Course code	: **CSE3009**
Course title	: **No SQL Data Bases**
Module	: **5**
Topic	: **2**

Cassandra

Objectives

This session will give the knowledge about

- Cassandra

Cassandra History

Cassandra was first developed at Facebook for inbox search.

Facebook open sourced it in July 2008.

Apache incubator accepted Cassandra in March 2009.

Cassandra is a top level project of Apache since February 2010.

The latest version of Apache Cassandra is 3.11.5.

Cassandra Architecture

Cassandra stores data on different nodes with a peer to peer distributed fashion architecture.

All the nodes exchange information with each other using Gossip protocol.

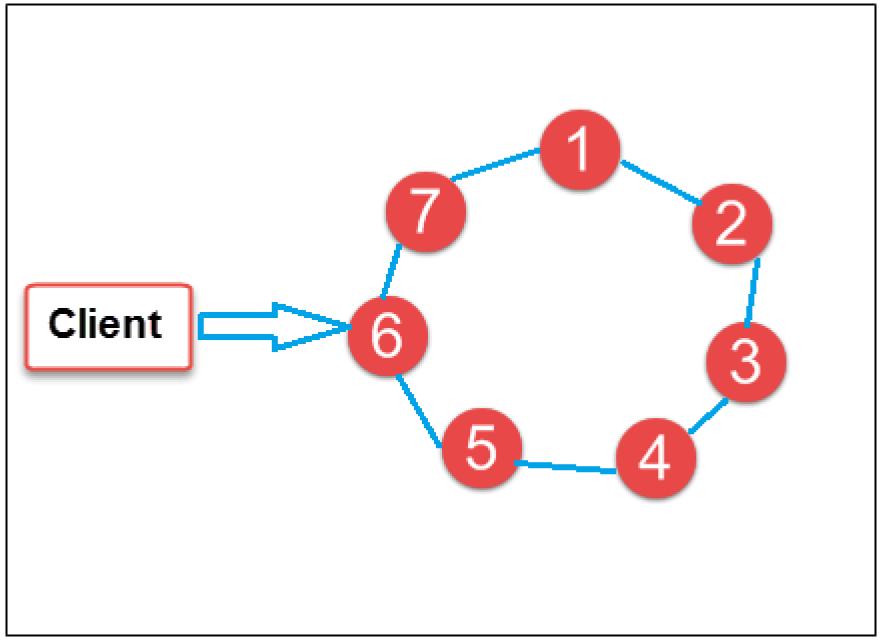
Gossip is a protocol in Cassandra by which nodes can communicate with each other.

Cassandra

Cassandra is a distributed database management system designed for handling a high volume of structured data across commodity servers.

Data is placed on different machines with more than one replication factor that provides high availability and no single point of failure.

In the Fig, circles are Cassandra nodes and lines between the circles shows distributed architecture, while the client is sending data to the node.

