SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER APPLICATIONS

23CAT701 – FULL STACK DEVELOPMENT

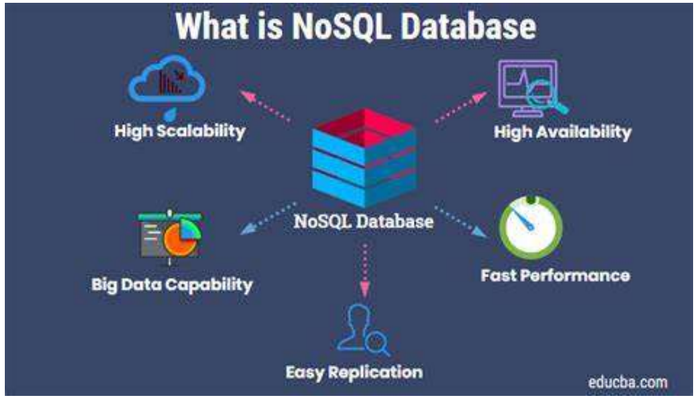
II YEAR III SEM

UNIT II – ADVANCED NODE JS AND DATABASE

TOPIC 1 - Introduction to NoSQL databases – MongoDB system overview

Introduction to NoSQL

1.Document databases: These databases store data as semi-structured documents, such as JSON or XML, and can be queried using document-oriented query languages.



2.Key-value stores: These databases store data as key-value pairs, and are optimized for simple and fast read/write operations.

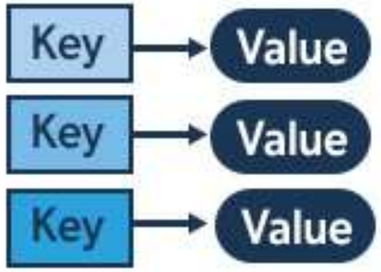
3.Column-family stores: These databases store data as column families, which are sets of columns that are treated as a single entity. They are optimized for fast and efficient querying of large amounts of data.

4.Graph databases: These databases store data as nodes and edges, and are designed to handle complex relationships between data.

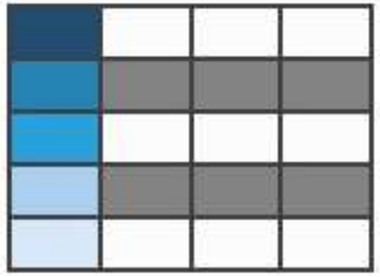
NoSQL

Memcached, Redis, Coherence

Key-Value

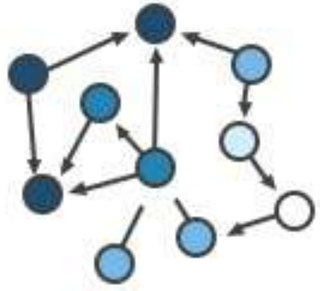


Column-Family



Hbase, Big Table, Accumulo

Graph



Amazon Neptune, Neo4j

Document

MongoDB, CouchDB, Cloudant

How do Document Databases Work?

