MongoDB

0.설치하기

for window

mongodb : https://www.mongodb.com/
try free -> server -> download -> install
* mongodb compass 는 check 해제

환경변수 path에 '설치경로\bin' 추가

C:\data\db 폴더 생성 [다른 경로에 생성 가능]
> mongod [--dbpath 다른 경로]
// mongo server(mongod) 실행

> mongo
// mongo client 실행

* api-ms-win-crt-runtime-l1-1-0.dll 에러 발생시 다음 링크에서 업데이트 다운로드 및 설치
https://support.microsoft.com/ko-kr/help/2999226/update-for-universal-c-runtime-in-windows

* mongod service 등록
mongod.exe [--dbpath 다른경로] --install --serviceName MongoDB --serviceDisplayName MongoDB

for mac (homebrew)

참조 : https://github.com/mongodb/homebrew-brew

$ brew tap mongodb/brew
$ brew install mongodb-community

<mongodb-community path>
* configuration file : /usr/local/etc/mongod.conf
* log directory path : /usr/local/var/log/mongodb
* data directory path : /usr/local/var/mongodb

$ brew services start mongodb-community
$ mongo

* service를 종료할 때는..
$ brew services stop mongodb-community

[database server / client]

	MongoDB	MySQL	Oracle	Informix	DB2
Database Server	mongod	mysqld	oracle	IDS	DB2 Server
Database Client	mongo	mysql	sqlplus	DB-Access	DB2 Client

1. MongoDB란?

관계형 데이터베이스(rdbms)가 아니라 문서(document) 지향 데이터베이스이다.
- data를 행(row) 대신 document에 저장한다.
- 고정된 형태의 스키마(schema)가 없이 구조화된 document를 데이터베이스에 저장할 수 있다.
* 문서의 형식은 JSON(JavaScript Object Notation)이다.

1-1. mongo shell
interactive javascript interface. shell은 javascript interpreter로, 임의의 javascript program을 실행 가능
- javascript library 기능 활용 가능 (function 활용 가능)

1-2. document
data record를 bson document로 저장하여 사용. mongodb의 기본 단위
- bson : JSON에 없는 type(ex. data, binary data, ...)을 추가
- "_id" : primary key. 생략 시 ObjectId type으로 자동 생성

1-3. collection
collection은 document의 모음이다. (관계형 데이터베이스의 table과 같다.)
- 하나의 collection 내 document들이 모두 다른 구조를 가질 수 있다.
- index는 collection 별로 정의한다.

collection 생성 방법
- db.createCollection(name,[option]) : name과 option을 지정하여 생성
- db.users.insert({name:"test"}) : insert 명령을 통해 자동으로 생성

1-4. database
database는 collection의 그룹이다.
- 하나의 database는 자체 권한을 가지고 있으며, 따로 분리된 파일로 디스크에 저장된다.
- 데이터베이스의 이름은 파일시스템 상에서 파일이 된다.

* 명령어
참조 : https://docs.mongodb.com/manual/reference/mongo-shell/
- 전체 database 확인
> show dbs
- 현재 사용하고 있는 database 확인
> db
- database 변경
> use [db name]
- 현재 database에서 생성한 collection 확인
> show collections

[예약된 database 이름들]
* admin: root database.
- admin에 사용자를 추가하면, 해당 사용자는 자동으로 모든 데이터베이스에 대한 사용 권한을 상속.
- 서버 전역에 걸쳐 실행하는 명령어들은 admin에서만 실행 가능
* config: 샤딩하는 경우, config는 내부적으로 샤드 정보를 저장하는데 사용
* local: 절대로 복제되지 않음. 특정 서버에만 저장하는 collection에 사용

2. CRUD
2-1. create
```
> db.users.insertOne(                 // collection
  {                                   // document
    name: "sue",                      // field : value
    age: 26,                          // field : value
    status: "pending"                 // field : value
  }
)
```

2-2. read
```
> db.users.find(                      // collection
  { age: { $gt: 18} },                // query criteria      (조건)
  { name: 1, address: false}          // projection          (select 할 field)
).limit(5)                            // cursor modifier     (출력될 개수)
```

* pretty() 를 사용하면 보기 편하게 출력
> db.users.find().pretty()

- query 부분에 조건을 넣으면 해당 조건에 맞는 document를 리턴
  and : {field: "값", ... }
  or : {$or: [{field: "값", ... }] }
  {}인 경우 모든 document 출력
- projection 부분에 출력하고 싶은 field와 1(true) or 0(false)를 넣으면 1(true)인 field 만 출력
- 정렬 : find().sort({field:1}) (1 : 오름차순, -1:내림차순)

2-3. update

```
> db.users.updateMany(                  // collection
  { age: { $lt: 18} },                  // update filter          (변경할 document 조건)
  { $set: { status: "reject"} }         // update action          (변경 값)
)
```

- filter 부분에 조건을 넣으면 해당 조건에 맞는 document를 수정
- 사용 메소드
  updateOne()
  updateMany()
  replaceOne()

