

Chapter 4 (CPTSD advanced)

• MGL operators (comparison)

- Update Operators

\$inc, \$set, \$unset...

\$ - multiple uses...

Precedes MGL operators

Precedes Aggregation pipeline stage

Allows Access to Field Values

— Query Op.

⑥ $\$eq \leftrightarrow \ne (not equal) (Comparison)

$\$gt \leftrightarrow \lt (less than)

$\$gte \leftrightarrow \lte (less than or equal to)

$\{ <field> : \{ <operator> : <value> \}$

ex) $\{ "tripduration" : \{ "lte" : 60 \} \}$

Explicitly use \$and when you need to include the same operator more than once in a query.

— Query OP. (logic)

— and, or, nor

({ <operator> : {

negation.

(query requires)

[{ statements } , { statements }]

...

{ \$not : { statements } }

— imply negation.

⑥ Implicit and.

{ sector : "Mobile Food Vendor-881", result :

"warning" }

In the same as:

{ "and" : [{ sector : "Mobile Food Vendor-881",

result : "warning" }] }

{ "student_id" : { \$gt : 25, "alt" : 100 } }

~~Query plan in Mongo DB (Explain)~~

Expressive Query Operator

\$expr : allows the use of aggregation expressions within the query language.

{ \$expr : { <expression> } }

\$expr allows us to use variables and conditional statements.

Same docu Compne 7/5

{ "\$expr" : { "\$eq" : ["\$start station id", "\$end station id"]] }

Operator

addresses the field value

field value is value
operator is value
operator is value

* A closer look.

{ "\$expr" : {
 "\$and" : [
 { "\$gt" : ["\$trip duration", 1200] },
 { "\$eq" : ["\$end station id", "\$start station id"] }
]
}

9

? \$set?

SQL syntax.

{ \$field > : { <operator> : { <value> } }

Aggregation syntax.

{ <operator> : { <field> { <value> } }

}

2

Array Operator

\$push

- ① Allows us to add an element to an array
- ② Turns a field into an array field if it was previously a different type.

```
{<array field> : { "$size": <number> } }  
→ { "x": 1, "y": 1, "z": 1 } & specified & returns a cursor
```

```
{<array field> : { "$all": <array> } }
```

which the specified array field contains all the given elements "regardless of their order in the array"

Querying an array field using

An array returns only exact array matches.

Where "field = value" → { "x": 1, "y": 1, "z": 1 } (documents 1/2)

Array Operator and Projection

db.distingsAndReviews.find(

{ "amenities": { "\$size": 20,

"ball": ["Internet", "wifi", "Kitchen",

...] } }

{ "price": 1, "address": "11" }, pretty.

the

true

* only include the price and address

fields in the cursor result

and condition.
→ projection condi?

{ "price": 1, "address": "11" }, pretty.

db.grades.find({ "class_id": 431 },

{ "scores": { "\$elemMatch": { "score": { "\$gt": 85 } } }, pretty.

\$elemMatch 78

→ scores parton 2024.10.10 2024.10.10
score

* \$elemMatch 3 조건여기까지 2024.10.10 2024.10.10

Querying Arrays and Sub-Documents

→ Dot notation (notation of sub-document)

* syntax.
db.companies.find({relationships: {person: {last_name: "Zuckerberg"}},
{"name": 1}).pretty()

0: position of the first array element

person: field name with a nested object as a value.

last_name: field name within the "person" sub-document

"Zuckerberg": value that we are looking for

{ "name": 1 } : projection to only include the company name in the resulting cursor.

* MongoDB command to find documents with a specific value in a field.

*Syntax 7/15

db.companies.find({relationships: {person: {first_name: "Mark",
"relationships.0.title": {~~\$regex~~: "CEO"}},
"name": "13").pretty() refer - C.8 (12) page

db.companies.find({relationships: {\$elemMatch: {

"is-past": true,

"person.first_name": "Mark" } }},

{ "name": "13").pretty() }

Summary

- MongoDB uses dot-notation to specify the address of nested elements in a document.

- To use dot-notation in arrays specify the position of the element in the array.

✱ db.collection.find("field 1", other field, also a field : "value")

10

{ "amenities": "Wifi" }

```
{ "price": 1, "address": 1, "id": 0 } .pretty
```

db. Distings. And Reviews, aggregated

```

} &match: { "amenities": "Wifi" } }

```

```
{ project: { "price": 1, "address": 1, "id": 0 } }
```

Pipe zlobolek

MongoDB Aggregation Framework

~~Group~~compute

recharge

12

Site filter

update

group:

?
-id: "address.country", // group by expression

<field>: { <accumulation1> : <expression1> },
 accumulator-1

3

$$Z$$

db. listings And Reviews.aggregate()

```
{ $project: { "address": 1, "_id": 0 } }  
{ $group: { "_id": "$address.country",  
  "count": { "$sum": 1 }  
}  
}
```

1)

* sort() and limit() aggregation sort() sort order?



```
db.zipcodes.find().sort({ "pop": 1 }) "city": -1 }
```



Decreasing
"city": "z" -> "A"

* Cursor methods
sort(), limit(), pretty(),
count(), skip()
가시어주는 result set 가시화
연산량. 객체 / 크기 보충
작업이 안됨!
* limit, limit은 도큐먼트 개수
제한
skip()은 포락할 디멘션(각
비율 선정)
sort -> limit
? limit sort vs

~~Index~~

easy syntax:

```
db.trips.createIndex(2"birth year":13)
```

- db better -

◦ Single field index

```
db.trips.createIndex(1"birth year":13)
```

◦ Not perfect for

```
db.trips.find(1"start station id":476).sort(2"birth year":1)
```

◦ Compound Index

```
db.trips.createIndex(1"start station id":1, 2"birth year":13)
```

- MongoDB Performance Course

◦ Build the right indexes

◦ Learn how MongoDB picks an index

◦ Improve query performance

◦ Learn about other index types

* MongoDB University > 데이터베이스 성능 향상

◦ 원본의 4가지 문제 or 구조를 searching 하는 것

Logic

Introduction to Data Modeling.

Rule: Data is stored in the way that it is used.

Data that is used together should be stored together.
Evolving application implies an evolving data model.

Topic: Insert - update or insert.

→ 72%

* 이론: modeling, transaction, performance. 실용.

진행, basic MongoDB is Done. (need to practice).