**What is Mongo DB?**

It is a NoSql DataBase.

NoSQL Database is a non-relational Data Management System, that does not require a fixed schema. It avoids joins, and is easy to scale. The major purpose of using a NoSQL database is for distributed data stores with humongous data storage needs. NoSQL is used for Big data and real-time web apps

**What is “Namespace” in MongoDB?**

MongoDB stores BSON (Binary Interchange and Structure Object Notation) objects in the collection. The concatenation of the collection name and database name is called a namespace.

**Advantages:**

1. Big Data Capability
2. No Single Point of Failure
3. Easy Replication
4. No Need for Separate Caching Layer
5. It provides fast performance and horizontal scalability.

**Disadvantages :**

1. No standardization rules
2. Limited query capabilities
3. Doesn't work as well with relational data
4. Complex Joins won’t support
5. No transaction management implemented.

**Why Use MongoDB?**

Below are the few of the reasons as to why one should start using MongoDB

**Document-oriented** – Since MongoDB is a NoSQL type database, instead of having data in a relational type format, it stores the data in documents. This makes MongoDB very flexible and adaptable to real business world situation and requirements.

**Ad hoc queries** - MongoDB supports searching by field, range queries, and regular expression searches. Queries can be made to return specific fields within documents.

**Indexing** - Indexes can be created to improve the performance of searches within MongoDB. Any field in a MongoDB document can be indexed.

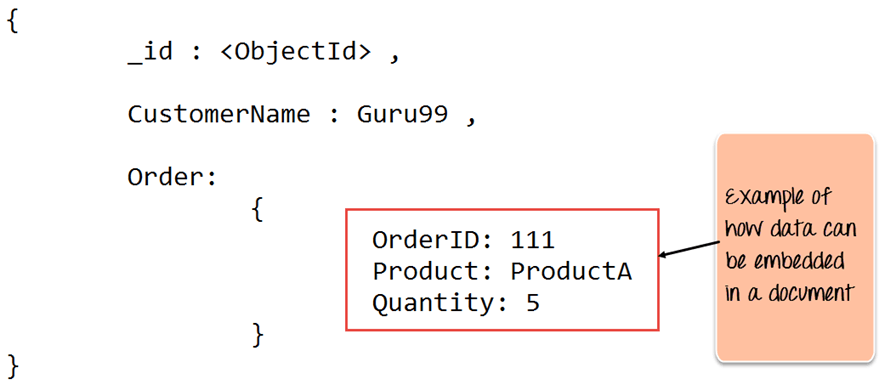
**Replication** - MongoDB can provide high availability with replica sets. A replica set consists of two or more mongo DB instances. Each replica set member may act in the role of the primary or secondary replica at any time. The primary replica is the main server which interacts with the client and performs all the read/write operations. The Secondary replicas maintain a copy of the data of the primary using built-in replication. When a primary replica fails, the replica set automatically switches over to the secondary and then it becomes the primary server.

**Load balancing [Sharding]-** MongoDB uses the concept of sharding to scale horizontally by splitting data across multiple MongoDB instances. MongoDB can run over multiple servers, balancing the load and/or duplicating data to keep the system up and running in case of hardware failure.

**Mongo DB Collection Explanation:**

The \_id field is added by MongoDB to uniquely identify the document in the collection.

What you can note is that the Order Data (OrderID, Product, and Quantity ) which in RDBMS will normally be stored in a separate table, while in MongoDB it is actually stored as an embedded document in the collection itself. This is one of the key differences in how data is modeled in MongoDB.



If you create a new document without an \_id field, MongoDB will automatically create the field.

| **\_Id** | **CustomerID** | **CustomerName** | **OrderID** |
| --- | --- | --- | --- |
| 563479cc8a8a4246bd27d784 | 11 | Guru99 | 111 |
| 563479cc7a8a4246bd47d784 | 22 | Trevor Smith | 222 |
| 563479cc9a8a4246bd57d784 | 33 | Nicole | 333 |

**Collection –** This is a grouping of MongoDB documents

**Cursor –** This is a pointer to the result set of a query.

**Database –** This is a container for collections like in RDMS wherein it is a container for tables. Each database gets its own set of files on the file system. A MongoDB server can store multiple databases.

