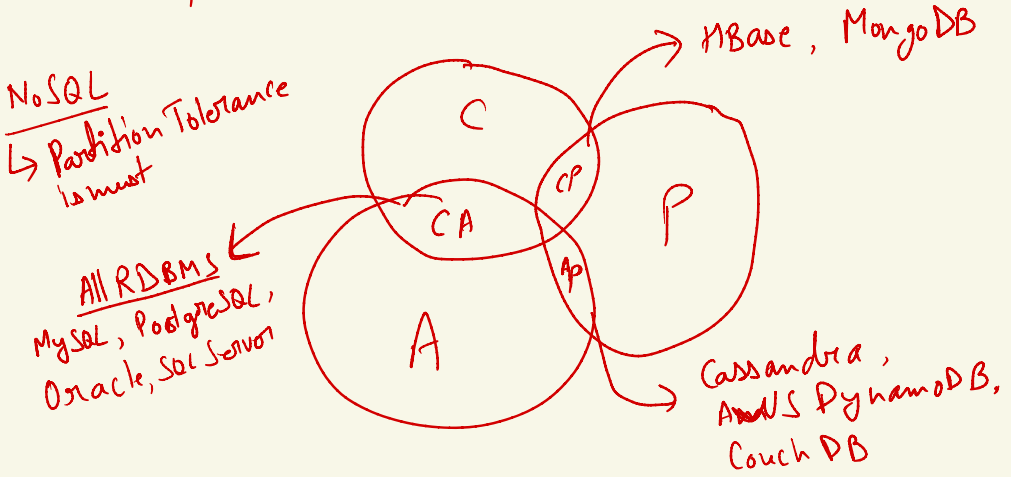
CAP Theorem → We can only achieve two properties from a database.

C → Consistency

A → Availability

P → Partition Tolerance



Bro (1)

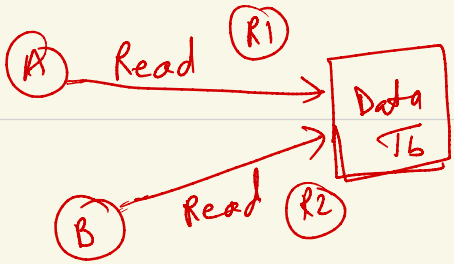
A 500 → INR 100 → B 400 → Current Consistent state

(2)

A 400 → INR 100 X (- - - -) B 400 → Inconsistent state

(3)

A 500 → 100 INR → B 400 → again Consistency



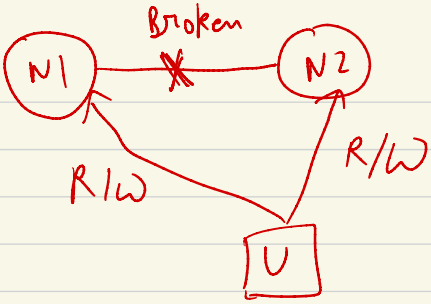
Availability

↳ our system should be up 24*7

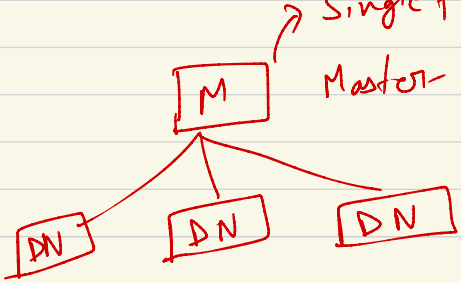
↳ If we have single point of failure then it is hard to achieve

Partition-tolerance

If there is network break in the system still it should operate and serve request.



