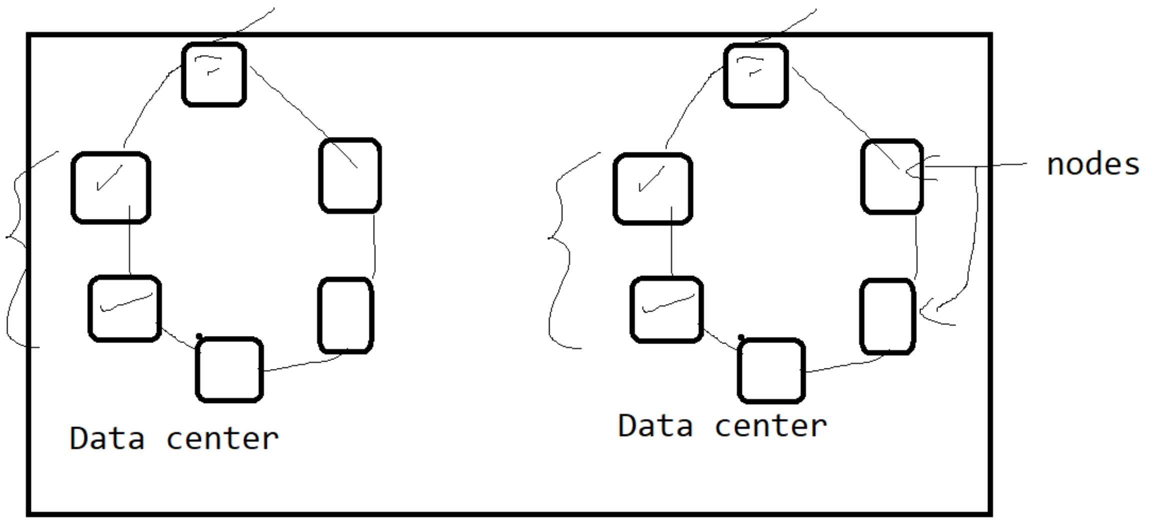
Features

- Highly scalable
- Highly available----→ fault tolerant
- Cassandra support CAP theorem
- It is structured as well as unstructured database.
- It stores the data in column-oriented manner.

Cluster

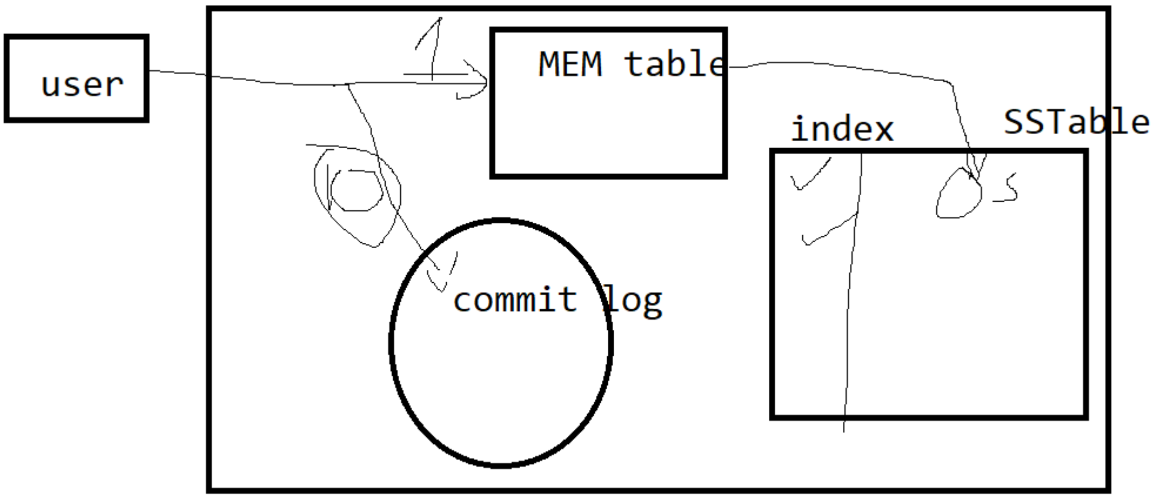


-

Components in Cassandra

- Data center-→ collection of nodes
- Node-→ it is a machine which stored the data
- Cluster-→collection of data centers
- Mem table---→ it is table which resides in RAM. After making entry in the commit log the data will be written in memtable, there can be more than one memtable
- SSTAble (sorted string table)---It is a disk file, in which data will be stored when mem table will reach to threshold value.

Write operation performed.