**Document - A** record in a MongoDB collection is basically called a document

**Field -** A name-value pair in a document

| **RDBMS** | **MongoDB** |
| --- | --- |
| Table | Collection |
| Row | Document |
| Column | Field |
| Joins | Embedded documents |

**Data Modelling in MongoDB :**

As we have seen from the Introduction section, the data in MongoDB has a flexible schema. Unlike in SQL databases, where you must have a table's schema declared before inserting data, MongoDB's collections do not enforce document structure. This sort of flexibility is what makes MongoDB so powerful.

When modeling data in Mongo, keep the following things in mind.

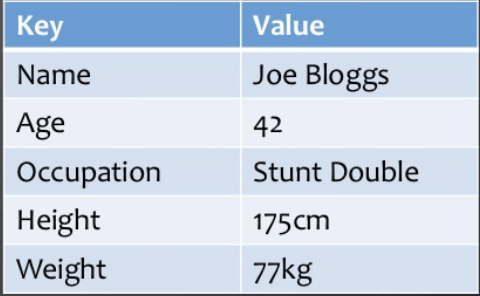
1.What are the needs of the application – Look at the business needs of the application and see what data and the type of data needed for the application. Based on this, ensure that the structure of the document is decided accordingly.

2.What are data retrieval patterns – If you foresee a heavy query usage then consider the use of indexes in your data model to improve the efficiency of queries.

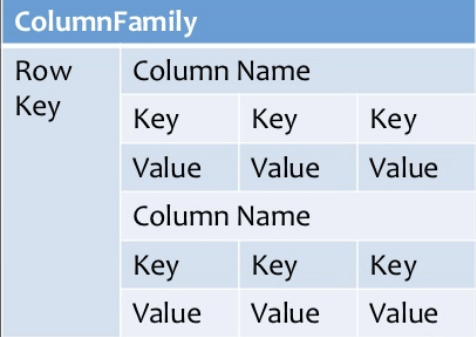
3.Are frequent inserts, updates and removals happening in the database? Reconsider the use of indexes or incorporate sharding if required in your data modeling design to improve the efficiency of your overall MongoDB environment.

**Types of NoSQL Databases:**

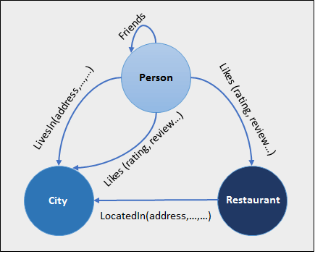
1. Key-value Pair Based : key =age and value=42



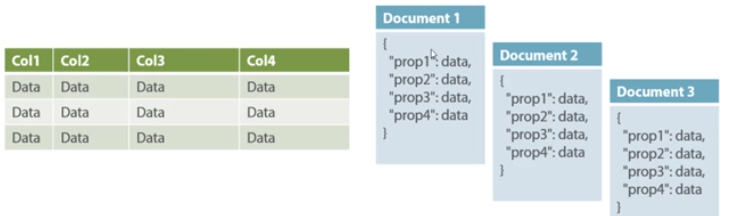
1. Column-oriented Graph : Column-oriented databases work on columns and are based on BigTable paper by Google



1. Graphs based : A graph type database stores entities as well the relations amongst those entities. The entity is stored as a node with the relationship as edges. An edge gives a relationship between nodes. Every node and edge has a unique identifier



1. Document-oriented: Document-Oriented NoSQL DB stores and retrieves data as a key value pair but the value part is stored as a document. The document is stored in JSON or XML formats. The value is understood by the DB and can be queried.



**How to Create Database & Collection in MongoDB :**

**Replica Set:**

What is MongoDB Replication?

Replication is referred to the process of ensuring that the same data is available on more than one Mongo DB Server. This is sometimes required for the purpose of increasing data availability.

Because if your main MongoDB Server goes down for any reason, there will be no access to the data. But if you had the data replicated to another server at regular intervals, you will be able to access the data from another server even if the primary server fails.

Step 1).Ensure that all mongo.exe instances can connect to each other. From ServerA, issue the below 2 commands

**mongo –host ServerB –port 27017**

**mongo –host ServerC –port 27017**

Similarly, do the same thing from the remaining servers.

Step 2) Start the first mongod.exe instance with the replSet option. This option provides a grouping for all servers which will be part of this replica set.

**mongo –replSet "Replica1"**

Where "Replica1" is the name of your replica set. You can choose any meaningful name for your replica set name.

To add ServerB and ServerC to the replica set issue the commands

**rs.add("ServerB")**

**rs.add("ServerC")**

Use the rs.remove command to remove the required server from the replica set

**rs.remove("ServerC")**

**Data Sharding:**

Shards are implemented by using clusters which are nothing but a group of MongoDB instances.

