NOsql

store large volumes of unstructured and semi-structured data

Unstructured data:
not a good fit for a mainstream relational database.
So for Unstructured data, there are alternative platforms for storing and managing
-Example: Word, PDF, Text, Media logs.

features of NOSQL
it is used in modern web scale database, when we collect the huge amount of the data
1 Schemaless : there is no "desc" command
2 horizontally scalable / scaling out
it is easy to add the nodes
adding a new server is easy
3 data is disributed
4 open source
5 Replication factor : 4 by default
the data is replicated in 4 nodes
6 adding the number of data is flexible in a table for different records
the sequeunce of the cols is not fixed
and the number of col values is flexible
7 there are not join , fk
8 there is no normalization
9 ad hoc query (complex query) can not be possible
10 the cost of the entire confituration is less, because the cost of the "commodity server" is less since these m/c are with low configuration
11 no create command,
pk --optional , if the user has not given pk , auto pk col will be created , the name of the col _id
12 non relational, anti RDBMS
13 easy replication support , simple API
14 supports all the languages :, c, c++, java, .net , python, javascript, perl, PHP etc
15 performance is more, and is faster
16 no commit and rollback statement , there is no transaction
17 no group by and order by

elational Database	NoSQL
It is used to handle data coming in low velocity.	It is used to handle data coming in high velocity.
It gives only read scalability.	It gives both read and write scalability.
It manages structured data.	It manages all type of data.
Data arrives from one or few locations.	Data arrives from many locations.
It supports complex transactions.	It supports simple transactions.
It has single point of failure.	No single point of failure.
It handles data in less volume.	It handles data in high volume.
Transactions written in one location.	Transactions written in many locations.
support ACID properties compliance	doesn't support ACID properties
Its difficult to make changes in database once it is defined	Enables easy and frequent changes to database
schema is mandatory to store the data	schema design is not required
Deployed in vertical fashion.	Deployed in Horizontal fashion.

CAP

consistency (among replicated copies), **availability** (of the system for read and write operations) and **partition tolerance** (in the face of the nodes in the system being partitioned by a network fault).

The theorem states that networked shared-data systems can only strongly support two of the following three properties:

Consistency refers to every client having the same view of the data.

response for all the read and write requests in a reasonable amount of time.

system continues to function and upholds its consistency guarantees in spite of network partitions.

BASE

Basically Available: ensure the availability of data by spreading and replicating it across the nodes of the database cluster.