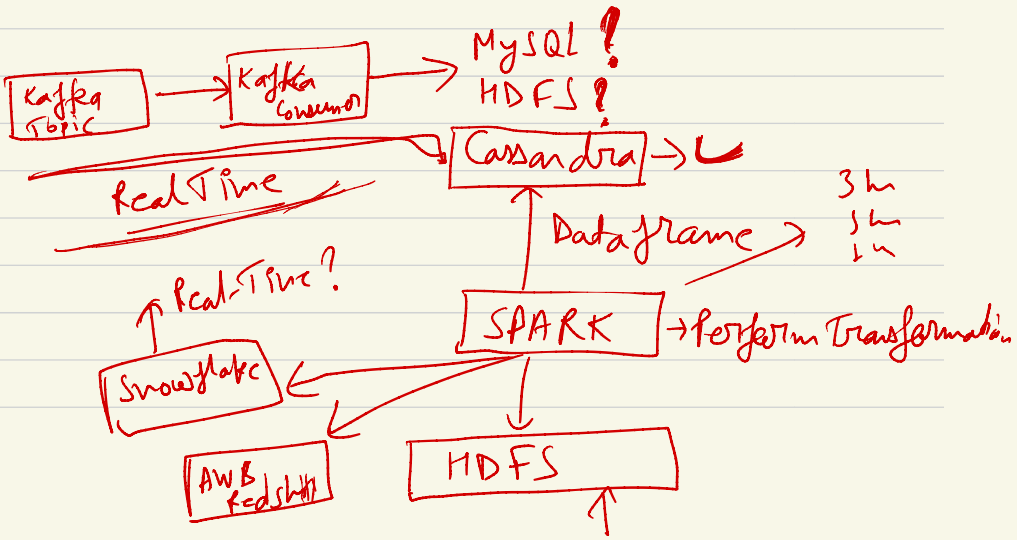
HBase



Cassandra

- It was developed by facebook
- It came as a open source project under Apache
- 3.0+ , 3.25 version
- Cassandra is not made for Analytical queries
- we can not perform joins, aggregation etc.
- It is beneficial for faster write operation
- Also beneficial for faster read search type queries
- Cassandra follow Ring Kind of Arch.

Real Time Data Pipeline (High Velocity)



Hive

HDFS → storage

hdd → HADOOP → Distributed computation

→ SPARK

C++, Java, Python, Scala

SPARK ⊂ HDFS → Storage file systems

SPARK → framework

DCE

Databases → RDBMS (Transactional)

Databases → NOSQL

Messaging-Queue → KAFKA

Data Warehousing → Means storing transformed structured Data!

↳ Hive, AWS Redshift, Snowflake, Azure Data Factory

Distributed Computation Engine

- Process data in distributed fashion on parallel processing
- HADOOP
- SPARK
- Flink

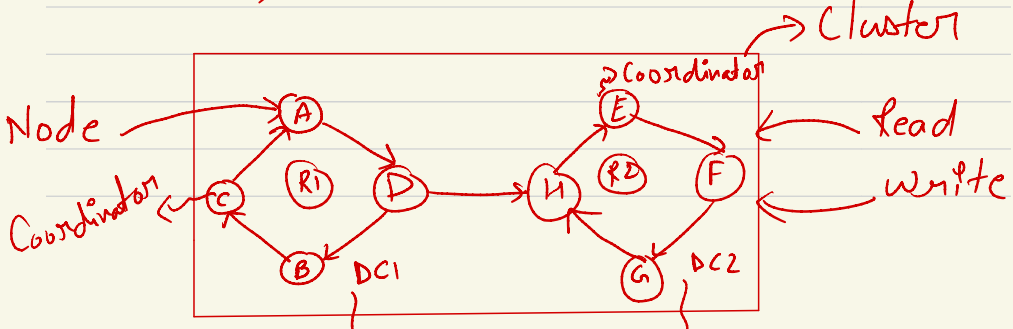
HADOOP

- HDFS → Storage
- MAP-Reduce → Processing
- YARN → Resource Management

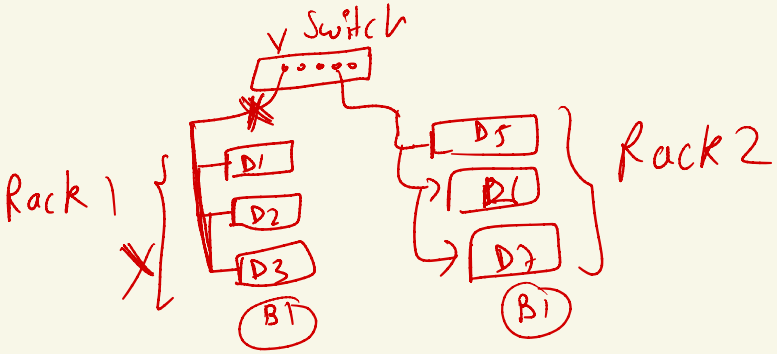
Datalake

- Means storing Raw Data (structured, un-structured, semi-s)
- HDFS, AWS S3, Azure Blob, GCP storage

