-> Not Only SQL / non SQL

-> NoSQL databases are databases that store data in a format other than relational tables

-> NoSQL databases come in a variety of types based on their data model.

-> The main types are document, key-value, wide-column, and graph.

-> They provide flexible schemas and scale easily with large amounts of big data and high user loads

**What is NoSQL Database?**

-> NoSQL databases store data in a more natural and flexible way.

-> NoSQL, as opposed to SQL, is a database management approach, whereas SQL is just a query language, similar to the query languages of NoSQL databases.

**Types of NoSQL databases**

**1) Document-oriented Databases** - store data in documents similar to JSON (JavaScript Object Notation) objects. Each document contains pairs of fields and values. The values can typically be a variety of types including things like strings, numbers, booleans, arrays, or objects.

**Examples: MongoDB, Couchbase**

**2) Key-value Databases** - are a simpler type of database where each item contains keys and values. Each key is unique and associated with a single value. They are used for caching and session management and provide high performance in reads and writes because they tend to store things in memory

**Examples: Amazon DynamoDB, Redis**

**3) Wide-column Stores** - store data in tables, rows, and dynamic columns. The data is stored in tables. However, unlike traditional SQL databases, wide-column stores are flexible, where different rows can have different sets of columns.

**Examples: Apache Cassandra and HBase**

**4) Graph databases** - store data in nodes and edges. Nodes typically store information about people, places, and things (like nouns), while edges store information about the relationships between the nodes. They work well for highly connected data, where the relationships or patterns may not be very obvious initially.

**Example: Neo4J and Amazon Neptune**. MongoDB also provides graph traversal capabilities using the $graphLookup stage of the aggregation pipeline

==> Difference between RDBMS and NoSQL databases -- <https://www.mongodb.com/nosql-explained>

**When to go for NonSQL Databases?**

-> When you want to store and retrieve huge amount of data - Bigdata

think about flights - thousands of flights, huge data

in a bank - how many transactions are happening in a day / year - huge amount of data

-> the relationship between the data you store is not that important

-> the data is not structured and changing over time

-> constraints and joins support is not required at database level

-> the data is growing continuously and you need to scale the database regularly to handle the data

ex: for a post of yours in Facebook - many people will give the comments in different formats like images, text etc - can we maintain a specific structure to store the comments

**MongoDB**

-> MongoDB is a new and popularly used database.

-> name from the English word - "Humongous"

-> It is a document based, non relational database provider.

-> MongoDB is a popular Non-SQL Database - used to deal with huge amount of unstructured data

-> MongoDB is open source, cross-platform, document-oriented database written in C++

-> MongoDB provides high performance, high availability, and automatic scaling

-> MongoDB is available under General Public License for free (community edition), and also available under Commercial License

-> MongoDB is developed and supported by a US based company 10Gen (2007) - name changed to MongoDB, Inc. August 27, 2013

**Mapping MongoDB with RDBMS**

**RDBMS**  **MongoDB**

Database ----> Database

Tables ----> Collections

Rows ----> Documents

Columns ----> Fields

**MongoDB Terminology**

-> Collections in MongoDB is equivalent to the tables in RDBMS

-> Documents in MongoDB is equivalent to the rows in RDBMS

-> Fields in MongoDB is equivalent to the Columns in RDBMS

Fields (key and value pairs) are stored in Document

Documents are stored in Collection Collections are stored in Database

**MongoDB Structure**

MongoDB Physical Database contains several logical databases.

Each Database contains several collections.

Collection is similar to Table in Relational Database

Ex:

**Database**: Shopping Cart

**Collections**: Customers, Products , Orders

**Customer Collection**: contains several documents

**document1: {**

"Name":"Sunny",

"age": 40,

"Salary": 10000

}

**document2:{**

"Name":"Sunil"

} ---- no specific structure / schema here

**Document3:{**

"Name":"Sanjay",

"age":30,

"address":{

"city":"Hyderabad"

},

"hobbies":[

{"name","cricket"}, {"name":"Swimming"}

]

}

-- the above things will be stored in the binary form (BSON) BSON to JSON --- JSON to BSON conversion is taking place internally

How data represented in MongoDB?

In JSON - BSON - Binary JSON

**Key Characteristics of MongoDB**

* MongoDB is schema less. It is a document database in which one collection holds different documents.
* There may be difference between number of fields, content and size of the document from one to other.
* Structure of a single object is clear in MongoDB.
* There are no complex joins in MongoDB.
* It uses internal memory for storing working sets and this is the reason of its fast access.
* All information related to a document will be stored in a single place. To retrieve data, it is not required to perform join operations and hence retrieval is very fast
* Documents are independent of each other and no schema; hence we can store unstructured data like videos, audio files
* Very easy to perform CRUD operations (inserting new document, updating an existing document etc)
* Retrieval data is in the form of JSON and is understandable for any programming language without any conversion
* We can store very huge amount of data
* More Scalable

**MongoDB Data Types**

String Integer Boolean Double Arrays Object Null Date

Symbol - it is generally used for languages that use a specific type.