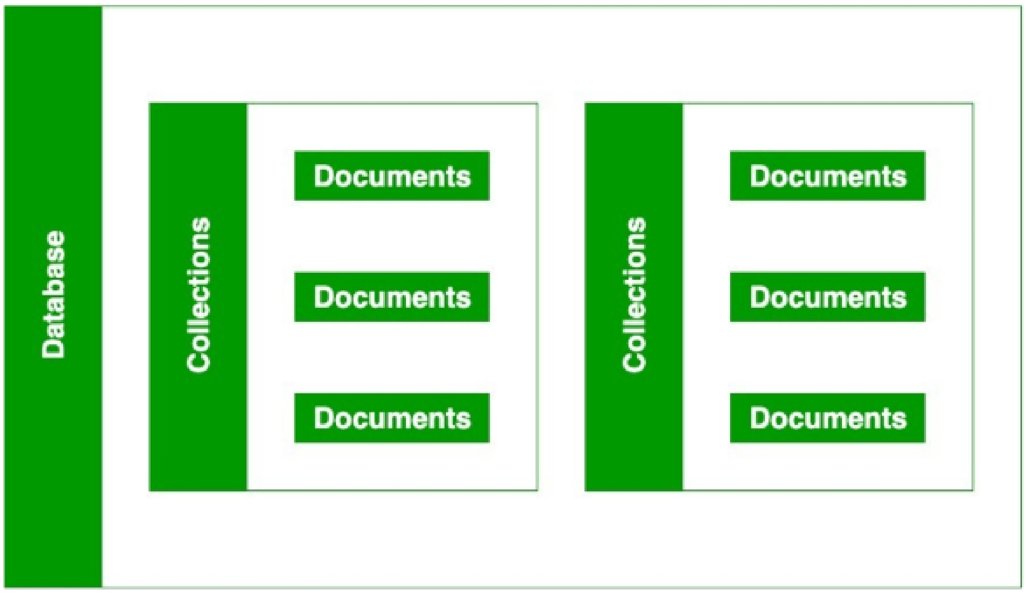
S**oft State:**

E**The fact tventually consistent: BASE does not obligates immediate consistency but it does not mean that it never achieves it.**

Types of NoSQL Databases

- Column family (Apache Cassandra, HBase)
    -
    - Scalability.
    - Compression.
    - Very responsive.
- Document (MongoDB, CouchDB)
    - it uses the documents to store the data in the database. A document database stores data in JSON, BSON, or XML documents.
    - Open formats
    - No foreign keys
    - Faster creation and maintenance
    - Flexible schema
- Graph (Neo4J, Titan)
    -
    - graph-based database, it is easy to identify the relationship between the data by using the links.
    - The Query’s output is real-time results.
    - The speed depends upon the number of relationships among the database elements.

    - Updating data is also easy,
- Key value (Redis, Riak)
    -
    - Simplicity.
    - Scalability.
    - Speed.



In MongoDB, the names of the database are case insensitive

    - For windows user, MongoDB database names cannot contain any of these following characters:

```
/\. "$*:|?
```

    - For linux, user, MongoDB database names cannot contain any of these following characters:

```
/\. "$
```

mongoDB database names cannot be empty and must contain less than 64 characters

collection

- Collection name must starts with an underscore or a character.
- Collection name does not contain $, empty string, null character and does not begin with system. prefix.
- The maximum length of the collection name is 120 bytes
- syntax; `db.collection_name.insertOne({..})`

Document

- The field names are of strings.
- The _*id* field name is reserved to use as a primary key. And the value of this field must be unique, immutable, and can be of any type other than an array.
- The field name cannot contain null characters.
- The top-level field names should not start with a dollar sign ($).
- The maximum size of the BSON document is 16MB.
- In MongoDB, the data records are stored as BSON documents. Here, BSON stands for binary representation of JSON documents,

- **Starting MongoDB server**

  C:\mongodb\bin\mongod.exe
- **Starting MongoDB shell**

  C:\mongodb\bin\mongo.exe
- **Viewing the name of the current database**

  db
- **Viewing the list of databases**

  show dbs
- **Creating a new database / using an existing database**

  use CDAC
- **Others**

  help

  db.dropDatabase()

- show collections

  lists all the collections
- db.persons.find()

  to find the recors
- db.persons.findOne()

  performs a read operation to return a single document(first document)
- db.persons.count()

  Returns a count of the documents
- db.persons.drop()

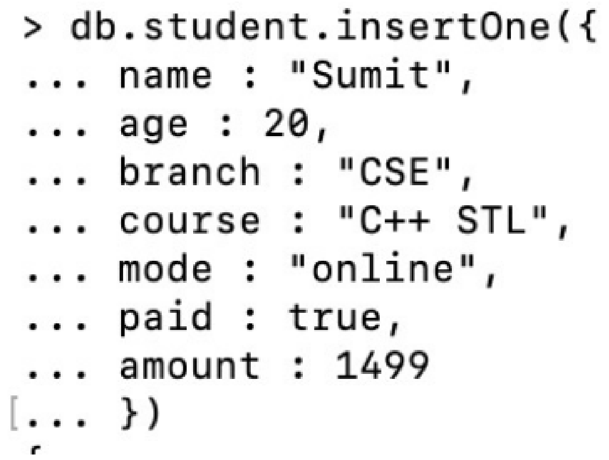
  method to drop the entire collection, including the indexes

