WEEK 8

TLM

DATA MODEL IN MONGODB

The key decision in designing data models for MongoDB applications revolves around the structure of documents and how the application represents relationships between data. MongoDB allows related data to be embedded within a single document.

Embedded Data

Embedded documents capture relationships between data by storing related data in a single document structure. MongoDB documents make it possible to embed document structures in a field or array within a document. These *denormalized* data models allow applications to retrieve and manipulate related data in a single database operation.

For many use cases in MongoDB, the denormalized data model is optimal.

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.

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.

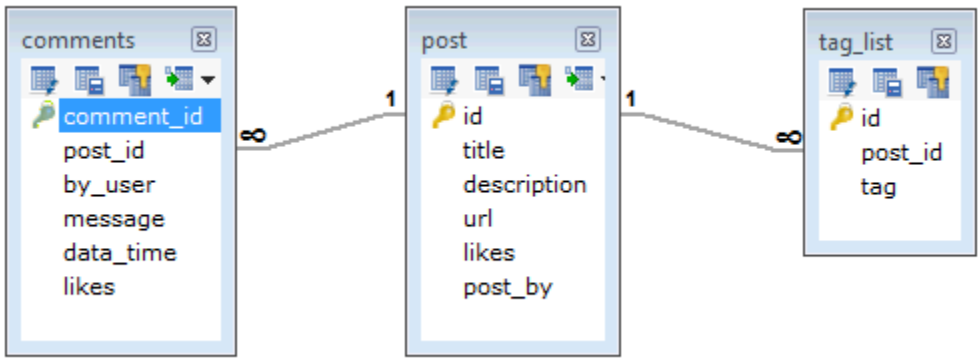
Every post can have one or more tags.

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

Example

If you want to use a database with name **<mydb>**, then **use DATABASE** statement would be as follows −

>use mydb

switched to db mydb

To check your currently selected database, use the command **db**

>db

mydb

If you want to check your databases list, use the command **show dbs**.

>show dbs

local 0.78125GB

test 0.23012GB

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"})
```

```
>show dbs
```

```
local      0.78125GB
mydb       0.23012GB
test       0.23012GB
```

In MongoDB default database is test. If you didn't create any database, then collections will be stored in test database.

Previous Page

he dropDatabase() Method

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

Syntax

Basic syntax of **dropDatabase()** command is as follows −

```
db.dropDatabase()
```

This will delete the selected database. If you have not selected any database, then it will delete default 'test' database.

Example

First, check the list of available databases by using the command, **show dbs**.

```
>show dbs
```

local 0.78125GB

mydb 0.23012GB

test 0.23012GB

>

If you want to delete new database **<mydb>**, then **dropDatabase()** command would be as follows −

>use mydb

switched to db mydb

>db.dropDatabase()

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

>

Now check list of databases.

>show dbs

local 0.78125GB

test 0.23012GB

>

create a collection using MongoDB.

The createCollection() Method

MongoDB **db.createCollection(name, options)** is used to create collection.

Syntax

Basic syntax of **createCollection()** command is as follows −

db.createCollection(name, options)

Examples

Basic syntax of **createCollection()** method without options is as follows −

>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

The drop() Method

MongoDB's **db.collection.drop()** is used to drop a collection from the database.

Syntax

Basic syntax of **drop()** command is as follows −

db.COLLECTION_NAME.drop()