Terminologies:

1. **Node :**
    - Node is a basic infrastructure. It is a fully functional machine that connects to other nodes in a cluster with high network speed.
    - The fully functional machines can be servers,ec2 instances, or any virtual machine.
    - All the nodes are connected or organized in ring network topology.
    - Every node works independently and has the same role in the topology. This arrangement of nodes is called a peer to peer structure.
    - Each node accepts read and write requests.
    - Each node is responsible for storing a portion of data and each node communicates with other nodes to ensure data consistency.
    - The number of nodes can be increased or decreased in a cluster to handle volume of traffic.
    - A node stores information of location of data centers, it keeps data like keyspace, table, schema.

2. **Racks :**
    - Rack is a logical group of nodes within the ring.
    - We can say a rack is a collection of servers.
    - A database uses rack to ensure that replicas are distributed among different logical nodes. As a result it sends operation not only to one node, but to multiple nodes.

3. **Data Centers**
    - A data center is a logical set of racks.
    - A data center should contain at least 1 rack.

- Data centers are used to provide fault tolerance, disaster recovery, and low latency access of data for users in different locations.
- Replication strategy is used to replicate data across data centers to ensure high availability of data.
- When we write any operation data is replicated to the number of nodes in the same data center and to a specified number of nodes in different data centers.
- This way it provides fault tolerance at data center level.

4. **Cluster:**
    - Cluster is a collection of nodes that works together to store and manage data.
    - A cluster can have one or more data centers within it.

5. **Seed node:**
    - A seed node is a node that is used to bootstrap new nodes into the cluster.
    - A new node is added to the cluster, it needs to discover the topology of the cluster and connect to other nodes in the cluster - This process is called bootstrapping.
    - It typically needs a list of initial nodes to connect.
    - Seed nodes play an important role in the bootstrapping process by providing necessary information to new nodes to join clusters and start participating in clusters.
    - Without a seed node, new nodes may not be able to discover other nodes in the cluster.
    - A seed node is a node that is designated in the configuration of the cluster to provide a list of initial nodes to new nodes joining the cluster.
    - A new node contacts the seed node to get the list of other nodes in the cluster.

6. **Gossip Protocol**
    - Gossip protocol is used for peer to peer communication.
    - Gossip informs a node about their state.
    - This protocol is used to communicate cluster state information between the nodes in the cluster.
    - Each node in the cluster periodically sends gossip messages to few other nodes in the cluster which contain information about itself and other nodes.
    - When a node receives a gossip message, it updates its view of the cluster based on information of the message.
    - If a node receives gossip about a new node, it contacts that node to learn about it and add it to the view of the cluster.
    - If a node receives a gossip that a node has failed, it marks the node down and propagates this information to other nodes in the cluster and starts raising the flag of severity level.
    - The nodes in the cluster gossip with each other and exchange information about states and update their view until they all have the same view.

Basic structure of tables in Cassandra

- Columns
    - Column is the smallest model in cassandra.
    - This column consists name,values and timestamp, time to live(ttl)
    - Time stamp is used to avoid conflict when any client makes write operations.
    - Time to live is a feature that allows you to set expiration time for data in a column when a time to live value is set for a column.

The data is automatically deleted from the family after the specified time period
    - TTL is used to manage the size of the column family and to remove outdated data automatically.
    - TTL value can be set at column or at row level.
    - If TTL value is set at column level, the value applies only to that specific column. If TTL value is zero of row level.
      The value applies to all the columns in that row.
- Rows:
    - Row consists of row_key, this is also known as primary key and set of column consists of row key
    - Each row may have different column name and there are no timestamp
    - A row must be stored on a node and cannot be separated in 2 or more nodes in a cluster.

- Column Family
    - Column family is a container of a set of rows. A row key in a column family must be unique to identify rows.
    - The column family can be an analogy to tables in relational databases.

- Keyspace
    - Keyspace is a namespace that defines replication factor or replication strategy
    - Keyspace is the highest level of organization in cassandra.
    - A keyspace is an analogous schema
    - A keyspace contains replication settings on how the data is replicated in the clusters. One cluster only contain one key space
- Tables
    - Tables in cassandra 3.x and later versions a table is a collection of rows with defined schema.

- A table is organized with a partition key which defines how data is distributed among the cluster and how data is defined within the partition.

7. **Partition**
    - Partitioning is a way through which we distribute the data across the nodes.
    - Each table has a primary key and the primary key is divided into partition key and clustering column. This clustering column is optional and there is no uniqueness constraint for any of the keys.
    - The partition key is used to index the data, all the rows which share a common partition key make a single data partition which is the basic unit of partitioning storage and retrieval of data.
    - A partitioning is a technique to distribute data across multiple nodes in a cluster.
    - The data is divided into partitions based on the partition key which is a part of the primary key of a table.
    - Each partition is stored on a specific node in a cluster based on a hashing algorithm.
    - When a query is made to retrieve data from the partition, Cassandra routes the query to the node that owns the partition. This allows …
    - A partition key is converted to a token by the partitioner.
    - Token values can range from -2^{63} to $(2^{63}-1)$

Replication

- Data in each keyspace is replicated with a replication factor and replication is a process of copying data across multiple nodes in a cluster to ensure data is available even if one node fails.

- There are many replication strategies
    - **Simple Strategy:**
      This replicates the data in the node across the same cluster.
    - **Network Topology Strategy:**
      Data is replicated across the node in different data centers.

Commit Log

- When data is written to cassandra, the data is stored in in-memory cache(MEM table) and also appended the same to commit log.

