①

∘ MongoDB. ~ Mongo University (chapter 1~2).

Flynn/Flynn.
("이[illegible] + 구글링")

NoSQL DB.
stored as documents"

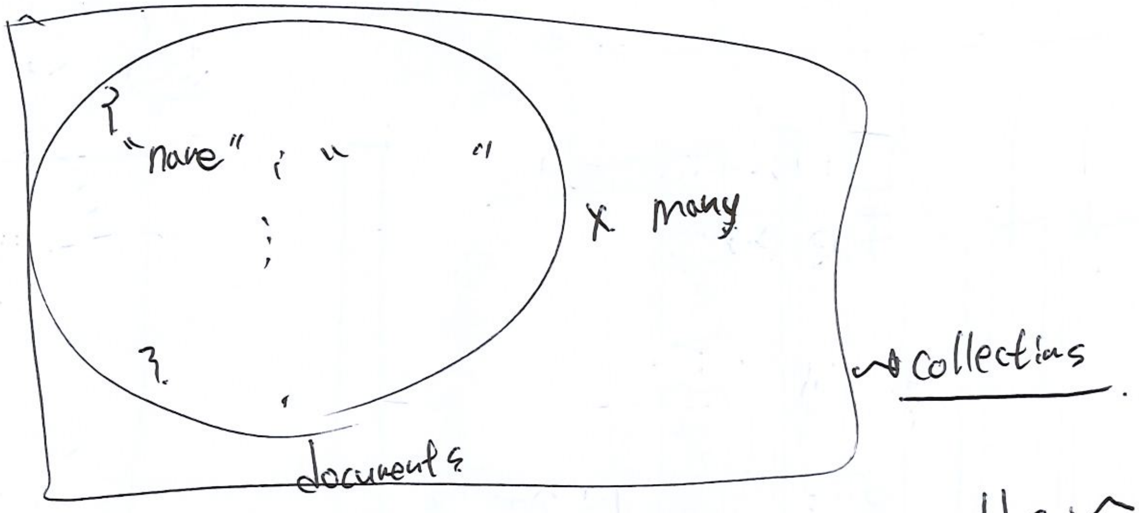
documents 들의 collection
(doesn't utilize tables (rows and columns)
(documents 들의 ~~I~~ collection. (structured way to store and access data)

∘ What is documents. ?

{
<field> : <value>,
:

ex)
"name" : "Flynn"
}

<key, value> 쌍으로 저장.



dbs ~ collections ~ docus.
:

②

JSON format (Encoding UTF-8.
Data Support (String
Boolean
Number
Array

{ key : value
key : value
:

{
key : value
key : value
:

Readability (Human and Machine)

key : value

}

→ String은 " ", 정수는 숫자는 없음.

keys must be surrounded by " "

key : {
key : value
:
}
→ Sub-Document

Pros	Cons
Friendly	Text-based
Readable	Space-Con
Familiar	Limited.

BSON (JSON format → Binary)

Binary JSON
(Speed
Space
Flexibility.

Encoding. Binary

Data support (String
Boolean
Number. (Integer, Long, Float, ...)
Array
Date
Raw Binary

* Export local machine
→ JSON < mongoimport
mongoexport

Export different System.
→ BSON. < mongorestore
mongodump. >

Export

mongodump --uri "<Atlas Cluster URI>" //BSON

mongoexport --uri "<Atlas Cluster URI>"

-- collection = <collection name>

-- out = <filename>.json.

// JSON. → import/Export 방법 다시정리 필요,,

--drop

o URI string

Added username password → mongodb+srv://user:password@clusterURI.mongodb.net/database

Atlas cluster, dbname

JSON Query {"state": "NY", "City": "ALBANY"}

NY ALBANY and → query가 날라감,

→? SQL style 아는? order by 는?

group by 등등은 document DB에선 어떻게 부를까?

~~Atlas~~ MongoDB Query, → 강의를 계속진행하면 나옴,,

질문을 한번에 하는게 좋은것 같음,,

show dbs

= du 명령어랑 비슷..

* find() command.

db.<collectionname>.find({"state": "NY"})

NoSQL query

How many ?

.count()

.pretty

~~.Pretty~~

it

Iterates through a cursor

cursor

A pointer to a result set of a query

pointer

A direct address of the memory location.

* find

↳ Documentation 찾아보기 !!

find documentation.

proto type

db.collection.find(query, projection)

이부분을 생략하거나, {}로 주면

select * from table.

모든 documents가 나옴.

projection

{<field1> : <value>, <field2>: <value>...}

prototype.

<field>: <1 or true>

<field>:<0 or false>

"<field>.$" : <1 or true>

<field> : <array projection>

<field> : <$meta expression>

<field>: <aggregation expression>

↳ need drill down.

Chapter 3 : Creating & Manipulating Documents. (atlas ui)

* Every document must have a unique _id value.

{ "_id : "1a" } { "_id" : "1b" } { "_id" : "4c" } ...

- definitely not a very good use of memory space.
- All doc

* M320 data modeling course" 수들을것.

ObjectId() : Default value for the _id field unless otherwise specified

⑥ Inserting Documents : Errors.

error code : 11000
└ Duplicate "_Id"

_Id를 지우고 insert하면 insert가 되나, pk지정하듯이 _Id가 지정되어 자동으로 붙어 되므로, Integrity 안깨지게 조심..

Tips : db.collection.findOne()

└ 컬렉션 shape 찾을때(볼때) 유용할듯.

⑥ Inserting Documents : Order.

* db.collectionname.insert(
[{ test : 1 }, { "test" : 2 }, { "test" : 3 }]
└ 배열에 doc를 담아서 insert가능.

[ ], { "ordered" : false }

Tips. ~~db~~ collection 네임 오타나면, 틀린 collection을 바로 만들어 버림. 주심할것.

→ _Id와 order 관계 // Insert가 왜 되는지 안되는지 확실히 check

⑥ Update Documents (MQL).

```
db. zips. updateMany({"city":"HUDSON"}, {"$inc": {"pop":10}})
```

increment.

(key) pop을 ⑩만큼 증가 (value.)

... 문법이익숙하진않음.. 연습.

```
db.zips.updateOne({"zip":"12534"}, {"$set":{"pop":17630}}) ... ok.
```

```
{"$set":{"population":17630}}) ...ok.
```

? update 과서가능한듯. population은 없는 field였어도 추가되버림..
"조심해야함.

```
db.grades.updateOne({"student_id":250,
                     "class_id":339}, {"$push":{"score":{"type":"extra credit", "score":100
                                                                                    }}})
```

* array 형태를가지는 element. add 할때.
adds an element to an array field.
? 만약에 element 형식이 맞지 않으면(같은 shape이 아니면). → googling.

◎ Deleting Documents.

* db 안에 collection을 모두 지우면, db도 삭제된다.

- db.collectionname.drop()

deleteOne() / deleteMany()

→ Caution!!

After these commands is issued the data is GONE.

* One / Many - function 비교. (find, insert, update, ~~del~~ drop)

→ document 찾아서 정리. 해야함.

→ 잘되어있음, 공식문서 → 정리할것.

- 비교연산자 / Indexing and Aggregation / modeling / transaction (3/5)