C1	C2	C3

Relational Data Model

Document Store Model

RELATIONAL

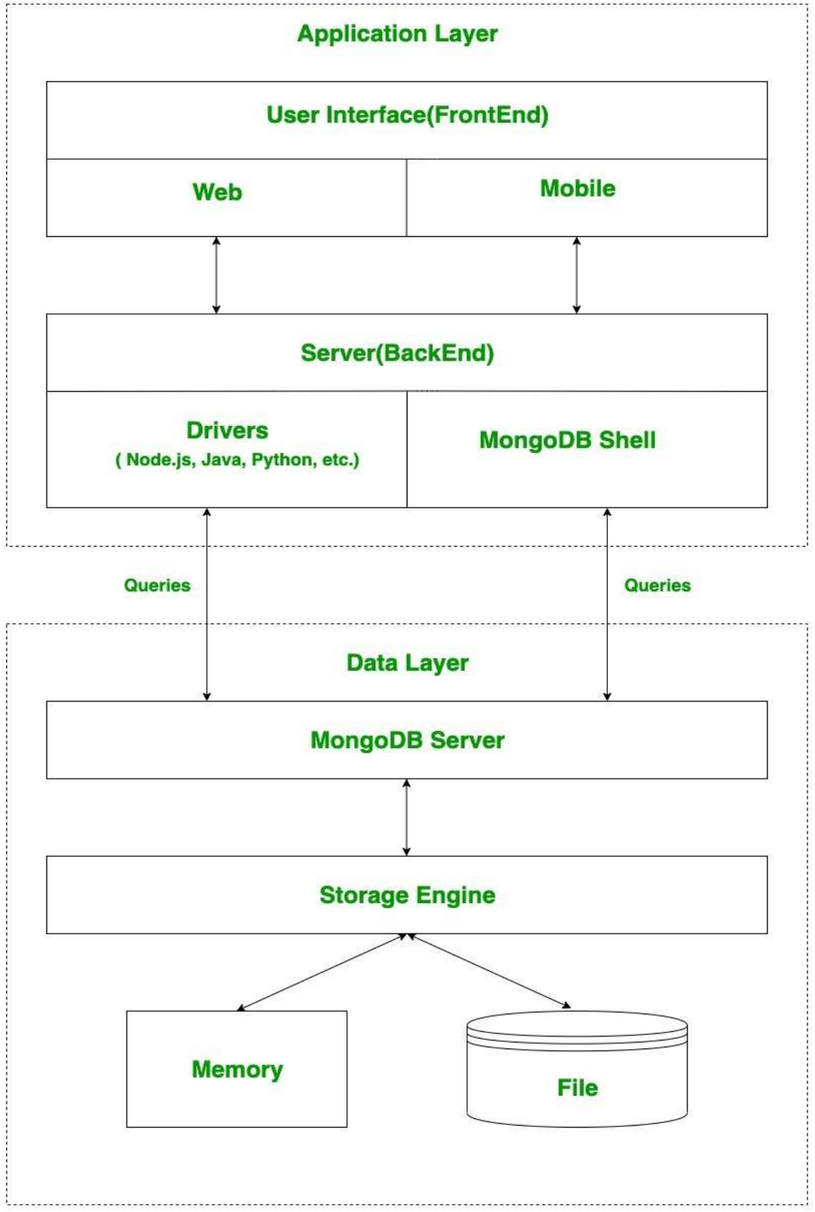
ID	first_name	last_name	cell	city	year_of_birth	location_x	location_y
1	'Mary'	'Jones'	'516-555-2048'	'Long Island'	1986	'-73.9876'	'40.7574'

ID	user_id	profession
10	1	'Developer'
11	1	'Engineer'

ID	user_id	name	version
20	1	'MyApp'	1.0.4
21	1	'DocFinder'	2.5.7

ID	user_id	make	year
30	1	'Bentley'	1973
31	1	'Rolls Royce'	1965

How MongoDB works ?



MongoDB: An introduction

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple or Row	Document
Column	Field
Table Join	Embedded Documents
Primary Key	Primary key / Default key
Mysqld / Oracle	mongod

http://www.mongodb.org/downloads

Use of the "use" command followed by the database_name for creating a database.

use my_project_db

can use the **show dbs** command.

```
> show dbs
admin   0.000GB
config  0.000GB
> db.movie.insert({"name":"Avengers: Endgame"})
WriteResult({ "nInserted" : 1 })
> show dbs
admin          0.000GB
config         0.000GB
my_project_db  0.000GB
>
```

db.movie.insert({"name":"Avengers: Endgame"})

db

db.dropDatabase()
{ "dropped": "my_project_db", "ok": 1}

```
> db
my_project_db
> db.dropDatabase()
{ "ok" : 1 }
> show dbs
admin   0.000GB
config  0.000GB
>
```

Integer

db.TestCollection.insert({"Integer example": 62})

```
> use my_project_db
switched to db my_project_db
> db.TestCollection.insert({"Integer example": 62})
WriteResult({ "nInserted" : 1 })
> db.TestCollection.find()
{ "_id" : ObjectId("5d08724bce0d9292a5121a78"), "Integer example" : 62 }
>
```

