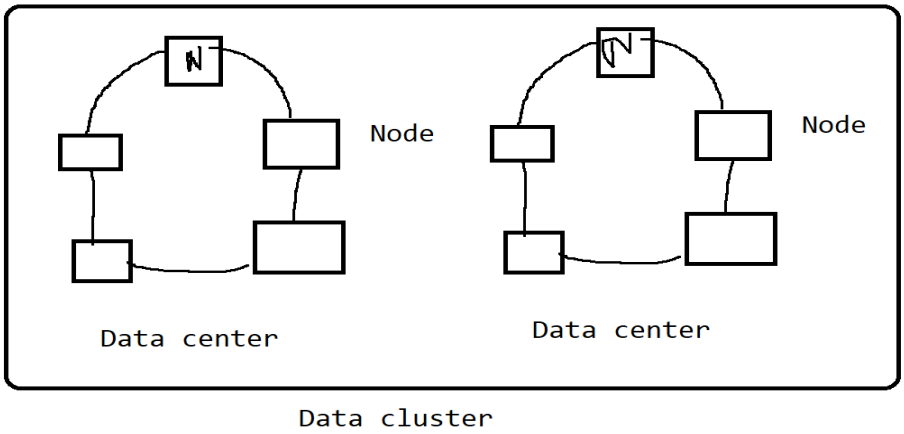
SQL	NO SQL
It supports ACID property	It support CAP principal
It is hostcentric	It is distributed
It doesnot support sharding	Sharding is possible
It uses vertical scaling	It uses horizontal scaling
Less available as compared to no sql	Highly available, because of replica fascility
It is slower as compared to no sql	It is faster as compared to sql
Usually stores data in row oriented form, which is good for OLTP	Usually stores data in column oriented form or document or key-value pair which is good for OLAP

Cassandra

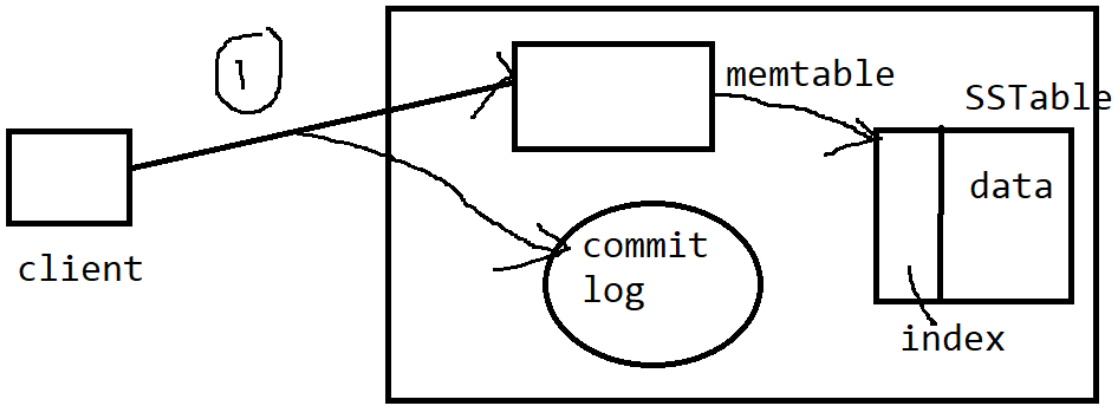
1. highly scalable
2. highly available , because of replica
3. Cassandra supports CAP principle
4. It is structured as well as unstructured
5. it stores data in column oriented manner



Terminology

1. Data center→ collection of multiple nodes connected to each other in ring, is called data center, to communicate with each other it uses a algorithm called as gossiping
2. Node-→ It is a machine ,which stores the data
3. Cluster-→ Collection of data centre is called as cluster
4. Mem-table-> it is a table which resides in RAMAfter making the entry in commit log the data will be written in mem table, there can be more than one mem table
5. SSTABLE(sorted string table)-→It is a disk filein which the data is stored when mem table will reach to a threshold value

Write operation



In Cassandra every table should have primary key.

primary key is formed by 2 parts

1. partition key
    a. partition key is used to find the node in the data centre, to understand where the data is stored.
2. cluster key
    a. cluster key -→ used to find data within the node found by partition key

    In Cassandra there are 3 types of read request
    1. direct read request
        a. when client wants to read data if partition key is given, then based on the key, the node on which the data is stored will be found by coordinator server and then the direct request will be send to the node for retrieval of data, this node returns full data
    2. digest read request
        If the consistency level given is n, specifies that there are n replicas, so the digest request will be send to all replicas, they return hashcode generated by the data in the row to the coordibnator node, coordinator node will calculate hashcode and check if all hashcode matches, if it matches means the data is up to date. otherwise it sends the rea repair request to the replica node whose hashcode deos not match

    3. read repair request -→ read repair request will make the entry up-to date.

It also has a replica placement strategy

1. simple strategy--→I we are working with one data center, with 1 copy of data, then it is called as simple strategy.
2. Network topology---→
    a. if we are working with more data centers, or clusters, then we use network topology, replication factor can be decided based on number of nodes in data center.

b. replication happens in clockwise direction.

