LAB2

Mongo Shell

Contents

- Create and Drop Database
- Create and Drop Collection
- Insert Document
- Update Document
  - set, unset, inc
- Import Dataset
- Backup and Restore
- Close

Get started

- Remind
  - Start MongoDB: sudo service mongodb start
  - Stop MongoDB: sudo service mongodb stop
  - Restart MongoDB: sudo service mongodb restart
  - Mongo shell: mongo

Get started

- help command
  - Case sensitive

```
> help
        db.help()                    help on db methods
        db.mycoll.help()             help on collection methods
        sh.help()                    sharding helpers
        rs.help()                    replica set helpers
        help admin                   administrative help
        help connect                 connecting to a db help
        help keys                    key shortcuts
        help misc                    misc things to know
        help mr                      mapreduce

        show dbs                     show database names
        show collections             show collections in current database
        show users                   show users in current database
        show profile                 show most recent system.profile entries with time >= 1ms
        show logs                    show the accessible logger names
        show log [name]              prints out the last segment of log in memory, 'global' is default
        use <db_name>                set current database
        db.foo.find()                list objects in collection foo
        db.foo.find( { a : 1 } )     list objects in foo where a == 1
        it                           result of the last line evaluated; use to further iterate
        DBQuery.shellBatchSize = x   set default number of items to display on shell
        exit                         quit the mongo shell
```

Create Database

- Syntax
  - Switch databases: Use <database_name>
  - Display a database: db
  - Check a database list: show dbs

Create Database

- Example

```
> use mydb
switched to db mydb
> db
mydb
> show dbs
admin  0.000GB
local  0.000GB
```

Create Database

- *mydb* is not present in list.
- To display database, insert at least one document into it.

```
> use mydb
switched to db mydb
> db
mydb
> show dbs
admin  0.000GB
local  0.000GB
> db.myname.insert({"name": "My name"})
WriteResult({ "nInserted" : 1 })
> show dbs
admin  0.000GB
local  0.000GB
mydb   0.000GB
```

Drop Database

- Syntax
  - Delete selected database: db.dropDatabase()

Drop Database

- Example

```
> show dbs
admin  0.000GB
local  0.000GB
mydb   0.000GB
> use mydb
switched to db mydb
> db.dropDatabase()
{ "dropped" : "mydb", "ok" : 1 }
> show dbs
admin  0.000GB
local  0.000GB
```

Create Collection

- Syntax
  - Create a collection: db.createCollection(name[, options])
    - name = The name of the collection to create: string
  - When you insert documents, Mongodb creates collection
- The mongo shell does not accept the name of the collection containing a **space, hyphen,** or **starts with a number**
- Use an alternate syntax to refer to the collection
  - db[“big-data”].find()
  - db.getCollection(“big data”).find()

Create Collection

- Example

```
> use mydb
switched to db mydb
> db.createCollection("mycollection")
{ "ok" : 1 }
> show collections
mycollection
> db.bigdata.insert({"name" : "big data"})
WriteResult({ "nInserted" : 1 })
> show collections
bigdata
mycollection
```

Drop Collection

- Syntax
  - db.collection_name.drop()
- Example

```
> use mydb
switched to db mydb
> show collections
bigdata
mycollection
> db.mycollection.drop()
true
> show collections
bigdata
```

Insert Document

- Syntax
  - db.collection_name.insert(document)
  - db.collection_name.insertOne(document)
  - db.collection_name.insertMany([<document 1> , <document 2>, ...])
    - db refers to the current database
    - myCollection is the name of the collection.

Multi-line Operations

- The mongo shell waits for the closing parenthesis, closing brace, or the closing bracket before evaluating the code
  - If you end a line with (, {, or [, the subsequent lines start with ... until the corresponding), }, or]

Insert Document

- Example

```
> db.mycol.insertOne({"title": "MongoDB Lab2",
... description: "MongoDB is a NoSQL database system",
... by: "Seoul National University",
... tags: ["MongoDB", "Database", "NoSQL"],
... })
```

  - Check

```
> db.mycol.find()
{ "_id" : ObjectId("59e353651e647c00ea5c13b2"), "title" : "MongoDB Lab2", "description" : "MongoDB is a NoSQL database system",
"by" : "Seoul National University", "tags" : [ "MongoDB", "Database", "NoSQL" ] }
```

Exercise 1

- Insert documents into *people (lab2* database).
  - Confirm the insertion by db.people.find()

```
{ “name”: “Kim”, “age”: 21 },
{ “name”: “Lee”, “age”: 22 },
{ “name”: “Jung”, “age”:27},
{ “name”: “Park”, “age”: 27, “skills”: [ “mongodb”, “python”] },
{ “name”: “Choi”, “age”: 22, “score”: 10 }
```

- Delete a document whose name is *Jung*.
  - Using db.collection_name.remove()

Update Document

- Syntax
  - db.collection.update(filter, update[, options])
  - db.collection.updateOne(filter, update[, options])
  - db.collection.updateMany(filter, update[, options])
    - query = The selection criteria for the update : document
    - update = The modifications to apply : document

Update Document

- Update
  - *$set* to set the value of a field
    - { $set: { <field1>: <value1>, ... } }
  - *$unset* to set the value of a field
    - { $unset: { <field1>: “”, ... } }
  - $inc to increment a counter atomically
    - { $inc: { <field1>: <amount1>, <field2>: <amount2>, ... } }

