**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.

**Database**

Database is a physical container for collections. Each database gets its own set of files on the file system. 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.

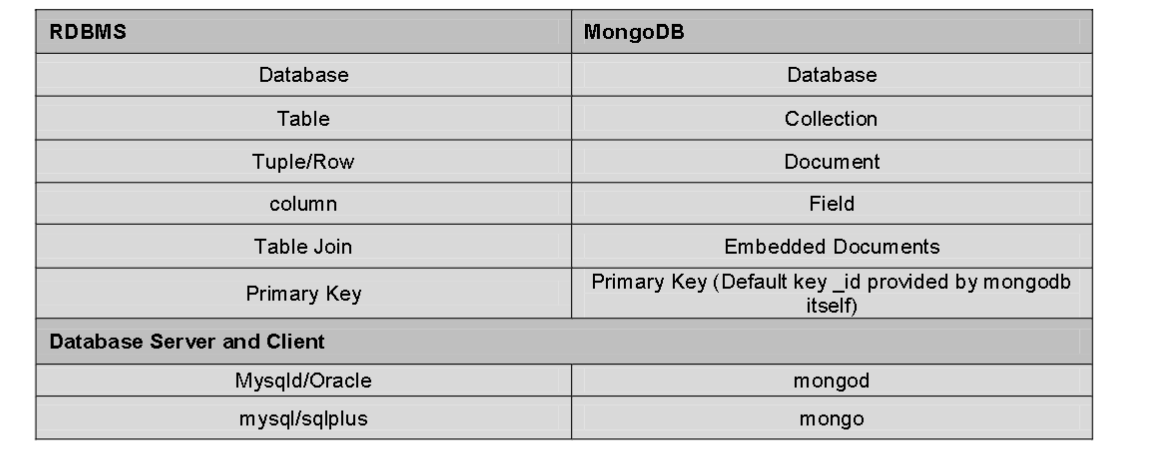
**Document**

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.

Below given table shows the relationship of RDBMS terminology with MongoDB



Below given example shows the document structure of a blog site which is simply a comma separated key value pair.

{

\_id: ObjectId(7df78ad8902c)

title: 'MongoDB Overview',

description: 'MongoDB is no sql database',

by: 'tutorials point',

url: 'http://www.tutorialspoint.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 100,

comments: [

{

user:'user1',

message: 'My first comment',

dateCreated: new Date(2011,1,20,2,15),

like: 0

},

{

user:'user2',

message: 'My second comments',

dateCreated: new Date(2011,1,25,7,45),

like: 5

}

]

}

\_id is a 12 bytes hexadecimal number which assures the uniqueness of every document. You can provide \_id

while inserting the document. If you didn't provide then MongoDB provide 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**

Any relational database has a typical schema design that shows number of tables and the relationship between

these tables. While in MongoDB there is no concept of relationship

**Advantages of MongoDB over RDBMS**

Schema less : MongoDB is document database in which one collection holds different different documents.

Number of fields, content and size of the document can be 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 should use MongoDB**

Document Oriented Storage : Data is stored in the form of JSON style documents

Index on any attribute

Replication & High Availability

Auto-Sharding

Rich Queries

Fast In-Place Updates

Professional Support By MongoDB

Where should use MongoDB?

Big Data

Content Management and Delivery

Mobile and Social Infrastructure

User Data Management

Data Hub

**MongoDB Environment**

Install MongoDB On Windows

To install the MongoDB on windows, first doownload the latest release of MongoDB from

http://www.mongodb.org/downloads Make sure you get correct version of MongoDB depending upon your windows

version. To get your windows version open command prompt and execute following command

C:\>wmic os get osarchitecture

OSArchitecture

64-bit

C:\>

32-bit versions of MongoDB only support databases smaller than 2GB and suitable only for testing and evaluation

purposes.

Now extract your downloaded file to c:\ drive or any other location. Make sure name of the extracted folder is

mongodb-win32-i386-[version] or mongodb-win32-x86\_64-[version]. Here [version] is the version of MongoDB

download.

Now open command prompt and run the following command

C:\>move mongodb-win64-\* mongodb

1 dir(s) moved.

C:\>

In case you have extracted the mondodb at different location, then go to that path by using command cd

FOOLDER/DIR and now run the above given process.

MongoDB requires a data folder to store its files. The default location for the MongoDB data directory is c:\data\db. So

you need to create this folder using the Command Prompt. Execute the following command sequence

C:\>md data

C:\md data\db

If you have install the MongoDB at different location, then you need to specify any alternate path for \data\db by

setting the path dbpath in mongod.exe. For the same issue following commands

In command prompt navigate to the bin directory present into 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"

This will show waiting for connections message on the console output indicates that the mongod.exe process is

running successfully.

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 }

>

This will show that mongodb is installed and run successfully. Next time when you run mongodb you need to issue

only commands

D:\set up\mongodb\bin>mongod.exe --dbpath "d:\set up\mongodb\data"

D:\set up\mongodb\bin>mongo.exe

Install MongoDB on Ubuntu

Run the following command to import the MongoDB public GPG Key:

sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10

Create a /etc/apt/sources.list.d/mongodb.list file using the following command.

echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen' | sudo tee

/etc/apt/sources.list.d/mongodb.list

Now issue the following command to update the repository:

sudo apt-get update

Now install the MongoDB by using following command:

apt-get install mongodb-10gen=2.2.3

In the above installation 2.2.3 is currently released mongodb version. Make sure to install latest version always. Now

mongodb is installed successfully.

Start MongoDB

sudo service mongodb start

Stop MongoDB

sudo service mongodb stop

Restart MongoDB

sudo service mongodb restart

To use mongodb run the following command

mongo

This will connect you to running mongod instance.

MongoDB Help

To get list of commands type db.help() in mongodb client. This will give you list of commands as follows:

To get stats about mongodb server type the command db.stats() in mongodb client. This will show the database

name, cumber of collection and documents in the database. Output the command is shown below:

MongoDB Data Modelling

Data in MongoDB has a flexible schema.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.

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.

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 have 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.

While in MongoDB schema design will have one collection post and has 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.

TUTORIALS POINT

Simply Easy Learning Page 15

MongoDB Create Database

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 create 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, then 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 atleast 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.