In Cassandra every table should have a primary key

Primary key is formed by 2 parts

1. Partition key and cluster key
    1. In the table below since the primary key contains only one field, it is treated as partition key

    Create table emp(empno int primary key,ename text)

    2. If the tale has composite key as shown below then the first column will be the partition key and all remaining columns will be cluster key
    Partion key -→ sid
    Cluster key-→ cname
    Create table stud_marks(
    Sid int,
    cname text,
    marks int,
    primary key(sid,cname))
    3. If the tale has composite key as shown below, then the first column will be the partition key and all remaining columns will be cluster key
    partition key-→ rid
    cluster key-→ rname, bk_date
    Partion key -→ sid
    Cluster key-→ cname
    Create table roombooking(
    Roomno int,
    Rname text
    Bk_date date
    Amount double(10,2)
    primary key(rid, rname,bkdate,))
    4. But what if you want to add more than one column as partition key

Then while defining primary add the partition key in one more bracket
partition key-→ rid, rname
cluster key --→ bk_date

Create table roombooking(

Roomno int,

Rname text

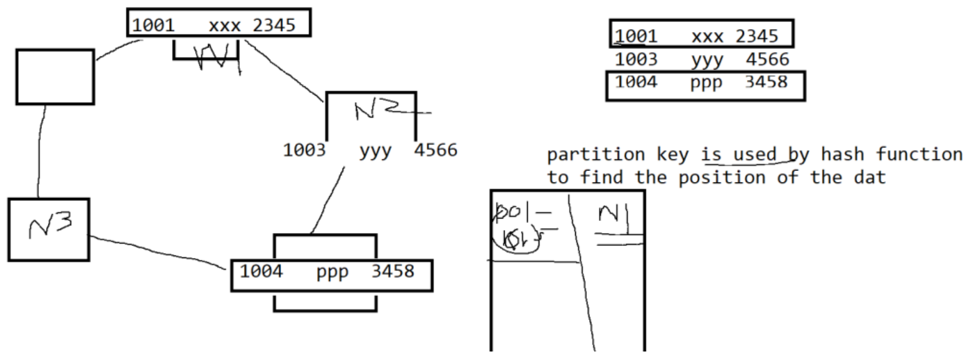
Bk_date date

Amount double(10,2)

primary key((rid, rname),bkdate,))

Partiotion key--→ helps to locate the node on which the data is stored.

Cluster key---→ helps to search data on the node faster, the data on the node is always stored in sorted order on cluster key.



Read operation

1. Direct read
2. Digest read
3. Read repair request

Replication factor--→Number of nodes in the data centre, on which the copy should be made, is called as replication factor.

Replica placement strategy

1. Simple strategy-→
   a. If we are working with the single data centre., then we will use this strategy
2. Network topology
   a. If we are working with 2 data centers then we use network strategy, replication factor will be decided based on number of nodes in one

data center, it will keep on replication data till it reaches to next data center.

b. Replication happens in clockwise direction.

Cassandra does not support joins, group by, aggregate functions.

In Cassandra to save the data we need to create keyspace, Keyspace are similar to database

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
timeuuid	uuids	Type 1 uuid

tuple	Read only and fixed size	A group of 2,3 fields
uuid	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

To create key space

CREATE KEYSPACE acts0923 WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'};

CREATE KEYSPACE acts0923 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 acts0923;

To create table

```
Create table customer( cno int primary key,
                       cname text,
                       mobile text);
```

to insert data into customer data

```
insert into customer(cno,cname,mobile) values (1,'Rajesh','333333');
insert into customer(cno,cname,mobile) values (2,'Rutuja','333333');
```

to see the table data

```
select * from customer;
```

In customer table cno is partition key, hence only used with = operator

Select * from customer where cno=1; ------right query

Select * from customer where cno> 1 ---→ wrong as it will try to find the node number where the data is stored.

Hence if you want to scan entire database, then use allow filtering.

```
select * from customer where cno>1 allow filtering;
```

1. To create an employee table, with empid, emp_salary as partition key
   And emp_firstname,emp_dob as cluster key

   ```
   Create table employee(
       Empid int,
   emp_salary int,
   emp_lastname varchar,
   emp_firstname text,
   emp_dob date ,
   emp_deptno int,
   emp_comm float,
   primary key((empid,emp_salary),emp_firstname,emp_dob)
   );
   ```

   To see the structure of the table
   Desc employee

   To alter the table
   Alter table employee add manager varchar

To modify the data
Alter table employee alter manager text

To drop the column
Alter table employee drop manager;

To delete the table
Drop table employee

Truncate the table
Truncate table employee;

Retrieval of data
Select * from employee;

To insert data
insert into
employee(empid,emp_salary,emp_lastname,emp_firstname,emp_dob,emp_
deptno,
emp_comm) values(12,1200, ‘Khadilkar’,’Kishori’,’2000-11-11’,10,345);

insert into
employee(empid,emp_salary,emp_lastname,emp_firstname,emp_dob,emp_
deptno,
emp_comm) values(12,1200, 'vaze', 'Amruta','2000-11-11',10,345);
insert into employee(empid,
emp_salary,emp_lastname,emp_firstname,emp_dob,emp_deptno,
... emp_comm) values(13,2000, 'Khadilkar','Rajan','2015-11-
11',10,345);

insert into
employee(empid,emp_salary,emp_lastname,emp_firstname,emp_dob,emp_
deptno,
... emp_comm) values(14,1500, 'kulkarni','Revati','2012-11-
11',20,500);