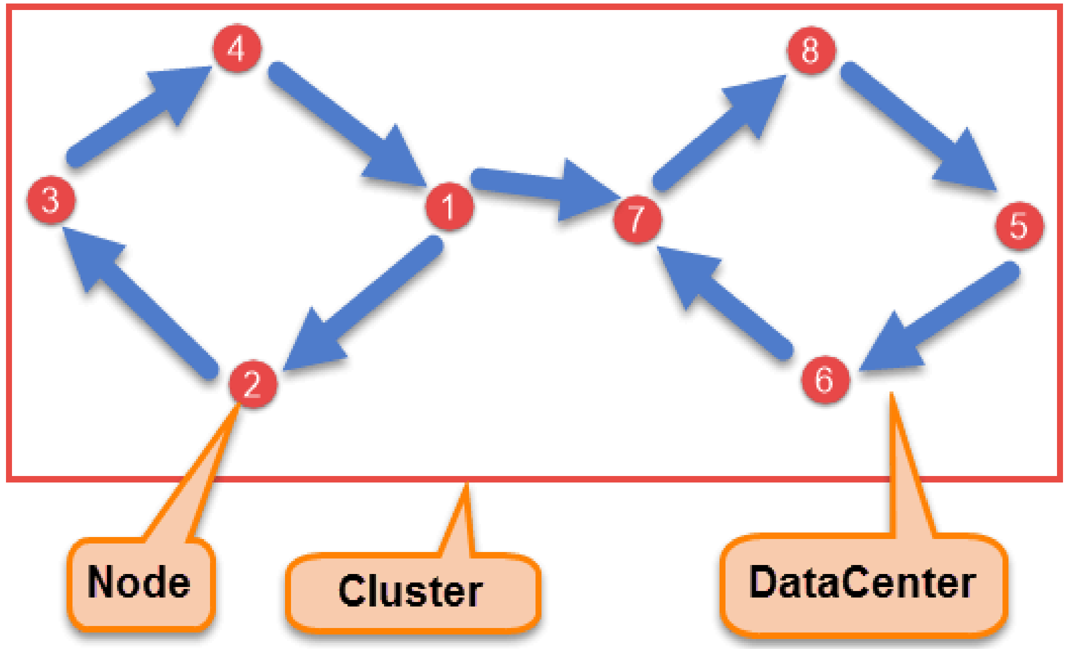


Components of Cassandra

Node: Node is the place where data is stored. It is the basic component of Cassandra.

Data Center: A collection of nodes are called data center. Many nodes are categorized as a data center.

Cluster: The cluster is the collection of many data centers.



Components of Cassandra

Commit Log: Every write operation is written to Commit Log. Commit log is used for crash recovery.

Mem-table: After data written in Commit log, data is written in Mem-table. Data is written in Mem-table temporarily.

SSTable: When Mem-table reaches a certain threshold, data is flushed to an SSTable disk file.

Bloom filter − These are quick, nondeterministic, algorithms for testing whether an element is a member of a set. It is a special kind of cache. Bloom filters are accessed after every query.

Data Replication

As hardware problem can occur or link can be down at any time during data process, a solution is required to provide a backup when the problem has occurred. So data is replicated for assuring no single point of failure.

Cassandra places replicas of data on different nodes based on these two factors.

- The place for next replica is determined by the Replication Strategy.
- The total number of replicas placed on different nodes is determined by the Replication Factor.

Data Replication

One Replication factor means that there is only a single copy of data while three replication factor means that there are three copies of the data on three different nodes.

For ensuring there is no single point of failure, replication factor must be three.

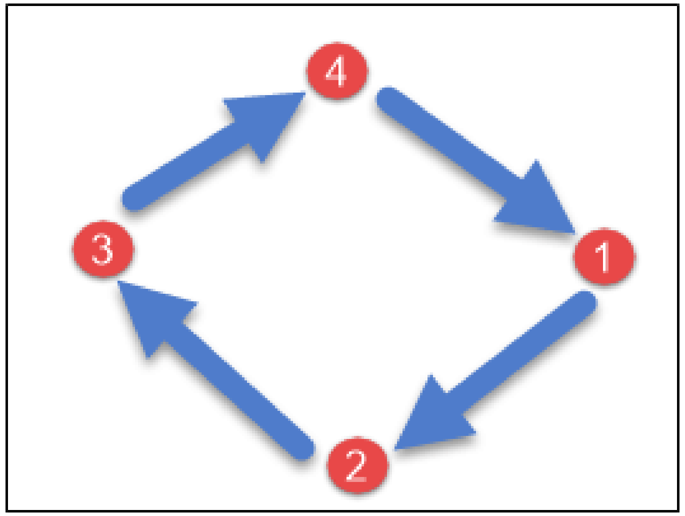
There are three kinds of replication strategies in Cassandra.

- simple strategy (rack-aware strategy),
- old network topology strategy (rack-aware strategy), and
- network topology strategy (datacenter-shared strategy)

SimpleStrategy

SimpleStrategy is used when you have just one data center.

