<https://www.tutorialspoint.com/mongodb/index.htm>

<https://github.com/sanjivnaik/mongodb>

Local installation using ZIP file: <https://mkyong.com/mongodb/how-to-install-mongodb-on-windows/>

**In one terminal**

C:\PERSONAL\study\mangoDB\mongodb\bin>mongod.exe --config C:\PERSONAL\study\mangoDB\mongodb\mongo.config

**Another terminal**

C:\PERSONAL\study\mangoDB\mongodb\bin>mongo

MongoDB

# *MongoDB - Overview*

**MongoDB** is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

A single MongoDB server typically has multiple databases.

**Collection**:Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

**Document**:A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

Relationship of RDBMS terminology with MongoDB:

**RDBMS** **MongoDB**

Database Database

Table Collection

Tuple/Row Document

column Field

Table Join Embedded Documents

Primary Key Primary Key (Default key \_id provided by MongoDB itself)

"**\_id** " is a 12 bytes hexadecimal number which assures the uniqueness of every document. You can provide \_id while inserting the document. If you don’t provide then MongoDB provides a unique id for every document. These 12 bytes first 4 bytes for the current timestamp, next 3 bytes for machine id, next 2 bytes for process id of MongoDB server and remaining 3 bytes are simple incremental VALUE.

# *MongoDB - Advantages*

## **Advantages of MongoDB over RDBMS:**

* **Schema less** − MongoDB is a document database in which one collection holds different documents. Number of fields, content and size of the document can differ from one document to another.
* Structure of a single object is clear.
* No complex joins.
* Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.
* Tuning.
* **Ease of scale-out** − MongoDB is easy to scale.
* Conversion/mapping of application objects to database objects not needed.
* Uses internal memory for storing the (windowed) working set, enabling faster access of data.

## **Why Use MongoDB?**

* **Document Oriented Storage** − Data is stored in the form of JSON style documents.
* Index on any attribute
* Replication and high availability
* Auto-Sharding
* Rich queries
* Fast in-place updates
* Professional support by MongoDB

## **Where to Use MongoDB?**

* Big Data
* Content Management and Delivery
* Mobile and Social Infrastructure
* User Data Management
* Data Hub

# *MongoDB - Data Modelling*

MongoDB provides two types of data models: — Embedded data model and Normalized data model. Based on the requirement, you can use either of the models while preparing your document.

### **Embedded Data Model:**

In this model, you can have (embed) all the related data in a single document, it is also known as de-normalized data model.

For example, assume we are getting the details of **employees** in three different documents namely, Personal\_details, Contact and, Address, you can embed all the three documents in a single one as shown below –

{

\_id: ,

Emp\_ID: "10025AE336"

Personal\_details:{

First\_Name: "Radhika",

Last\_Name: "Sharma",

Date\_Of\_Birth: "1995-09-26"

},

Contact: {

e-mail: "radhika\_sharma.123@gmail.com",

phone: "9848022338"

},

Address: {

city: "Hyderabad",

Area: "Madapur",

State: "Telangana"

}

}

### **Normalized Data Model:**

In this model, you can refer the sub documents in the original document, using references. For example, you can re-write the above document in the normalized model as:

Employee:

{

\_id: <ObjectId101>,

Emp\_ID: "10025AE336"

}

Personal\_details:

{

\_id: <ObjectId102>,

empDocID: " ObjectId101",

First\_Name: "Radhika",

Last\_Name: "Sharma",

Date\_Of\_Birth: "1995-09-26"

}

Contact:

{

\_id: <ObjectId103>,

empDocID: " ObjectId101",

e-mail: "radhika\_sharma.123@gmail.com",

phone: "9848022338"

}

Address:

{

\_id: <ObjectId104>,

empDocID: " ObjectId101",

city: "Hyderabad",

Area: "Madapur",

State: "Telangana"

}

## **Considerations while designing Schema in MongoDB:**

* Design your schema according to user requirements.
* Combine objects into one document if you will use them together. Otherwise separate them (but make sure there should not be need of joins).
* Duplicate the data (but limited) because disk space is cheap as compare to compute time.
* Do joins while write, not on read.
* Optimize your schema for most frequent use cases.
* Do complex aggregation in the schema.

# *MongoDB - Create Database*

“***use DATABASE\_NAME***” is used to create database. The command will create a new database if it doesn't exist, otherwise it will return the existing database.

> use mydb

switched to db mydb

> db

mydb

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

Your created database (mydb) is not present in list. To display database, you need to insert at least one document into it.

> db.movie.insert({"name":"tutorials point"})

WriteResult({ "nInserted" : 1 })

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

mydb 0.000GB

# *MongoDB - Drop Database*

MongoDB “***db.dropDatabase()***” command is used to drop a existing database.

> use mydb

switched to db mydb

> db.dropDatabase()

{ "dropped" : "mydb", "ok" : 1 }