Mongodb

MongoDB는 C++로 작성된 오픈소스 NoSql 데이터 베이스 입니다. NoSql 중에서는 가장 많이 사용되고 있는 데이터 베이스 입니다. NoSql은 관계형 데이터 베이스 RDBMS가 아니기 때문에 고정된 스키마 및 JOIN이 없습니다.

install

1. mac

brew를 이용해서 mongodb을 설치 합니다.

Reference
- https://gist.github.com/nrollr/9f523ae17ecdbb50311980503409aeb3

(1) 다운로드 및 설치

```
$ brew install mongodb
```

(2) 설정

```
$ brew tap homebrew/services
$ brew services start mongodb
$ brew services list
$ mongod --version
```

(3) mongo shell 실행

```
$ mongo
```

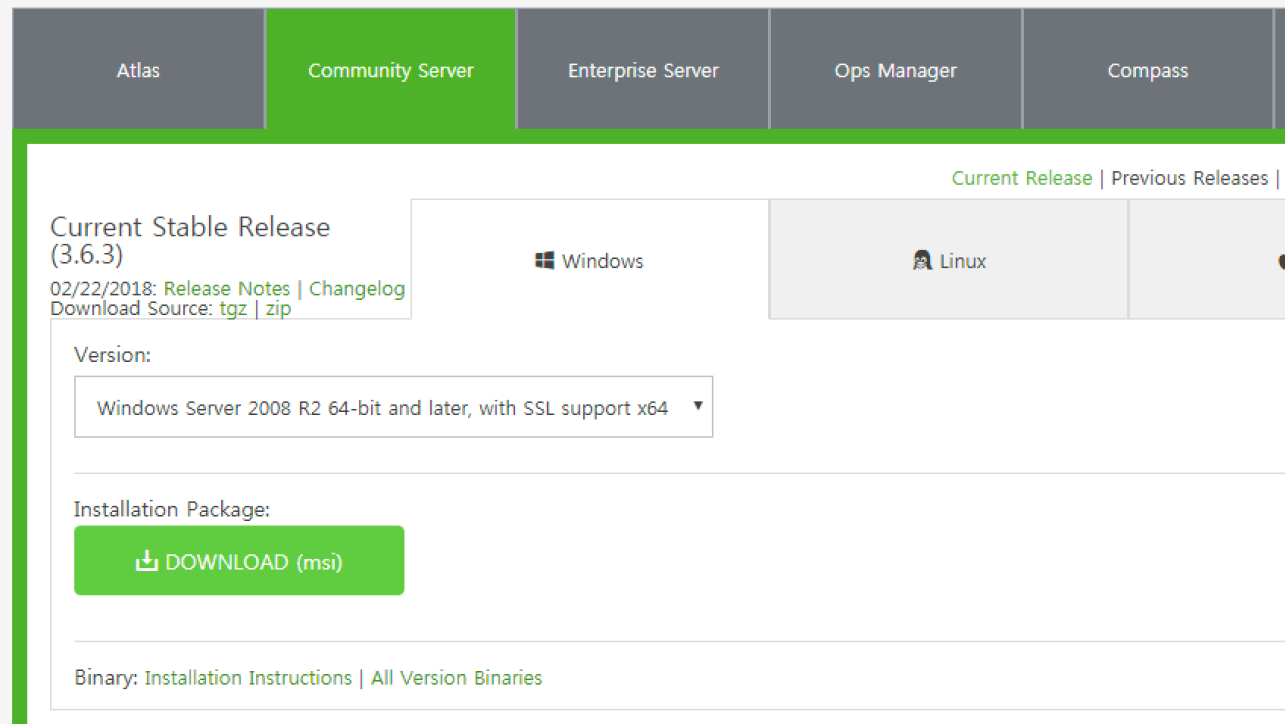
(4) mongodb 설정 확인
```
> db.serverCmdLineOpts()
```

2. windows

(1) 다운로드 및 설치

https://www.mongodb.com/download-center

위의 mongodb 공식 홈페이지에서 msi 파일을 다운 받아서 실행 시켜 설치 합니다.



* 설치 하실때 mongodb compass 를 설치 하지 않으셔도 됩니다.

(2) 환경변수 추가

설치를 하시고 나면 C:\Program Files\MongoDB\Server\3.6.\bin 에 설치가 됩니다. 설치가 완료된후 설치된 디렉토리를 환경변수에 추가해 줍니다.