3. Cassandra does not support joins, aggregate functions, group by, and hence lot of redundancy of data is there, it does not support normalization

to store the data in Cassandra we need to create a keyspace

to create a key space

CREATE KEYSPACE iacsd0324 WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'};

CREATE KEYSPACE iacsd0324 WITH replication = {'class': 'NetworTopologyStartegy', 'DC1':1,'DC2':3} and durable_writes=false;

By default durable_writes=true; which means that write in the commit log;

to change the keyspace

use iacsd0324;

to create a table

create table customer(cno int primary key,

cname text,

mobile text);

to see the data from the table

select * from customer;

to insert data in the table

insert into customer(cno,cname,mobile) values(1,'rajesh','3333');

insert into customer(cno,cname,mobile) values(2,'Nitin','44444');

In customer table cno is a partition key

select * from customer where cno=1 ----- right query

select * from customer where cno>1 ------ wrong query

partiotion key specifies the location but > 1 means multiple keys, hence to search data we need to search entire database, so we need to add allow filtering.

select * from customer where cno>1 allow filtering

To see all keyspaces

SELECT * FROM system_schema. keyspaces;

1. Get keyspaces info
2. SELECT * FROM system_schema.keyspaces; Get tables info

   SELECT columnfamily_name FROM system_schema.columnfamilies WHERE keyspace_name = 'keyspace name';

3. Get table info

   SELECT column_name, type, validator FROM system.schema_columns WHERE keyspace_name = 'keyspace name' AND columnfamily_name = 'table name';

   cqlsh> SELECT * FROM system.schema_columns WHERE keyspace_name = 'cycling' AND columnfamily_name = 'cyclist_name';

Data types

CQL Type	Constants	Description
ascii	Strings	US-ascii character string
bigint	Integers	64-bit signed long
blob	blobs	Arbitrary bytes in hexadecimal
boolean	Booleans	True or False
counter	Integers	Distributed counter values 64 bit
decimal	Integers, Floats	Variable precision decimal
double	Integers, Floats	64-bit floating point
float	Integers, Floats	32-bit floating point
frozen	Tuples, collections, user defined types	stores cassandra types
inet	Strings	IP address in ipv4 or ipv6 format
int	Integers	32 bit signed integer
list	Duplicate values are allowed , represented in []	Collection of elements
map	Keys are unique, represented in {}	JSON style collection of elements
set	Only unique values are allowed,{}	Collection of elements
text	strings	UTF-8 encoded strings
timestamp	Integers, Strings	ID generated with date plus time

uuid	uuids	Type 1 uuid
tuple	Read only and fixed size	A group of 2,3 fields
timeuuid	uuids	Standard uuid
varchar	strings	UTF-8 encoded string
varint	Integers	Arbitrary precision integer

CQL type compatibility

CQL data types have strict requirements for conversion compatibility. The following table shows the allowed alterations for data types:

Data type may be altered to:	Data type
ascii, bigint, boolean, decimal, double, float, inet, int, timestamp, timeuuid, uuid, varchar, varint	blob
int	varint
text	varchar
timeuuid	uuid
varchar	text

Clustering columns have even stricter requirements, because clustering columns mandate the order in which data is written to disk. The following table shows the allow alterations for data types used in clustering columns:

Data type may be altered to:	Data type
int	varint
text	varchar
varchar	text

create employee table

```
create table employee(
empid int,
emp_firstname text,
emp_lastname text,
emp_sal decimal,
emp_dob date,
emp_deptno int,
```

emp_comm float,

primary key((empid,emp_sal),emp_firstname,emp_dob))

In employee table empid+emp_salary is a partion key and emp_firstname+emp_lastname will be the cluster key

insert into employee(empid, emp_sal ,emp_firstname,emp_lastname, ,emp_dob,emp_deptno,emp_comm) values(12,1200,'Kishori','Khadilkar','2000-11-11',10,345);

Rules for where clause

1. You have to use all partition key columns with = operator, other wise to scan entire database use allow filtering.
   Select * from employee where empid=13; ------error
   Select * from employee where empid=13 and emp_salary=2000; ------right

2. Use cluster columns in the same sequence in which it is created.
   Select * from employee where empid=13 and emp_salary=2000 and emp_dob='2000-11-11';--------wrong, because we are skipping emp_firstname

   Select * from employee where empid=13 and emp_salary=2000 and emp_dob='2000-11-11' and emp_firstname='Rajan'; ------ right

   Select * from employee where empid=13 and emp_salary=2000 and emp_firstname='Rajan';-------right

   Select * from employee where empid=13 and emp_salary=2000 and emp_firstname='Rajan' and emp_dob='2000-11-11';-----right

3. You cannot use = operator only on non primary key columns without partition key and if you want to use it then add allow filtering

Select * from employee where emp_comm=345------ error

Select * from employee where emp_comm=345 ALLOW FILTERING.----right

4. You cannot use = operator on only cluster key columns without partition key
   Select * from employee where emp_firstname='Rajan' and emp_dob='2000-11-11';-----wrong

   Select * from employee where empid=13 and emp_salary=2000 and emp_firstname='Rajan' and emp_dob='2000-11-11';-----right

   Select * from employee where emp_firstname='Rajan' and emp_dob='2000-11-11' allow filtering ------right

5. In operator is allowed on all the columns of partition key but it slows the performance

Select * from employee where empid in (12,13) and emp_salary in (1200,2000); ---right
Select * from employee where empid in (12,13) ; ----wrong
Select * from employee where emp_salary in (1200,1110);----wrong

6. > , <, <=,>= operators are not allowed on partition key

Select * from emp where empid = 1100 and emp_salary = 1200

7. > , <, <=,>= operators can be used only on cluster key columns, and partition key columns

Select * from emp

Where empid=100 and emp_salary=1002 and emp_firstname=”xxxx” and emp_dob=’1999-12-11’ and emp_comm>1; ----wrong

Select * from employee

Where empid=12 and emp_salary=1110 and emp_firstname='kishori' and emp_dob>'1998-12-11';

8. Order by can used only on cluster key but where clause should have equality condition based on partition key;

select * from employee where empid=13 and emp_salary=1110 order by emp_firstname desc,emp_dob desc;

select * from employee where empid=13 and emp_salary=1110 order by emp_firstname desc;

select * from employee where empid=13 and emp_salary=1110 order by emp_dob desc;---wrong