Cassandra

Day 2

Day 2 - Overview

- CREATE, ALTER and DELETE Keyspace
- Creating Table
- Defining Columns and Data Types
- Primary Key
- Partition Key
- Clustering Key
- INSERT
- SELECT
- UPDATE
- DELETE

Keyspace in Cassandra

A keyspace is a data container in Cassandra, similar to a database in relational database management systems (RDMBS). A cluster has one keyspace per application, as many as needed, depending on requirements and system usage. Keyspaces are entirely separate entities, and the data they contain is unrelated to each other.

In a Cassandra cluster, a keyspace is an outermost object that determines how data replicates on nodes. Keyspaces consist of core objects called column families (which are like tables in RDBMS), rows indexed by keys, data types, data center awareness, replication factor, and keyspace strategy.
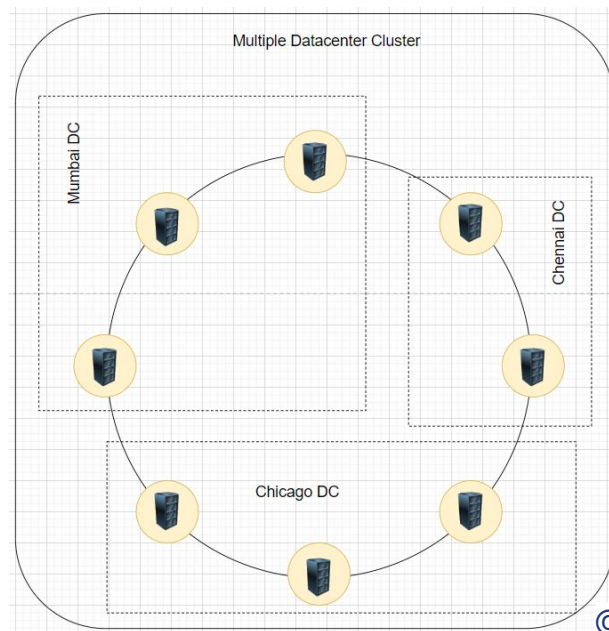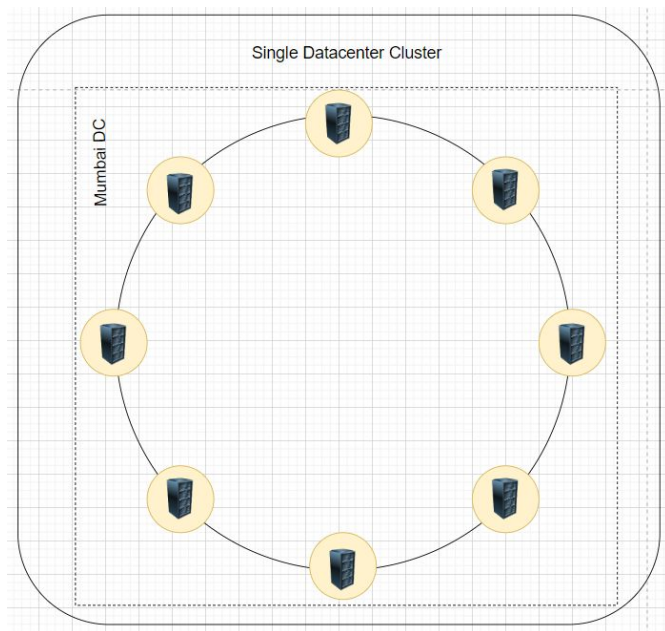Replication Strategy

When defining a keyspace, the replication strategy specifies the nodes where replicas will be placed. By using multiple nodes to place replicas, you achieve fault tolerance, high availability, and reliability.

There are two possible strategies:

Simple Strategy. Use this strategy for test and development environments, and if you do not intend to deploy a cluster to more than one data center. The replication factor applies to the whole cluster. The partitioner decides where to put the first replica on a node. Then, other replicas are distributed clockwise on the next nodes irrespective of data center or location.

Network Topology Strategy. This strategy is suitable when you need to deploy your cluster to multiple data centers. However, you can use it even with a single data center so you can expand later. Network Topology Strategy works for both production and development. It tends to place replicas on nodes that are not in the same rack to avoid issues when one rack goes down. Each data center can have a separate replication factor by using this option.