(3) 데이터 베이스 저장 디렉토리 생성

mongodb를 설치하신 후에 C:\data\db 디렉토리를 만들어 주셔야 mongodb를 실행할수 있습니다. C:\data\db 경로는 mongdb가 데이터를 저장하는 디렉토리인데 mongodb를 설치하시면 디폴트로 데이터 저장경로가 C:\data\db로 설정이 되어있습니다.

경로 생성 방법

- cmd 창 열기
C:\User\~~> cd \
> mkdir data
data> cd data
> mkdir db

(4) mongodb 실행

cmd 창에서 아래와 같이 실행

> mongod

(5) mongodb shell

cmd 창에서 아래와 같이 실행
>mongo

basic syntax

자세한 내용은 아래 레퍼런스를 참고 하시면 됩니다.
https://docs.mongodb.com/v3.6/reference/

1. CRUD

(1) make database

dss 라는 이름의 데이터 베이스 생성
> use dss

현재 사용중인 데이터 베이스 확인
> db

database 리스트 확인
> show dbs
* 데이터 베이스 리스트에서 확인을 하려면 database에 최소 1개의 document를 추가해줘야 합니다.

document 생성
현재 사용중인 database에 user라는 collection을 만들고 document를 저장한다.
> db.user.insert({"name":"alice", "age":20, "email":"alice.gmail.com"})

dss라는 데이터 베이스가 생성된것을 확인
> show dbs

(2) delete database

현재 사용중이 데이터 베이스 삭제
> db.dropDatabase()
* 데이터 베이스를 삭제하기 전에 (> use <database name>) 으로 삭제 할 데이터 베이스가 선택된 상태에서 삭제 해야 합니다.

(3) make collection

reference
- https://docs.mongodb.com/v3.6/reference/method/db.createCollection/

name : collection 이름,
> db.createCollection(name, [option])

option
capped : true로 설정하면 collection의 최대 용량을 설정하여 collection의 최대 용량(size 옵션으로 설정)을 넘어서 데이터가 들어가면 오래된 데이터부터 자동으로 삭제
autoIndex : true로 설정하면 _id 필드에 index를 자동으로 생성
size : 숫자 데이터를 사용하며 collection의 최대 사이즈를 byte 단위로 지정
max : 숫자 데이터를 사용하며 최대 document 갯수를 설정
실습

dss 데이터 베이스를 생성
> use dss

옵션 없이 user 컬렉션을 생성
> db.createCollection("user")

autoIndex와 max 옵션을 설정하여 info 컬렉션을 생성
> db.createCollection("info", { autoIndexId:true, max:5 })

createCollection을 사용하지 않고 article 컬렉션을 생성
> db.articles.insert({"title":"data science", "contents":"mongodb" })

collection 리스트 확인
> show collections

(4) delete collection

> db.<collection name>.drop()

articles 컬렉션 삭제
> db.articles.drop()

(5) make document

> db.<collection_name>.insert(<document>)

info 컬렉션에 document를 추가
> db.info.insert({ "subject":"python", "level":3 })
> db.info.insert({ "subject":"web", "level":1 })
> db.info.insert({ "subject":"sql", "level":2 })

여래개의 document를 한번에 추가
> db.info.insert([{ "subject":"python", "level":3 }, { "subject":"web", "level":1 }, { "subject":"sql", "level":2 }])

collection의 document를 확인
> db.info.find()

(6) delete document

> db.<collection_name>.remove(<conditions>)

level이 2인 데이터 삭제
> db.info.remove({level:2})

2. find

reference
https://docs.mongodb.com/manual/reference/method/db.collection.find/index.html

db.collection.find(query, projection)

query : document 조회 조건을 설정. 모든 document를 조회 할때는 ({})를 사용
projection : document를 조회할때 보여지는 필드(컬럼)를 정의

query

(1) 기본 document 조회

```
# dss 데이터 베이스 선택
> use dss

# 추가 데이터 입력
> db.info.insert([
{ "subject":"r", "level":3, "comments": [{"name":"jin","msg":"bad"}, {"name":"alice","msg":"hi"}] },
{ "subject":"java", "level":1, "comments": [{"name":"po","msg":"check"}] },
{ "subject":"javascript", "comments": [{"name":"jin","msg":"nice"}] },
{ "subject":"html", "level":3, "comments": [] },
{ "subject":"css", "level":1, "comments": [{"name":"alice", "msg":"hello"}] },
{ "subject":"python", "level":2, "comments": [{"name":"alice", "msg":"hello"}] },
{ "subject":"python", "level":5, "comments": [{"name":"jin","msg":"nice"}] }
])

# info 컬렉션에 있는 모든 document 조회
> db.info.find()

# document를 깔끔하게 조회
> db.info.find().pretty()

# subject가 python인 document를 조회
> db.info.find({"subject": "python"})
```