MethodDescription

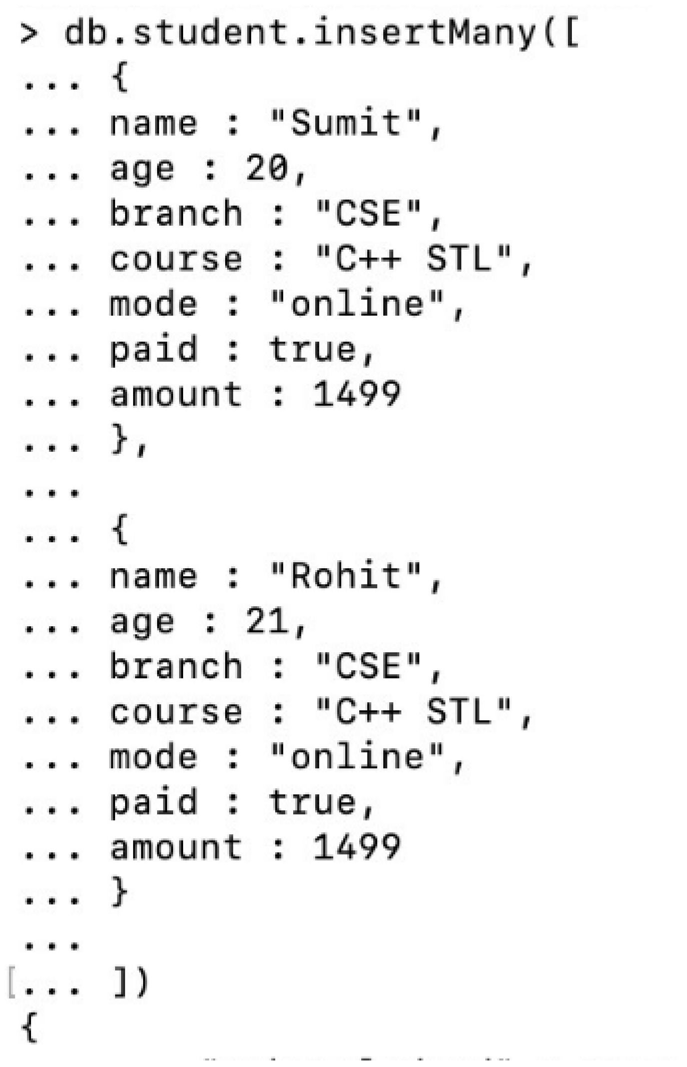
db.collection.insertOne(): It is used to insert a single document in the collection.

db.collection.insertMany(): It is used to insert multiple documents in the collection.

db.createCollection(): It is used to create an empty collection.



```
> db.student.insertOne({
... name : "Sumit",
... age : 20,
... branch : "CSE",
... course : "C++ STL",
... mode : "online",
... paid : true,
... amount : 1499
[... })
```