**The components of a Shard include:**

**A Shard** – This is the basic thing, and this is nothing but a MongoDB instance which holds the subset of the data. In production environments, all shards need to be part of replica sets.

**Config server** – This is a mongodb instance which holds metadata about the cluster, basically information about the various mongodb instances which will hold the shard data.

**A Router** – This is a mongodb instance which basically is responsible to re-directing the commands send by the client to the right servers.

**Step by Step Sharding Cluster Example**

**Step 1)** Create a separate database for the config server.

**mkdir /data/configdb**

**Step 2)** Start the mongodb instance in configuration mode. Suppose if we have a server named Server D which would be our configuration server, we would need to run the below command to configure the server as a configuration server.

**mongod –configdb ServerD: 27019**

**Step 3)** Start the mongos instance by specifying the configuration server

**mongos –configdb ServerD: 27019**

**Step 4)** From the mongo shell connect to the mongo's instance

**mongo –host ServerD –port 27017**

**Step 5)** If you have Server A and Server B which needs to be added to the cluster, issue the below commands

**sh.addShard("ServerA:27017")**

**sh.addShard("ServerB:27017")**

**Step 6)** Enable sharding for the database. So if we need to shard the Employeedb database, issue the below command

**sh.enableSharding(Employeedb)**

**Step 7)** Enable sharding for the collection. So if we need to shard the Employee collection, issue the below command

**Sh.shardCollection("db.Employee" , { "Employeeid" : 1 , "EmployeeName" : 1})**

**Spring Boot with Bongo DB:**

[**https://www.journaldev.com/18156/spring-boot-mongodb**](https://www.journaldev.com/18156/spring-boot-mongodb)

**1.pom.xml:**

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-mongodb</artifactId> </dependency>

**2.All below statement in SpringBootApplication.java**

@EnableMongoRepositories(basePackageClasses = IssueRepository.class)

**3.Create Document:**

@Document

public class User {

@Id

private String userId;

private String name;

private Date creationDate = new Date(); private Map<String, String> userSettings = new HashMap<>();

**}**

**4.Create Mongo Repository:**

@Repository public interface UserRepository extends MongoRepository<User, String> {

}

**5.Propertiesfile :**

**#Local MongoDB config**

spring.data.mongodb.authentication-database=admin

spring.data.mongodb.username=root

spring.data.mongodb.password=root

spring.data.mongodb.database=user\_db

spring.data.mongodb.port=27017

spring.data.mongodb.host=localhost

**# App config**

server.port=8102

spring.application.name=BootMongo

server.context-path=/user

**6.Repository:**

*@Repository*

*public class UserDALImpl implements UserDAL {*

*@Autowired*

*private MongoTemplate mongoTemplate;*

*@Override*

*public List<User> getAllUsers() {*

*return mongoTemplate.findAll(User.class);*

*}*

*@Override*

*public User getUserById(String userId) {*

*Query query = new Query();*

*query.addCriteria(Criteria.where("userId").is(userId));*

*return mongoTemplate.findOne(query, User.class);*

*}*

*@Override*

*public User addNewUser(User user) {*

***mongoTemplate.save(user);***

*// Now, user object will contain the ID as well*

*return user;*

*}*

*@Override*

*public Object getAllUserSettings(String userId) {*

*Query query = new Query();*

*query.addCriteria(Criteria.where("userId").is(userId));*

*User user = mongoTemplate.findOne(query, User.class);*

*return user != null ? user.getUserSettings() : "User not found.";*

*}*

*@Override*

*public String getUserSetting(String userId, String key) {*

*Query query = new Query();*

*query.fields().include("userSettings");*

*query.addCriteria(Criteria.where("userId").is(userId).andOperator(Criteria.where("userSettings." + key).exists(true)));*

*User user = mongoTemplate.findOne(query, User.class);*

*return user != null ? user.getUserSettings().get(key) : "Not found.";*

*}*

*@Override*

*public String addUserSetting(String userId, String key, String value) {*

*Query query = new Query();*

*query.addCriteria(Criteria.where("userId").is(userId));*

*User user = mongoTemplate.findOne(query, User.class);*

*if (user != null) {*

*user.getUserSettings().put(key, value);*

*mongoTemplate.save(user);*

*return "Key added.";*

*} else {*

*return "User not found.";*

*}*

*}*

*}*