MongoDB Drop Database

The dropDatabase () Method

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

Syntax:

Basic syn tax 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 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

>

MongoDB Create Collection

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 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 name of the collection. Following is the list of options you

can use:

Field Type Description

capped Boolean

(Optional) If true, enables a capped collection. Capped

collection is a collection fixed size collecction that

automatically overwrites its oldest entries when it reaches

its maximum size. If you specify true, you need to

specify size parameter also.

autoIndexID Boolean (Optional) If true, automatically create index on \_id field.s

Default value is false.

size number

(Optional) Specifies a maximum size in bytes for a capped

collection. If If capped is true, then you need to specify

this field also.

max number (Optional) Specifies the maximum number of documents

allowed in the capped collection.

While inserting the document, MongoDB first checks size field of capped collection, then it checks max field.

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

mycollection

system.indexes

Following example shows the syntax of createCollection() method with few important options:

>db.createCollection("mycol", { capped : true, autoIndexID : true, size : 6142800, max : 10000 } )

{ "ok" : 1 }

>

In mongodb you don't need to create collection. MongoDB creates collection automatically, when you insert some

document.

>db.tutorialspoint.insert({"name" : "tutorialspoint"})

>show collections

mycol

mycollection

system.indexes

tutorialspoint

>

**MongoDB Drop Collection**

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()

Example:

First, check the available collections into your database mydb

>use mydb

switched to db mydb

>show collections

mycol

mycollection

system.indexes

tutorialspoint

>

Now drop the collection with the name mycollection

>db.mycollection.drop()

true

>

Again check the list of collections into database

>show collections

mycol

system.indexes

tutorialspoint

>

drop() method will return true, if the selected collection is dropped successfully otherwise it will return false

CHAPTER

8

TUTORIALS POINT

Simply Easy Learning Page 20

MongoDB Datatypes

MongoDB supports many datatypes whose list is given below:

String : This is most commonly used datatype to store the data. String in mongodb must be UTF-8 valid.

Integer : This type is used to store a numerical value. Integer can be 32 bit or 64 bit depending upon your server.

Boolean : This type is used to store a boolean (true/ false) value.

Double : This type is used to store floating point values.

Min/ Max keys : This type is used to compare a value against the lowest and highest BSON elements.

Arrays : This type is used to store arrays or list or multiple values into one key.

Timestamp : ctimestamp. This can be handy for recording when a document has been modified or added.

Object : This datatype is used for embedded documents.

Null : This type is used to store a Null value.

Symbol : This datatype is used identically to a string however, it's generally reserved for languages that use a

specific symbol type.

Date : This datatype is used to store the current date or time in UNIX time format. You can specify your own date

time by creating object of Date and passing day, month, year into it.

Object ID : This datatype is used to store the document’s ID.

Binary data : This datatype is used to store binay data.

Code : This datatype is used to store javascript code into document.

Regular expression : This datatype is used to store regular expression

CHAPTER

9

TUTORIALS POINT

Simply Easy Learning Page 21

MongoDB - Insert Document

The insert() Method

To insert data into MongoDB collection, you need to use MongoDB's insert() or save()method.

Syntax

Basic syntax of insert() command is as follows:

>db.COLLECTION\_NAME.insert(document)

Example

>db.mycol.insert({

\_id: ObjectId(7df78ad8902c),

title: 'MongoDB Overview',

description: 'MongoDB is no sql database',

by: 'tutorials point',

url: 'http://www.tutorialspoint.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 100

})

Here mycol is our collection name, as created in previous tutorial. If the collection doesn't exist in the database, then

MongoDB will create this collection and then insert document into it.

In the inserted document if we don't specify the \_id parameter, then MongoDB assigns an unique ObjectId for this

document.

\_id is 12 bytes hexadecimal number unique for every document in a collection. 12 bytes are divided as follows:

\_id: ObjectId(4 bytes timestamp, 3 bytes machine id, 2 bytes process id, 3 bytes incrementer)

To insert multiple documents in single query, you can pass an array of documents in insert() command.

Example

CHAPTER

10

TUTORIALS POINT

Simply Easy Learning Page 22

>db.post.insert([

{

title: 'MongoDB Overview',

description: 'MongoDB is no sql database',

by: 'tutorials point',

url: 'http://www.tutorialspoint.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 100

},

{

title: 'NoSQL Database',

description: 'NoSQL database doesn't have tables',

by: 'tutorials point',

url: 'http://www.tutorialspoint.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 20,

comments: [

{

user:'user1',

message: 'My first comment',

dateCreated: new Date(2013,11,10,2,35),

like: 0

}

]

}

])

To insert the document you can use db.post.save(document) also. If you don't specify \_id in the document

then save() method will work same as insert() method. If you specify \_id then it will replace whole data of

document containing \_id as specified in save() method.

TUTORIALS POINT

Simply Easy Learning Page 23

MongoDB - Query Document

The find() Method

To query data from MongoDB collection, you need to use MongoDB's find() method.

Syntax

Basic syntax of find() method is as follows

>db.COLLECTION\_NAME.find()

find() method will display all the documents in a non structured way.

The pretty() Method

To display the results in a formatted way, you can use pretty() method.

Syntax:

>db.mycol.find().pretty()

Example

>db.mycol.find().pretty()

{

"\_id": ObjectId(7df78ad8902c),

"title": "MongoDB Overview",

"description": "MongoDB is no sql database",

"by": "tutorials point",

"url": "http://www.tutorialspoint.com",

"tags": ["mongodb", "database", "NoSQL"],

"likes": "100"

}

>

Apart from find() method there is findOne() method, that reruns only one document.