(2) 비교 연산자

* 연산자 관련 reference
https://docs.mongodb.com/v3.6/reference/operator/query/

```
# level이 2 이하인 document를 조회
> db.info.find({"level": {$lte: 2} })

# level이 3 이상인 document를 조회
> db.info.find({"level": {$gte: 3} })

# subject가 java와 python을 포함하는 document 조회
> db.info.find( {"subject": {$in: ["java", "python"]}} )
```

(3) 논리 연산자

$or : 조건중 하나라도 true이면 true
$and : 모든 조건이 true이면 true
$not : 조건중 하나라도 false이면 true
$nor : 모든 조건이 false이면 true (or와 반대 개념)

subject가 python이고 level이 4이상인 document 조회
> db.info.find({ **$and**: [{ "subject":"python" }, { "level": {$gte: 4} }] })

subject가 python이거나 level이 4이상이 아닌 document 조회
> db.info.find({ **$nor**: [{ "subject":"python" }, { "level": {$gte: 4} }] })

level이 2보다 크지 않은 document 조회 (2 포함)
> db.info.find({ "level": { **$not**: {$gt: 2} } })

(4) $where 연산자

$where 연산자를 사용하면 자바스크립트 표현식 사용이 가능합니다.

comments가 없는 document 조회
> db.info.find({ **$where**: "this.comments.length == 0"})

comments가 1개 이상인 document 조회
> db.info.find({ **$where**: "this.comments.length >= 1"})

level이 1인 document 조회
> db.info.find({ **$where**: "this.level == 1"})

(5) $elemMatch 연산자

$elemMatch 연산자를 사용하면 document 안에 있는 element 조회가 가능합니다.

comments의 작성자가 alice인 document 조회
> db.info.find({"comments": {**$elemMatch**: {"name":"alice"}}})

comments 작성자가 alice이거나 jin인 document 조회
> db.info.find({"comments": {**$elemMatch**: {$or: [{"name":"alice"},{"name":"jin"}] }}})

아래와 같이 find로도 가능
> db.info.find({"comments.name": "alice"})
> db.info.find({$or: [{"comments.name": "alice"},{"comments.name": "jin"}]})

(6) $exists

$exists를 사용하면 특정 키값이 있거나 없는 document 조회가 가능합니다.

level 키값이 없는 document 조회
> db.info.find({ level: {$exists: false} })

projection

document를 조회할때 보여지는 필드(컬럼)를 정의합니다.

(1) basic

```
# subject와 comments만 출력되도록 find
# 설정을 true 값을 설정하던가 false 값을 설정합니다. (조건에 true와 false가 섞이면 에러)
> db.info.find({},{"_id":false, "level":false})
> db.info.find({},{"subject":true, "comments":true})
```

(2) $slice 연산자

document를 읽어올때 출력 갯수를 설정

```
# docuemnt에서 comments는 1개까지 나오도록 출력
> db.info.find({},{comments:{$slice:1}})
```

(3) $elemMatch 연산자

```
# coments 이름에 alice가 있는 document를 subject와 comments필드만 조회( 다른 사람 comment도 조회됨)
> db.info.find({ "comments": {$elemMatch: {"name":"alice"}}}, {"subject":true, "comments":true} )

# 조회된 document에서 alice의 comments 만 출력
> db.info.find(
        { "comments": {$elemMatch: {"name":"alice"}}},
        { "subject":true, "comments": {$elemMatch: {"name":"alice"}}}
)
```

3. find method

find method를 사용하면 find를 사용한 document의 결과를 가공하여 출력할수 있습니다.