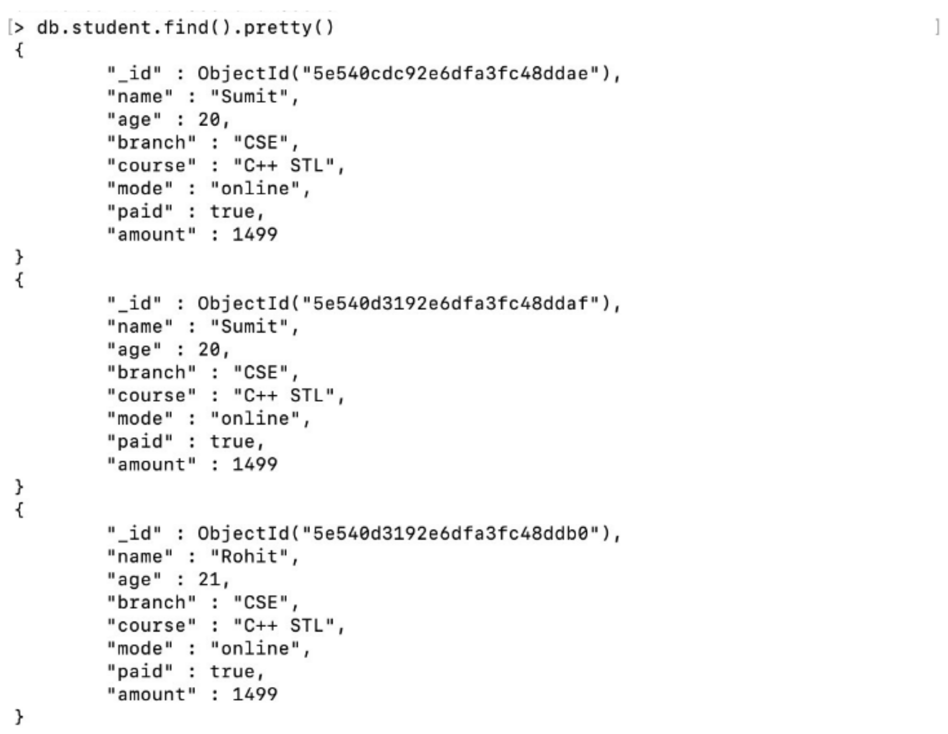
```
> db.student.insertMany([
... {
... name : "Sumit",
... age : 20,
... branch : "CSE",
... course : "C++ STL",
... mode : "online",
... paid : true,
... amount : 1499
... },
...
... {
... name : "Rohit",
... age : 21,
... branch : "CSE",
... course : "C++ STL",
... mode : "online",
... paid : true,
... amount : 1499
... }
...
[... ])
{
```

Read Operations –

The Read operations are used to retrieve documents from the collection

db.collection.find()	**It is used to retrieve documents from the collection.**

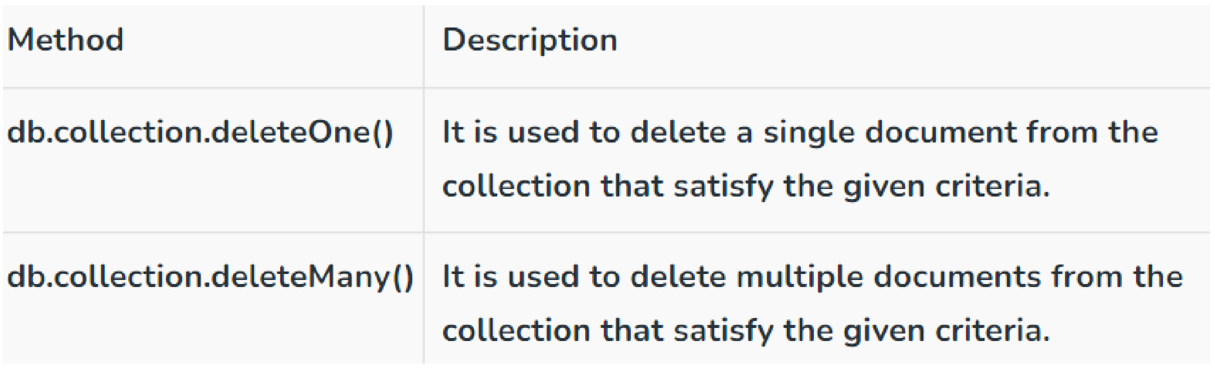
<u>.pretty() :</u> this method is used to decorate the result such that it is easy to read.



```
> db.student.find().pretty()
{
        "_id" : ObjectId("5e540cdc92e6dfa3fc48ddae"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddaf"),
        "name" : "Sumit",
        "age" : 20,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
{
        "_id" : ObjectId("5e540d3192e6dfa3fc48ddb0"),
        "name" : "Rohit",
        "age" : 21,
        "branch" : "CSE",
        "course" : "C++ STL",
        "mode" : "online",
        "paid" : true,
        "amount" : 1499
}
```

Delete Operations –

The delete operation are used to delete or remove the documents from a collection



Method	Description
db.collection.deleteOne()	It is used to delete a single document from the collection that satisfy the given criteria.
db.collection.deleteMany()	It is used to delete multiple documents from the collection that satisfy the given criteria.

index

Indexes are special data structures that stores some information related to the documents such that it becomes easy for MongoDB to find the right data file. The indexes are order by the value of the field specified in the index.

