# ADVANCED WEB APPLICATION DEVELOPMENT WEEK 7 ASSIGNMENT

Submitted by : Deepthi Jayadevan(EA2232251010046)

1. Explain Schema and Model in MONGODB in details
   * 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.
   * Example
   * Suppose a client needs a database design for his blog/website and see the
   * differences between RDBMS and MongoDB schema design. Website has the
   * following requirements.
   * Every post has the unique title, description and url.
   * Every post can have one or more tags.
   * Every post has the name of its publisher and total number of likes.

●

* + Every post has comments given by users along with their name, message, data-
  + time and likes.

●

* + On each post, there can be zero or more comments.
  + In RDBMS schema, design for above requirements will have minimum three tables.

create a database in MongoDB. The use Command

MongoDB 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.

Syntax

Basic syntax of use DATABASE statement is as follows − use DATABASE\_NAME

create a database in MongoDB. The use Command

MongoDB 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.

Syntax

Basic syntax of use DATABASE statement is as follows − use DATABASE\_NAME

1. Write a MongoDB Database to display all the documents in the collection library. 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.

>use test

switched to db test

>db.createCollection("mycollection")

{ "ok" : 1 }

>

You can check the created collection by using the command show collections.

>show collections

1. How to define the Schema in mongoose?

Everything in Mongoose starts with a Schema. Each schema maps to a MongoDB collection and defines the shape of the documents within that collection.

import mongoose from 'mongoose'; const { Schema } = mongoose;

const blogSchema = new Schema({

title: String, // String is shorthand for {type: String}

author: String,

body: String,

comments: [{ body: String, date: Date }],

date: { type: Date, default: Date.now },

hidden: Boolean,

meta: {

votes: Number,

favs: Number

}

});

If you want to add additional keys later, use the [Schema#add](https://mongoosejs.com/docs/api.html#schema_Schema-add) method.

Each key in our code blogSchema defines a property in our documents which will be cast to its associated [SchemaType](https://mongoosejs.com/docs/api.html#schematype_SchemaType). For example, we've defined a property title which will be cast to the [String](https://mongoosejs.com/docs/api.html#schema-string-js) SchemaType and property date which will be cast to a Date SchemaType.

Notice above that if a property only requires a type, it can be specified using a shorthand notation (contrast the title property above with the date property).

Keys may also be assigned nested objects containing further key/type definitions like the meta property above. This will happen whenever a key's value is a POJO that doesn't have a type property.

In these cases, Mongoose only creates actual schema paths for leaves in the tree. (like meta.votes and meta.favs above), and the branches do not have actual paths. A side-effect of this is that meta above cannot have its own validation. If validation is needed up the tree, a path needs to be created up the tree - see the [Subdocuments](https://mongoosejs.com/docs/subdocs.html) section for more information on how to do this. Also read the [Mixed](https://mongoosejs.com/docs/schematypes.html) subsection of the SchemaTypes guide for some gotchas.

The permitted SchemaTypes are:

* [String](https://mongoosejs.com/docs/schematypes.html#strings)
* [Number](https://mongoosejs.com/docs/schematypes.html#numbers)
* [Date](https://mongoosejs.com/docs/schematypes.html#dates)
* [Buffer](https://mongoosejs.com/docs/schematypes.html#buffers)
* [Boolean](https://mongoosejs.com/docs/schematypes.html#booleans)
* [Mixed](https://mongoosejs.com/docs/schematypes.html#mixed)
* [ObjectId](https://mongoosejs.com/docs/schematypes.html#objectids)
* [Array](https://mongoosejs.com/docs/schematypes.html#arrays)
* [Decimal128](https://mongoosejs.com/docs/api.html#mongoose_Mongoose-Decimal128)
* [Map](https://mongoosejs.com/docs/schematypes.html#maps)