InvalidRequest: Error from server: code=2200 [Invalid query] message="Invalid INTEGER constant (1) for "value(hobby)" of type text"
```

Collections – Map: Key, Value pair

- ***Alter the table Student_Info to add a column Routine as a Map.***
- ***And Update the table Student_Info to provide the values for Routine:{FirstPeriod:'Big Data', SecondPeriod:Cloud' }.***

```
cqlsh:student> alter table Student_info Add Routine map<text,text>;
cqlsh:student> update Student_Info Set Routine={'FirstPeriod':'Big Data','SecondPeriod':'Cloud'} where sid=5;
cqlsh:student> select * from student_info;

 sid | address     | age | hobby                    | name    | routine                                                 |
-----+-------------+-----+--------------------------+---------+---------------------------------------------------------+
   5 |     Simpani |  23 |                     null |   Shyam | {'FirstPeriod': 'Big Data', 'SecondPeriod': 'Cloud'} |
```

- ***And Update the table Student_Info to provide the values for Routine:{ThirdPeriod:'Project'}.***

```
cqlsh:student> update Student_Info Set Routine=Routine +{'ThirdPeriod':'Project'} where sid=5;
cqlsh:student> select * from student_info;

 sid | address     | age | hobby                    | name    | routine                                                                                |
-----+-------------+-----+--------------------------+---------+----------------------------------------------------------------------------------------+
   5 |     Simpani |  23 |                     null |   Shyam | {'FirstPeriod': 'Big Data', 'SecondPeriod': 'Cloud', 'ThirdPeriod': 'Project'} |
```

Collections – Map: Key, Value pair

- ***Display Routine and Name from Student_Info table which has a key "FirstPeriod"***

```
cqlsh:student> select Routine, name from Student_info where Routine contains key 'FirstPeriod' allow filtering;

 routine                                                                | name
------------------------------------------------------------------------+-------
 {'FirstPeriod': 'Big Data', 'SecondPeriod': 'Cloud', 'ThirdPeriod': 'Project'} | Shyam
```

- ***Delete an element from the map which has a key "FirstPeriod" and sid=5.***

```
cqlsh:student> delete routine['FirstPeriod'] from Student_info where sid=5;
cqlsh:student> select Routine, name from Student_info
           ... ;

 routine                                                | name
--------------------------------------------------------+---------
 {'SecondPeriod': 'Cloud', 'ThirdPeriod': 'Project'} |   Shyam
                                                   null |   Padam
                                                   null | Bhimesh
                                                   null |  Biplav
```

Aggregation Functions

```
cqlsh:student> select sum(age) as TotalSum from Student_info
           ... ;

 totalsum
----------
       92

(1 rows)

Warnings :
Aggregation query used without partition key

cqlsh:student> select avg(age) as AvgSum from Student_info;

 avgsum
--------
     23
```

```
cqlsh:student> select max(age) as Maximum_Age , name from Student_info
           ... ;

 maximum_age | name
-------------+-------
          24 | Shyam

(1 rows)

Warnings :
Aggregation query used without partition key

cqlsh:student> select count(*) as Total_Rows from Student_info;

 total_rows
------------
          4
```

Import and Export to/From csv file.

- ***Export Command:***

```
cqlsh:student> copy Student_Info(sid,Name,Age,Hobby,Subjects,Address,year) to 'd:\Student.csv';
Using 7 child processes

Starting copy of student.student_info with columns [sid, name, age, hobby, subjects, address, year].
Processed: 4 rows; Rate:       5 rows/s; Avg. rate:       4 rows/s
4 rows exported to 1 files in 1.142 seconds.
cqlsh:student>
```

- ***Import Command:***

```
cqlsh:student> copy ImportedFile(sid,Name,Age,Hobby,Subjects,Address,year) from 'd:\Student.csv';
Column family 'importedfile' not found
cqlsh:student> Truncate Student_Info;
cqlsh:student> select * from Student_info;

 sid | address | age | hobby | name | routine | subjects | year
-----+---------+-----+-------+------+---------+----------+------

(0 rows)
cqlsh:student> copy Student_Info(sid,Name,Age,Hobby,Subjects,Address,year) from 'd:\Student.csv';
Using 7 child processes
```

Thank You

???