3. 쿼리 작성하기

차례

1. 쿼리 구조

- 실습준비

https://github.com/Karoid/mongodb_tutorials.git

area.json, # 전국 지역 정보와 인구수
by_month.json, # 각 지역 교통사고 통계 월별 정리
by_road_type.json, # 각 지역의 도로 형태별 사고 통계
by_type.json # 각 지역 사고 형태별 통계
- 다운받아 nosqlboster에 import 함

1. 쿼리 구조

- 값으로 쿼리하기

```
db.containerBox.find({name:'cat'})
```

- 연산자로 쿼리하기

```
{<filter> :{<operator> : <value>},....}
{height : {$gte: 175, $lte :180}
{height : {$gte: 175}, weight:{$lte :180}}
{$or [{ status : 'A'}, {qty:{$lt:30}}]}
{'name.first" : "Karoid", 'name.last': 'Jeong'}
```

1. 쿼리 구조

- 실습하기
  - 강릉시 교차로 내에서 일어난 총 사고 숫자를 출력하라
  - 전국에서 도로 종류 중에 '기타단인로'에서 사만자지수가 0인 지역을 출력한다.

```
db.by_road_type.find({county:'강릉시'},{'교차로내.accident_count':1})

db.by_road_type.find({'기타단일로.death_toll':0}, {city_or_province:1, county:1})
```

2. 논리, 비교 연산자

- 비교 및 논리 연산자

종류	유형	
비교 연산자	$cmp	두 값을 비교하여 앞의 값이 크면 양수, 작으면 음수, 같으면 0을 return 함
	$eq	두 값이 같으면 True, 다르면 False
	$gt	앞의 값이 크면 True, 작거나 같으면 False
	$get	앞의 값이 크거나 같으면 True, 작으면 False
	$lt	앞의 값이 작으면 True, 크거나 같으면 False
	$lte	앞의 값이 작거나 같으면 True, 크면 False
	$ne	두 값이 다르면 True, 같으면 False
	$in	주어진 배열 안에 속하는 값
	$nin	(not in)
Boolean 연산자	$and	모두 True이면 True
	$or	하나라도 True이면 True
	$not	해당 결과값의 반대
	$nor	주어진 값들 중에 하나라도 만족하지 않는 값을 찾는다.

2. 논리, 비교 연산자

- 실습

```
db.inventory.insertMany([
 { item: "journal", qty: 25, tags: ["blank", "red"] },
 { item: "notebook", qty: 50, tags: ["red", "blank"] },
 { item: "paper", qty: 100, tags: },
 { item: "planner", qty: 75, tags: ["blank", "red"] },
 { item: "postcard", qty: 45, tags: ["blue"] }
]);
```

```
db.inventory.find({item:{$eq:'journal'}})
db.inventory.find({item:'journal'})
db.inventory.find({tags: {$in : ['red']}})
db.inventory.find({tags: {$nin:["blank","blue"]}})
db.inventory.find({tags: {$in : [/^[a-z]*d/]}})
db.inventory.find({tags: {$in : [/^b/]}})
db.inventory.find({qty:{$not:{$gt:50}}})
db.inventory.find({qty:{$lte:50}})
db.inventory.find({$or:[{qty:{$gt:90}},{qty:{$lt:50}}]})
db.inventory.find({$and:[{qty:{$gt:50}},{qty:{$lt:90}}]})
db.inventory.find({qty:{$gt:50,$lt:90}})
```

2. 논리, 비교 연산자

- 실습하기
  - 전국의 '차대차' 사고에서 100회 이상 사고가 났지만 사망자 수가 0회인 지역을 추출한다.
  - 전국의 '차대사람' 사고 중에서 사망자수가 0회이거나 중상자수가 0회인 지역을 출력하라.

3. 문자열 연산자

- 연산자

```
$regex 정규식
$text 문자열 검색
```

- $regex 연산자

```
{<field> : { $regex : /pattern/, $options : '<options>'}}
{<field> : { $regex : 'pattern', $options : '<options>'}}
{<field> : { $regex : /pattern/<options>}}
```

```
db.article.find({"title": /article0[1-9]/i})
db.inventory.find({item:/^p/i}) #m 사용
```

- $regex 연산자

```
i : 대소문자 무시
m: 대소문자 구분
x: 정규표현식에 있는 공백 모두 무시
s : 점(.)을 하용할 때 \n을 포함해서 매치
```