Creating an Index :

MongoDB provides a method called createIndex() that allows user to create an index

`db.COLLECTION_NAME.createIndex({KEY:1})`

The key determines the field on the basis of which you want to create an index and 1 (or -1) determines the order in which these indexes will be arranged(ascending or descending).

Drop an index:

In order to drop an index, MongoDB provides the dropIndex() method.

Syntax –

```
db.NAME_OF_COLLECTION.dropIndex({KEY:1})
```

The dropIndex() methods can only delete one index at a time. In order to delete (or drop) multiple indexes from the collection, MongoDB provides the dropIndexes() method that takes multiple indexes as its parameters.

Syntax –

```
db.NAME_OF_COLLECTION.dropIndexes({KEY1:1, KEY2: 1})
```

et description of all indexes :

The getIndexes() method in MongoDB gives a description of all the indexes that exists in the given collection.

Syntax –

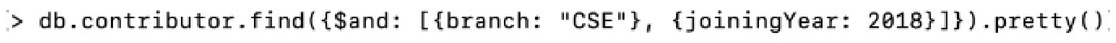
```
db.NAME_OF_COLLECTION.getIndexes()
```

MongoDB AND operator ($and)

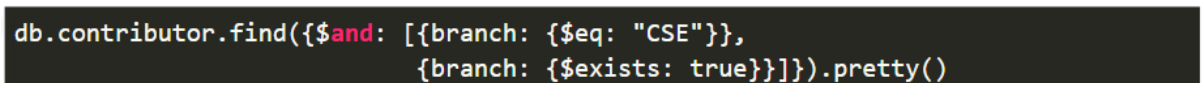
- perform logical AND operation on the array of one or more expressions and select or retrieve only those documents that match all the given expression in the array.
- f the first expression of $and operator evaluates to false, then MongoDB will not evaluate the remaining expressions in the array.
- You can also use AND operation implicitly with the help of comma(,).

Syntax:

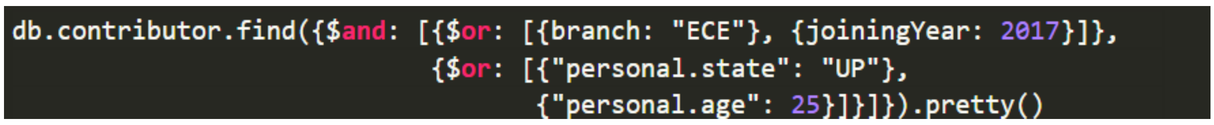
```
{ $and: [ { Expression1 }, { Expression2 }, ..., { Expressi
onN } ] }
or
{ { Expression1 }, { Expression2 }, ..., { ExpressionN }}
```



```
> db.contributor.find({$and: [{branch: "CSE"}, {joiningYear: 2018}]}).pretty()
```



```
db.contributor.find({$and: [{branch: {$eq: "CSE"}},
                            {branch: {$exists: true}}]}).pretty()
```



```
db.contributor.find({$and: [{$or: [{branch: "ECE"}, {joiningYear: 2017}]},
                            {$or: [{"personal.state": "UP"},
                                   {"personal.age": 25}]}]}).pretty()
```

`$match` : Filters documents based on specified criteria.

Query Operators:

- `$eq` : Matches values that are equal to a specified value.
- `$ne` : Matches values that are not equal to a specified value.
- `$gt` : Matches values that are greater than a specified value.