Cassandra Architecture (Ring)



Data Center Data Center



Rack Awareness

* **Node** → where data stored

Cluster → Combination of Nodes

Data Center → Large location to Contain CNodes

Coordinator → Handles current Read/Write request

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 will be written in Mem-Table. Data is written in Mem-Table Temporarily.

SSTable ⇒ When mem-table reaches a certain threshold, data is flushed to an an SSTable disk.

Data Replication in Cassandra

↳ Means making multiple copy of Data and store it on different nodes for fault tolerance.

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

① where to place next replica is determined by replication strategy.

② While the total number of replicas placed on different nodes is determined by Replication factor.

Replication Factor = How many copies of same data will be created?

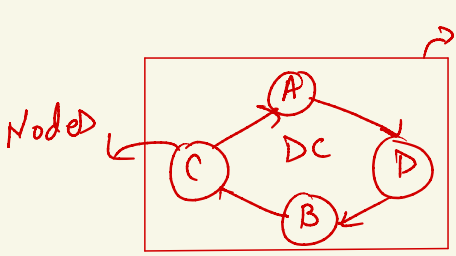
Two type of Replication Strategy

① Simple Strategy

② Network Topology Strategy

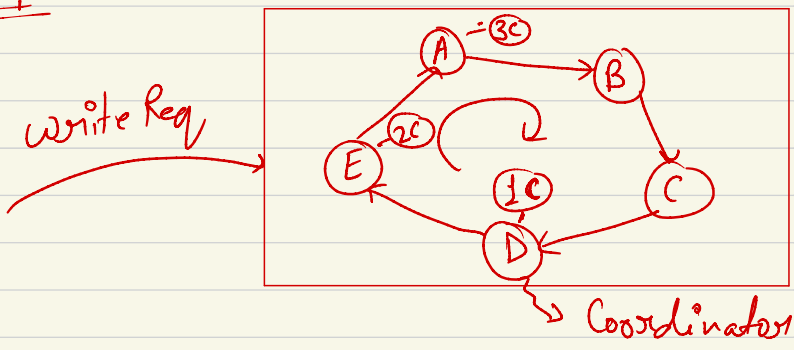
* Simple Strategy

↳ it will be used when we will one Data Center.



↳ first copy will be created on the coordinator Node and after that remaining copies will be created in other nodes in clock-wise direction

Example : Replication factor = 3 , it will be done on D → E → A



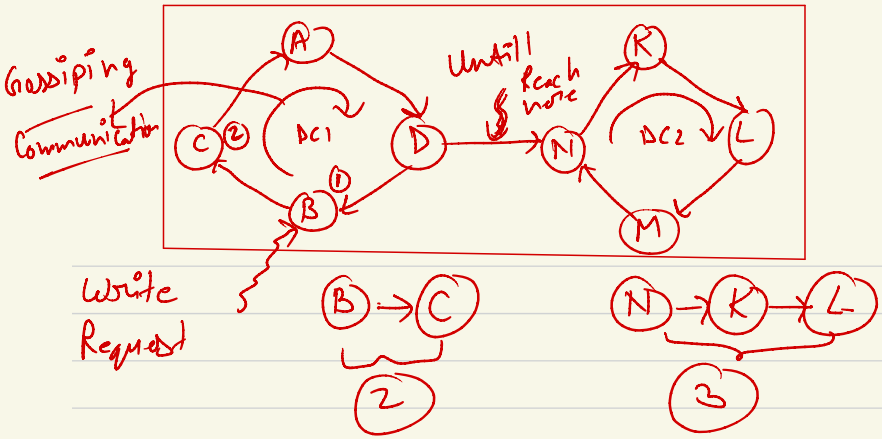
⊛ NetworkTopology Strategy

↳ It is used when we have more than one data Centers.