RDBMS Where Clause Equivalents in MongoDB

To query the document on the basis of some condition, you can use following operations

Operation Syntax Example RDBMS

Equivalent

Equality {<key>:<value>} db.mycol.find({"by":"tuto

rials point"}).pretty()

where by =

'tutorials point'

Less Than {<key>:{$lt:<value>}} db.mycol.find({"likes":{$l

t:50}}).pretty() where likes < 50

Less Than

Equals

{<key>:{$lte:<value>}} db.mycol.find({"likes":{$l

te:50}}).pretty()

where likes <=

50

Greater Than {<key>:{$gt:<value>}} db.mycol.find({"likes":{$

gt:50}}).pretty() where likes > 50

Greater Than

Equals

{<key>:{$gte:<value>}} db.mycol.find({"likes":{$

gte:50}}).pretty()

where likes >=

50

Not Equals {<key>:{$ne:<value>}} db.mycol.find({"likes":{$

ne:50}}).pretty()

where likes !=

50

AND in MongoDB

Syntax:

In the find() method if you pass multiple keys by separating them by ',' then MongoDB treats it AND condition. Basic

syntax of AND is shown below:

>db.mycol.find({key1:value1, key2:value2}).pretty()

Example

Below given example will show all the tutorials written by 'tutorials point' and whose title is 'MongoDB Overview'

>db.mycol.find({"by":"tutorials point","title": "MongoDB Overview"}).pretty()

{

"\_id": ObjectId(7df78ad8902c),

"title": "MongoDB Overview",

"description": "MongoDB is no sql database",

"by": "tutorials point",

"url": "http://www.tutorialspoint.com",

"tags": ["mongodb", "database", "NoSQL"],

"likes": "100"

}

>

For the above given example equivalent where clause will be ' where by='tutorials point' AND title='MongoDB

Overview' '. You can pass any number of key, value pairs in find clause.

OR in MongoDB

Syntax:

To query documents based on the OR condition, you need to use $or keyword. Basic syntax of OR is shown below:

>db.mycol.find(

{

$or: [

{key1: value1}, {key2:value2}

]

}

).pretty()

Example

Below given example will show all the tutorials written by 'tutorials point' or whose title is 'MongoDB Overview'

>db.mycol.find({$or:[{"by":"tutorials point"},{"title": "MongoDB Overview"}]}).pretty()

{

"\_id": ObjectId(7df78ad8902c),

"title": "MongoDB Overview",

"description": "MongoDB is no sql database",

"by": "tutorials point",

"url": "http://www.tutorialspoint.com",

"tags": ["mongodb", "database", "NoSQL"],

"likes": "100"

}

>

Using AND and OR together

Example

Below given example will show the documents that have likes greater than 100 and whose title is either 'MongoDB

Overview' or by is 'tutorials point'. Equivalent sql where clause is 'where likes>10 AND (by = 'tutorials point' OR title

= 'MongoDB Overview')'

>db.mycol.find("likes": {$gt:10}, $or: [{"by": "tutorials point"}, {"title": "MongoDB Overview"}] }).pretty()

{

"\_id": ObjectId(7df78ad8902c),

"title": "MongoDB Overview",

"description": "MongoDB is no sql database",

"by": "tutorials point",

"url": "http://www.tutorialspoint.com",

"tags": ["mongodb", "database", "NoSQL"],

"likes": "100"

}

>

**MongoDB Update Document**

MongoDB's update() and save() methods are used to update document into a collection. The update() method

update values in the existing document while the save() method replaces the existing document with the document

passed in save() method.

MongoDB Update() method

The update() method updates values in the existing document.

Syntax:

Basic syntax of update() method is as follows

>db.COLLECTION\_NAME.update(SELECTIOIN\_CRITERIA, UPDATED\_DATA)

Example

Consider the mycol collectioin has following data.

{ "\_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}

{ "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}

{ "\_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}

Following example will set the new title 'New MongoDB Tutorial' of the documents whose title is 'MongoDB Overview'

>db.mycol.update({'title':'MongoDB Overview'},{$set:{'title':'New MongoDB Tutorial'}})

>db.mycol.find()

{ "\_id" : ObjectId(5983548781331adf45ec5), "title":"New MongoDB Tutorial"}

{ "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}

{ "\_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}

>

By default mongodb will update only single document, to update multiple you need to set a paramter 'multi' to true.

>db.mycol.update({'title':'MongoDB Overview'},{$set:{'title':'New MongoDB Tutorial'}},{multi:true})

**MongoDB Save() Method**

The save() method replaces the existing document with the new document passed in save() method

Syntax

Basic syntax of mongodb save() method is shown below:

>db.COLLECTION\_NAME.save({\_id:ObjectId(),NEW\_DATA})

Example

Following example will replace the document with the \_id '5983548781331adf45ec7'

>db.mycol.save(

{

"\_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point New Topic", "by":"Tutorials

Point"

}

)

>db.mycol.find()

{ "\_id" : ObjectId(5983548781331adf45ec5), "title":"Tutorials Point New Topic", "by":"Tutorials

Point"}

{ "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}

{ "\_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}

>

**MongoDB Delete Document**

The remove() Method

MongoDB's remove() method is used to remove document from the collection. remove() method accepts two

parameters. One is deletion criteria and second is justOne flag

1. deletion criteria : (Optional) deletion criteria according to documents will be removed.

2. justOne : (Optional) if set to true or 1, then remove only one document.

Syntax:

Basic syntax of remove() method is as follows

>db.COLLECTION\_NAME.remove(DELLETION\_CRITTERIA)

Example

Consider the mycol collectioin has following data.

{ "\_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}

{ "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}

{ "\_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}

Following example will remove all the documents whose title is 'MongoDB Overview'

>db.mycol.remove({'title':'MongoDB Overview'})

>db.mycol.find()

{ "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}

{ "\_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}

>

Remove only one

If there are multiple records and you want to delete only first record, then set justOne parameter in remove() method

>db.COLLECTION\_NAME.remove(DELETION\_CRITERIA,1)

Remove All documents

If you don't specify deletion criteria, then mongodb will delete whole documents from the collection. This is equivalent

of SQL's truncate command.

>db.mycol.remove()

>db.mycol.find()

>

**MongoDB Projection**

In mongodb projection meaning is selecting only necessary data rather than selecting whole of the data of a

document. If a document has 5 fields and you need to show only 3, then select only 3 fields from them.

The find() Method

MongoDB's find() method, explained in MongoDB Query Document accepts second optional parameter that is list of

fields that you want to retrieve. In MongoDB when you execute find() method, then it displays all fields of a document.

To limit this you need to set list of fields with value 1 or 0. 1 is used to show the filed while 0 is used to hide the field.

Syntax:

Basic syntax of find() method with projection is as follows

>db.COLLECTION\_NAME.find({},{KEY:1})

Example

Consider the collection myycol has the following data

{ "\_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}

{ "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}

{ "\_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}

Following example will display the title of the document while quering the document.