Min/Max Keys - this datatype compare a value against the lowest and highest bson elements.

BSON is just binary JSON (a superset of JSON with some more data types, most importantly binary byte array). It is a serialization format used in MongoDB.

**Installation of MongoDB on Windows**

-> Download MongoDB

https://www.mongodb.com

Products > Community Edition > Download Community > MongoDB Community Server > Download

mongodb-windows-x86\_64-8.0.9-signed.msi

-> Install It --- Start Compass

Compass is the Graphical Tool for MongoDB

mongodb://localhost:27017

**set PATH environment variable for C:\Program Files\MongoDB\Server\8.0\bin**

MongoDB **mongos** instances route queries and write operations to shards in a sharded cluster. mongos provides the only interface to a sharded cluster from the perspective of applications. Applications never connect or communicate directly with the shards.

**A shard is a single MongoDB instance** that holds a subset of the sharded data. Shards can be deployed as replica sets to increase availability and provide redundancy. The combination of multiple shards creates a complete data set.

**check in Services.msc** if MongoDB is not added as a service then

mongod --dbpath "" --logpath "" --install --serviceName "MongoDB"

**open CMD in Run As Administrator**

**cmd> mongod** --dbpath "C:\Program Files\MongoDB\Server\7.0\data" --logpath "C:\Program Files\MongoDB\Server\7.0\bin\mongod.log" --install --serviceName "MongoDB"

**Practicals**

**Open MongoDB Shell**

cmd> **mongo**

**\*\* If installed MongoDB version is 6.0 or above, mongo command will not work on Powershell/cmd.**

To run mongo commands, you have to install **MongoDB Shell**

**Mongo Shell** - Mongo Shell is used to perform operations with Mongo Database

Google “mongoDB shell download”

https://www.mongodb.com/try/download/shell > Download > Extract > set path for /bin

C:\Users\Sunil Joseph\Downloads\mongosh-2.5.1-win32-x64\bin --- Set in PATH environment

**cmd > mongosh** You're all set to work with MongoDB **test> show dbs**

**Show All Databases** > show dbs

**Show Current Database** > db

**Create Or Switch Database** > use wipro

**Drop Database** > db.dropDatabase()

**Show Collections** > show collections

**Create Collection** > db.createCollection('posts') > show collections

**Check on MongoDB Compass**

**mongod --port 27017**

**Insert Row / Document** -- open notepad

db.posts.insert({

title: 'Post One',

body: 'Body of Post One',

category: 'News',

likes: 12,

tags: ['news','events'],

user: {

name: 'Sunil Joseph',

status: 'Author'

},

date: Date()

})

> paste it

**Insert Multiple Rows**

-- open notepad

**db.posts.insertMany**([

{

title: 'Post Four',

body: 'Body of Post Four',

category: 'Entertainment',

date: Date()

}

])

{

title: 'Post Three',

body: 'Body of Post Three',

category: 'News',

date: Date()

},

[{

title: 'Post Two',

body: 'Body of Post Two',

category: 'Technology',

date: Date()

},

> paste it

**Get All Rows**  > db.posts.find()

**Get All Rows Formatted** > db.posts.find().pretty()

**Find Rows** > db.posts.find({category: 'News'})

> db.posts.find({category: 'News'}).pretty()

**Sort Rows** # asc

> db.posts.find().sort({ title: 1 }).pretty()

# desc

> db.posts.find().sort({ title: -1 }).pretty()

**Count Rows** > db.posts.find().count()

> db.posts.find({ category: 'news' }).count()

**Limit Rows** > db.posts.find().limit(2).pretty()

**Chaining** > db.posts.find().limit(2).sort({ title: 1 }).pretty()

**Foreach**

> **db.posts.find().forEach(function(doc){**

print('Blog Post:' + doc.title)

})

**Find One Row** > db.posts.findOne({category: 'News'})

**Find Specific Fields**

db.posts.find({ title: 'Post One' }, {

title: 1,

author: 1

})

**Update Row** -- notepad

**Update Specific Field**

> db.posts.find({title: 'Post Two'}).pretty()

-- notepad

**db.posts.update({ title: 'Post Two' },**

{

$set: {

body: 'Body for post 2',

category: 'Technology'

}

})

db.posts.update({ title: 'Post Two' },

{

title: 'Post Two',

body: 'New body for post 2',

date: Date()

},

{

upsert: true

})

**Rename Field** -- to rename 'likes' field to 'views'

db.posts.update({ title: 'Post Two' },

{

$rename: {

likes: 'views'

}

})

**> db.posts.find({title: 'Post One'}).pretty()**

**> db.posts.find({title: 'Post Two'}).pretty()**

**Increment Field ($inc)** -- increase the likes by 2

db.posts.update({ title: 'Post Two' },

{

$inc: {

likes: 2

}

})

**> db.posts.find({title: 'Post One'}).pretty()**

**Delete Row** > db.posts.find().pretty()

> db.posts.remove({title: 'Post Four' })

> db.posts.find().pretty()