SimpleStrategy places the first replica on the node selected by the partitioner. After that, remaining replicas are placed in clockwise direction in the Node ring.



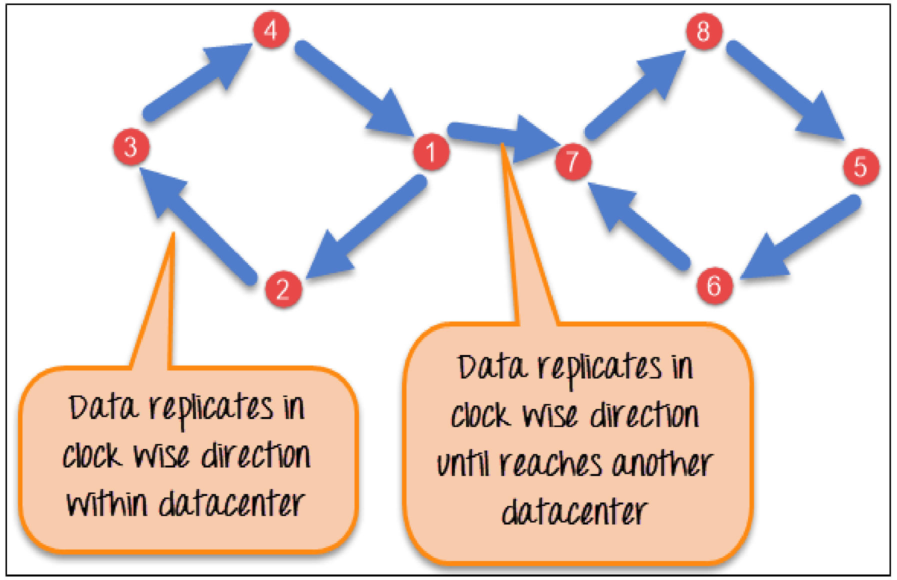
NetworkTopologyStrategy

NetworkTopologyStrategy is used when you have more than two data centers.

In NetworkTopologyStrategy, replicas are set for each data center separately. NetworkTopologyStrategy places replicas in the clockwise direction in the ring until reaches the first node in another rack.

This strategy tries to place replicas on different racks in the same data center.

Here is the pictorial representation of the Network topology strategy


4
8
3
1
7
5
2
6
Data replicates in clock wise direction within datacenter
Data replicates in clock wise direction until reaches another datacenter

Write Operation

The coordinator sends a write request to replicas. If all the replicas are up, they will receive write request regardless of their consistency level.

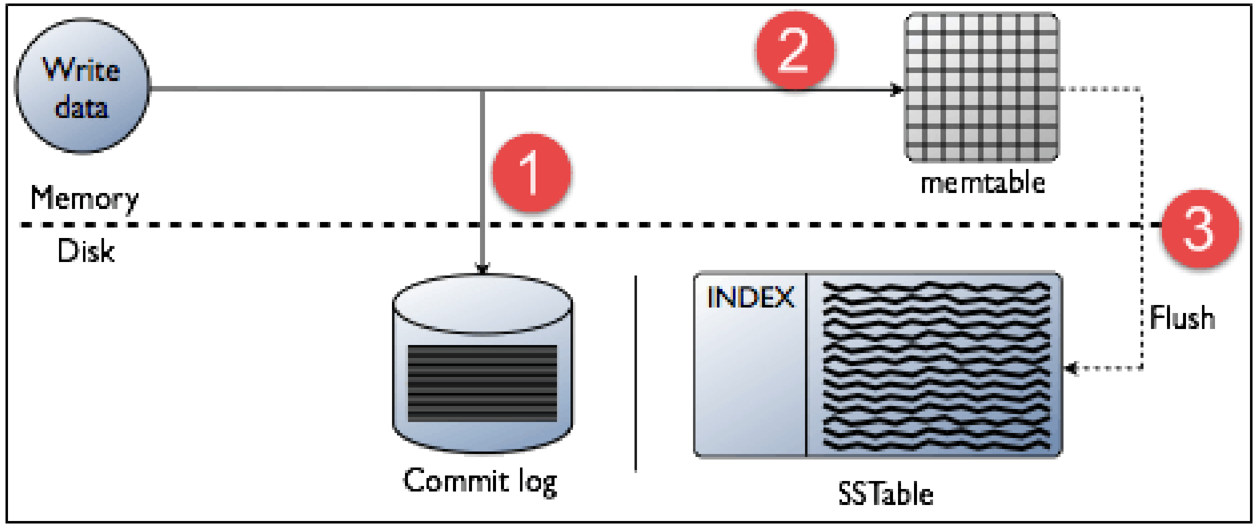
Consistency level determines how many nodes will respond back with the success acknowledgment.