>db.mycol.find({},{"title":1,\_id:0})

{"title":"MongoDB Overview"}

{"title":"NoSQL Overview"}

{"title":"Tutorials Point Overview"}

>

Please note \_id field is always displayed while executing find() method, if you don't want this field, then you need to

set it as 0

**MongoDB Limit Records**

The Limit() Method

To limit the records in MongoDB, you need to use limit() method. limit() method accepts one number type argument,

which is number of documents that you want to displayed.

Syntax:

Basic syntax of limit() method is as follows

>db.COLLECTION\_NAME.find().limit(NUMBER)

Example

Consider the collection myycol has the following data

{ "\_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}

{ "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}

{ "\_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}

Following example will display only 2 documents while quering the document.

>db.mycol.find({},{"title":1,\_id:0}).limit(2)

{"title":"MongoDB Overview"}

{"title":"NoSQL Overview"}

>

If you don't specify number argument in limit() method then it will display all documents from the collection.

MongoDB Skip() Method

Apart from limit() method there is one more method skip() which also accepts number type argument and used to

skip number of documents.

Syntax:

Basic syntax of skip() method is as follows

>db.COLLECTION\_NAME.find().limit(NUMBER).skip(NUMBER)

Example:

Following example will only display only second document.

>db.mycol.find({},{"title":1,\_id:0}).limit(1).skip(1)

{"title":"NoSQL Overview"}

>

Please note default value in skip() method is 0

**MongoDB Sort Documents**

The sort() Method

To sort documents in MongoDB, you need to use sort() method. sort() method accepts a document containing list of

fields along with their sorting order. To specify sorting order 1 and -1 are used. 1 is used for ascending order while -1

is used for descending order.

Syntax:

Basic syntax of sort() method is as follows

>db.COLLECTION\_NAME.find().sort({KEY:1})

Example

Consider the collection myycol has the following data

{ "\_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}

{ "\_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}

{ "\_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}

Following example will display the documents sorted by title in descending order.

>db.mycol.find({},{"title":1,\_id:0}).sort({"title":-1})

{"title":"Tutorials Point Overview"}

{"title":"NoSQL Overview"}

{"title":"MongoDB Overview"}

>

Please note if you don't specify the sorting preference, then sort() method will display documents in ascending order.

CHAPTER

**MongoDB Indexing**

Indexes support the efficient resolution of queries. Without indexes, MongoDB must scan every document of a

collection to select those documents that match the query statement. This scan is highly inefficient and require the

mongod to process a large volume of data.

Indexes are special data structures, that store a small portion of the data set in an easy to traverse form. The index

stores the value of a specific field or set of fields, ordered by the value of the field as specified in index.

The ensureIndex() Method

To create an index you need to use ensureIndex() method of mongodb.

Syntax:

Basic syntax of ensureIndex() method is as follows()

>db.COLLECTION\_NAME.ensureIndex({KEY:1})

Here key is the name of filed on which you want to create index and 1 is for ascending order. To create index in

descending order you need to use -1.

Example

>db.mycol.ensureIndex({"title":1})

>

In ensureIndex() method you can pass multiple fields, to create index on multiple fields.

>db.mycol.ensureIndex({"title":1,"description":-1})

>

ensureIndex() method also accepts list of options (which are optional), whose list is given below:

**Parameter Type Description**

background Boolean

Builds the index in the background so that building an index

does not block other database activities. Specify true to build

in the background. The default value is false.

unique Boolean

Creates a unique index so that the collection will not accept

insertion of documents where the index key or keys match

an existing value in the index. Specify true to create a unique

index. The default value is false.

name string

The name of the index. If unspecified, MongoDB generates

an index name by concatenating the names of the indexed

fields and the sort order.

dropDups Boolean

Creates a unique index on a field that may have duplicates.

MongoDB indexes only the first occurrence of a key and

removes all documents from the collection that contain

subsequent occurrences of that key. Specify true to create

unique index. The default value is false.

sparse Boolean

If true, the index only references documents with the

specified field. These indexes use less space but behave

differently in some situations (particularly sorts). The default

value is false.

expireAfterSeconds integer Specifies a value, in seconds, as a TTL to control how long

MongoDB retains documents in this collection.

v index

version

The index version number. The default index version

depends on the version of mongod running when creating

the index.

weights document

The weight is a number ranging from 1 to 99,999 and

denotes the significance of the field relative to the other

indexed fields in terms of the score.

default\_language string

For a text index, the language that determines the list of stop

words and the rules for the stemmer and tokenizer. The

default value is english.

language\_override string

For a text index, specify the name of the field in the

document that contains, the language to override the default

language. The default value is language.

**MongoDB Aggregation**

Aggregations operations process data records and return computed results. Aggregation operations group

values from multiple documents together, and can perform a variety of operations on the grouped data to return a

single result. In sql count(\*) and with group by is an equivalent of mongodb aggregation.

The aggregate() Method

For the aggregation in mongodb you should use aggregate() method.

Syntax:

Basic syntax of aggregate() method is as follows

>db.COLLECTION\_NAME.aggregate(AGGREGATE\_OPERATION)

Example:

In the collection you have the following data:

{

\_id: ObjectId(7df78ad8902c)

title: 'MongoDB Overview',

description: 'MongoDB is no sql database',

by\_user: 'tutorials point',

url: 'http://www.tutorialspoint.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 100

},

{

\_id: ObjectId(7df78ad8902d)

title: 'NoSQL Overview',

description: 'No sql database is very fast',

by\_user: 'tutorials point',

url: 'http://www.tutorialspoint.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 10

},

{

\_id: ObjectId(7df78ad8902e)

title: 'Neo4j Overview',

description: 'Neo4j is no sql database',

by\_user: 'Neo4j',

url: 'http://www.neo4j.com',

tags: ['neo4j', 'database', 'NoSQL'],

likes: 750

},

Now from the above collection if you want to display a list that how many tutorials are written by each user then you

will use aggregate() method as shown below:

> db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$sum : 1}}}])

{

"result" : [

{

"\_id" : "tutorials point",

"num\_tutorial" : 2

},

{

"\_id" : "tutorials point",

"num\_tutorial" : 1

}

],

"ok" : 1

}

>

TUTORIALS POINT

Simply Easy Learning Page 37

Sql equivalent query for the above use case will be select by\_user, count(\*) from mycol group by by\_user

In the above example we have grouped documents by field by\_user and on each occurance of by\_user previous

value of sum is incremented. There is a list available aggregation expressions .