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

---

```
Create table customer( cno int primary key,
                       cname text,
                       mobile text);
```

to update data

update customer

set mobile='333333'

where cno=1;

to delete the data

delete from customer where cno=1;

delete mobile from customer where cno=1;

Data type

Collection

1. Set ---
    a. stores unique values
    b. Represent in {}
    c. Mutable

        d. No indexing is possible
2. List
3. Map
4. Tuple

Create table student(

  Id int primary key,

 Name text,

Courses set<text>);

1. Insert into student(id,name,courses) values(123,'Rajas',{'Python','Perl','PHP'});

2. To overwrite existing courses

Update student

Set courses={‘a’,’b’,’c’,’d’}

Where id=123

3. To add in existing courses

Update student

Set courses=courses+{‘a’,’b’,’c’,’d’}

Where id=131sdfj

1. To delete from the set

Update student

Set courses=courses-{python}

Where id=131sdfj

2. To remove all elements from the set

   Update student

   Set courses={}

   Where id=131sdfj

   Delete courses from student where id=131sdfj

3. List
   a. Uses []
   b. Duplicates are allowed.
   c. Ordered collection, indexing is possible
   d. Mutable

   Create table emp11(

Id int primary key,

Name text,

Companies list<text>)

1. To insert data

Insert into employee(id,name,companies)
values(11,’Rajan’,[‘cognizant’,’cabgemin’,’Tech-M’])

2. Update list add at the beginning
   Update employee set companies=[‘Wipro’]+companies where id=1

3. Update list add at the end
   Update employee set companies=companies+[‘Wipro’] where id=1

4. Update list at particular location, value will be overwritten
   Update emp11 set companies[2]= ‘Wipro’ where id=1

5. to remove the element from particular position
   delete companies[2] from employee where id=1;

4. Map ------
   a. Store data in key value format
   b. Keys should be unique
   c. Data is stored in {}
   d. Values can bw retrieved by using keys.

Create table empdata(id int, name text, companies map<text,int>)

1. Insert into empdata(id,name,companies)
   values(111,’xxx’,{‘cognizant’:5,’igate’:3})
2. Update companies to add new key value pair
   Update empdata set companies=companies+{‘acceture’:5}
3. Update value of a particular key in maps
   Update empdata set companies[‘igate’]=5 where id=111;
4. Delete a particular key value pair
   Delete companies[‘igate’] from employee where id=111;
5. Delete multiple keys also
   Update set companies=companies-{‘cognizant’,’igate’} where id=111;

Tuple

1. Duplicates are allowed
2. Uses()
3. Ordered collection,Indexing is possible
4. Tuples are immutable

Create table studdata(id int primary key,name text, marks tuple<int,text,int>);

insert into studdata(id,name,marks)values(1,'a',(1,'a',1))

to update marks for id 1

update studdata set marks=(1,’xx’,2);

Create table trainee(id int primary key,name text, marks tuple<int,text,tuple<int,int,int>>);

insert into trainee(id,name,marks) values(11,'Revati',(90,'A',(20,40,80)))

delete marks from trainee where id=11;

update trainee set marks=(10,'a',(12,13,15)) where id=11;

-----------create user defined data type

Create type address(street text,zipcode text,city text)

To create a table friend

Create table friend(fid int primary key, name text,loc address);

Insert into friend(fid,name,loc) values(12,’Ashwini’,{street:’kothrud’, zipcode:’234456’, city:’Pune’})

Use FROZEN keyword when in user defined data type, then part of the type is not allowed to change but entire value can be overwritten

Create table friend(fid int, name text,loc FROZEN<address>);

Create table friend(fid int, name text,loc LIST<FROZEN<address>>);

To alter the data type

Alter type address add country text;

To rename field of a type

Alter type address rename country to cont;

Batch operations in cassandara

Begin batch

Insert into empdata(id,name,companies) values(111,’xxx’,{‘cognizant’:5,’igate’:3});

Insert into empdata(id,name,companies) values(112,yyy’,{‘TechM’:5,’igate’:3});

Delete companies from empdata where id=123;

Apply batch;

Indexes in Cassandra

Create index idxname on empdata(name)

To add data in json format

```
INSERT INTO Registration JSON'

{

"Emp_id": 2000,

"current_address":  { "Emp_id":1,"h_no" : "A 210", "city" :

                    "delhi", "state" : "DL",

                    "pin_code" :"201307",

                    "country" :"india"},

"Emp_Name": "Ashish Rana"}' ;
```