Exercise 2

- Use the *people* collection in *lab2*
  - Replace a document that has the name *Lee* to
    - The name is Lim and the age is 25
  - Update a document that has the name *Kim*, setting *age* to 20
  - Remove the *skills* field in a document that has the name *Park*
  - Decrease the *score* field by 2 in a document that has the name *Choi*

Import Example Dataset

- import data from files in JSON, CSV or TSV
- Syntax
  - mongoimport –d *(database name)* –c *(collection name)* < *file name*
    - Note that you should check the file path

Backup & Restore

- Syntax
  - Backup: mongodump
  - Restore: mongorestore

Exercise 3

- Download the *fruit.zip* file
  - http://bdi.snu.ac.kr/academy/portal/index.php/bigdata-engineering-4-notice
  - Decompress a file using *unzip fruit.zip*
- *mongoimport –d lab2 –c apples < apple.json*
  - Go into the Mongo shell
  - Print all documents

Exercise 4

- Download the *dump.zip* file
  - http://bdi.snu.ac.kr/academy/portal/index.php/bigdata-engineering-4-notice
  - Decompress a file using *unzip dump.zip*
- *mongorestore dump*
  - Go into the Mongo shell
  - Perform find() method on the collection called *bigdata* in *lab2*. That will return a one document.
  - Check the value corresponding to the *answer* key from the document returned.

Tips

- Command history
  - Use the up/down arrow keys to scroll
- Tab Completion
  - Use <Tab> to autocomplete or to list the completion possibilities
  - The following example which uses <Tab> to complete the method name starting with the letter 'c‘:

```
> db.mycol.c
db.mycol.clean(                 db.mycol.convertToSingleObject(  db.mycol.createIndex(
db.mycol.constructor            db.mycol.copyTo(
db.mycol.convertToCapped(       db.mycol.count(
```

  - The <Tab> will list the various methods that start with 'c'.

Exit the shell

- Type quit() or exit
- Use the <Ctrl-C>

PyMongo

Contents

- Connecting to MongoDB with Python
- Insert Document
- Update Document
  - set, unset, inc

PyMongo?

- Python + MongoDB
- A Python distribution containing tools for working with MongoDB
- The recommended way to work with MongoDB from Python

Connecting to MongoDB with Python

- Create a MongoClient to the running mongod instance

  ```
  from pymongo import MongoClient
  client = MongoClient() or
  client = MongoClient("localhost", 27017)
  ```

  - By default, the *MongoClient* will connect to a MongoDB server on *localhost* at port *27017*

Connecting to MongoDB with Python

- Database list
  - client.database_name()

- Getting a database
  - db = client.test_database
  - db = client['test_database']

Exercise

- Print your database list

Connecting to MongoDB with Python

- Collection list
  - db.collection_names()

- Getting a collection
  - collection = db.test_collection
  - collection = db['test_collection']

Exercise

- Print collection list in *lab2*

Documents

- Using dictionaries to represent documents

```
import datetime
post = {"author": "Mike",
        "text": "My first blog post!",
        "tags": [“MongoDB", “Python", “PyMongo"],
        "date": datetime.datetime.utcnow()}
```

Insert documents

- When a document is inserted without *_id*, it is automatically added
  - The value of "_id" must be unique across the collection
- Syntax
  - db.collection.insert_one(document)
  - db.collection.insert_many([document1, document2,…])
  - MongoDB creates a collection if the collection does not exist

Exercise 5

- mongo_helloworld.py
  - Insert a document using *insert_one* into the *posts* collection (*lab2* database)
    - Author: your name
    - Text: Hello World!
  - Confirm the insertion by db.posts.find() using the mongo shell.

Exercise 6

- Insert documents into *pypeople* (*lab2*).
  - Using *insert_many*
  - Confirm the insertion by db.pypeople.find() using the mongo shell.

{ “name”: “Kim”, “age”: 21 }
{ “name”: “Lee”, “age”: 22 }
{ “name”: “Park”, “age”: 27, “skills”: [“mongodb”, “python”] }
{ “name”: “Choi”, “age”: 22, “score”: 10 }

Update Documents

- Syntax
  - db.collection.update(query, update[, options])
  - db.collection.update_one(query, update[, options])
  - db.collection.update_many(query, update[, options])
    - query = The selection criteria for the update : document
    - update = The modifications to apply : document

Update Documents

- Update
  - *$set* to set the value of a field
    - { “$set”: { <field1>: <value1>, … } }
  - *$unset* to set the value of a field
    - { “$unset”: { <field1>: “”, … } }
  - $inc to increment a counter atomically
    - { “$inc”: { <field1>: <amount1>, <field2>: <amount2>, … } }

Exercise 7

- Use the *pypeople* collection in *lab2*
  - Replace a document that has the name *Lee* to
    - The name is Lim and the age is 25
  - Update a document that has the name *Kim*, setting *age* to 20
  - Remove the *skills* field in a document that has the name *Park*
  - Decrease the *score* field by 2 in a document that has the name *Choi*

Exercise 8-1

- Grading
  - Given a file that has a zero or one (grade.txt)
  - If there are consecutive ones, the student takes +1 point

e.g.) total score = 11

1	0	1	1	0	1	1	1	0	1
1	0	1	2	0	1	2	3	0	1

  - The first line means the number of students
  - Insert the total score into the *grade* collection in *lab2*
    - Documents have *sid* (line number in the file and starting from zero) and *score*

Exercise 8-2

- Grading
  - Update documents (HINT: update_many)
    - If students get the perfect score, they take extra points (+5)
    - The student whose sid is 5 takes a minus point (-1)