Expression Description Example

$sum Sums up the defined value from all documents in the

collection.

db.mycol.aggregate([{$group : {\_id

: "$by\_user", num\_tutorial : {$sum :

"$likes"}}}])

$avg Calculates the average of all given values from all

documents in the collection.

db.mycol.aggregate([{$group : {\_id

: "$by\_user", num\_tutorial : {$avg :

"$likes"}}}])

$min Gets the minimum of the corresponding values from all

documents in the collection.

db.mycol.aggregate([{$group : {\_id

: "$by\_user", num\_tutorial : {$min :

"$likes"}}}])

$max Gets the maximum of the corresponding values from all

documents in the collection.

db.mycol.aggregate([{$group : {\_id

: "$by\_user", num\_tutorial : {$max :

"$likes"}}}])

$push Inserts the value to an array in the resulting document. db.mycol.aggregate([{$group : {\_id

: "$by\_user", url : {$push: "$url"}}}])

$addToSet Inserts the value to an array in the resulting document

but does not create duplicates.

db.mycol.aggregate([{$group : {\_id

: "$by\_user", url : {$addToSet :

"$url"}}}])

$first

Gets the first document from the source documents

according to the grouping. Typically this makes only

sense together with some previously applied “$sort”-

stage.

db.mycol.aggregate([{$group : {\_id

: "$by\_user", first\_url : {$first :

"$url"}}}])

$last

Gets the last document from the source documents

according to the grouping. Typically this makes only

sense together with some previously applied “$sort”-

stage.

db.mycol.aggregate([{$group : {\_id

: "$by\_user", last\_url : {$last :

"$url"}}}])

Pipeline Concept

In UNIX command shell pipeline means the possibility to execute an operation on some input and use the output as

the input for the next command and so on. MongoDB also support same concept in aggregation framework. There is

a set of possible stages and each of those is taken a set of documents as an input and is producing a resulting set of

documents (or the final resulting JSON document at the end of the pipeline). This can then in turn again be used for

the next stage an so on.

Possible stages in aggregation framework are following:

$project: Used to select some specific fields from a collection.

$match: This is a filtering operation and thus this can reduce the amount of documents that are given as input to

the next stage.

$group: This does the actual aggregation as discussed above.

$sort: Sorts the documents.

$skip: With this it is possible to skip forward in the list of documents for a given amount of documents.

$limit: This limits the amount of documents to look at by the given number starting from the current position.s

$unwind: This is used to unwind document that are using arrays. when using an array the data is kind of pre-

joinded and this operation will be undone with this to have individual documents again. Thus with this stage we

will increase the amount of documents for the next stage.

**MongoDB Replication**

Replication is the process of synchronizing data across multiple servers. Replication provides redundancy and

increases data availability with multiple copies of data on different database servers, replication protects a database

from the loss of a single server. Replication also allows you to recover from hardware failure and service interruptions.

With additional copies of the data, you can dedicate one to disaster recovery, reporting, or backup.

Why Replication?

To keep your data safe

High (24\*7) availability of data

Disaster Recovery

No downtime for maintenance (like backups, index rebuilds, compaction)

Read scaling (extra copies to read from)

Replica set is transparent to the application

How replication works in MongoDB

MongoDB achieves replication by the use of replica set. A replica set is a group of mongod instances that host the

same data set. In a replica one node is primary node that receives all write operations. All other instances,

secondaries, apply operations from the primary so that they have the same data set. Replica set can have only one

primary node.

1. Replica set is a group of two or more nodes (generally minimum 3 nodes are required).

2. In a replica set one node is primary node and remaining nodes are secondary.

3. All data replicates from primary to secondary node.

4. At the time of automatic failover or maintenance, election establishes for primary and a new primary node is

elected.

5. After the recovery of failed node, it again join the replica set and works as a secondary node.

A typical diagram of mongodb replication is shown in which client application always interact with primary node and

primary node then replicate the data to the secondary nodes.

**Replica set features**

A cluster of N nodess

Anyone node can be primary

All write operations goes to primary

Automatic failover

Automatic Recovery

Consensus election of primary

Set up a replica set

In this tutorial we will convert standalone mongod instance to a replica set. To convert to replica set follow the below

given steps:

Shutdown already running mongodb server.

Now start the mongodb server by specifying --replSet option. Basic syntax of --replSet is given below:

mongod --port "PORT" --dbpath "YOUR\_DB\_DATA\_PATH" --replSet "REPLICA\_SET\_INSTANCE\_NAME"

Example

mongod --port 27017 --dbpath "D:\set up\mongodb\data" --replSet rs0

It will start a mongod instance with the name rs0, on port 27017. Now start the command prompt and connect to this

mongod instance. In mongo client issue the command rs.initiate() to initiate a new replica set. To check the replica

set configuration issue the command rs.conf(). To check the status of replica sete issue the command rs.status().

Add members to replica set

To add members to replica set, start mongod instances on multiple machines. Now start a mongo client and issue a

command rs.add().

Synttax:

Basic syntax of rs.add() command is as follows:

>rs.add(HOST\_NAME:PORT)

Example

Suppose your mongod instance name is mongod1.net and it is running on port 27017. To add this instance to replica

set issue the command rs.add() in mongo client.

>rs.add("mongod1.net:27017")

>

You can add mongod instance to replica set only when you are connected to primary node. To check whether

you are connected to primary or not issue the command db.isMaster() in mongo client.

TUTORIALS POINT

Simply Easy Learning Page 41

MongoDB Sharding

Sharding

Sharding is the process of storing data records across multiple machines and it is MongoDB's approach to meeting

the demands of data growth. As the size of the data increases, a single machine may not be sufficient to store the

data nor provide an acceptable read and write throughput. Sharding solves the problem with horizontal scaling. With

sharding, you add more machines to support data growth and the demands of read and write operations.

Why Sharding?

In replication all writes go to master node

Latency sensitive queries still go to master

Single replica set has limitation of 12 nodes

Memory can't be large enough when active dataset is big

Local Disk is not big enough

Vertical scaling is too expensive

Sharding in MongoDB

Below given diagram shows the sharding in MongoDB using sharded cluster.

In the above given diagram there are three main components which are described below:

Shards: Shards are used to store data. They provide high availability and data consistency. In production

environment each shard is a separate replica set.