Mem-table

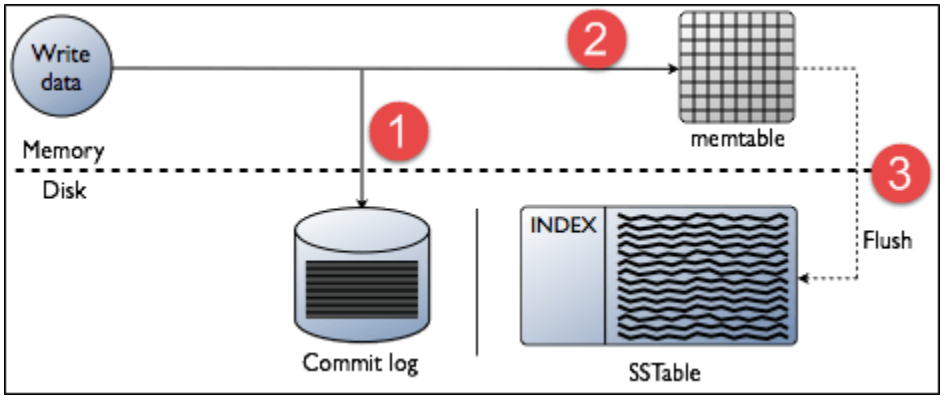
- MEM table is an in-memory cache that holds data row before they are flushed to SS table

SS Table

- SST table is a file that stores a set of immutable data rows or key values stored by row key.

Write Operation in Cassandra

1. When you write a request to a node first it will write to commit log.
2. Then cassandra writes to the MEM table in the memory table. Data is written in the MEM table on each write request. Also writes in commit log separately
3. MEM table is temporarily stored data in memory while commit log logs the transaction record for backup process.
4. When the MEM table is full data is flushed to SS table data file.



List of commands that **doesn't support** CQL

1. Aggregation queries like min,max,avg,sum,count
2. Group by and having queries
3. Joins
4. Or query
5. Wildcard
6. Union,intersection
7. Greater than and less than query
   This is only supported on clustering columns.

- This is the reason why cassandra is not suitable for analytical functions because there are so many limitations.
- Table column cannot be filtered without creating index

Default partitioner in Cassandra > **Murmur3Partitioner**

Commands:

describe cluster

This describes the current cluster of cassandra and information about the cluster

1. **CREATE KEYSPACE company**

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

This row level:

Insert into table_name(col_1,col_2,col_3) values(val1,val2,val3) using TTL 3600;

This is at column level:

Insert into table_name(col_1,col_2,col_3) values(val1,val2,val3) using TTL 3600 where col1=val1;

1. **USE company;**

1. **CREATE TABLE employee (**

employee_id int PRIMARY KEY,
first_name text,
last_name text,
email text,
phone text,
hire_date date,
salary float,
department text
);

```
CREATE TABLE orders (
    customer_id int,
    order_date timestamp,
    order_id uuid,
    product_name text,
    quantity int,
    PRIMARY KEY ((customer_id), order_date, order_id)
);
```

```
CREATE TABLE employees (
    department text,
```

```
    last_name text,
    first_name text,
    hire_date timestamp,
    PRIMARY KEY (department, hire_date, last_name, first_name)
) WITH CLUSTERING ORDER BY (hire_date DESC, last_name ASC, first_name ASC);
```

```
INSERT INTO employee (employee_id, first_name, last_name, email, phone, hire_date, salary, department) VALUES (1, 'John', 'Doe', 'john.doe@example.com', '555-1234', '2022-04-01', 50000, 'Sales');
INSERT INTO employee (employee_id, first_name, last_name, email, phone, hire_date, salary, department)VALUES (2, 'Jane', 'Smith', 'jane.smith@example.com', '555-5678', '2022-03-15', 60000, 'Engineering');

INSERT INTO employee (employee_id, first_name, last_name, email, phone, hire_date, salary, department) VALUES (3, 'Bob', 'Johnson', 'bob.johnson@example.com', '555-9876', '2022-02-01', 45000, 'Sales');

INSERT INTO employee (employee_id, first_name, last_name, email, phone, hire_date, salary, department) VALUES (4, 'Alice', 'Williams', 'alice.williams@example.com', '555-4321', '2022-01-01', 55000, 'Marketing');
```

```
INSERT INTO employees (department, last_name, first_name, hire_date)
VALUES ('Sales', 'Smith', 'John', '2022-01-01 10:00:00');

INSERT INTO employees (department, last_name, first_name, hire_date)
VALUES ('Sales', 'Johnson', 'Amy', '2022-01-02 11:00:00');

INSERT INTO employees (department, last_name, first_name, hire_date)
VALUES ('Sales', 'Garcia', 'Juan', '2022-01-03 12:00:00');

INSERT INTO employees (department, last_name, first_name, hire_date)
VALUES ('Marketing', 'Lee', 'Jennifer', '2022-01-04 13:00:00');

INSERT INTO employees (department, last_name, first_name, hire_date)
VALUES ('Marketing', 'Wong', 'Alice', '2022-01-06 15:00:00');
```

```
INSERT INTO employees (department, last_name, first_name, hire_date)
VALUES ('HR', 'Chen', 'Wei', '2022-01-07 16:00:00');

 INSERT INTO employees (department, last_name, first_name, hire_date)
VALUES ('IT', 'Park', 'Ji-hoon', '2022-01-10 19:00:00');


select * from employee where employee_id = 3;

SELECT * FROM employees WHERE department = 'Sales' ORDER BY hire_date DESC,
last_name ASC, first_name ASC;
```
