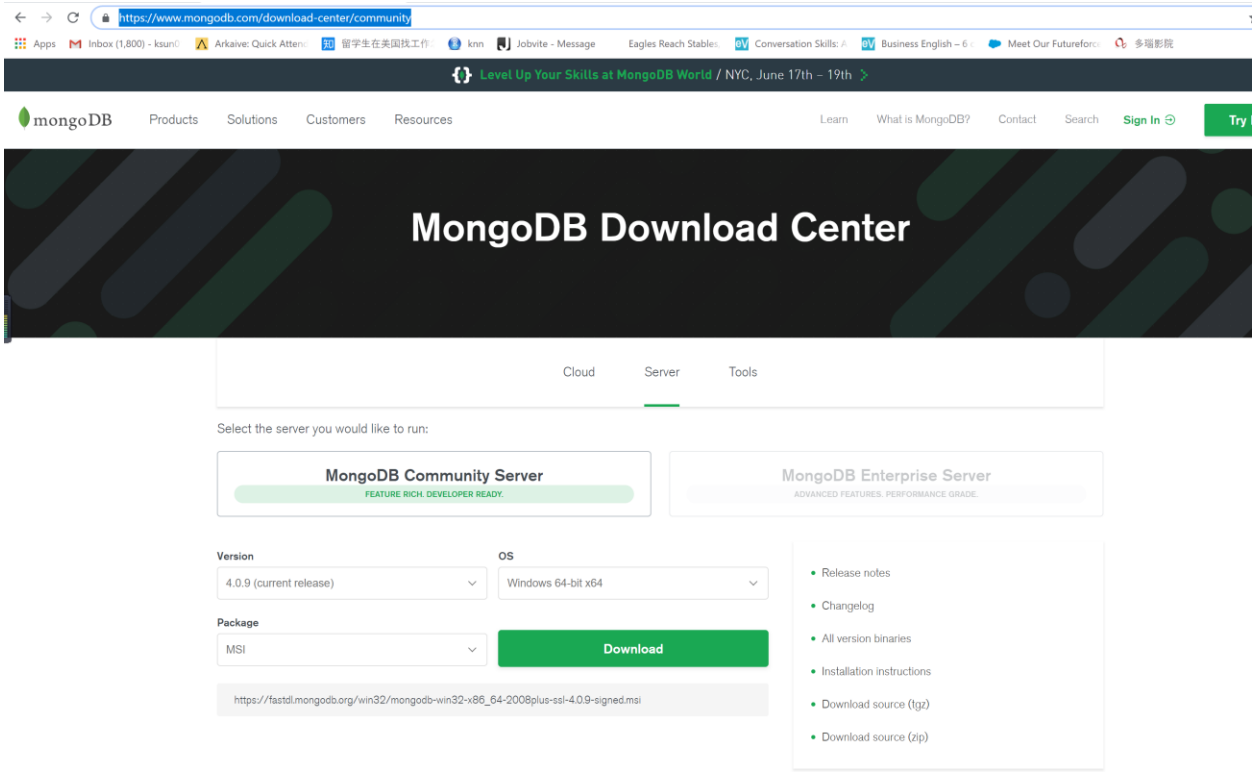
MongoDB

Learning Notes by Kaitan Sun

Content

Installation

- Install MongoDB Community Server (Windows) from official website: https://www.mongodb.com/download-center/community



- Check the path of the installed file and copy it
  - C:\Program Files\MongoDB\Server\4.0\bin
- Create a file named “data” and a file named “db” inside the file named “data”.
  - C:\data
  - C:\data\db
- Open Commend Prompt
  - Enter the copied file path: “cd C:\Program Files\MongoDB\Server\4.0\bin”
  - Type “mongod”
- Open a new window of Commend Prompt
  - Type “cd C:\Program Files\MongoDB\Server\4.0\bin”
  - Type “mongo”
  - Type “show dbs”
- Create a path in control panel
  - Open “Control Panel”
  - Select “System and Security”
  - Select “System”

- Select “system properties”, “advanced”, “environment variables”
- create a new path “C:\Program Files\MongoDB\Server\4.0\bin” under “system variables”

Basic Knowledge

- Database – “Database” in Relational Database
- Collection – “Table” in Relational Database
- Document – “Record” in Relational Database

Task 1: Enter/exit from MongoDB

Open Commend Prompt, then type:

```
> mongo  //Enter MongoDB program in Commend Prompt
> help   //get help file
> exit   //exit from MongoDB
```

Task 2: Create databases/collections

```
> mongo  //Enter MongoDB program in Commend Prompt
> show dbs; //View database
> use Kaitan; //Create a new database called " Kaitan"
> show collections; //View collections
> db.createCollection("posts"); // Create collections in “Kaitan” database
> db.createCollection("categories");
> db.createCollection("tags");
> show collections;
> show dbs;
> db.stats();     //View the parameters of the current database
> db.dropDatabase(); //Drop the current database
> show dbs;
```

