MONGODB NO SQL DOCUMENT DATABASE

Michel Halim Ibrahim
Numéro dossier : 10652 f
Michel.ibrahim@isae.edu.com

TABLE OF CONTENTS

What is MongoDB?

•Scalable High-Performance Open-source, Document-orientated database.

- Built for Speed
- Rich Document based queries for Easy readability.
- Full Index Support for High Performance.
- Replication and Failover for High Availability.
- Auto Sharding for Easy Scalability.
- Map / Reduce for Aggregation.

WHY USE MONGODB?

- SQL was invented in the 70's to store data.

- MongoDB stores documents (or) objects.

- Now-a-days, everyone works with objects (Python/Ruby/Java/etc.)

- And we need Databases to persist our objects. Then why not store objects directly ?

- Embedded documents and arrays reduce need for joins. No Joins and No-multi document transactions.

WHERE TO USE MONGODB?

- Big Data.
- Content Management and Delivery
- Mobile and Social Infrastructure
- User Data Management
- Data Hub

ADVANTAGES

- RDBMS replacement for Web Applications.
- Semi-structured Content Management.
- Real-time Analytics &High-Speed Logging.
- Caching and High Scalability

Web 2.0, Media, SAAS, Gaming HealthCare, Finance, Telecom, Government

DISADVANTAGES

- Highly Transactional Applications.
- Problems requiring SQL.

Some Companies using MongoDB in Production

craigslist

IGN

GILT GROUPE

Eventbrite Events Made Easy

github SOCIAL CODING

intuit

BUSINESS INSIDER

ENVIRONMENT

- Download the latest release of MongoDB from http://www.mongodb.org/downloads
- Extract your downloaded file to c:\ drive or any other location.
- Open the command prompt and run the following command :

```
move mongodb-win64-* mongodb
```

- Navigate to the bin directory present in the MongoDB installation folder. Suppose my installation folder is **D:\set up\mongodb**

```
C:\Users\XYZ>d:
D:\>cd "set up"
D:\set up>cd mongodb
D:\set up\mongodb>cd bin
D:\set up\mongodb\bin>mongod.exe --dbpath "d:\set up\mongodb\data"
```

- Now to run the MongoDB, you need to open another command prompt and issue the following command

```
D:\set up\mongodb\bin>mongo.exe
MongoDB shell version: 2.4.6
connecting to: test
>db.test.save( { a: 1 } )
>db.test.find()
{ "_id" : ObjectId(5879b0f65a56a454), "a" : 1 }
>
```

LET’S DIVE IN!

Data in MongoDB has a flexible schema.documents in the same collection.

Some 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.

A. Every post has the unique title, description and URL.

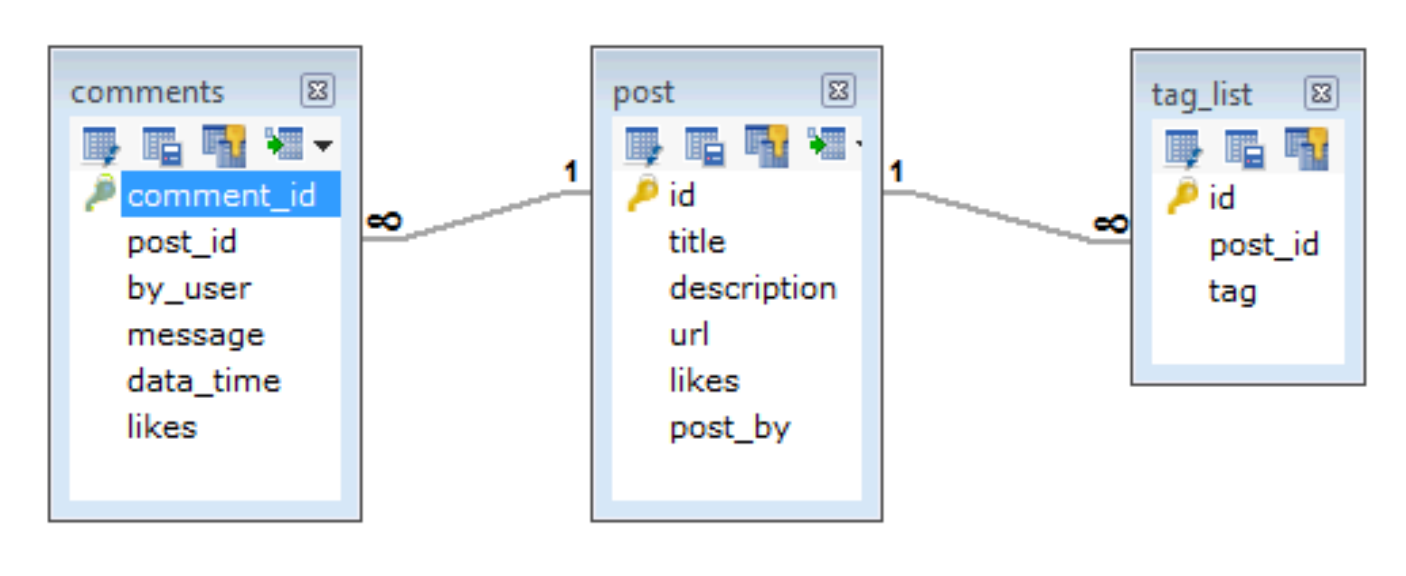
B. Every post can have one or more tags.

