UNIT-II

Types of database:

OLTP database -> Online Transaction processing

* Large number of short online transaction like create, insert, delete, updata.

* Fast Query processing is done to maintain the data integrity.

* effectiveness is measured by number of transaction per second

eg: oracle, mysql

OLAP -> Online Analytical processing database

* Lower number transaction but capable of doing complex queries.

* effectiveness is measured based on the response time.

eg: Oracle express.

NoSQL database -> Not only SQL database

* No sql database is traditionally relational database.

* Database structure may vary

* It stores the data as key value pack

MongoDB cursor:

find the document present in the given collection, then find and returned a pointer which known as cursor.

db.student.find().pretty()

```
{
    "_id": ObjectId("60256f44f1965zdb63812e94"),
    "studentId": 1,
    "studentName": "Gaurav",
    "studentAge": 20
}
{
    "_id": ObjectId("60256f44f1965zdb63812e95"),
    "studentId": 2,
    "studentAge": 20
}
{
    "_id" : ObjectId("60256f44f1965zdb63812e96"),
    "studentId": 3,
    "studentName": "Geek",
    "studentAge": 20
}
```

Manual iterating:

```
Var name = db.collection_name.find()

Var mycursor = db.student.find({studentId:3}).pretty().

                                        mycursor
{
    "_id": ObjectId("60256f44f1965zdb63812e96"),
    "studentId": 3,
    "StudentName": "Geeks",
    "StudentAge": 20
}
```

Database

SQL

NO-SQL

Users

id	Nam	Pho
1	A	
2	B	
3	C	

Product

Id	Prod
1	cha
2	Tab

Order

Id	Userid	Product id
1	1	2
2		
3		

relational

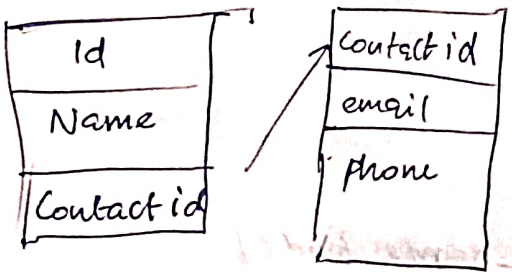
Types of relation DB

1) One to One

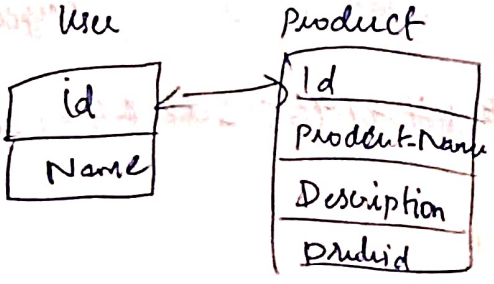
User

Id	Name
1	A
2	B

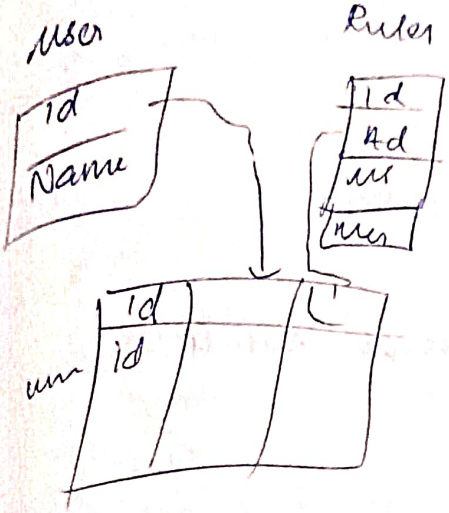
Contact



II) One - Many



Many to Many.



SQL	No SQL
i) Relational	i) Non-relational
ii) Fixed Schema	ii) No fixed schema
iii) Data is not distributed	iii) Data is distributed (replication)
iv) Vertical Scaling	iii) Horizontal
v) Scalability (less)	v) Scalability (upscale)
vi) 1000/s sys go down	v) 1000/s query Available

eg: MySQL, MariaDB

JSON:

```
{
  id: ___
  Name: ___,
  Class: ___
}
{
  id ___
  Name: ___,
}
```

id → Hashed

No fixed schema

I) No SQL

-> key value
-> Document
-> Column
-> Graph

I) key value :

Using key id get the fastest result

eg: Redis

II) Document :

Data can be found using parameter

{

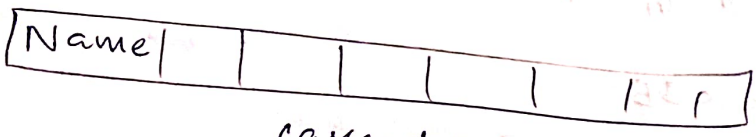
Name : " Ashwin "

dept : " "

}

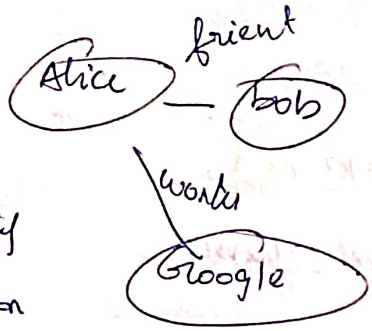
eg: Mongodb

III) Column :



Cassandra

IV) Graph db :



→ entity

--- Relation

mongodb

Database → collection → document

Create a new collection

1) use demo → shell

CRUD

INSERT

```
db.students.insertOne({
name: "Alice Johnson",
age: 21,
major: "CSE"
})
```

{acknowledged: true
insertedId: }

insertMany

Read:

find(), findOne()

projection:

$gte → greater than equals

$lte → less than equal to

$set →

$gt

Update:

Set: to update

```
db.student.updateOne(                    // updateMany()
{name: "Alice Johnson"},
{$set: {age: 22}}
)
```

IV) deleteOne

deleteMany

db.students.drop() → drop the whole collection

Aggregation

I) $match
match
~~group~~ the values

II) $group

group the values

III) $sort

IV) $project

Project the val

Sort:
db.students.find()
.sort({age:-1})

↓ descending order

skip:

skip the value :.skip(1)

limit:

db.students.find()
.skip(1).(limit(5)→.