- `$lt` : Matches values that are less than a specified value.
- `$gte` : Matches values that are greater than or equal to a specified value.
- `$lte` : Matches values that are less than or equal to a specified value.
- `$in` : Matches any of the values specified in an array.
- `$nin` : Matches none of the values specified in an array.
- `$set` : Sets the value of a field in a document.
- `$unset` : Removes a field from a document.
- `$inc` : Increments the value of a field by a specified amount

sort() Method

- specifies the order in which the query returns the matching documents from the given collection
- The value is 1 or -1 specifying an ascending or descending sort respectively.
- MongoDB can find the result of the sort operation using indexes
- If a sort returns the same result every time we perform on same data, then such type of sort is known as a stable sort.
- If a sort returns a different result every time we perform on same data, then such type of sort is known as unstable sort.
- MongoDB generally performs a stable sort unless sorting on a field that holds duplicate values.
- We can use limit() method with sort() method, it will return first m documents, where m is the given limit.

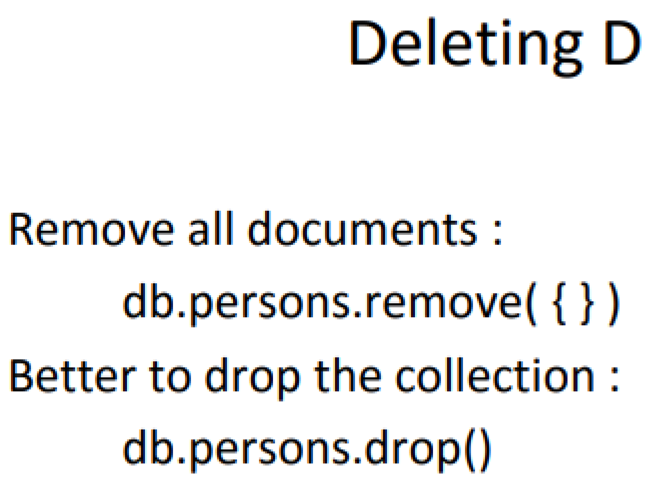
```
db.Collection_Name.sort({field_name:1 or -1})
```

```
db.student.find().sort({age:1})
```

MongoDB – Replication and Sharding

- Replication can be simply understood as the duplication of the data-set
- Replication is the method of duplication of data across multiple servers.
- Replicating your database means you make imagers of your data-set.
- The primary server receives all write operations and record all the changes to the data i.e, oplog.
- All the secondary nodes are connected with the primary nodes
- **Why Replication?**
  - High Availability of data disasters recovery
  - No downtime for maintenance (like backups index rebuilds and compaction)
  - Read Scaling (Extra copies to read from)

- sharding is partitioning the data-set into discrete parts.
- By sharding, you divided your collection into different parts.
- Sharding is a method for allocating data across multiple machines.
- MongoDB used sharding to help deployment with very big data sets and large throughput the operation.
- Sharding determines the problem with horizontal scaling breaking the system dataset and store over multiple servers, adding new servers to increase the volume as needed.



Deleting Data

Remove all documents :

```
db.persons.remove( { } )
```

Better to drop the collection :

```
db.persons.drop()
```

- To find with female and the first field to display

  ```
  db.persons.find( {gender: 'F'}, {name: 1} )
  ```

- To hide the id column

  ```
  db.persons.find( {gender: 'F'},
  {name: 1, _id: 0} )
  ```

The $in Operator

- Find all people living in Mumbai or Jaipur

```
db.persons.find ( {livesIn: {$in: ['Mumbai',
'Jaipur'] } } )
```

```
db.persons.find ( {livesIn: {$in: ['Mumbai',
'Jaipur'] } }, {name: 1, _id: 0} )
```

- All persons not living in Jaipur

db.persons.find ({livesIn: {$ne: 'Jaipur'} }, {name: 1, livesIn: 1})

- Find all persons who have visited India

db.persons.find ({countriesVisited: 'India'}, {name: 1})

db.persons.ensureIndex ({yearOfMarriage : 1})

- Creates an index on the yearOfMarriage field in ascending order. For descending order, specify -1.
- Order of indexing matters only if you use the sort clause for query output.