Boolean

db.TestCollection.insert({"Nationality Indian": true})

```
> db.TestCollection.insert({"Nationality Indian": true})
WriteResult({ "nInserted" : 1 })
> db.TestCollection.find()
{ "_id" : ObjectId("5d08724bce0d9292a5121a78"), "Integer example" : 62 }
{ "_id" : ObjectId("5d087412ce0d9292a5121a79"), "Nationality Indian" : true }
>
```

Double

```
db.TestCollection.insert({"double data type": 3.1415})
```

String

```
db.TestCollection.insert({"string data type" : "This is a sample message."})
```

Arrays

```
var degrees = ["BCA", "BS", "MCA"]
db.TestCollection.insert({" Array Example" : " Here is an example of array",
" Qualification" : degrees})
```

Object

```
var embeddedObject={"English" : 94, "ComputerSc." : 96, "Maths" : 80,
    "GeneralSc." : 85}

    db.TestCollection.insert({"Object data type" : "This is Object",
"Marks" : embeddedObject})
```

Date

```
    var date=new Date()
        var date2=ISODate()
        var month=date2.getMonth()
    db.TestCollection.insert({"Date":date, "Date2":date2, "Month":month})
```

Creation of collection

db.createCollection
(collection_name, options)

db.createCollection(<collection_name>, { capped: <boolean>,
autoIndexId: <boolean>,
size: <number>,
max: <number>,
storageEngine: <document>,
validator: <document>,
validationLevel: <string>,
validationAction: <string>,
indexOptionDefaults: <document>,
viewOn: <string>,
pipeline: <pipeline>,
collation: <document>,
writeConcern: <document>})

db.createCollection("MyCollection")

db.movie.insert({"name":"Avengers: Endgame"})

db.collection_name.find()

db.collection_name.drop()

```
use my_project_db
db.movie.drop()
show collections
```

Insert Documents

insertOne()

```
db.posts.insertOne({
title: "Post Title 1",
body: "Body of post.",
category: "News",
likes: 1, tags:
["news", "events"],
date: Date() })
```

insertMany()

```
db.posts.insertMany([ { title: "Post
Title 2", body: "Body of post.",
category: "Event", likes: 2, tags:
["news", "events"], date: Date() },
 { title: "Post Title 3", body: "Body
of post.", category: "Technology",
likes: 3, tags: ["news", "events"],
date: Date() }}]
```

Update Document

updateOne()

```
db.posts.updateOne( { title: "Post Title 1" }, { $set: { likes: 2 } } )
```

updateMany()

```
db.posts.updateMany({}, { $inc: { likes: 1 } })
```

Delete Documents

deleteOne()

```
db.posts.deleteOne({ title: "Post Title 5" })
```

deleteMany()

```
db.posts.deleteMany({ category: "Technology" })
```

MongoDB Query Operators

Comparison

The following operators can be used in queries to compare values:

- $eq: Values are equal
- $ne: Values are not equal
- $gt: Value is greater than another value
- $gte: Value is greater than or equal to another value
- $lt: Value is less than another value
- $lte: Value is less than or equal to another value
- $in: Value is matched within an array

MongoDB Query Operators

Logical

The following operators can logically compare multiple queries.

- $and: Returns documents where both queries match
- $or: Returns documents where either query matches
- $nor: Returns documents where both queries fail to match
- $not: Returns documents where the query does not match

Evaluation

The following operators assist in evaluating documents.

- $regex: Allows the use of regular expressions when evaluating field values
- $text: Performs a text search
- $where: Uses a JavaScript expression to match documents

MongoDB Query Operators

Comparison

The following operators can be used in queries to compare values:

- $eq: Values are equal
- $ne: Values are not equal
- $gt: Value is greater than another value
- $gte: Value is greater than or equal to another value
- $lt: Value is less than another value
- $lte: Value is less than or equal to another value
- $in: Value is matched within an array

Reference

- MongoDB Query Operators (w3schools.com)
- MongoDB mongosh Insert (w3schools.com)
- Overview of MongoDB (w3schools.in)
- Insert Documents in MongoDB (w3schools.in)

- https://www.mongodb.com/**nosql**-explained

- www.couchbase.com/**nosql**-resources/what-is-**no-sql**

- http://nosql-database.org/ "NoSQL DEFINITION: Next Generation Databases mostly addressing some of the points: being non-relational, distributed, open- source and horizontally scalable“