The node will respond back with the success acknowledgment if data is written successfully to the commit log and memTable.

For example, in a single data center with replication factor equals to three, three replicas will receive write request. If consistency level is one, only one replica will respond back with the success acknowledgment, and the remaining two will remain dormant.

Suppose if remaining two replicas lose data due to node downs or some other problem, Cassandra will make the row consistent by the built-in repair mechanism in Cassandra.

Write Operation - Steps



1. When write request comes to the node, first of all, it logs in the commit log.
2. Then Cassandra writes the data in the mem-table. Data written in the mem-table on each write request also writes in commit log separately. Mem-table is a temporarily stored data in the memory while Commit log logs the transaction records for back up purposes.
3. When mem-table is full, data is flushed to the SSTable data file.

Read Operation

There are three types of read requests that a coordinator sends to replicas.

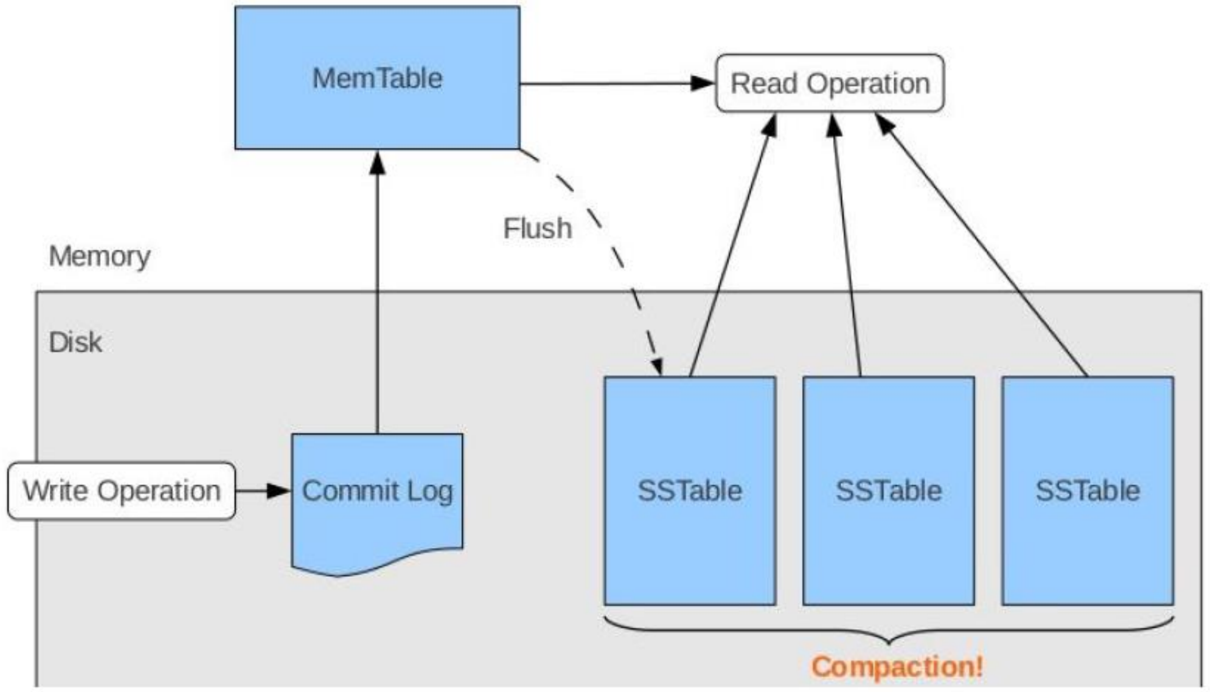
- Direct request
- Digest request
- Read repair request