Config Servers: Config servers store the cluster's metadata. This data contains a mapping of the cluster's data

set to the shards. The query router uses this metadata to target operations to specific shards. In production

environment sharded clusters have exactly 3 config servers.

Query Routers: Query Routers are basically mongos instances, interface with client applications and direct

operations to the appropriate shard. The query router processes and targets operations to shards and then

returns results to the clients. A sharded cluster can contain more than one query router to divide the client request

load. A client sends requests to one query router. Generally a sharded cluster have many query routers.

**MongoDB Create Backup**

Dump MongoDB Data

To create backup of database in mongodb you should use mongodump command. This command will dump all data

of your server into dump directory. There are many options available by which you can limit the amount of data or

create backup of your remote server.

Syntax:

Basic syntax of mongodump command is as follows

>mongodump

Example

Start your mongod server. Assuming that your mongod server is running on localhost and port 27017. Now open a

command prompt and go to bin directory of your mongodb instance and type the command mongodump

Consider the mycol collectioin has following data.

>mongodump

The command will connect to the server running at 127.0.0.1 and port 27017 and back all data of the server to

directory /bin/dump/. Output of the command is shown below:

There are a list of available options that can be used with the mongodump command.

This command will backup only specified database at specified path

Syntax Description Example

mongodump --host HOST\_NAME –port

PORT\_NUMBER

This commmand will

backup all databases of

specified mongod instance.

mongodump --host

tutorialspoint.com –

port 27017

mongodump --dbpath DB\_PATH --out

BACKUP\_DIRECTORY

mongodump --dbpath

/data/db/ --out

/data/backup/

mongodump --collection COLLECTION

--db DB\_NAME

This command will backup

only specified collection of

specified database.

mongodump --collection

mycol --db test

Restore data

To restore backup data mongodb's mongorerstore command is used. This command restore all of the data from the

back up directory.

Syntax

Basic syntax of mongorestore command is

>mongorestore

Output of the command is shown below:

MongoDB Deployment

When you are preparing a MongoDB deployment, you should try to understand how your application is going

to hold up in production. It’s a good idea to develop a consistent, repeatable approach to managing your deployment

environment so that you can minimize any surprises once you’re in production.

The best approach incorporates prototyping your set up, conducting load testing, monitoring key metrics, and using

that information to scale your set up. The key part of the approach is to proactively monitor your entire system - this

will help you understand how your production system will hold up before deploying, and determine where you will

need to add capacity. Having insight into potential spikes in your memory usage, for example, could help put out a

write-lock fire before it starts.

To monitor your deployment MongoDB provides some commands that are shown below:

mongostat

This command checks the status of all running mongod instances and return counters of database operations. These

counters include inserts, queries, updates, deletes, and cursors. Command also shows when you’re hitting page

faults, and showcase your lock percentage. This means that you're running low on memory, hitting write capacity or

have some performance issue.

To run the command start your mongod instance. In another command prompt go to bin directory of your mongodb

installation and type mongostat.

D:\set up\mongodb\bin>mongostat

Output of the command is shown below:

**Mongotop**

This command track and report the read and write activity of MongoDB instance on a collection basis. By default

mongotop returns information in each second, by you can change it accordingly. You should check that this read

and write activity matches your application intention, and you’re not firing too many writes to the database at a time,

reading too frequently from disk, or are exceeding your working set size.

To run the command start your mongod instance. In another command prompt go to bin directory of your mongodb

installation and type mongotop.

D:\set up\mongodb\bin>mongotop

Output of the command is shown below:

TUTORIALS POINT

Simply Easy Learning Page 47

To change mongotop command to return information less frequently specify a specific number after the mongotop

command.

D:\set up\mongodb\bin>mongotop 30

The above example will return values every 30 seconds.

Apart from the mongodb tools, 10gen provides a free, hosted monitoring service MongoDB Management Service

(MMS), that provides a dashboard and gives you a view of the metrics from your entire cluster.

**MongoDB Java**

Installation

Before we start using MongoDB in our Java programs, we need to make sure that we have MongoDB JDBC Driver

and Java set up on the machine. You can check Java tutorial for Java installation on your machine. Now, let us check

how to set up MongoDB JDBC driver.

You need to download the jar from the path Download mongo.jar. Make sure to download latest release of it.

You need to include the mongo.jar into your classpath.

Connect to database

To connect database, you need to specify database name, if database doesn't exist then mongodb creates it

automatically.

Code snippets to connect to database would be as follows:

import com.mongodb.MongoClient;

import com.mongodb.MongoException;

import com.mongodb.WriteConcern;

import com.mongodb.DB;

import com.mongodb.DBCollection;

import com.mongodb.BasicDBObject;

import com.mongodb.DBObject;

import com.mongodb.DBCursor;

import com.mongodb.ServerAddress;

import java.util.Arrays;

public class MongoDBJDBC{

public static void main( String args[] ){

try{

// To connect to mongodb server

MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

// Now connect to your databases

DB db = mongoClient.getDB( "test" );

System.out.println("Connect to database successfully");

boolean auth = db.authenticate(myUserName, myPassword);

System.out.println("Authentication: "+auth);

}catch(Exception e){

System.err.println( e.getClass().getName() + ": " + e.getMessage() );

}

}

}

Now, let's compile and run above program to create our database test. You can change your path as per your

requirement. We are assuming current version of JDBC driver mongo-2.10.1.jar is available in the current path

$javac MongoDBJDBC.java

$java -classpath ".:mongo-2.10.1.jar" MongoDBJDBC

Connect to database successfully

Authentication: true

If you are going to use Windows machine, then you can compile and run your code as follows:

$javac MongoDBJDBC.java

$java -classpath ".;mongo-2.10.1.jar" MongoDBJDBC

Connect to database successfully

Authentication: true

Value of auth will be true, if the user name and password are valid for the selected database.

Create a collection

To create a collection, createCollection() method of com.mongodb.DB class is used.