↳ Replicas are set for each data center separately

↳ Network Topology places replicas in the clockwise direction in the ring untill reaches the first node in another Data center.

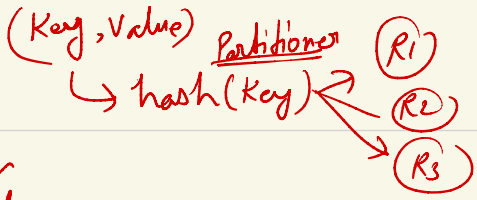
Replication Factor for DC1 = 2, for DC2 = 3



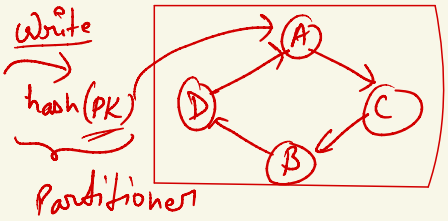
⑧ Concept of Keys in Cassandra

↳ Primary Key
↳ Partition Key
↳ Composite Partition Key
↳ Clustering Key

* Partition Key:



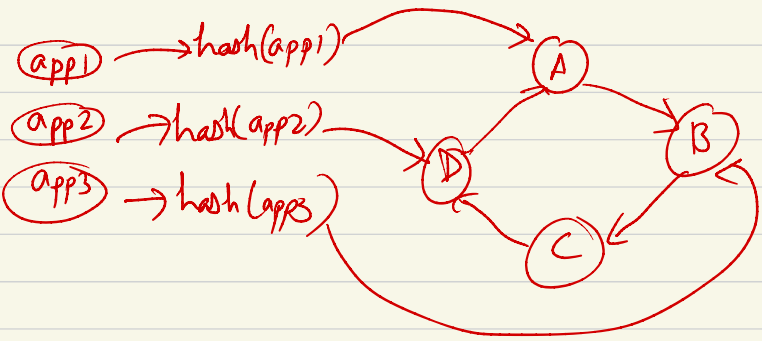
- A partition key is for data placement apart from uniquely identify a record.
- Partition key will decide on which node Data will be stored.



Ex:

```
Create table application_logs
(   id int,
    app_name varchar,
    env varchar,
    hostname varchar,
    log_datetime Timestamp,
    log_message Text,
    PRIMARY KEY (app_name)
);
```

Partition Key



⑧ Composite Partition Key

↳ If we need to combine more than one column value to form a single partition key that will be known as Composite Partition Key

exam

```
Create table application_logs
(
    id,
    app_name,
    host_name,
    env,
    log_message,
    PRIMARY KEY ((app_name, env))
)
```

(app_name, env) — Partition Key

* Clustering Key

↳ The columns which will be used to order the data inside a partition.

Ex

```
Create table app_logs
(
    id,
    app_name,
    host_name,
    log_datetime,
    env,
```

log_message,

PRIMARY KEY((app_name, env), hostname, log_datetime)

);

Partition Key — Clustering Key

OR if appname is only Partition Key

PRIMARY KEY(app_name, hostname, log_datetime)

Partition Key — Clustering Key

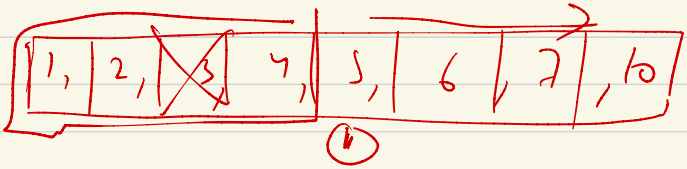
Partition

app_name	env	log_message	host_name	log_datetime
appl	Prod	appIIFO	host1	2022-11-20 16:04
appl	Prod	appIIFO2	host2	2022-11-20 16:20

Q? Why id is helpful?

so for search or filter related queries it can optimize.

Key = 7



Q? how to change the order for clustering?

Create table app_logs

(

id,

```
    app_name,
    host_name,
    env,
    log_datetime,
    log_message,
    primary key ((app_name, env), host_name, log_datetime)
) with clustering order by (hostname ASC, log_datetime DESC);
```

* Primary Key
  - ↳ It consists of one or more partition key and zero or more clustering key.