Task 3: Rename a collection, drop collections/databases

```
> show dbs;
> use Kaitan;
> show collections;
> db.createCollection("users");
> show collections;
> db.users.renameCollection("staff");      //rename a collection
> show collections;
> db.staff.drop();        //drop a collection
> show collections;
> db.dropDatabase();
> show dbs;
```

Task 4: Create/view/count/delete documents in collections

```
> mongo
> use kaitan;
> show collections;
> db.createCollection("posts");
> db.posts.insert(
... {
...     title: "blog 1.",
...     content: "It is a great start. I am excited."
... }
... );   //insert a document in collection “posts”

> show collections;
> db.posts.find();        //view documents in collection “posts”
> db.posts.insert(
... {
...     title: "blog 2.",
...     content: "What should I write?",
...     tag: ["Not classified"]
... }
... );

> db.posts.find();
> for(var i = 3; i <=10; i++ ) {
...     db.posts.insert({
...         title: "Blog " + i + "."
...     });
... }    //use a loop to create multiple documents in collection “posts”
```

```
> db.posts.find();
> db.posts.count();          //count how many documents are there in this collection
> db.posts.remove({});       //remove all the documents in the collection
> db.posts.count();
> db.posts.find();
```

Task 5: Retrieve documents

- db.[collection_name].find({"":""}) //retrieve documents with one condition
- $gte, $gt, $lte, $lt //greater than or equal to, greater than, less than or equal to, less than
- $eq, $ne //equal to, not equal to
- Regular expression: /k/, /^k/
- db.[collection_name].distinct("field_name"); //like "select"

```
> mongo
> use kaitan;
> db.posts.remove({});  //delete all documents in collection "posts"
> db.posts.insert({title:"How do I learn deep learning?","rank":2,"tag":"AI"});
> db.posts.insert({title:"Be a smart engineer","rank":1,"tag":"CS"});
> db.posts.insert({title:" Which will be the next language","rank":3,"tag":"IT"});
> db.posts.insert({title:" Will automation lead to loss of
jobs","rank":4,"tag":"IT"});
> db.posts.insert({title:" Who is leading in AI research","rank":7,"tag":"AI"});
> db.posts.insert({title:" Is deep learning overhyped","rank":4,"tag":"AI"});
> db.posts.find({"tag": "AI"});  //in collection "posts", select
> db.posts.find({"rank": {$gte: 4}});  //find documents whose rank is greater than or equal to
4
> db.posts.find({"rank": {$gt: 4}});
> db.posts.find({"rank": {$lte: 4}});
> db.posts.find({"rank": {$lt: 4}});
> db.posts.find({"title": /u/});  //find documents that contains "u" in its "title"
> db.posts.find({"title": /^B/}); //find documents that starts with "B" in "title"
> db.posts.find({"title": /^W/}); //find documents that starts with "W" in "title"
> db.posts.distinct("tag");  //list all the unique "tag"
```

- db.[collection_name].find({"":"", "":""}) // retrieve documents with multiple conditions
- db.[collection_name].find({$or:[{...},{...}]}); // retrieve documents if any of the following conditions is satisfied

- db.[collection_name].find({"": {$in: [...]}}); // retrieve documents if certain values are included in certain tags
- db.[collection_name].find({"": {$exists: true}}); // retrieve documents if certain tags exist in certain collections

```
> mongo
> use kaitan;
> db.posts.find();
> db.posts.find({"title": /u/, "rank":{$gte:5} });          //select where the following  two conditions are satisfied
> db.posts.find({$or: [{"title": /u/}, {"rank":{$gte:4}}] });       //select where any of the following conditions is satisfied
> db.posts.find({"rank": {$in: [3,4]} }); //select if "rank" contains 3 or 4
> db.posts.insert({ "title":"The greatest Transaction", "istop": true });  //insert a new document with a new attribute "istop"
> db.posts.find({"istop": {$exists: true} });      // retrieve document if it contains the attribute "istop"
```

Task 6: Show certain parts of documents

```
> mongo
> use kaitan;
> db.posts.find();
> db.posts.find({}, {title:true, rank:1});          //true=1, show title and rank
> db.posts.find({}, {title:true, rank:1, _id:0}); //0, not show id
```

Task 7: Sort, skip, limit

```
> mongo
> use kaitan;
> db.posts.find();
> db.posts.find({}, {_id:0}).sort({rank:1});       //sort in default sequence without showing "_id"
> db.posts.find({}, {_id:0}).sort({rank:-1});      //sort in reversed sequence
> db.posts.find({}, {_id:0}).limit(3);     //show first three results
> db.posts.find({}, {_id:0}).sort({rank:-1}).limit(3);      //show last three results
```

```
> db.posts.findOne({}, {_id:0});     //show the first result
> db.posts.find({}, {_id:0});       //show all the results
> db.posts.find({}, {_id:0}).limit(3);      //show first three results
> db.posts.find({}, {_id:0}).skip(3).limit(3);       //skip the first three results, show next three results
```