3. 문자열 연산자

- $text 연산자

```
{
  $text: {
    $search: <string>,
    $language: <string>,
    $caseSensitive: <boolean>,
    $diacriticSensitive: <boolean>
  }
}
```

- 실습

```
db.stores.insert(
   [
     { _id: 1, name: "Java Hut", description: "Coffee and cakes" },
     { _id: 2, name: "Burger Buns", description: "Gourmet hamburgers" },
     { _id: 3, name: "Coffee Shop", description: "Just coffee" },
     { _id: 4, name: "Clothes Clothes Clothes", description: "Discount clothing" },
     { _id: 5, name: "Java Shopping", description: "Indonesian goods" }
   ]
);

#Text index
db.stores.createIndex({name:"text",description:"text"})
```

$text Operation

```
db.stores.find( { $text: { $search: "java coffee shop" } } )
```

- $text 연산자
  - ***$text Operation***

```
db.stores.find( { $text: { $search: "java coffee shop" } } )
```

  - Exact Phrase : 정확하게 일치하는 문서 찾기

```
db.stores.find( { $text: { $search: "\"coffee shop\"" } } )
```

  - Term Exclustion : "-"연산자를 사용하여 검색에 제외할 텀을 지정

```
db.stores.find( { $text: { $search: "java shop -coffee" } } )
```

  - sort

```
db.stores.find(
...     { $text: { $search: "java coffee shop" } },

...     { score: { $meta: "textScore" } }

... ).sort( { score: { $meta: "textScore" } } )
```

4. 배열 연산자

- 배열 연산자 소개

$all 순서와 상관없이 있으면 선택
$elemMatch 조건과 맞는 배열 속 요소를 가진 것을 선택
$size 해당 배열의 크기와 같은 것 선택

```
item: "book", tags: ["blank", "red"]}
{item: "book", tags: ["red", "blank"]}
{item: "book", tags: ["blue"]}

// tags에 red가 들어간 도큐먼트 전부 출력
db.inventory.find({tags: "red"})

// tags가 "red", "blank" 둘 다 주어진 순서대로 가진 도큐먼트 전부 출력
db.inventory.find({tags: ["red", "blank"]})
```

4. 배열 연산자

- $elemMatch
  - **배열의 요소 별로 해당 조건을 적용**

```
db.collection.find({tags: {$gt: 10, $lt: 5}})
db.collection.find({tags: {$elemMatch: {$gt: 10, $lt: 5}}})



db.by_month.find({$and: [
    {month_data: {$elemMatch: {month: "01월", heavy_injury: 0}}},
    {month_data: {$elemMatch: {month: "02월", death_toll: 0}}}
 ]})
```

4. 배열 연산자

- $all

```
// 순서가 중요함
b.inventory.find({tags: ["red", "blank"]})

// $all을 쓰면 "red", "blank" 순서와 상관 없이 해당 요소가 있는지만 확인
db.collection.find({tags: {$all: ["red", "blank"]}})
```

- $size

```
db.collection.find({tags: {$size: 3}}) //배열 lenth가 3인 문서
```

5. 프로젝션 연산자

- 프로젝션 연산자의 역할
  - 특정 필드만 가져옴

$slice : 배열 필드에 주어진 범위
$elemMatch 배열 필드의 조건에 맞는 것만
$ 첫번째 요소만

```
db.people.insetMany(
{name: {first: "철수", last: "김"}}
{name: {first: "영희", last: "김"}}
{name: {first: "수영", last: "박"}}
{name: {first: "희영", last: "이"}}
)

db.collection.find({}, {"name.first": 1})
```

5. 프로젝션 연산자

- $slice

```
// {item: "book", tags: ["red", "blank"]}
// 잘못됨. tags의 첫번째 인자[0]가 아니라 tags 배열의 0이란 원소를 출력하라는 의미
db.collection.find({}, {"tags.0": 1})

// tags 배열의 [0], [1]을 출력하라 (앞에서 부터 2개를 출력하라)
db.collection.find({}, {tags: {$slice: 2}})

// tags 배열의 [2:3] 을 출력하라
db.collection.find({}, {tags: {$slice: [2, 3]}})
```

5. 프로젝션 연산자

- $elemMatch

```
#특정 조건에 부합하는 필드만 출력하라
db.collection.find({}, {$elemMatch: {$regex: /^b/}})
```

- $ 연산자

```
#특정 조건에 부합하는 첫번째 데이터만 출력하라
db.collection.find({{tags: “red”}, {“tags.$:true}})
```