Code snippets to create a collection:

import com.mongodb.MongoClient;

import com.mongodb.MongoException;

import com.mongodb.WriteConcern;

import com.mongodb.DB;

import com.mongodb.DBCollection;

import com.mongodb.BasicDBObject;

import com.mongodb.DBObject;

import com.mongodb.DBCursor;

import com.mongodb.ServerAddress;

import java.util.Arrays;

public class MongoDBJDBC{

public static void main( String args[] ){

try{

// To connect to mongodb server

MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

// Now connect to your databases

DB db = mongoClient.getDB( "test" );

System.out.println("Connect to database successfully");

boolean auth = db.authenticate(myUserName, myPassword);

System.out.println("Authentication: "+auth);

DBCollection coll = db.createCollection("mycol");

System.out.println("Collection created successfully");

}catch(Exception e){

System.err.println( e.getClass().getName() + ": " + e.getMessage() );

}

}

}

When program is compiled and executed, it will produce the following result::

Connect to database successfully

Authentication: true

Collection created successfully

Getting/ selecting a collection

To get/select a collection from the database, getCollection() method of com.mongodb.DBCollection class is used.

Code snippets to get/select a collection:

import com.mongodb.MongoClient;

import com.mongodb.MongoException;

import com.mongodb.WriteConcern;

import com.mongodb.DB;

import com.mongodb.DBCollection;

import com.mongodb.BasicDBObject;

import com.mongodb.DBObject;

import com.mongodb.DBCursor;

import com.mongodb.ServerAddress;

import java.util.Arrays;

public class MongoDBJDBC{

public static void main( String args[] ){

try{

// To connect to mongodb server

MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

// Now connect to your databases

DB db = mongoClient.getDB( "test" );

System.out.println("Connect to database successfully");

boolean auth = db.authenticate(myUserName, myPassword);

System.out.println("Authentication: "+auth);

DBCollection coll = db.createCollection("mycol");

System.out.println("Collection created successfully");

DBCollection coll = db.getCollection("mycol");

System.out.println("Collection mycol selected successfully");

}catch(Exception e){

System.err.println( e.getClass().getName() + ": " + e.getMessage() );

}

}

}

When program is compiled and executed, it will produce the following result::

Connect to database successfully

Authentication: true

Collection created successfully

Collection mycol selected successfully

Insert a document

To insert a document into mongodb, insert() method of com.mongodb.DBCollection class is used.

Code snippets to insert a documents :

import com.mongodb.MongoClient;

import com.mongodb.MongoException;

import com.mongodb.WriteConcern;

import com.mongodb.DB;

import com.mongodb.DBCollection;

import com.mongodb.BasicDBObject;

import com.mongodb.DBObject;

import com.mongodb.DBCursor;

import com.mongodb.ServerAddress;

import java.util.Arrays;

public class MongoDBJDBC{

public static void main( String args[] ){

try{

// To connect to mongodb server

MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

// Now connect to your databases

DB db = mongoClient.getDB( "test" );

System.out.println("Connect to database successfully");

boolean auth = db.authenticate(myUserName, myPassword);

System.out.println("Authentication: "+auth);

DBCollection coll = db.getCollection("mycol");

System.out.println("Collection mycol selected successfully");

BasicDBObject doc = new BasicDBObject("title", "MongoDB").

append("description", "database").

append("likes", 100).

append("url", "http://www.tutorialspoint.com/mongodb/").

append("by", "tutorials point");

coll.insert(doc);

System.out.println("Document inserted successfully");

}catch(Exception e){

System.err.println( e.getClass().getName() + ": " + e.getMessage() );

}

}

}

When program is compiled and executed, it will produce the following result: :

Connect to database successfully

Authentication: true

Collection mycol selected successfully

Document inserted successfully

Retrieve all documents

To select all documents from the collection, find() method of com.mongodb.DBCollection class is used. This

method returns a cursor, so you need to iterate this cursor.

Code snippets to select all documents:

import com.mongodb.MongoClient;

import com.mongodb.MongoException;

import com.mongodb.WriteConcern;

import com.mongodb.DB;

import com.mongodb.DBCollection;

import com.mongodb.BasicDBObject;

import com.mongodb.DBObject;

import com.mongodb.DBCursor;

import com.mongodb.ServerAddress;

import java.util.Arrays;

public class MongoDBJDBC{

public static void main( String args[] ){

try{

// To connect to mongodb server

MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

// Now connect to your databases

DB db = mongoClient.getDB( "test" );

System.out.println("Connect to database successfully");

boolean auth = db.authenticate(myUserName, myPassword);

System.out.println("Authentication: "+auth);

DBCollection coll = db.getCollection("mycol");

System.out.println("Collection mycol selected successfully");

DBCursor cursor = coll.find();

int i=1;

while (cursor.hasNext()) {

System.out.println("Inserted Document: "+i);

System.out.println(cursor.next());

i++;

}

}catch(Exception e){

System.err.println( e.getClass().getName() + ": " + e.getMessage() );

}

}

}

When program is compiled and executed, it will produce the following result::

Connect to database successfully

Authentication: true

Collection mycol selected successfully

Inserted Document: 1

{

"\_id" : ObjectId(7df78ad8902c),

"title": "MongoDB",

"description": "database",

"likes": 100,

"url": "http://www.tutorialspoint.com/mongodb/",

"by": "tutorials point"

}

Update document

To update document from the collection, update() method of com.mongodb.DBCollection class is used.

Code snippets to select first document:

import com.mongodb.MongoClient;

import com.mongodb.MongoException;

import com.mongodb.WriteConcern;

import com.mongodb.DB;

import com.mongodb.DBCollection;

import com.mongodb.BasicDBObject;

import com.mongodb.DBObject;

import com.mongodb.DBCursor;

import com.mongodb.ServerAddress;

import java.util.Arrays;

public class MongoDBJDBC{

public static void main( String args[] ){

try{

// To connect to mongodb server

MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

// Now connect to your databases

DB db = mongoClient.getDB( "test" );

System.out.println("Connect to database successfully");

boolean auth = db.authenticate(myUserName, myPassword);

System.out.println("Authentication: "+auth);

DBCollection coll = db.getCollection("mycol");

System.out.println("Collection mycol selected successfully");

DBCursor cursor = coll.find();

while (cursor.hasNext()) {

DBObject updateDocument = cursor.next();

updateDocument.put("likes","200")

col1.update(updateDocument);

}

System.out.println("Document updated successfully");

cursor = coll.find();

int i=1;

while (cursor.hasNext()) {

System.out.println("Updated Document: "+i);

System.out.println(cursor.next());

i++;

}

}catch(Exception e){

System.err.println( e.getClass().getName() + ": " + e.getMessage() );

}

}

}

When program is compiled and executed, it will produce the following result:

Connect to database successfully

Authentication: true

Collection mycol selected successfully

Document updated successfully

Updated Document: 1

{

"\_id" : ObjectId(7df78ad8902c),

"title": "MongoDB",

"description": "database",

"likes": 100,

"url": "http://www.tutorialspoint.com/mongodb/",

"by": "tutorials point"

}

Delete first document

To delete first document from the collection, you need to first select the documents using findOne() method and then

remove method of com.mongodb.DBCollection class.