The coordinator sends direct request to one of the replicas. After that, the coordinator sends the digest request to the number of replicas specified by the consistency level and checks whether the returned data is an updated data.

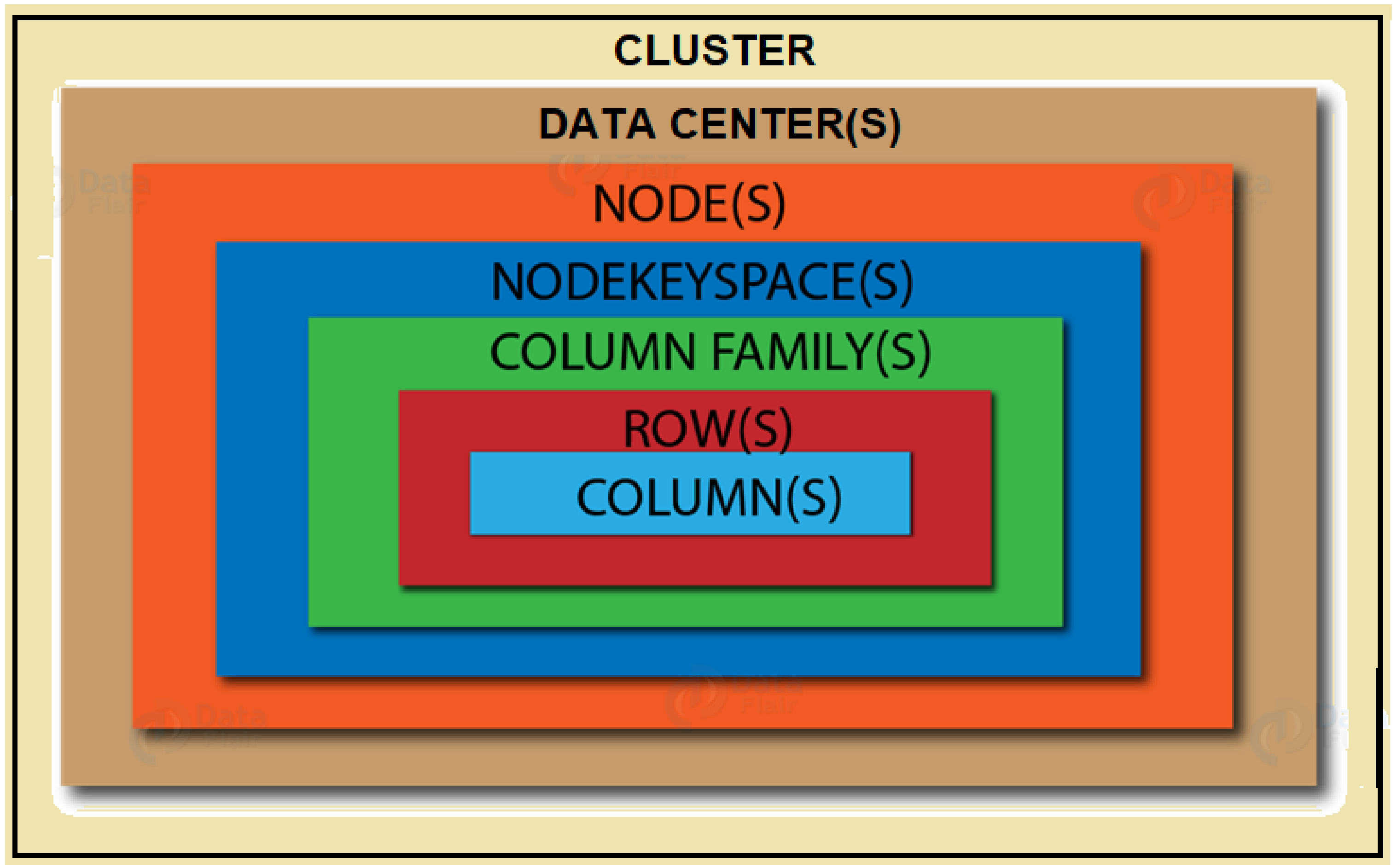
After that, the coordinator sends Read repair request to all the remaining replicas. If any node gives out of date value, a background read repair request will update that data. This process is called read repair mechanism.