(1) sort

document를 정렬시켜 줍니다.
'sort({key: value})' 와 같은 포멧으로 사용을 하며 key는 정렬할 필드명을 작성하고, value는 오름차순은 1, 내림차순을 -1을 넣어주면 됩니다.

```
# info 컬렉션의 document를 level 오름차순으로 정렬
> db.info.find().sort({"level":1})

# info 컬렉션의 document를 level 내림차순으로 정렬
> db.info.find().sort({"level":-1})

# level을 기준으로 내림차순으로 정렬한 후 id를 기준으로 오름차순으로 정렬
> db.info.find().sort({"level":-1, "_id":1})
```

(2) limit

limit을 사용하면 document출력 결과의 수룰 제한할수 있습니다.

```
# document의 결과를 5개 까지만 출력
> db.info.find().limit(5)

# document의 결과를 level로 내림차순으로 정렬하고 3개까지만 출력
> db.info.find().sort({"level":-1}).limit(3)
```

(3) skip

skip을 검색한 document의 결과의 시작부분을 설정할때 사용합니다.

```
# document를 2번째 부터 출력
> db.info.find().skip(2)
```

4. update

reference
https://docs.mongodb.com/manual/reference/command/update/index.html

db.collection.update(query, update, { upsert: <bool>, multi: <bool> })

upsert : insert와 update의 합성어 (데이터가 있으면 update, 없으면 insert 한다는 의미)
multi : true로 설정되면 여려개의 document를 수정합니다. 기본값은 false

```
# 특정 document를 새로운 document로 수정하기
> db.info.update(
        { "subject": "css" },
        {"subject":"sass", "level":2, "comments":{"name":"alice", "msg":"hello"}}
)

> db.info.update(
        { "subject": "less" },
        { "subject":"less", "level":2, "comments":{"name":"alice", "msg":"hello"}},
        { "upsert":true}
)
```

(1) $set, $unset

$set을 사용하면 특정 document의 필드를 수정할수 있습니다.
$unset를 사용하면 특정 document의 필드 제거할수 있습니다.

```
# python의 level을 3으로 수정 (한개의 데이터만 수정)
> db.info.update( { subject: "python" }, { $set: { level: 3 } } )

# python의 level을 4으로 수정 (여러개의 데이터 수정)
> db.info.update(
        { subject: "python" },
        { $set: { level: 4 } },
        { multi: true }
)

# subject가 java인 document의 level필드 삭제
> db.info.update(
        { subject: "java" },
        { $unset: {level: 1} }
```

```
)
* level: 1의 1은 true를 의미합니다.

# level이 3이하인 데이터를 1로 수정하기
> db.info.update(
        { level: {$lte: 3}  },
        { $set: {level: 1} },
        { multi: 1 }
)

# level이 없는 데이터 level 추가하기
> db.info.update(
        { level: {$exists: false}  },
        { $set: {level: 3} },
        { multi: 1 }
)
```

(2) $push, $each

$push를 사용하면 배열을 추가할수 있습니다.
$each를 사용하면 여러개의 배열 값을 추가할수 있습니다.

```
# subject가 java인 document의 comments 리스트에 comment 추가
> db.info.update(
        { subject: "java" },
        { $push: { "comments": {"name":"peter", "msg":"java is good programing language!!" } } }
)

# subject가 java인 document의 comments 리스트에 comment 추가
> db.info.update(
        { subject: "java" },
        { $push: { "comments": {
                        $each: [{"name":"jin", "msg":"echo jin"},{"name":"alice", "msg":"i'm alice"}]
                        }
                }
        }
)
```

(3) $pull, $in

$pull은 배열의 값을 제거할때 사용합니다.
$in은 여러개의 배열 값을 선택할수 있습니다.

```
# subject가 java인 document에서 comments에서 name이 jin comment를 삭제합니다.
> db.info.update(
        { subject: "java" },
        { $pull: { "comments": {"name":"po"} }
        }
)

# subject가 java인 document에서 comments에서 name이 alice, po comment를 삭제합니다.
> db.info.update(
        { subject: "java" },
        { $pull: { "comments":
                        { $in: [{"name":"jin", "msg":"echo jin"}, {"name":"alice", "msg":"i'm alice"}] }
                }
        }
```

)

5. function

자바스크립트 문법으로 함수를 작성하여 사용이 가능합니다.

```
# skip 함수
var showSkip = function(start){
        return db.info.find().skip(start)
}

# limit 함수
var showLimit = function(end){
        return db.info.find().skip(end)
}
```

6. quiz

(1) page와 block을 설정해서 결과를 나타내는 함수를 작성

```
var showPage = function(page, block){
        # TODO
}
```

아래와 같이 함수를 호출하면 2페이지의 페이지당 document는 3개이므로 index가 3,4,5인 세개의 document가 출력되어야 합니다.

```
showPage(2, 3)
```

3. robomongo

https://robomongo.org/download

위의 주소에서 본인에게 맞는 OS의 robomongo 설치 파일을 다운 받아 실행해 줍니다.