2-4. delete

```
> db.users.deleteMany(                  // collection
  { status: "reject" }                  // delete filter
)
```

- filter 부분에 조건을 넣으면 해당 조건에 맞는 document를 삭제
- 사용 메소드
  deleteOne()
  deleteMany()

operators : https://docs.mongodb.com/manual/reference/operator/query/

3. 집계

3-1. 집계 (aggregate stage)

pipeline : 이전 단계의 연산결과를 다음 단계의 연산에 사용 가능. document의 흐름을 받아들이고,변환하여 변환 결과를 전달하고 마지막 pipeline operator의 결과 리턴

- pipeline stage는 배열로 나타낸다. document는 순서대로 stage를 통과하며, $out 및 $geoNear 를 제외한 모든 stage는 여러 번 사용 가능하다
  stage : https://docs.mongodb.com/manual/meta/aggregation-quick-reference/
  $field : field 참조 시 사용 ($$current.field) 와 같다
  pipeline aggregate operators : https://docs.mongodb.com/manual/reference/operator/aggregation/#aggregation-expression-operators

```
db.users.aggregate([
  {
    $match: {age:{$gt:30}}                          // first stage
  },
  {
    $group: {_id:"avg_id", avg:{$avg:"$age"}}       // second stage
  }
])
```

SQL	Aggregation
WHERE	$match
GROUP BY	$group
HAVING	$match

SELECT	$project
ORDER BY	$sort
LIMIT	$limit
SUM()	$sum
COUNT()	$sum

* operator의 순서에 따라 효율이 달라진다. ($sort -> $match 보다 $match 후 $sort 등)

3-2. 맵리듀스(map reduce)
query 조건을 통해 입력된 document를 제어하고, javascript function을 통해 map reduce를 수행할 수 있다.
map-reduce 연산은 javascript function으로 사용
- 순서 : query -> map -> reduce -> out

```
> var map = function(){
  emit(this.class, this)
}

> var reduce = function(key, values){
  var result = {class:key, avg:0};
  var cnt = 0;
  values.forEach(function(v){
    result.avg += (v.kor+v.eng+v.math)/3
    cnt +=1
  })
  return result.avg/cnt
}

> db.score.mapReduce(
  map,
  reduce,
  {
    query: {class: {$ne: "s"}},
    out: "score_avg"
  }
)
```

- map : 입력된 document의 특정 값을 key에 mapping (grouping)
- reduce : mapping된 값들을 연산
- query : collection에서 특정 document를 입력으로 사용
- out : collection으로 출력

4. 인덱스
collection의 데이터 집합 중 작은 부분을 횡단하기 쉬운 형태로 저장하는 특수한 데이터 구조.
- _id field는 자동으로 index 설정
- index는 b-tree 구조 사용

4-1. 타입
- single field : 해당 field에 index 설정
```
> db.score.createIndex({name:1})                  // 1:오름차순, -1: 내림차순
```
- compound index : 복합 인덱스
```
> db.score.createIndex({name:1, avg:1})           // name으로 정렬 후 각 name값 내에서 avg로 정렬
```
- multikey index : field type이 배열일 때 사용
- geospatial index : 2d, 2dsphere 지정하여 index 설정(geoJSON 객체 지원)
- text index : 중지 단어(the, a, or 등)을 제외하고 index 설정 (효율적인 문자열 검색)
- hashed index : field값의 hash를 설정

4-2. 속성
- Unique : 유일한 하나의 값만 존재 가능
- Partial : 일부 document에만 인덱스 적용
- Sparse : index로 선언된 key값이 없는 경우 무시
- TTL : 특정 시간 후 document 자동 제거

5. python + mongodb
pymongo : https://api.mongodb.com/python/current/

```
> pip install pymongo

import pymongo
# MongoClient('ip',port)
client = pymongo.MongoClient('127.0.0.1', 27017)
# client.db / client[db]
db = client.test
# db.collection / db[collection]
score = db['score']
# cursor 안에 document가 모두 들어있다.
result = score.find()
# cursor 안에 있는 document 반복하여 출력
for res in result:
    print(res)
```

6. spring + mongodb
mongo java driver : https://mongodb.github.io/mongo-java-driver/

6-1. pom.xml
```
<dependency>
  <groupId>org.mongodb</groupId>
  <artifactId>mongodb-driver-sync</artifactId>
  <version>3.10.2</version>
</dependency>
<dependency>
  <groupId>org.springframework.data</groupId>
  <artifactId>spring-data-mongodb</artifactId>
  <version>2.1.10.RELEASE</version>
</dependency>
```

6-2. mongodb-context.xml
```
<mongo:mongo-client id="mongo" host="127.0.0.1" port="27017"></mongo:mongo-client>
<bean id="mongoTemplate" class="org.springframework.data.mongodb.core.MongoTemplate">
  <constructor-arg ref="mongo"/>
  <constructor-arg name="databaseName" value="test">
</bean>
```

6-3. DTO
@Id private String id (ObectjId 값 받을 field)

6-4. DAO
mongoTemplate.findAll(dto.class, collectionName)