Cassandra Cluster

Creating Keyspace

Simple

      CREATE KEYSPACE simple_keyspace

      WITH replication = {'class':'SimpleStrategy', 'replication_factor' : 3};

Network Topology

      CREATE KEYSPACE network_topology_keyspace

      WITH replication = {'class':'NetworkTopologyStrategy', 'datacenter1' : 3};

Validate Existing Keyspace

Fetch All Keyspaces

select *

from system_schema.keyspaces;

Fetch a single Keyspace

select *

from system_schema.keyspaces

where keyspace_name = 'simple_keyspace';

Alter Keyspace

The only thing you cannot change is the keyspace name. Other than that you can alter the replication strategy, replication factor, and durable writes.

ALTER KEYSPACE simple_keyspace

WITH replication = {'class':'SimpleStrategy', 'replication_factor' : 1};

Deleting Keyspace

Deleting Keyspace without exists check

    DROP KEYSPACE simple_keyspace_2;

Optional Drop, if exists

    DROP KEYSPACE if exists simple_keyspace_2;

Data Types

Type	Constants supported	Description
ascii	string	ASCII character string and uses 1 byte per character
bigint	integer	64-bit signed long : range -9 quintillion to 9 quintillion
blob	blob	Arbitrary bytes (no validation)
boolean	boolean	Either true or false
counter	integer	Counter column (64-bit signed value). Can only be incremented or decremented
date	integer, string	A date (with no corresponding time value). Example yyyy-mm-dd
decimal	integer, float	Variable-precision decimal
double	integer float	64-bit IEEE-754 floating point
duration	duration,	A duration with nanosecond precision. Time elapsed or time since use case - 89h4m48s
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.

Data Types - Continued

Type	Constants supported	Description
int	integer	32-bit signed int - ranging from approximately -2.1 billion to 2.1 billion
smallint	integer	16-bit signed int - ranging from -32,768 to 32,767
text	string	String - Alias for varchar
time	integer, string	A time (with no corresponding date value) with nanosecond precision. 08:12:54.123456789
timestamp	integer, string	A timestamp (date and time) with millisecond precision. 2011-02-03 04:05+0000
tinyint	integer	8-bit signed int - ranging from -128 to 127
uuid	uuid	A UUID (of any version)
varchar	string	string
varint	integer	Arbitrary-precision integer - it has no minimum or maximum value

Collections

Cassandra supports three kinds of collections: maps, sets and lists. The types of those collections is defined by:

collection_type::= MAP '<' cql_type',' cql_type'>'

      | SET '<' cql_type '>'

      | LIST '<' cql_type'>'

Partition Key and Clustering Key

```
CREATE TABLE user (
        email          text,
        first_name        text,
        last_name         text,
        age          smallint,
        birth_city        text,
        birth_state        text,
        hobbies      list<text>,
        created_datetime       timestamp,
        PRIMARY KEY ((birth_state , birth_city), email)
);
```

List all tables in a Keyspace

SELECT table_name

FROM system_schema.tables

WHERE keyspace_name = 'simple_keyspace';

List All Columns for a Table and Keyspace

SELECT *

FROM system_schema.columns

WHERE keyspace_name = 'simple_keyspace'

AND table_name = 'user';

Insert record into table

INSERT INTO user (email, first_name, last_name, age, birth_city, birth_state, hobbies, created_datetime)
  VALUES ('saurav@gmail.com', 'Saurav', 'Samantray', 33, 'Bhubaneswar', 'Odisha', ['reading', 'cycling'], toUnixTimestamp(now())
);

Fetch Records from table

SELECT * from user;

SELECT * from user where email='saurav@gmail.com' AND birth_city='Bhubaneswar' AND birth_state='Odisha';

SELECT * from user where email='saurav@gmail.com';

SELECT * from user where email='saurav@gmail.com' ALLOW FILTERING;

Update Record

Without Partitioning key- Error
UPDATE user SET hobbies = ['Reading','Cycling','Programming'] WHERE email = 'saurav@gmail.com';

With partitioning key
UPDATE user SET first_name='Saurav2' WHERE birth_state='Odisha' AND birth_city='Bhubaneswar' AND email='saurav@gmail.com';

UPDATE user SET hobbies = ['Reading','Cycling','Programming'] WHERE birth_state='Odisha' AND birth_city='Bhubaneswar' AND email='saurav@gmail.com';

DELETE hobbies[0] FROM user WHERE birth_state='Odisha' AND birth_city='Bhubaneswar' AND email='saurav@gmail.com';

© Saurav Samantray

Deleting Record

DELETE FROM user where birth_state='Odisha' AND birth_city='Bhubaneswar' AND email='saurav@gmail.com';

Q and A