Read Operation

During read operations, Cassandra gets values from the mem-table and checks the bloom filter to find the appropriate SSTable that holds the required data.



Cassandra - Data Model



Cassandra - Data Model

The data model of Cassandra is significantly different from what we normally see in an RDBMS.

Cluster

Cassandra database is distributed over several machines that operate together. The outermost container is known as the Cluster.

For failure handling, every node contains a replica, and in case of a failure, the replica takes charge. Cassandra arranges the nodes in a cluster, in a ring format, and assigns data to them.

Keyspace

Keyspace is the outermost container for data in Cassandra. The basic attributes of a Keyspace in Cassandra are:

Replication factor − It is the number of machines in the cluster that will receive copies of the same data.

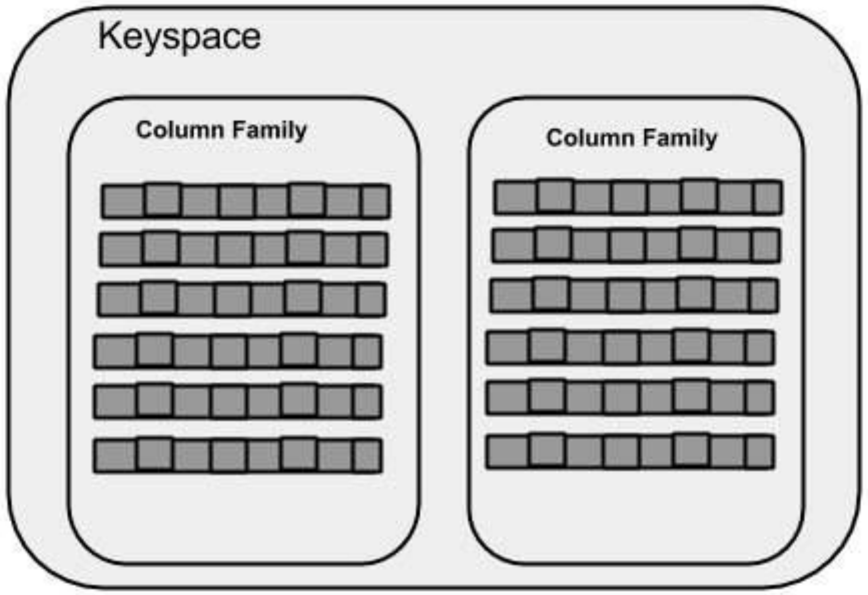
Replica placement strategy − It is the strategy to place replicas in the ring. We have strategies such as simple strategy (rack-aware strategy), old network topology strategy (rack-aware strategy), and network topology strategy (datacenter-shared strategy).

Column families − Keyspace is a container for a list of one or more column families. A column family, in turn, is a container of a collection of rows. Each row contains ordered columns. Column families represent the structure of your data. Each keyspace has at least one and often many column families.

Keyspace

The syntax of creating a Keyspace is as follows:

CREATE KEYSPACE Keyspace name WITH replication = {'class': 'SimpleStrategy', 'replication_factor' : 3};



Column Family

A column family is a container for an ordered collection of rows. Each row, in turn, is an ordered collection of columns. The following table lists the points that differentiate a column family from a table of relational databases.