C. Every post has the name of its publisher and total number of likes.

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

E. On each post, there can be zero or more comments.

In RDBMS schema, design for above requirements will have minimum three tables.



While in MongoDB schema, design will have one collection post and the following structure:

```
{
   _id: POST_ID
   title: TITLE_OF_POST,
   description: POST_DESCRIPTION,
   by: POST_BY,
   url: URL_OF_POST,
   tags: [TAG1, TAG2, TAG3],
   likes: TOTAL_LIKES,
   comments: [
      {
         user:'COMMENT_BY',
         message: TEXT,
         dateCreated: DATE_TIME,
         like: LIKES
      },
      {
         user:'COMMENT_BY',
         message: TEXT,
         dateCreated: DATE_TIME,
         like: LIKES
      }
   ]
}
```

So while showing the data, in RDBMS you need to join three tables and in MongoDB, data will be shown from one collection only.

CREATE DATABASE AND DROP DATABASE

The use Command :

- **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

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

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

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

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

The 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.

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)
```

In the command, **name** is name of collection to be created. **Options** is a document and is used to specify configuration of collection.

Parameter	Type	Description
Name	String	Name of the collection to be created
Options	Document	(Optional) Specify options about memory size and indexing

Options parameter is optional, so you need to specify only the name of the collection.

The DropCollection() Method

MongoDB **db. Collection.drop()** is used to removes the specified collection as well as removes any indexes associated with the dropped collection from the database.

MongoDB — Datatypes

Types	Number	Used for
Double	1	store floating point values.
String	2	easily store most international characters
Object	3	stores embedded documents
Array	4	store multiples of values and data types
Binary data	5	store the binary data in it.
Undefined	6	stores the undefined values.
Object Id	7	stores the unique key Id of documents stored
Boolean	9	stores Boolean values i.e. true/false.

MongoDB — Datatypes

Types	Number	Used for
Date	10	stores current date or time.
Null	11	stores a null value in it.
Regular Expression	12	maps directly to JavaScript RegExp
JavaScript	13	store the JavaScript data without a scope
symbol	14	similar to the string data type.
JavaScript with scope	15	store JavaScript data with a scope.
Integer	16 and 18	store an integer value.
timestamp	10	store a timestamp.
Min key	255	compares the value of the lowest BSON element
Max key	127	compares the value of the highes BSON element

MongoDB — Insert Document

db.collection.insertOne() inserts a single document into a collection.

Example:

```
db.inventory.insertOne(
   { item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" }
  }
)
```

The following example inserts a new document into the inventory collection. If the document does not specify an _id field, MongoDB adds the _id field with an ObjectId value to the new document

To retrieve the document that you just inserted, query the collection:

```
db.inventory.find( { item: "canvas" } )
```

MongoDB — Insert Multiple Document

db.collection.insertMany() can insert multiple documents into a collection.
Pass an array of documents to the method.

Example:

```
db.inventory.insertMany([
   { item: "journal", qty: 25, tags: ["blank", "red"], size: { h: 14, w: 21, uom: "cm" } },
   { item: "mat", qty: 85, tags: ["gray"], size: { h: 27.9, w: 35.5, uom: "cm" } },
   { item: "mousepad", qty: 25, tags: ["gel", "blue"], size: { h: 19, w: 22.85, uom: "cm" } }
])
```

The following example inserts three new documents into the inventory collection. If the documents do not specify an _id field, MongoDB adds the _id field with an ObjectId value to each document.

MongoDB – Query Document

Select All Documents in a Collection

To select all documents in the collection, pass an empty document as the query filter parameter to the find method. The query filter parameter determines the select criteria:

```
db.inventory.find( {} )
```

This operation corresponds to the following SQL statement:

```
SELECT * FROM inventory
```

MongoDB — Query Document

The following example selects from the inventory collection all documents where the status equals "D":

```
db.inventory.find( { status: "D" } )
```

This operation corresponds to the following SQL statement:

```
SELECT * FROM inventory WHERE status = "D"
```

MongoDB – Query Document

Specify Conditions Using Query Operators

The following example retrieves all documents from the inventory collection where status equals either "A" or "D":

```
db.inventory.find( { status: { $in: [ "A", "D" ] } } )
```

The operation corresponds to the following SQL statement:

```
SELECT * FROM inventory WHERE status in ("A", "D")
```

MongoDB — Query Document

Specify AND Conditions

The following example retrieves all documents in the inventory collection where the status equals "A" and qty is less than ($lt) 30:

```
db.inventory.find( { status: "A", qty: { $lt: 30 } } )
```

The operation corresponds to the following SQL statement:

```
SELECT * FROM inventory WHERE status = "A" AND qty < 30
```

MongoDB — Query Document

Specify OR Conditions

The following example retrieves all documents in the collection where the status equals "A" or qty is less than ($lt) 30:

```
db.inventory.find( { $or: [ { status: "A" }, { qty: { $lt: 30 } } ] } )
```

The operation corresponds to the following SQL statement:

```
SELECT * FROM inventory WHERE status = "A" OR qty < 30
```

MongoDB — Query Document

Specify AND as well as OR Conditions

In the following example, the compound query document selects all documents in the collection where the status equals "A" and either qty is less than ($lt) 30 or item starts with the character p:

```
db.inventory.find( {
    status: "A",
    $or: [ { qty: { $lt: 30 } }, { item: /^p/ } ]
    } )
```

The operation corresponds to the following SQL statement:

```
SELECT * FROM inventory WHERE status = "A" AND ( qty < 30 OR
                                                 item LIKE "p% ")
```

MongoDB — Update Document

Update a Single Document

The following example uses the db.collection.updateOne() method on the inventory collection to update the first document where item equals "paper":

```
db.inventory.updateOne(
   { item: "paper" },
   {
    $set: { "size.uom": "cm", status: "P" },
    $currentDate: { lastModified: true }
   }
)
```

MongoDB — Update Document

Update Multiple Documents

The following example uses the db.collection.updateMany() method on the inventory collection to update all documents where qty is less than 50:

```
db.inventory.updateMany(
   { "qty": { $lt: 50 } },
   {
     $set: { "size.uom": "in", status: "P" },
     $currentDate: { lastModified: true }
   }
)
```

- $set operator to update the value of the size.uom field to "in" and the value of the status field to "P“
- $currentDate operator to update the value of the lastModified field to the current date. If lastModified field does not exist, $currentDate will create the field.

MongoDB – Delete Document

1. The following example deletes all documents from the inventory collection:

```
db.inventory.deleteMany({})
```

2. The following example removes all documents from the inventory collection where the status field equals "A":

```
db.inventory.deleteMany({ status : "A" })
```

3. The following example deletes the first document where status is "D":

```
db.inventory.deleteOne( { status: "D" } )
```