To fetch limited records from a collection.

Indexes

It supports the efficient resolution of queries

createIndex()

db.collection.name.createIndex({KEY:1})

background:

Boolean → Builds the index in the background

unique:

Boolean: create a unique index, no duplicate data.

name: String

MongoDB generate a indexname by concatenating the name of indexed.

sparse: Boolean:

If true, the index only reference documents with the specified field.

expireAfterSeconds: integer:

specifies a value, in seconds as TTL to control how long MongoDB retains

weight: document:

The weight is a number ranges from 1 to 99,999

default_language: string:

To determine the list of stop words

language_override: string

dropIndex()

dropIndex to delete the key index

db.collection_name.dropIndex({key:1})

getIndex():

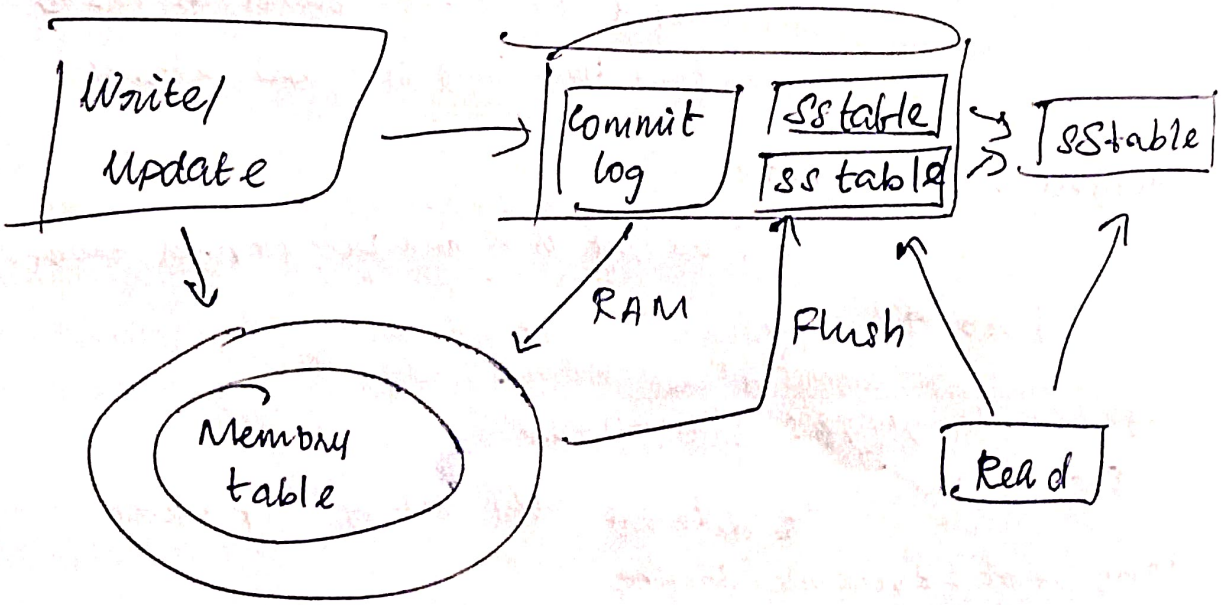
It list the data of all index.

db.collection_Name.getIndexes()

CAP -> Consistency, Availability, Partition tolerance

Cassandra is a open source distributed no sql database

* It is a column oriented database or otherwise called as wide angle stored in db
* db that stores the value of each column together rather than storing the value of each row together
* column oriented db are well suited for bigdata processing, business intelligence and analytics



In cassandra commit log is a crash recovery mechanish

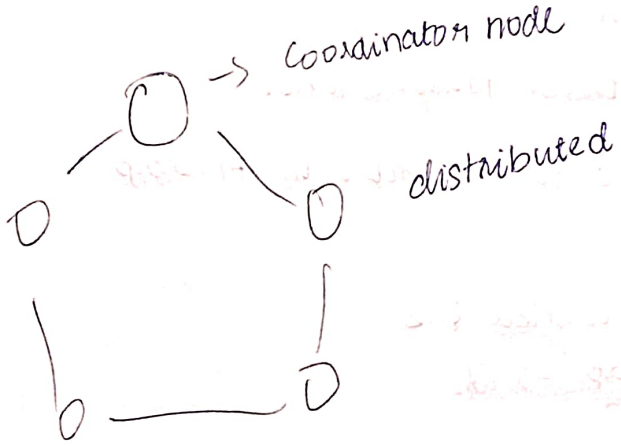
Every right operation is written to the commit logs

A .mem table is a memory resident data structure

After commit logs the data will be return to the mem table

SS table where the commit log get filled a flush is triggred and the content of the mem table are return to this into a SS table

after the completion of this process the mem table is cleared and the commit log is recycled



CRUD

Create:

```
INSERT INTO STUD(ID, NAME, DEPT)
VALUE(1, "ASH", "CSE")
```

Read
~~Select~~:

```
SELECT * FROM STUD
WHERE ID=1
```

UPDATE

```
UPDATE ~~STUD~~ DEPT from STUD where DEPT
                                = Mech
```

Delete:

```
DELETE ID ~~DEPT~~ from STUD where ID=1
```