Code snippets to delete first document:

import com.mongodb.MongoClient;

import com.mongodb.MongoException;

import com.mongodb.WriteConcern;

import com.mongodb.DB;

import com.mongodb.DBCollection;

import com.mongodb.BasicDBObject;

import com.mongodb.DBObject;

import com.mongodb.DBCursor;

import com.mongodb.ServerAddress;

import java.util.Arrays;

public class MongoDBJDBC{

public static void main( String args[] ){

try{

// To connect to mongodb server

MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

// Now connect to your databases

DB db = mongoClient.getDB( "test" );

System.out.println("Connect to database successfully");

boolean auth = db.authenticate(myUserName, myPassword);

System.out.println("Authentication: "+auth);

DBCollection coll = db.getCollection("mycol");

System.out.println("Collection mycol selected successfully");

DBObject myDoc = coll.findOne();

col1.remove(myDoc);

DBCursor cursor = coll.find();

int i=1;

while (cursor.hasNext()) {

System.out.println("Inserted Document: "+i);

System.out.println(cursor.next());

i++;

}

System.out.println("Document deleted successfully");

}catch(Exception e){

System.err.println( e.getClass().getName() + ": " + e.getMessage() );

}

}

}

When program is compiled and executed, it will produce the following result::

Connect to database successfully

Authentication: true

Collection mycol selected successfully

Document deleted successfully

Remaining mongodb methods save(), limit(), skip(), sort() etc works same as explained in subsequent

tutorial.

MongoDB PHP

To use mongodb with php you need to use mongodb php driver. Download the driver from the url Download

PHP Driver. Make sure to download latest release of it. Now unzip the archive and put php\_mongo.dll in your PHP

extension directory ("ext" by default) and add the following line to your php.ini file:

extension=php\_mongo.dll

Make a connection and Select a database

To make a connection, you need to specify database name, if database doesn't exist then mongodb creates it

automatically.

Code snippets to connect to database would be as follows:

<?php

// connect to mongodb

$m = new MongoClient();

echo "Connection to database successfully";

// select a database

$db = $m->mydb;

echo "Database mydb selected";

?>

When program is executed, it will produce the following result::

Connection to database successfully

Database mydb selected

Create a collection

Code snippets to create a collection would be as follows:

<?php

// connect to mongodb

$m = new MongoClient();

echo "Connection to database successfully";

// select a database

$db = $m->mydb;

echo "Database mydb selected";

$collection = $db->createCollection("mycol");

echo "Collection created succsessfully";

?>

When program is executed, it will produce the following result: :

Connection to database successfully

Database mydb selected

Collection created succsessfully

Insert a document

To insert a document into mongodb, insert() method is used.

Code snippets to insert a documents :

<?php

// connect to mongodb

$m = new MongoClient();

echo "Connection to database successfully";

// select a database

$db = $m->mydb;

echo "Database mydb selected";

$collection = $db->mycol;

echo "Collection selected succsessfully";

$document = array(

"title" => "MongoDB",

"description" => "database",

"likes" => 100,

"url" => "http://www.tutorialspoint.com/mongodb/",

"by", "tutorials point"

);

$collection->insert($document);

echo "Document inserted successfully";

?>

When program is executed, it will produce the following result::

Connection to database successfully

Database mydb selected

Collection selected succsessfully

Document inserted successfully

Find all documents

To select all documents from the collection, find() method is used.

Code snippets to select all documents:

<?php

// connect to mongodb

$m = new MongoClient();

echo "Connection to database successfully";

// select a database

$db = $m->mydb;

echo "Database mydb selected";

$collection = $db->mycol;

echo "Collection selected succsessfully";

$cursor = $collection->find();

// iterate cursor to display title of documents

foreach ($cursor as $document) {

echo $document["title"] . "\n";

}

?>

When program is executed, it will produce the following result:

Connection to database successfully

Database mydb selected

Collection selected succsessfully

{

"title": "MongoDB"

}

Update a document

To update a document , you need to use update() method.

In the below given example we will update the title of inserted document to MongoDB Tutorial. Code snippets to

update a document:

TUTORIALS POINT

Simply Easy Learning Page 60

<?php

// connect to mongodb

$m = new MongoClient();

echo "Connection to database successfully";

// select a database

$db = $m->mydb;

echo "Database mydb selected";

$collection = $db->mycol;

echo "Collection selected succsessfully";

// now update the document

$collection->update(array("title"=>"MongoDB"), array('$set'=>array("title"=>"MongoDB Tutorial")));

echo "Document updated successfully";

// now display the updated document

$cursor = $collection->find();

// iterate cursor to display title of documents

echo "Updated document";

foreach ($cursor as $document) {

echo $document["title"] . "\n";

}

?>

When program is executed, it will produce the following result:

Connection to database successfully

Database mydb selected

Collection selected succsessfully

Document updated successfully

Updated document

{

"title": "MongoDB Tutorial"

}

Delete a document

To delete a document , you need to use remove() method.

In the below given example we will remove the documents that has title MongoDB Tutorial. Code snippets to delete

document:

<?php

// connect to mongodb

$m = new MongoClient();

echo "Connection to database successfully";

// select a database

$db = $m->mydb;

echo "Database mydb selected";

$collection = $db->mycol;

echo "Collection selected succsessfully";

// now remove the document

$collection->remove(array("title"=>"MongoDB Tutorial"),false);

echo "Documents deleted successfully";

// now display the available documents

$cursor = $collection->find();

// iterate cursor to display title of documents

echo "Updated document";

foreach ($cursor as $document) {

echo $document["title"] . "\n";

}

?>

When program is executed, it will produce the following result: :

Connection to database successfully

Database mydb selected

Collection selected succsessfully

Documents deleted successfully

In the above given example second parameter is boolean type and used for justOne field of remove() method.

Remaining mongodb methods findOne(), save(), limit(), skip(), sort() etc works same as explained in above tutorial.