Task 8: Update

Format: db.posts.update({"tag":"it"}, {$set: {"rank": 60}}, {multi: true});

- filter
- content
- multiple/single

Reference: https://docs.mongodb.com/manual/reference/method/db.collection.update

```
> mongo
> use kaitan;
> db.posts.findOne({"title":"Greatest Work"});      //check original document
> db.posts.update({"title":" Greatest Work"}, {$set: {"rank": 10} });        //set the rank of it to be 10
> db.posts.find();        //check the updated document
> db.posts.update({"title":" Greatest Work"}, {"rank": 99});         //if "$set" is not written, the whole document will be changed and destroyed!
> db.posts.find();
> db.posts.update({"tag":"it"}, {$set: {"rank": 50}});       //if multiple documents satisfied the filter condition, only the first one will be updated
> db.posts.find();
> db.posts.update({"tag":"it"}, {$set: {"rank": 60}}, {multi: true});        //to update all the documents that satisfy the condition, use the third parameter "multi:true" to do so
> db.posts.find();
```

Task 9: Inc, mul, rename, set, unset

- $inc: increase
- $mul: multiple
- $rename: rename
- $set: create/update
- $unset: delete

```
> mongo
> use kaitan;
> db.posts.find({title:"Greatest Work"}, {_id:0});
> db.posts.update({title:"Greatest Work"}, {$inc: {rank: 1}});      //add 1 to "rank"
> db.posts.find({title:"Greatest Work"}, {_id:0});
> db.posts.update({title:"Greatest Work"}, {$mul: {rank: 2}});      //multiple 2 to "rank"
> db.posts.find({title:"Greatest Work"}, {_id:0});
> db.posts.update({title:"Greatest Work"}, {$rename: {"rank": "score"}});   //rename
"rank" to "score", only for "Greatest Work"
> db.posts.find({title:"Greatest Work"}, {_id:0});
> db.posts.update({title:"Greatest Work"}, {$set: {"istop": true}});        //If "istop" is
not a current attribute of this document, create it automatically. If exists, update it.
> db.posts.find({title:"Greatest Work"}, {_id:0});
> db.posts.update({title:"Greatest Work"}, {$unset: {"istop": true}});      //Delete the
"istop"
> db.posts.find({title:"Greatest Work"}, {_id:0});
```

Task 10: Upsert, remove

- upsert: update and insert
- remove: delete with conditions

```
> mongo
> use komablog;
> db.posts.find({}, {_id:0});
> db.posts.update({title:"Deep Learning Update"}, {title:"Deep Learning Update",
"rank":5,"tag":"game"}); //update only if the filter condition is satisfied
> db.posts.find({}, {_id:0});
> db.posts.update({title:"Deep Learning Update"},{title:"Deep Learning Update",
"rank":5,"tag":"game"}, {upsert:true});    //with "upsert: true", if filter condition is not satisfied,
insert this document
> db.posts.find({}, {_id:0});
> db.posts.update({title:"Deep Learning Update"},{title:"Deep Learning Update",
"rank":7,"tag":"game"}, {upsert:true});
> db.posts.find({}, {_id:0});
> db.posts.remove({title:"Deep Learning Update"});          //delete certain documents
> db.posts.find({}, {_id:0});
```

Task 11: Get, create, drop indexes

- getIndexes()
- createIndex({...}, {...})
- dropIndex({...})

```
> mongo
> use komablog;
> db.posts.getIndexes(); //view all the indexes (there are some default indexes)
> db.posts.createIndex({rank:-1});          //create index by rank in descent order
> db.posts.getIndexes();
> db.posts.dropIndex({rank:-1}); //delete the index
> db.posts.getIndexes();
> db.posts.createIndex({title:1}, {unique:true}); //create index by unique title in ascent order
> db.posts.getIndexes();
> db.posts.find({}, {_id:0});
> db.posts.insert({title:"Greatest Work"});          //if unique index/key is created, no duplicate index can be inserted
```

Task 12: Mongodump, mongorestore

```
$ mongo
> show dbs;
> use kaitan;
> db.posts.find({}, {_id:0});
> exit                    //check the original collections and exit monogoDB
$ mkdir dbbak             //create a new folder named dbbak
$ cd dbbak                //enter the folder
$ mongodump -d kaitan     //back up the database "kaitan"
$ ls                      // lists the files in the current working directory
$ mongo kaitan            // shortcut key to enter a database of monogoDB
> db.posts.find({}, {_id:0});
> db.posts.remove({});    //remove all the documents in "posts" collection
> db.posts.find({}, {_id:0});
> exit
$ mongorestore --drop     //drop the current data and restore backup data
$ mongo kaitan
> db.posts.find({}, {_id:0});      //check whether restored
> exit
$ mongodump --help
```

Reference

- https://www.youtube.com/watch?v=N513d7QjIvg
- https://www.bilibili.com/video/av24311263/?p=1