A Cassandra column family has the following attributes −

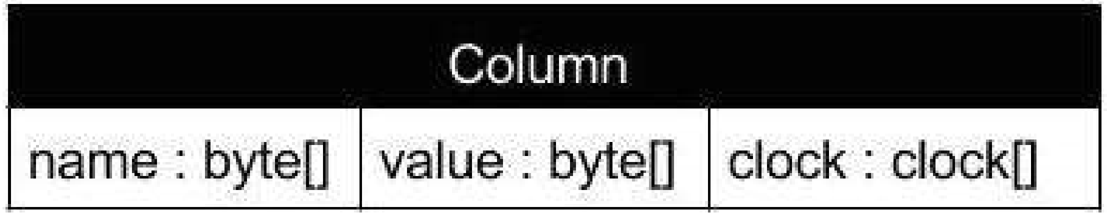
- keys_cached − It represents the number of locations to keep cached per SSTable.
- rows_cached − It represents the number of rows whose entire contents will be cached in memory.
- preload_row_cache − It specifies whether you want to pre-populate the row cache.

Note − Unlike relational tables where a column family's schema is not fixed, Cassandra does not force individual rows to have all the columns.

Column and Super Column

Column

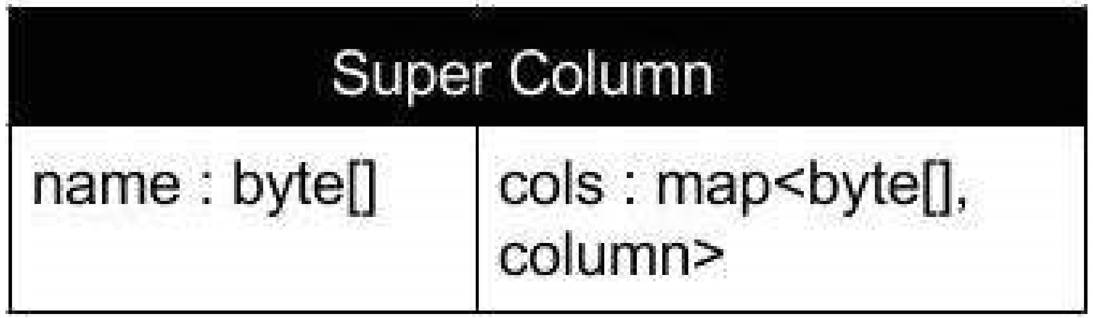
A column is the basic data structure of Cassandra with three values, namely key or column name, value, and a time stamp. Given below is the structure of a column.



Column		
name : byte[]	value : byte[]	clock : clock[]

SuperColumn

A super column is a special column, therefore, it is also a key-value pair. But a super column stores a map of sub-columns.



Super Column	
name : byte[]	cols : map<byte[], column>

Cassandra Features

- Massively Scalable Architecture: Cassandra has a masterless design where all nodes are at the same level which provides operational simplicity and easy scale out.
- Masterless Architecture: Data can be written and read on any node.
- Linear Scale Performance: As more nodes are added, the performance of Cassandra increases.
- No Single point of failure: Cassandra replicates data on different nodes that ensures no single point of failure.
- Fault Detection and Recovery: Failed nodes can easily be restored and recovered.

Cassandra Features

- Flexible and Dynamic Data Model: Supports datatypes with Fast writes and reads.
- Data Protection: Data is protected with commit log design and build in security like backup and restore mechanisms.
- Tunable Data Consistency: Support for strong data consistency across distributed architecture.
- Multi Data Center Replication: Cassandra provides feature to replicate data across multiple data center.
- Data Compression: Cassandra can compress up to 80% data without any overhead.
- Cassandra Query language: Cassandra provides query language that is similar like SQL language. It makes very easy for relational database developers moving from relational database to Cassandra.

Summary

This session will give the knowledge about

- Cassandra